
Amazon Relational Database Service

User Guide

API Version 2014-10-31



Amazon Relational Database Service: User Guide

Copyright © 2019 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is Amazon RDS?	1
Overview	1
DB Instances	1
Regions and Availability Zones	2
Security	2
Monitoring an Amazon RDS DB Instance	3
Amazon RDS Interfaces	3
AWS Management Console	3
Command Line Interface	3
Programming with Amazon RDS	3
How You Are Charged for Amazon RDS	3
What's Next?	4
Getting Started	4
Database Engine-Specific Topics	4
Setting Up	5
Sign Up for AWS	5
Create an IAM User	5
Determine Requirements	7
Provide Access to Your DB Instance in Your VPC by Creating a Security Group	8
Getting Started	10
Creating a MariaDB DB Instance and Connecting to a Database	10
Creating a MariaDB Instance	10
Connecting to a Database on a DB Instance Running MariaDB	15
Deleting a DB Instance	17
Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance	18
Creating a Sample SQL Server DB Instance	18
Connecting to Your Sample DB Instance	23
Exploring Your Sample DB Instance	25
Deleting Your Sample DB Instance	27
Creating a MySQL DB Instance and Connecting to a Database	28
Creating a MySQL DB Instance	28
Connecting to a Database on a DB Instance Running MySQL	33
Deleting a DB Instance	35
Creating an Oracle DB Instance and Connecting to a Database	36
Creating a Sample Oracle DB Instance	36
Connecting to Your Sample DB Instance	39
Deleting Your Sample DB Instance	41
Creating a PostgreSQL DB Instance and Connecting to a Database	41
Creating a PostgreSQL DB Instance	41
Connecting to a PostgreSQL DB Instance	46
Deleting a DB Instance	49
Tutorial: Create a Web Server and an Amazon RDS Database	50
Step 1: Create a DB Instance	51
Step 2: Create a Web Server	56
Tutorials	69
Best Practices for Amazon RDS	70
Amazon RDS Basic Operational Guidelines	70
DB Instance RAM Recommendations	71
Amazon RDS Security Best Practices	71
Using Enhanced Monitoring to Identify Operating System Issues	71
Using Metrics to Identify Performance Issues	72
Viewing Performance Metrics	72
Evaluating Performance Metrics	73
Tuning Queries	75

Best Practices for Working with MySQL Storage Engines	75
Best Practices for Working with MariaDB Storage Engines	76
Best Practices for Working with Oracle	76
Best Practices for Working with PostgreSQL	77
Loading Data into a PostgreSQL DB Instance	77
Working with the fsync and full_page_writes database parameters	77
Working with the PostgreSQL Autovacuum Feature	77
Best Practices for Working with SQL Server	78
Working with DB Parameter Groups	79
Amazon RDS Best Practices Presentation Video	79
DB Instances	80
DB Instance Class	82
DB Instance Class Types	82
Specifications for All Available DB Instance Classes	82
Changing Your DB Instance Class	87
Configuring the Processor for a DB Instance Class	87
DB Instance Status	98
Regions and Availability Zones	101
.....	101
DB Instance Storage	103
Storage Types	103
General Purpose SSD Storage	103
Provisioned IOPS Storage	105
Magnetic storage	106
Monitoring storage performance	107
Factors That Affect Storage Performance	107
High Availability (Multi-AZ)	109
Modifying a DB Instance to be a Multi-AZ Deployment	110
Failover Process for Amazon RDS	110
Related Topics	111
DB Instance Lifecycle	112
Creating a DB Instance	113
Connecting to a DB Instance	114
Modifying a DB Instance	115
Maintaining a DB Instance	117
Upgrading the Engine Version	123
Renaming a DB Instance	126
Rebooting a DB Instance	129
Stopping a DB Instance	131
Starting a DB Instance	133
Deleting a DB Instance	135
Tagging RDS Resources	138
Overview	138
AWS Management Console	139
CLI	141
API	141
.....	142
Working with Read Replicas	143
Overview	143
Creating a Read Replica	145
Promoting a Read Replica	146
Creating a Read Replica in a Different AWS Region	148
Monitoring Read Replication	154
Working with Option Groups	156
Option Groups Overview	156
Creating an Option Group	157
Making a Copy of an Option Group	159

Adding an Option to an Option Group	160
Listing the Options and Option Settings for an Option Group	163
Modifying an Option Setting	164
Removing an Option from an Option Group	167
Working with Parameter Groups	169
Creating a DB Parameter Group	170
Modifying Parameters in a DB Parameter Group	171
Copying a DB Parameter Group	174
Listing DB Parameter Groups	175
Viewing Parameter Values for a DB Parameter Group	176
Comparing DB Parameter Groups	177
DB Parameter Values	177
Working with ARNs	181
Constructing an ARN	181
Getting an Existing ARN	184
Working with Storage	188
Increasing DB Instance Storage Capacity	188
Change Storage Type	189
Modify Provisioned IOPS	191
DB Instance Billing for Amazon RDS	193
On-Demand DB Instances	194
Reserved DB Instances	195
Backing Up and Restoring	207
Working With Backups	208
Backup Storage	208
Backup Window	208
Backup Retention Period	209
Disabling Automated Backups	209
Enabling Automated Backups	211
Retaining Automated Backups	212
Automated Backups with Unsupported MySQL Storage Engines	214
Automated Backups with Unsupported MariaDB Storage Engines	215
.....	215
Creating a DB Snapshot	216
Restoring from a DB Snapshot	218
Parameter Groups	218
Security Groups	218
Option Groups	218
Microsoft SQL Server	218
Oracle	219
Restoring from a Snapshot	219
Copying a Snapshot	221
Limitations	221
Snapshot Retention	221
Shared Snapshots	221
Encryption	221
Copying Snapshots Across AWS Regions	222
Option Groups	222
Parameter Groups	222
Copying a DB Snapshot	223
Sharing a Snapshot	230
Sharing an Encrypted Snapshot	231
Sharing a Snapshot	233
Point-in-Time Recovery	237
Tutorial: Restore a DB Instance from a DB Snapshot	239
Prerequisites for Restoring a DB Instance from a DB Snapshot	239
Restoring a DB Instance from a DB Snapshot	240

Modifying a Restored DB Instance	241
Monitoring	244
Overview of Monitoring	245
Monitoring Tools	246
Monitoring with CloudWatch	247
Publishing to CloudWatch Logs	253
Enhanced Monitoring	256
Enhanced Monitoring Availability	256
Differences Between CloudWatch and Enhanced Monitoring Metrics	256
Setting Up for and Enabling Enhanced Monitoring	257
Viewing Enhanced Monitoring	258
Viewing Enhanced Monitoring by Using CloudWatch Logs	260
Performance Insights	266
Enabling Performance Insights	267
Access Control for Performance Insights	271
Using the Performance Insights Dashboard	272
Additional User Interface Features	278
Performance Insights API	279
Metrics Published to CloudWatch	279
Logging Performance Insights Operations by Using AWS CloudTrail	280
Using Amazon RDS Recommendations	282
Responding to Recommendations	283
Using Amazon RDS Event Notification	287
Amazon RDS Event Categories and Event Messages	288
Subscribing to Amazon RDS Event Notification	294
Listing Your Amazon RDS Event Notification Subscriptions	297
Modifying an Amazon RDS Event Notification Subscription	299
Adding a Source Identifier to an Amazon RDS Event Notification Subscription	301
Removing a Source Identifier from an Amazon RDS Event Notification Subscription	302
Listing the Amazon RDS Event Notification Categories	303
Deleting an Amazon RDS Event Notification Subscription	304
Viewing Amazon RDS Events	305
AWS Management Console	305
CLI	305
API	305
.....	306
Database Log Files	307
Viewing and Listing Database Log Files	307
Downloading a Database Log File	307
Watching a Database Log File	309
Publishing to CloudWatch Logs	309
Reading Log File Contents Using REST	309
MariaDB Database Log Files	311
Microsoft SQL Server Database Log Files	319
MySQL Database Log Files	320
Oracle Database Log Files	328
PostgreSQL Database Log Files	334
Logging Amazon RDS API Calls with AWS CloudTrail	338
Amazon RDS Information in CloudTrail	338
Understanding Amazon RDS Log File Entries	338
Configuring Security	342
Authentication and Access Control	342
Authentication	343
Access Control	343
Overview of Managing Access	344
Using Identity-Based Policies (IAM Policies)	347
Amazon RDS API Permissions Reference	351

Using Conditions	367
IAM Database Authentication for MySQL and PostgreSQL	372
Encrypting Amazon RDS Resources	389
Overview of Encrypting Amazon RDS Resources	390
Enabling Amazon RDS Encryption for a DB Instance	390
Availability of Amazon RDS Encryption	391
Managing Amazon RDS Encryption Keys	391
Limitations of Amazon RDS Encrypted DB Instance	392
Using SSL to Encrypt a Connection	392
Intermediate Certificates	393
Controlling Access with Security Groups	394
DB Security Groups	394
VPC Security Groups	394
DB Security Groups vs. VPC Security Groups	395
Security Group Scenario	395
Creating a VPC Security Group	396
Associating with a DB Instance	396
Deleting DB VPC Security Groups	396
DB Security Groups on EC2-Classic	399
Master User Account Privileges	407
Service-Linked Roles	409
Service-Linked Role Permissions for Amazon RDS	409
Creating a Service-Linked Role for Amazon RDS	410
Editing a Service-Linked Role for Amazon RDS	410
Deleting a Service-Linked Role for Amazon RDS	410
Using Amazon RDS with Amazon VPC	412
Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform	412
Scenarios for Accessing a DB Instance in a VPC	414
Working with a DB Instance in a VPC	420
Updating the VPC for a DB Instance	425
Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance	427
MariaDB on Amazon RDS	432
Common Management Tasks	432
MariaDB Versions	434
Version and Feature Support	434
MariaDB 10.3 Support	434
MariaDB 10.2 Support	435
MariaDB 10.1 Support	435
MariaDB 10.0 Support	436
Features Not Supported	436
Supported Storage Engines	437
MariaDB Security	437
SSL Support	439
Cache Warming	440
Dumping and Loading the Buffer Pool on Demand	441
Database Parameters	441
Common DBA Tasks	441
Local Time Zone	441
Creating a DB Instance Running MariaDB	443
AWS Management Console	443
CLI	446
API	447
Available Settings	448
Related Topics	451
Connecting to a DB Instance Running MariaDB	452
Connecting from the mysql Utility	454
Connecting with SSL	454

Maximum MariaDB Connections	455
Related Topics	455
Modifying a DB Instance Running MariaDB	456
Available Settings	457
Related Topics	463
Upgrading the MariaDB DB Engine	464
Overview	464
Upgrading a MariaDB DB Instance	465
Migrating Data from a MySQL DB Snapshot to a MariaDB DB Instance	466
Incompatibilities Between MariaDB and MySQL	466
AWS Management Console	466
CLI	468
API	469
Related Topics	469
Working with MariaDB Replication	470
Working with MariaDB Read Replicas	470
Configuring GTID-Based Replication	473
Importing Data into a MariaDB DB Instance	476
Options for MariaDB	477
MariaDB Audit Plugin Support	477
Parameters for MariaDB	480
MariaDB on Amazon RDS SQL Reference	485
mysql.rds_set_external_master_gtid	485
mysql.rds_kill_query_id	487
Microsoft SQL Server on Amazon RDS	488
Common Management Tasks	488
Limits	490
DB Instance Class Support	491
Security	492
Compliance Programs	492
HIPAA	493
SSL Support	493
Version and Feature Support	493
SQL Server 2017 Support	493
SQL Server 2016 Support	494
SQL Server 2014 Support	494
SQL Server 2012 Support on Amazon RDS	495
SQL Server 2008 R2 Support on Amazon RDS	496
Engine Version Management	497
CDC Support	497
Features Not Supported and Features with Limited Support	498
Multi-AZ Deployments	498
Using TDE	499
Local Time Zone	499
Supported Time Zones	500
Licensing SQL Server on Amazon RDS	503
Restoring License-Terminated DB Instances	503
Using SQL Server Developer Edition on AWS	503
Related Topics	503
Creating a DB Instance Running SQL Server	504
AWS Management Console	504
CLI	508
API	509
Available Settings	510
Related Topics	513
Connecting to a DB Instance Running SQL Server	514
Connecting to Your DB Instance with SSMS	514

Connecting to Your DB Instance with SQL Workbench/J	517
Security Group Considerations	519
Troubleshooting	520
.....	520
Modifying a DB Instance Running SQL Server	521
Available Settings	522
Related Topics	528
Upgrading the SQL Server DB Engine	529
Overview	529
Major Version Upgrades	529
Multi-AZ and In-Memory Optimization Considerations	530
Option and Parameter Group Considerations	530
Testing an Upgrade	531
Upgrading a SQL Server DB Instance	532
Importing and Exporting SQL Server Databases	533
Limitations and Recommendations	533
Setting Up	534
Using Native Backup and Restore	536
Compressing Backup Files	539
Troubleshooting	540
Related Topics	541
Importing and Exporting SQL Server Data Using Other Methods	542
Multi-AZ for SQL Server	551
Adding Multi-AZ to a SQL Server DB Instance	551
Notes and Recommendations	552
Determining the Location of the Secondary	554
Migrating to Always On	554
Using SSL with a SQL Server DB Instance	555
Forcing SSL	555
Encrypting Specific Connections	556
Options for SQL Server	559
Native Backup and Restore	559
Transparent Data Encryption	561
Common DBA Tasks for SQL Server	564
Accessing the tempdb Database	565
Analyzing Your Database Workload with SQL Server Tuning Advisor	567
Collations and Character Sets	569
Determining a Recovery Model	570
Dropping a Multi-AZ Database	570
Using CDC	570
Renaming a Multi-AZ Database	572
Resetting the db_owner Role Password	573
Restoring License-Terminated DB Instances	573
Transitioning a Database from OFFLINE to ONLINE	574
Using SQL Server Agent	574
Working with SQL Server Logs	575
Working with Trace and Dump Files	576
Advanced Administrative Tasks and Concepts for SQL Server	577
Using Windows Authentication with a SQL Server DB Instance	578
MySQL on Amazon RDS	586
Common Management Tasks	586
MySQL Versions	589
MySQL Features Not Supported By Amazon RDS	590
Supported Storage Engines	591
MySQL Security	591
SSL Support	592
Using memcached and Other Options with MySQL	594

InnoDB Cache Warming	594
Dumping and Loading the Buffer Pool on Demand	595
Local Time Zone	595
Known Issues and Limitations	596
Creating a DB Instance Running MySQL	597
AWS Management Console	597
CLI	600
API	601
Available Settings	601
Related Topics	605
Connecting to a DB Instance Running MySQL	606
Connecting from the MySQL Utility	608
Connecting with SSL	608
Maximum MySQL connections	609
.....	609
Modifying a DB Instance Running MySQL	610
Available Settings	611
Related Topics	617
Upgrading the MySQL DB Engine	618
Overview	618
Major Version Upgrades	618
Testing an Upgrade	620
Upgrading a MySQL DB Instance	621
Upgrading a MySQL Database with Reduced Downtime	621
Upgrading a MySQL DB Snapshot	623
Upgrading a MySQL DB Snapshot	623
CLI	623
API	624
Importing Data into a MySQL DB Instance	625
Overview	625
Importing Data Considerations	627
Restoring a Backup into an Amazon RDS MySQL DB Instance	631
Importing Data from a MySQL or MariaDB DB to an Amazon RDS MySQL or MariaDB DB Instance	638
Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime	640
Importing Data From Any Source to a MySQL or MariaDB DB Instance	652
Working with MySQL Replication	657
Working with MySQL Read Replicas	657
Using GTID-Based Replication	663
Replication with a MySQL or MariaDB Instance Running External to Amazon RDS	667
Exporting Data From a MySQL DB Instance	673
Prepare an Instance of MySQL External to Amazon RDS	673
Prepare the Replication Source	674
Copy the Database	674
Complete the Export	676
Related Topics	676
Options for MySQL	677
MariaDB Audit Plugin	678
memcached	681
Common DBA Tasks for MySQL	685
Killing a Session or Query	685
Skipping the Current Replication Error	685
Working with InnoDB Tablespaces to Improve Crash Recovery Times	686
Managing the Global Status History	687
Known Issues and Limitations	689
Inconsistent InnoDB Buffer Pool Size	689
Index Merge Optimization Returns Wrong Results	689

Log File Size	690
MySQL Parameter Exceptions for Amazon RDS DB Instances	690
MySQL File Size Limits	690
MySQL on Amazon RDS SQL Reference	692
Overview	692
SQL Reference Conventions	693
mysql.rds_set_master_auto_position	693
mysql.rds_set_external_master	694
mysql.rds_set_external_master_with_delay	696
mysql.rds_set_external_master_with_auto_position	698
mysql.rds_reset_external_master	700
mysql.rds_import_binlog_ssl_material	701
mysql.rds_remove_binlog_ssl_material	702
mysql.rds_set_source_delay	703
mysql.rds_start_replication	704
mysql.rds_start_replication_until	704
mysql.rds_start_replication_until_gtid	705
mysql.rds_stop_replication	706
mysql.rds_skip_transaction_with_gtid	707
mysql.rds_skip_repl_error	707
mysql.rds_next_master_log	708
mysql.rds_innodb_buffer_pool_dump_now	710
mysql.rds_innodb_buffer_pool_load_now	710
mysql.rds_innodb_buffer_pool_load_abort	710
mysql.rds_set_configuration	711
mysql.rds_show_configuration	712
mysql.rds_kill	713
mysql.rds_kill_query	713
mysql.rds_rotate_general_log	714
mysql.rds_rotate_slow_log	714
mysql.rds_enable_gsh_collector	714
mysql.rds_set_gsh_collector	715
mysql.rds_disable_gsh_collector	715
mysql.rds_collect_global_status_history	715
mysql.rds_enable_gsh_rotation	715
mysql.rds_set_gsh_rotation	716
mysql.rds_disable_gsh_rotation	716
mysql.rds_rotate_global_status_history	716
Oracle on Amazon RDS	717
Common Management Tasks	717
Licensing	719
License Included	719
Bring Your Own License (BYOL)	719
Licensing Oracle Multi-AZ Deployments	720
Migrating Between Oracle Editions	720
DB Instance Class Support	720
DB Instance Class Deprecation	722
Security	723
SSL Support	723
Oracle 12c	724
Oracle 12c Version 12.2.0.1	724
Oracle 12c Version 12.1.0.2	727
Oracle 11g	733
Oracle 11g Supported Features	733
Oracle 11g Features Not Supported	734
Amazon RDS Parameters for Oracle 11g	734
Engine Version Management	735

Deprecation of Oracle 11.2.0.2	735
Deprecation of Oracle 11.2.0.3	735
Deprecation of Oracle 12.1.0.1	736
Using Huge Pages	736
Using <code>utl_http</code> , <code>utl_tcp</code> , and <code>utl_smtp</code>	738
Using OEM, APEX, TDE, and Other Options	739
Using Extended Data Types	739
Enabling Extended Data Types for a New DB Instance	740
Enabling Extended Data Types for an Existing DB Instance	740
Creating a DB Instance Running Oracle	742
AWS Management Console	742
CLI	745
API	746
Available Settings	746
Related Topics	750
Connecting to a DB Instance Running Oracle	751
Finding the Endpoint	751
SQL Developer	753
SQL*Plus	755
Security Group Considerations	756
Dedicated and Shared Server Processes	756
Troubleshooting	757
.....	757
Modifying a DB Instance Running Oracle	758
Available Settings	759
Modifying Oracle <code>sqlnet.ora</code> Parameters	767
Upgrading the Oracle DB Engine	770
Overview	770
Major Version Upgrades	771
Minor Version Upgrades	771
SE2 Upgrade Paths	772
Option and Parameter Group Considerations	772
Testing an Upgrade	773
Upgrading an Oracle DB Instance	773
Upgrading an Oracle DB Snapshot	774
AWS Management Console	774
CLI	774
API	775
Related Topics	775
Importing Data into Oracle on Amazon RDS	777
Oracle SQL Developer	777
Oracle Data Pump	777
Oracle Export/Import Utilities	781
Oracle SQL*Loader	781
Oracle Materialized Views	782
Oracle Character Sets	784
Options for Oracle	787
Application Express (APEX)	788
Enterprise Manager	795
Java Virtual Machine (JVM)	804
Label Security	807
Locator	810
Multimedia	813
Native Network Encryption (NNE)	815
Secure Sockets Layer (SSL)	817
Spatial	823
SQLT	825

Statspack	830
Time Zone	833
Transparent Data Encryption (TDE)	836
UTL_MAIL	838
XML DB	840
Common DBA Tasks for Oracle	842
System Tasks	845
Database Tasks	854
Log Tasks	866
Miscellaneous Tasks	873
Related Topics	874
Tools and Third-Party Software for Oracle	875
Setting Up	875
Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys	886
Using Oracle GoldenGate	902
Using the Oracle Repository Creation Utility	913
Installing a Siebel Database on Oracle on Amazon RDS	918
Oracle Database Engine Release Notes	921
Oracle Version 12.2.0.1	921
Oracle Versions 12.1.0.2 and 11.2.0.4	921
Database Engine: 12.2.0.1	922
Database Engine: 12.1.0.2	924
Database Engine: 11.2.0.4	944
PostgreSQL on Amazon RDS	965
Common Management Tasks for PostgreSQL on Amazon RDS	965
Creating a DB Instance Running PostgreSQL	969
Create a PostgreSQL DB Instance	969
CLI	974
API	975
Connecting to a DB Instance Running the PostgreSQL Database Engine	976
Using pgAdmin to Connect to a PostgreSQL DB Instance	976
Using psql to Connect to a PostgreSQL DB Instance	977
Troubleshooting Connection Issues	978
Modifying a DB Instance Running PostgreSQL	979
Available Settings	980
Related Topics	987
Upgrading the PostgreSQL DB Engine	988
Overview	988
Major Version Upgrades	989
Automatic Minor Version Upgrades	991
Upgrading a PostgreSQL DB Instance	991
Working with PostgreSQL Read Replicas	992
Read Replica Configuration with PostgreSQL	992
Monitoring PostgreSQL Read Replicas	993
Read Replica Limitations with PostgreSQL	993
Replication Interruptions with PostgreSQL Read Replicas	993
Troubleshooting a PostgreSQL Read Replica Problem	993
Importing Data into PostgreSQL on Amazon RDS	996
Importing a PostgreSQL Database from an Amazon EC2 Instance	997
Using the \copy Command to Import Data to a Table on a PostgreSQL DB Instance	998
Common DBA Tasks for PostgreSQL	1000
Creating Roles	1000
Managing PostgreSQL Database Access	1001
Working with PostgreSQL Parameters	1001
Working with PostgreSQL Autovacuum	1009
Audit Logging for a PostgreSQL DB Instance	1017
Working with the pgaudit Extension	1018

Working with the pg_repack Extension	1019
Working with PostGIS	1020
Using pgBadger for Log Analysis with PostgreSQL	1022
Viewing the Contents of pg_config	1022
Working with the orafce Extension	1023
Accessing External Data with the postgres_fdw Extension	1024
Using a Custom DNS Server for Outbound Network Access	1025
Restricting Password Management	1026
Working with the Database Preview Environment	1026
Features Not Supported in the Preview Environment	1027
PostgreSQL Extensions Supported in the Preview Environment	1027
Creating a New DB Instance in the Preview Environment	1029
PostgreSQL Versions and Extensions	1029
Supported PostgreSQL Database Versions	1030
Supported Features and Extensions	1043
Limits	1071
Limits in Amazon RDS	1071
Naming Constraints in Amazon RDS	1072
File Size Limits in Amazon RDS	1074
MySQL File Size Limits in Amazon RDS	1074
MariaDB File Size Limits in Amazon RDS	1075
Troubleshooting	1077
Cannot Connect to DB Instance	1077
Testing the DB Instance Connection	1077
Troubleshooting Connection Authentication	1078
Security Issues	1078
Error Message "Failed to retrieve account attributes, certain console functions may be impaired."	1078
Resetting the DB Instance Owner Role Password	1079
DB Instance Outage or Reboot	1079
Parameter Changes Not Taking Effect	1080
DB Instance Out of Storage	1080
Insufficient DB Instance Capacity	1081
MySQL Issues	1081
Index Merge Optimization Returns Wrong Results	1082
Diagnosing and Resolving Lag Between Read Replicas	1082
Diagnosing and Resolving a MySQL or MariaDB Read Replication Failure	1083
Creating Triggers with Binary Logging Enabled Requires SUPER Privilege	1084
Diagnosing and Resolving Point-In-Time Restore Failures	1086
Slave Down or Disabled Error	1086
Read Replica Create Fails or Replication Breaks With Fatal Error 1236	1087
Oracle GoldenGate Issues	1087
Retaining Logs for Sufficient Time	1087
Cannot Connect to SQL Server DB Instance	1087
Cannot Connect to PostgreSQL DB Instance	1088
Cannot set backup retention to 0	1088
Amazon RDS API Reference	1089
Using the Query API	1089
Query Parameters	1089
Query Request Authentication	1089
Troubleshooting Applications	1090
Retrieving Errors	1090
Troubleshooting Tips	1090
Document History	1091
Earlier Updates	1098

What Is Amazon Relational Database Service (Amazon RDS)?

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

Note

This guide covers non-Aurora Amazon RDS database engines. For information about using Amazon Aurora, see the [Amazon Aurora User Guide](#).

Overview of Amazon RDS

Why do you want a managed relational database service? Because Amazon RDS takes over many of the difficult or tedious management tasks of a relational database:

- When you buy a server, you get CPU, memory, storage, and IOPS, all bundled together. With Amazon RDS, these are split apart so that you can scale them independently. If you need more CPU, less IOPS, or more storage, you can easily allocate them.
- Amazon RDS manages backups, software patching, automatic failure detection, and recovery.
- To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.
- You can have automated backups performed when you need them, or manually create your own backup snapshot. You can use these backups to restore a database. The Amazon RDS restore process works reliably and efficiently.
- You can get high availability with a primary instance and a synchronous secondary instance that you can fail over to when problems occur. You can also use MySQL, MariaDB, or PostgreSQL Read Replicas to increase read scaling.
- You can use the database products you are already familiar with: MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server.
- In addition to the security in your database package, you can help control who can access your RDS databases by using AWS Identity and Access Management (IAM) to define users and permissions. You can also help protect your databases by putting them in a virtual private cloud.

If you are new to AWS products and services, begin learning more with the following resources:

- For an overview of all AWS products, see [What is Cloud Computing?](#).
- Amazon Web Services provides a number of database services. For guidance on which service is best for your environment, see [Running Databases on AWS](#).

DB Instances

The basic building block of Amazon RDS is the *DB instance*. A DB instance is an isolated database environment in the cloud. A DB instance can contain multiple user-created databases, and you can access it by using the same tools and applications that you use with a stand-alone database instance. You

can create and modify a DB instance by using the AWS Command Line Interface, the Amazon RDS API, or the AWS Management Console.

Each DB instance runs a *DB engine*. Amazon RDS currently supports the MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server DB engines. Each DB engine has its own supported features, and each version of a DB engine may include specific features. Additionally, each DB engine has a set of parameters in a DB parameter group that control the behavior of the databases that it manages.

The computation and memory capacity of a DB instance is determined by its *DB instance class*. You can select the DB instance that best meets your needs. If your needs change over time, you can change DB instances. For information, see [DB Instance Class \(p. 82\)](#).

Note

For pricing information on DB instance classes, go to the Pricing section of the [Amazon RDS](#) product page.

DB instance storage comes in three types: Magnetic, General Purpose (SSD), and Provisioned IOPS (PIOPS). They differ in performance characteristics and price, allowing you to tailor your storage performance and cost to the needs of your database. Each DB instance has minimum and maximum storage requirements depending on the storage type and the database engine it supports. It's important to have sufficient storage so that your databases have room to grow and that features for the DB engine have room to write content or log entries. For more information, see [DB instance storage \(p. 103\)](#).

You can run a DB instance on a virtual private cloud using the Amazon Virtual Private Cloud (VPC) service. When you use a virtual private cloud, you have control over your virtual networking environment: you can select your own IP address range, create subnets, and configure routing and access control lists. The basic functionality of Amazon RDS is the same whether it is running in a VPC or not; Amazon RDS manages backups, software patching, automatic failure detection, and recovery. There is no additional cost to run your DB instance in a VPC. For more information on VPC and RDS, see [Amazon Virtual Private Cloud \(VPCs\) and Amazon RDS \(p. 412\)](#).

Amazon RDS uses Network Time Protocol (NTP) to synchronize the time on DB Instances.

Regions and Availability Zones

Amazon cloud computing resources are housed in highly available data center facilities in different areas of the world (for example, North America, Europe, or Asia). Each data center location is called a region.

Each region contains multiple distinct locations called Availability Zones, or AZs. Each Availability Zone is engineered to be isolated from failures in other Availability Zones, and to provide inexpensive, low-latency network connectivity to other Availability Zones in the same region. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location. For more information, see [Regions and Availability Zones \(p. 101\)](#).

You can run your DB instance in several Availability Zones, an option called a Multi-AZ deployment. When you select this option, Amazon automatically provisions and maintains a secondary standby DB instance in a different Availability Zone. Your primary DB instance is synchronously replicated across Availability Zones to the secondary instance to provide data redundancy, failover support, eliminate I/O freezes, and minimize latency spikes during system backups. For more information, see [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#).

Security

A security group controls the access to a DB instance. It does so by allowing access to IP address ranges or Amazon EC2 instances that you specify.

Amazon RDS uses DB security groups, VPC security groups, and EC2 security groups. In simple terms, a DB security group controls access to a DB instance that is not in a VPC, a VPC security group controls access to a DB instance inside a VPC, and an Amazon EC2 security group controls access to an EC2 instance and can be used with a DB instance. For more information about security groups, see [Configuring Security in Amazon RDS \(p. 342\)](#).

Monitoring an Amazon RDS DB Instance

There are several ways that you can track the performance and health of a DB instance. You can use the free Amazon CloudWatch service to monitor the performance and health of a DB instance; performance charts are shown in the Amazon RDS console. You can subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB Snapshot, DB parameter group, or DB security group. For more information, see [Monitoring Amazon RDS \(p. 244\)](#).

Amazon RDS Interfaces

There are several ways that you can interact with Amazon RDS.

AWS Management Console

The AWS Management Console is a simple web-based user interface. You can manage your DB instances from the console with no programming required. To access the Amazon RDS console, sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.

Command Line Interface

You can use the AWS Command Line Interface (AWS CLI) to access the Amazon RDS API interactively. To install the AWS CLI, see [Installing the AWS Command Line Interface](#). To begin using the AWS CLI for RDS, see [AWS Command Line Interface Reference for Amazon RDS](#).

Programming with Amazon RDS

If you are a developer, you can access the Amazon RDS programmatically. For more information, see [Amazon RDS Application Programming Interface \(API\) Reference \(p. 1089\)](#).

For application development, we recommend that you use one of the AWS Software Development Kits (SDKs). The AWS SDKs handle low-level details such as authentication, retry logic, and error handling, so that you can focus on your application logic. AWS SDKs are available for a wide variety of languages. For more information, see [Tools for Amazon Web Services](#).

AWS also provides libraries, sample code, tutorials, and other resources to help you get started more easily. For more information, see [Sample Code & Libraries](#).

How You Are Charged for Amazon RDS

When you use Amazon RDS, you can choose to use on-demand DB instances or reserved DB instances. For more information, see [DB Instance Billing for Amazon RDS \(p. 193\)](#).

For Amazon RDS pricing information, see the [Amazon RDS product page](#).

What's Next?

The preceding section introduced you to the basic infrastructure components that RDS offers. What should you do next?

Getting Started

Create a DB instance using instructions in the [Getting Started with Amazon RDS \(p. 10\)](#) section.

Database Engine–Specific Topics

You can review information specific to a particular DB engine in the following sections:

- [MariaDB on Amazon RDS \(p. 432\)](#)
- [Microsoft SQL Server on Amazon RDS \(p. 488\)](#)
- [MySQL on Amazon RDS \(p. 586\)](#)
- [Oracle on Amazon RDS \(p. 717\)](#)
- [PostgreSQL on Amazon RDS \(p. 965\)](#)

Setting Up for Amazon RDS

Following, you can find how to set up Amazon Relational Database Service (Amazon RDS) for the first time. If you already have an AWS account, know your Amazon RDS requirements, and prefer to use the defaults for IAM and VPC security groups, skip ahead to [Getting Started \(p. 4\)](#).

A couple things you should know about Amazon Web Services (AWS):

- When you sign up for AWS, your AWS account automatically has access to all services in AWS, including Amazon RDS. However, you are charged only for the services that you use.
- With Amazon RDS, you pay only for the RDS instances that are active. The Amazon RDS DB instance that you create is live (not running in a sandbox). You incur the standard Amazon RDS usage fees for the instance until you terminate it. For more information about Amazon RDS usage rates, see the [Amazon RDS product page](#).

Topics

- [Sign Up for AWS \(p. 5\)](#)
- [Create an IAM User \(p. 5\)](#)
- [Determine Requirements \(p. 7\)](#)
- [Provide Access to Your DB Instance in Your VPC by Creating a Security Group \(p. 8\)](#)

Sign Up for AWS

If you have an AWS account already, skip to the next section, [Create an IAM User \(p. 5\)](#).

If you don't have an AWS account, you can use the following procedure to create one. If you are a new AWS customer, you can get started with Amazon RDS for free; for more information, see [AWS Free Usage Tier](#).

To create a new AWS account

1. Open <https://aws.amazon.com/>, and then choose **Create an AWS Account**.

Note

If you previously signed in to the AWS Management Console using AWS account root user credentials, choose **Sign in to a different account**. If you previously signed in to the console using IAM credentials, choose **Sign-in using root account credentials**. Then choose **Create a new AWS account**.

2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code using the phone keypad.

Create an IAM User

After you create an AWS account and successfully connect to the AWS Management Console, you can create an AWS Identity and Access Management (IAM) user. Instead of signing in with your AWS root account, we recommend that you use an IAM administrative user with Amazon RDS.

One way to do this is to create a new IAM user and grant it administrator permissions. Alternatively, you can add an existing IAM user to an IAM group with Amazon RDS administrative permissions. You can then access AWS from a special URL using the credentials for the IAM user.

If you signed up for AWS but haven't created an IAM user for yourself, you can create one using the IAM console.

To create an IAM user for yourself and add the user to an Administrators group

1. Use your AWS account email address and password to sign in as the *AWS account root user* to the IAM console at <https://console.aws.amazon.com/iam/>.

Note

We strongly recommend that you adhere to the best practice of using the **Administrator** IAM user below and securely lock away the root user credentials. Sign in as the root user only to perform a few [account and service management tasks](#).

2. In the navigation pane of the console, choose **Users**, and then choose **Add user**.
3. For **User name**, type **Administrator**.
4. Select the check box next to **AWS Management Console access**, select **Custom password**, and then type the new user's password in the text box. You can optionally select **Require password reset** to force the user to create a new password the next time the user signs in.
5. Choose **Next: Permissions**.
6. On the **Set permissions** page, choose **Add user to group**.
7. Choose **Create group**.
8. In the **Create group** dialog box, for **Group name** type **Administrators**.
9. For **Filter policies**, select the check box for **AWS managed - job function**.
10. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.
11. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
12. Choose **Next: Tags** to add metadata to the user by attaching tags as key-value pairs.
13. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users, and to give your users access to your AWS account resources. To learn about using policies to restrict users' permissions to specific AWS resources, go to [Access Management](#) and [Example Policies](#).

To sign in as the new IAM user, first sign out of the AWS Management Console. Then use the following URL, where *your_aws_account_id* is your AWS account number without the hyphens. For example, if your AWS account number is 1234-5678-9012, your AWS account ID is 123456789012.

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

Type the IAM user name and password that you just created. When you're signed in, the navigation bar displays "*your_user_name @ your_aws_account_id*".

If you don't want the URL for your sign-in page to contain your AWS account ID, you can create an account alias. From the IAM dashboard, choose **Customize** and type an alias, such as your company name. To sign in after you create an account alias, use the following URL.

```
https://your_account_alias.signin.aws.amazon.com/console/
```

To verify the sign-in link for IAM users for your account, open the IAM console and check under **AWS Account Alias** on the dashboard.

You can also create access keys for your AWS account. These access keys can be used to access AWS through the AWS Command Line Interface (AWS CLI) or through the Amazon RDS API. For more information, see [Managing Access Keys for Your AWS Account](#), [Installing the AWS Command Line Interface](#), and the [Amazon RDS API Reference](#).

Determine Requirements

The basic building block of Amazon RDS is the DB instance. In a DB instance, you create your databases. A DB instance provides a network address called an *endpoint*. Your applications use this endpoint to connect to your DB instance. When you create a DB instance, you specify details like storage, memory, database engine and version, network configuration, security, and maintenance periods. You control network access to a DB instance through a security group.

Before you create a DB instance and a security group, you must know your DB instance and network needs. Here are some important things to consider:

- **Resource requirements** – What are the memory and processor requirements for your application or service? You use these settings to help you determine what DB instance class to use. For specifications about DB instance classes, see [DB Instance Class \(p. 82\)](#).
- **VPC, subnet, and security group** – Your DB instance is most likely in a virtual private cloud (VPC). To connect to your DB instance, you need to set up security group rules. These rules are set up differently depending on what kind of VPC you use and how you use it: in a default VPC, in a user-defined VPC, or outside of a VPC.

Note

Some legacy accounts don't use a VPC. If you are accessing a new AWS Region or you are a new RDS user (after 2013), you are most likely creating a DB instance inside a VPC.

For information on how to determine if your account has a default VPC in a particular AWS Region, see [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#).

The following list describes the rules for each VPC option:

- **Default VPC** – If your AWS account has a default VPC in the current AWS Region, that VPC is configured to support DB instances. If you specify the default VPC when you create the DB instance, do the following:
 - Create a *VPC security group* that authorizes connections from the application or service to the Amazon RDS DB instance with the database. Use the [Amazon EC2 API](#) or the **Security Group** option on the VPC console to create VPC security groups. For information, see [Step 4: Create a VPC Security Group \(p. 424\)](#).
 - Specify the default DB subnet group. If this is the first DB instance you have created in this AWS Region, Amazon RDS creates the default DB subnet group when it creates the DB instance.
- **User-defined VPC** – If you want to specify a user-defined VPC when you create a DB instance, be aware of the following:
 - Make sure to create a *VPC security group* that authorizes connections from the application or service to the Amazon RDS DB instance with the database. Use the [Amazon EC2 API](#) or the **Security Group** option on the VPC console to create VPC security groups. For information, see [Step 4: Create a VPC Security Group \(p. 424\)](#).
 - The VPC must meet certain requirements in order to host DB instances, such as having at least two subnets, each in a separate availability zone. For information, see [Amazon Virtual Private Cloud \(VPCs\) and Amazon RDS \(p. 412\)](#).
 - Make sure to specify a DB subnet group that defines which subnets in that VPC can be used by the DB instance. For information, see the DB subnet group section in [Working with a DB Instance in a VPC \(p. 420\)](#).
- **No VPC** – If your AWS account doesn't have a default VPC and you don't specify a user-defined VPC, create a DB security group. A *DB security group* authorizes connections from the devices and Amazon

RDS instances running the applications or utilities to access the databases in the DB instance. For more information, see [Working with DB Security Groups \(EC2-Classic Platform\) \(p. 399\)](#).

- **High availability:** Do you need failover support? On Amazon RDS, a Multi-AZ deployment creates a primary DB instance and a secondary standby DB instance in another Availability Zone for failover support. We recommend Multi-AZ deployments for production workloads to maintain high availability. For development and test purposes, you can use a deployment that isn't Multi-AZ. For more information, see [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#).
- **IAM policies:** Does your AWS account have policies that grant the permissions needed to perform Amazon RDS operations? If you are connecting to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS operations. For more information, see [Authentication and Access Control \(p. 342\)](#).
- **Open ports:** What TCP/IP port does your database listen on? The firewall at some companies might block connections to the default port for your database engine. If your company firewall blocks the default port, choose another port for the new DB instance. When you create a DB instance that listens on a port you specify, you can change the port by modifying the DB instance.
- **AWS Region:** What AWS Region do you want your database in? Having your database in close proximity to your application or web service can reduce network latency.
- **DB disk subsystem:** What are your storage requirements? Amazon RDS provides three storage types:
 - Magnetic (Standard Storage)
 - General Purpose (SSD)
 - Provisioned IOPS (PIOPS)

Magnetic storage offers cost-effective storage that is ideal for applications with light or burst I/O requirements. General purpose, SSD-backed storage, also called *gp2*, can provide faster access than disk-based storage. Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, which are sensitive to storage performance and consistency in random access I/O throughput. For more information on Amazon RDS storage, see [DB instance storage \(p. 103\)](#).

When you have the information you need to create the security group and the DB instance, continue to the next step.

Provide Access to Your DB Instance in Your VPC by Creating a Security Group

VPC security groups provide access to DB instances in a VPC. They act as a firewall for the associated DB instance, controlling both inbound and outbound traffic at the instance level. DB instances are created by default with a firewall and a default security group that protect the DB instance.

Before you can connect to your DB instance, you must add rules to security group that enable you to connect. Use your network and configuration information to create rules to allow access to your DB instance.

Note

If your legacy DB instance was created before March 2013 and isn't in a VPC, it might not have associated security groups. If your DB instance was created after this date, it might be inside a default VPC.

For example, suppose that you have an application that accesses a database on your DB instance in a VPC. In this case, you must add a custom TCP rule that specifies the port range and IP addresses that your application uses to access the database. If you have an application on an Amazon EC2 instance, you can use the VPC or EC2 security group that you set up for the Amazon EC2 instance.

To create a VPC security group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc>.
2. In the top right corner of the AWS Management Console, choose the AWS Region where you want to create your VPC security group and DB instance. In the list of Amazon VPC resources for that AWS Region, you should see at least one VPC and several subnets. If you don't, you don't have a default VPC in that AWS Region.
3. In the navigation pane, choose **Security Groups**.
4. Choose **Create Security Group**.
5. In the **Create Security Group** window, type **Name tag**, **Group name**, and **Description** values for your security group. For **VPC**, choose the VPC that you want to create your DB instance in. Choose **Yes, Create**.
6. The VPC security group that you created should still be selected. If not, locate it in the list, and choose it. The details pane at the bottom of the console window displays the details for the security group, and tabs for working with inbound and outbound rules. Choose the **Inbound Rules** tab.
7. On the **Inbound Rules** tab, choose **Edit**.
 - a. For **Type**, choose **Custom TCP Rule**.
 - b. For **Port Range**, type the port value to use for your DB instance.
 - c. For **Source**, choose a security group name or type the IP address range (CIDR value) from where you access the instance.
8. Choose **Add another rule** if you need to add more IP addresses or different port ranges.
9. (Optional) Use the **Outbound Rules** tab to add rules for outbound traffic. By default, all outbound traffic is allowed.

You can use the VPC security group that you just created as the security group for your DB instance when you create it. If your DB instance isn't going to be in a VPC, see [Working with DB Security Groups \(EC2-Classic Platform\) \(p. 399\)](#) to create a DB security group to use when you create your DB instance.

Note

If you use a default VPC, a default subnet group spanning all of the VPC's subnets is created for you. When you create a DB instance, you can select the default VPC and use **default** for **DB Subnet Group**.

Once you have completed the setup requirements, you can launch a DB instance using your requirements and security group. For information on creating a DB instance, see the relevant documentation in the following table.

Database Engine	Documentation
MariaDB	Creating a MariaDB DB Instance and Connecting to a Database on a MariaDB DB Instance (p. 10)
Microsoft SQL Server	Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance (p. 18)
MySQL	Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance (p. 28)
Oracle	Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance (p. 36)
PostgreSQL	Creating a PostgreSQL DB Instance and Connecting to a Database on a PostgreSQL DB Instance (p. 41)

Getting Started with Amazon RDS

This section shows you how to create and connect to a DB instance using Amazon Relational Database Service (Amazon RDS). You can create, or launch, a DB instance that uses MySQL, Oracle, PostgreSQL, Microsoft SQL Server, or MariaDB.

Important

You must complete the tasks in the [Setting Up for Amazon RDS \(p. 5\)](#) section before you can create or connect to a DB instance.

Creating a DB instance and connecting to a database on a DB instance is slightly different for each of the DB engines. Choose the DB engine following that you want to use for detailed information on creating and connecting to the DB instance. After you have created and connected to your DB instance, there are instructions to help you delete the DB instance.

Topics

- [Creating a MariaDB DB Instance and Connecting to a Database on a MariaDB DB Instance \(p. 10\)](#)
- [Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance \(p. 18\)](#)
- [Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance \(p. 28\)](#)
- [Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance \(p. 36\)](#)
- [Creating a PostgreSQL DB Instance and Connecting to a Database on a PostgreSQL DB Instance \(p. 41\)](#)
- [Tutorial: Create a Web Server and an Amazon RDS Database \(p. 50\)](#)

Creating a MariaDB DB Instance and Connecting to a Database on a MariaDB DB Instance

The easiest way to create a MariaDB DB instance is to use the Amazon RDS console. Once you have created the DB instance, you can use command line tools such as `mysql` or standard graphical tools such as HeidiSQL to connect to a database on the DB instance.

Important

You must complete the tasks in the [Setting Up for Amazon RDS \(p. 5\)](#) section before you can create or connect to a DB instance.

Topics

- [Creating a MariaDB Instance \(p. 10\)](#)
- [Connecting to a Database on a DB Instance Running the MariaDB Database Engine \(p. 15\)](#)
- [Deleting a DB Instance \(p. 17\)](#)

Creating a MariaDB Instance

The basic building block of Amazon RDS is the DB instance. This environment is where you run your MariaDB databases.

In this example, you create a DB instance running the MariaDB database engine called `mariadb-instance1`, with a `db.t2.small` DB instance class, 20 GiB of storage, and automated backups enabled with a retention period of one day.

To create a MariaDB DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the upper-right corner of the Amazon RDS console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the upper left to open it.
4. Choose **Create database**. The **Select engine** page opens.

Select engine

Engine options

Amazon Aurora
Amazon Aurora

MySQL


MariaDB


PostgreSQL


Oracle


Microsoft SQL Server


MariaDB
MariaDB Community Edition is a MySQL-compatible database with strong support from the open source community, and extra features and performance optimizations.

- Supports database size up to 16 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.
- Supports global transaction ID (GTID) and thread pooling.
- Developed and supported by the MariaDB open source community.

Only enable options eligible for RDS Free Usage Tier [info](#)

[Cancel](#) **Next**

5. Choose **MariaDB**, and then choose **Next**.
6. The **Choose use case** page asks if you plan to use the DB instance you are creating for production. Because this is an example instance, choose **Dev/Test - MariaDB**. Then choose **Next**.

Note

If you create a production instance, you typically choose **Production - MariaDB** on this page to enable the failover option Multi-AZ and the Provisioned IOPS storage option.

7. On the **Specify DB details** page, specify your DB instance information. The following table shows settings for an example DB instance. When the settings are as you want them, choose **Next**.

For This Parameter	Do This
License model	Choose the default, general-public-license , to use the GNU General Public License, version 2 for MariaDB. MariaDB has only one license model.
DB engine version	Choose the version of MariaDB that you want to use.
DB instance class	Choose db.t2.small for a configuration that equates to 2 GiB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.
Multi-AZ deployment	<p>Choose Create replica in a different zone to have a standby replica of your DB instance created in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose No.</p> <p>For more information, see High Availability (Multi-AZ) for Amazon RDS (p. 109).</p>
Storage type	Choose the storage type General Purpose (SSD) . For more information about storage, see DB instance storage (p. 103) .
Allocated storage	Enter 20 to allocate 20 GiB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon RDS Features .
DB instance identifier	Enter a name for the DB instance that is unique for your account in the AWS Region you chose. You can add some intelligence to the name, such as including the AWS Region and DB engine you chose, for example mariadb-instance1 .
Master username	Enter a name using 1–16 alphanumeric characters to use as the master user name to log on to your DB instance. You use this user name to log on to your database on the DB instance for the first time.
Master password and Confirm password	Enter a password that contains from 8–41 printable ASCII characters (excluding /, ", and @) for your master user password. You use this password with the user name when you log on to your database. Enter the password again for Confirm Password .

Specify DB details

Instance specifications

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#) 

DB engine
MariaDB Community Edition

License model [Info](#)
▼

DB engine version [Info](#)
▼

Free tier
The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GiB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#).

Only enable options eligible for RDS Free Usage Tier [Info](#)

DB instance class [Info](#)
▼

8. On the **Configure advanced settings** page, provide additional information that RDS needs to launch the MariaDB DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then choose **Create database**.

For This Parameter	Do This
Virtual Private Cloud (VPC)	Choose the name of the Amazon Virtual Private Cloud (Amazon VPC) to host your MariaDB DB instance. For more information about using VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 412) .
Subnet group	Choose Create new DB subnet group .
Public accessibility	Choose Yes .
Availability zone	Determine if you want to specify a particular availability zone. For more information about Availability Zones, see Regions and Availability Zones (p. 101) .
VPC security groups	Choose Create new VPC security group .
Database name	Enter a name for your default database that is 1–64 alphanumeric characters. If you don't provide a name,

For This Parameter	Do This
	Amazon RDS doesn't automatically create a database on the DB instance you are creating. To create additional databases, connect to the DB instance and use the SQL command <code>CREATE DATABASE</code> . For more information about connecting to the DB instance, see Connecting to a DB Instance Running the MariaDB Database Engine (p. 452) .
Database port	Keep the default value of 3306 unless you have a specific port you want to access the database through. MariaDB installations default to port 3306.
DB parameter group	Accept the default value of default.mariadb10.0 unless you created your own DB parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 169) .
Option group	Accept the default value.
Copy tags to snapshots	Choose this option to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .
Encryption	Choose Disable encryption . Note You usually choose Enable encryption for production instances to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 389) .
Backup retention period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to 1 day .
Backup window	Unless you have a specific time that you want to have your database back up, use the default of No Preference .
Enhanced Monitoring	Unless you want to enable gathering metrics in real time for the operating system that your DB instance runs on, use the default of Disable enhanced monitoring .
Log exports	Select General log . For more information, see MariaDB Database Log Files (p. 311) .
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available.
Maintenance window	Choose the 30-minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose No preference .

9. Choose **Create database**.
10. Choose **View DB instance details**.

On the RDS console, the details for new DB instance appear. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.

mariadb-instance1

Summary

Engine MariaDB 10.1.26	DB instance class info db.t2.small	DB instance status creating
---------------------------	---	--------------------------------

CloudWatch (17) [Add instance to compare](#)

Legend: mariadb-instance1

Connecting to a Database on a DB Instance Running the MariaDB Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to a database on the DB instance. In this example, you connect to a database on a MariaDB DB instance using the `mysql` command-line tool. One GUI-based application you can use to connect is HeidiSQL; for more information, go to the [Download HeidiSQL](#) page. For more information on using MariaDB, go to the [MariaDB documentation](#).

To connect to a database on a DB instance using the `mysql` command-line tool

1. Find the endpoint (DNS name) and port number for your DB Instance.
 - a. Open the RDS console and then choose **Databases** to display a list of your DB instances.
 - b. Choose the MariaDB DB instance name to display its details.
 - c. On the **Connectivity** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.

mymariadb

Summary

DB Name

mymariadb

Role

Instance

Connectivity

Monitoring

Logs & events

Configuration

Maintenance

Connectivity

Endpoint & port

Endpoint

mymariadb [REDACTED] rds.amazonaws.com

Port

3306

- Enter the following command at a command prompt on a client computer to connect to a database on a MariaDB DB instance. Substitute the DNS name (endpoint) for your DB instance for `<endpoint>`, the master user name you used for `<mymasteruser>`, and provide the master password you used when prompted for a password.

```
PROMPT> mysql -h <endpoint> -P 3306 -u <mymasteruser> -p
```

After you enter the password for the user, you should see output similar to the following.

```
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 272  
Server version: 5.5.5-10.0.17-MariaDB-log MariaDB Server
```

```
Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.
```

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql >

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
3. Choose the DB instance you want to delete.
4. For **Actions**, choose **Delete**.
5. For **Create final snapshot?**, choose **No**, and select the acknowledgment.
6. Choose **Delete**.

Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance

The basic building block of Amazon RDS is the DB instance. Your Amazon RDS DB instance is similar to your on-premises Microsoft SQL Server. After you create your SQL Server DB instance, you can add one or more custom databases to it.

Important

You must have an AWS account before you can create a DB instance. If you don't have an AWS account, open <https://aws.amazon.com/>, and then choose **Create an AWS Account**.

In this topic, you create a sample SQL Server DB instance. You then connect to the DB instance and run a simple query. Finally you delete the sample DB instance.

Creating a Sample SQL Server DB Instance

In this procedure you use the AWS Management Console to create a sample DB instance. Because you are only creating a sample DB instance, each setting is not fully explained. For a full explanation of each setting, see [Creating a DB Instance Running the Microsoft SQL Server Database Engine \(p. 504\)](#).

To create a DB instance running the Microsoft SQL Server DB engine

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the upper-right corner of the Amazon RDS console, choose the AW Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
4. Choose **Create database**.

The **Select engine** page appears.

Select engine

Engine options

Amazon Aurora
Amazon Aurora

MySQL


MariaDB


PostgreSQL


Oracle
ORACLE

Microsoft SQL Server


Microsoft SQL Server

Edition

SQL Server Express Edition
Affordable database management system that supports database sizes up to 10 GiB.

SQL Server Web Edition
In accordance with Microsoft's licensing policies, it can only be used to support public and Internet-accessible webpages, websites, web applications, and web services.

SQL Server Standard Edition
Core data management and business intelligence capabilities for mission-critical applications and mixed workloads.

SQL Server Enterprise Edition
Comprehensive high-end capabilities for mission-critical applications with demanding database workloads and business intelligence requirements.



Aurora global database feature is now available.
This feature is now available in our new database creation flow.

[Try it now](#)

Only enable options eligible for RDS Free Usage Tier [Info](#)

[Cancel](#) **Next**

5. Choose the SQL Server icon, and then choose **Select** for the **SQL Server Express** edition.

The **Specify DB Details** page appears.

Specify DB details

Instance specifications

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#) 

DB engine

Microsoft SQL Server Express Edition

License model [Info](#)

license-included

DB engine version [Info](#)

SQL Server 2017 14.00.3035.2.v1



Free tier

The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GiB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#).

Only enable options eligible for RDS Free Usage Tier [Info](#)

DB instance class [Info](#)

db.t2.small — 1 vCPU, 2 GiB RAM

Time zone (optional)

No preference

Storage type [Info](#)

6. On the **Specify DB Details** page, provide the information for your DB instance as shown in the following table.

For This Parameter	Do This
License Model	Choose license-included to use the general license agreement for Microsoft SQL Server.
DB Engine Version	Choose the most recent version of SQL Server available in the list.
DB Instance Class	Choose db.t2.micro . This instance class is appropriate for testing.
Time Zone	Don't choose a time zone. If you don't choose a time zone, your DB instance uses the default time zone.
Storage Type	Choose the storage type General Purpose (SSD) .

For This Parameter	Do This
Allocated Storage	Enter 20 to allocate 20 GiB of storage for your database. There is a warning that you should consider allocating more storage, but because this is a sample DB instance, 20 GiB is sufficient.
DB Instance Identifier	Enter sample-instance .
Master Username	Enter a name to use as the master user name to log on to your DB Instance with all database privileges. The master user name is a SQL Server Authentication login.
Master Password and Confirm Password	Enter a password for your master user password. It must contain between 8–128 printable ASCII characters (excluding /", and @).

7. Choose **Next** to continue.

The **Configure Advanced Settings** page appears.

Configure advanced settings

Network & Security

Virtual Private Cloud (VPC) [Info](#)

VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-2aed394c) 



Only VPCs with a corresponding DB subnet group are listed.

Subnet group [Info](#)

DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

default 

Public accessibility [Info](#)

Yes

EC2 instances and devices outside of the VPC hosting the DB instance will connect to the DB instances. You must also select one or more VPC security groups that specify which EC2 instances and devices can connect to the DB instance.

No

DB instance will not have a public IP address assigned. No EC2 instance or devices outside of the VPC will be able to connect.

Availability zone [Info](#)

No preference 

VPC security groups

Security groups have rules authorizing connections from all the EC2 instances and devices that need to access the DB instance.

Create new VPC security group

Choose existing VPC security groups

8. On the **Configure Advanced Settings** page, provide the information for your DB instance as shown in the following table.

For This Parameter	Do This
VPC	Choose Create new VPC .
Subnet Group	Choose Create new DB Subnet Group .
Publicly Accessible	Choose Yes .
Availability Zone	Choose No Preference .
VPC Security Group	Choose Create new Security Group .
Database Port	Keep the default value of 1433 unless you have a specific port that you want to access the database through. SQL Server installations default to port 1433, but in some cases a firewall might block this port. If in doubt, ask your network administrator what port you should use.
DB Parameter Group	Keep the default value.
Option Group	Keep the default value.
Copy Tags To Snapshots	Keep this setting unselected.
Backup Retention Period	Choose 7 .
Backup Window	Choose No Preference .
Enable Enhanced Monitoring	Choose No .
Auto Minor Version Upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available.
Maintenance Window	Choose No Preference .

9. Choose **Create database**.
10. Choose **View Your DB Instances**.

On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.

The screenshot shows the 'Databases' section of the AWS RDS console. At the top, there's a breadcrumb navigation: 'RDS > Databases'. Below that is a search bar labeled 'Filter databases'. A table lists five database instances:

DB Name	Role	Engine	Region
mymariadb	Instance	MariaDB	US-East-1
myoracledb	Instance	Oracle Enterprise Edition	US-East-1
mypostgresql	Instance	PostgreSQL	US-East-1
mysqldb	Instance	SQL Server Express Edition	US-East-1
testauroramysql-cl	Regional	Aurora MySQL	US-East-1

Connecting to Your Sample SQL Server DB Instance

In this procedure you connect to your sample DB instance by using Microsoft SQL Server Management Studio (SSMS). To download a stand-alone version of this utility, see [Download SQL Server Management Studio \(SSMS\)](#) in the Microsoft documentation.

To connect to a DB Instance using SSMS

1. Find the DNS name and port number for your DB Instance.
 - a. Open the RDS console and then choose **Databases** to display a list of your DB instances.
 - b. Choose the name of your SQL Server DB instance to display the summary information for the DB instance.

mysqldb

Summary

DB Name	CPU	Info
mysqldb		Backing-up
Role	Current activity	Engine
Instance	0 Connections	SQL Server Express Edition

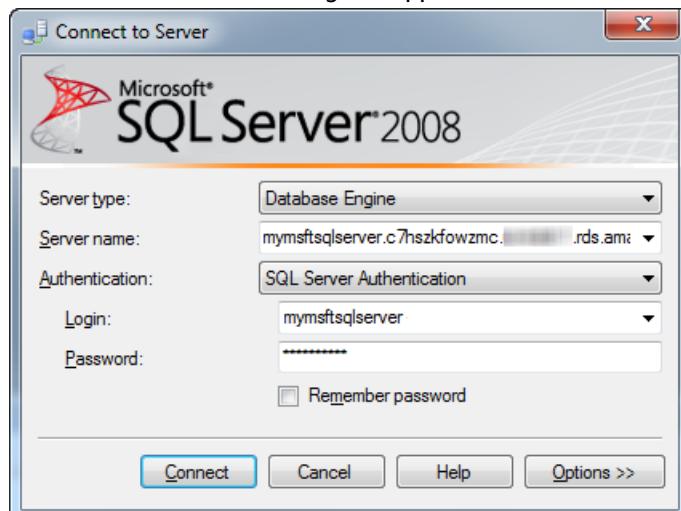
Connectivity Monitoring Logs & events Configuration Maintenance & backups Tags

Connectivity

Endpoint & port	Networking
Endpoint mysqldb.rds.amazonaws.com	Availability zone us-west-2b
Port 1433	VPC vpc-2aed394c
	Subnet group

- c. On the **Connectivity** tab, copy the endpoint. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, and the part following the colon is the port number. Copy both parts.
2. Start SQL Server Management Studio.

The **Connect to Server** dialog box appears.



3. Provide the information for your sample DB instance.
 - a. For **Server type**, choose **Database Engine**.
 - b. For **Server name**, enter the DNS name and port number of your sample DB instance, separated by a comma.

Important

Change the colon between the DNS name and port number to a comma.

For example, your server name should look like the following.

```
sample-instance.cg034hpkmmjt.us-east-1.rds.amazonaws.com,1433
```

- c. For **Authentication**, choose **SQL Server Authentication**.
 - d. For **Login**, enter the master user name that you chose earlier for your sample DB instance.
 - e. For **Password**, enter the password that you chose earlier for your sample DB instance.
4. Choose **Connect**.

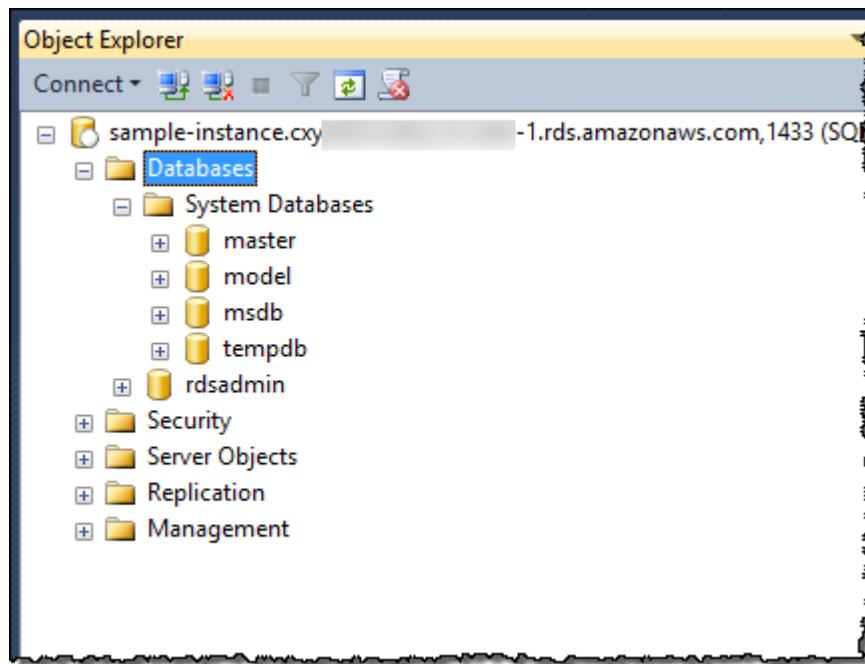
After a few moments, SSMS connects to your DB instance. If you can't connect to your DB instance, see [Troubleshooting the Connection to Your SQL Server DB Instance \(p. 520\)](#).

Exploring Your Sample SQL Server DB Instance

In this procedure you continue the previous procedure and explore your sample DB instance by using Microsoft SQL Server Management Studio (SSMS).

To explore a DB Instance using SSMS

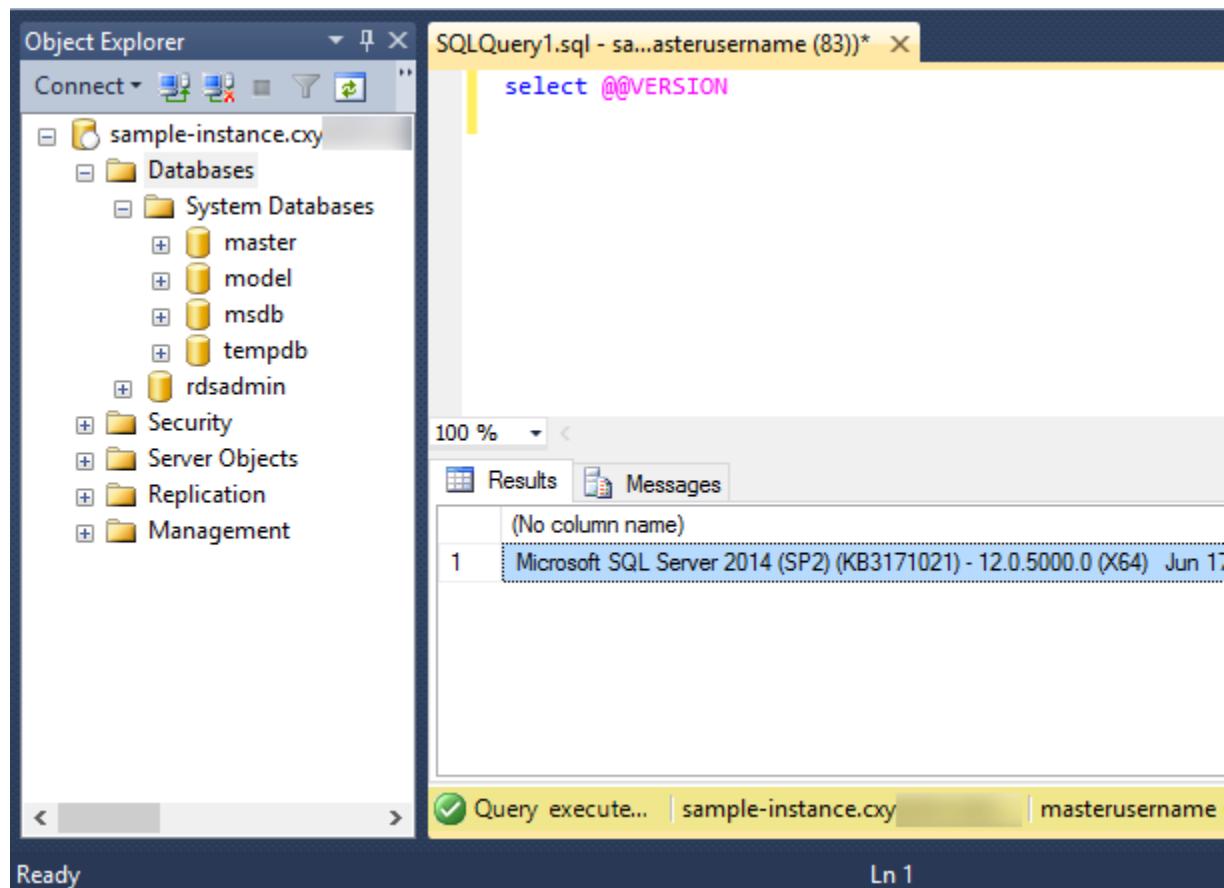
1. Your SQL Server DB instance comes with SQL Server's standard built-in system databases (master, model, msdb, and tempdb). To explore the system databases, do the following:
 - a. In SSMS, on the **View** menu, choose **Object Explorer**.
 - b. Expand your DB instance, expand **Databases**, and then expand **System Databases** as shown following.



2. Your SQL Server DB instance also comes with a database named `rdsadmin`. Amazon RDS uses this database to store the objects that it uses to manage your database. The `rdsadmin` database also includes stored procedures that you can run to perform advanced tasks.
3. You can now start creating your own databases and running queries against your DB instance and databases as usual. To run a test query against your sample DB instance, do the following:
 - a. In SSMS, on the **File** menu point to **New** and then choose **Query with Current Connection**.
 - b. Enter the following SQL query.

```
select @@VERSION
```

- c. Run the query. SSMS returns the SQL Server version of your Amazon RDS DB instance.



Deleting Your Sample DB Instance

Once you are done exploring the sample DB instance that you created, you should delete the DB instance so that you are no longer charged for it.

To delete a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose your sample DB instance.
4. For **Actions**, choose **Delete**.
5. For **Create final Snapshot**, choose **No**.

Note

You should create a final snapshot for any production DB instance that you delete.

6. Choose **Delete**.

Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance

The easiest way to create a DB instance is to use the AWS Management Console. Once you have created the DB instance, you can use standard MySQL utilities such as MySQL Workbench to connect to a database on the DB instance.

Important

You must complete the tasks in the [Setting Up for Amazon RDS \(p. 5\)](#) section before you can create or connect to a DB instance.

Topics

- [Creating a MySQL DB Instance \(p. 28\)](#)
- [Connecting to a Database on a DB Instance Running the MySQL Database Engine \(p. 33\)](#)
- [Deleting a DB Instance \(p. 35\)](#)

Creating a MySQL DB Instance

The basic building block of Amazon RDS is the DB instance. This is the environment in which you run your MySQL databases.

In this example, you create a DB instance running the MySQL database engine called *mysql-instance1*, with a *db.m1.small* DB instance class, 20 GiB of storage, and automated backups enabled with a retention period of one day.

To create a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the Amazon RDS console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
4. Choose **Create database**. The **Select engine** page opens.

Select engine

Engine options

Amazon Aurora
Amazon Aurora

MySQL


MariaDB


PostgreSQL


Oracle


Microsoft SQL Server


MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 16 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.

Only enable options eligible for RDS Free Usage Tier [info](#)

[Cancel](#) [Next](#)

5. Choose **MySQL**, and then choose **Next**.
6. The **Choose use case** page asks if you are planning to use the DB instance you are creating for production. Choose **Dev/Test** and then choose **Next**.
7. On the **Specify DB Details** page, specify your DB instance information. The following table shows settings for an example DB instance. When the settings are as you want them, choose **Next**.

For This Parameter	Do This
License model	Choose the default, general-public-license , to use the general license agreement for MySQL. MySQL has only one license model.
DB engine version	Choose the default version of MySQL. Amazon RDS supports multiple versions of MySQL in some AWS Regions.
DB instance class	Choose db.m1.small .
Multi-AZ deployment	Choose Yes to have a standby replica of your DB instance created in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to

For This Parameter	Do This
	<p>maintain high availability. For development and testing, you can choose No.</p> <p>For more information, see High Availability (Multi-AZ) for Amazon RDS (p. 109).</p>
Storage type	Choose the storage type General Purpose (SSD) . For more information about storage, see DB instance storage (p. 103) .
Allocated storage	Enter 20 to allocate 20 GiB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon RDS Features .
DB instance identifier	Enter a name for the DB instance that is unique for your account in the AWS Region you chose. You can add some intelligence to the name, such as including the AWS Region and DB engine you chose, for example mysql-instance1 .
Master username	Enter a name using alphanumeric characters to use as the master user name to log on to your DB instance. This is the user name you use to log on to your database on the DB instance for the first time.
Master password and Confirm password	Enter a password that contains from 8 to 41 printable ASCII characters (excluding /, ", and @) for your master user password. This is the password to use when you use the user name to log on to your database. Then type the password again in the Confirm Password box.

Specify DB details

Instance specifications

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#)

DB engine

MySQL Community Edition

License model [Info](#)

general-public-license

DB engine version [Info](#)

MySQL 5.6.40



Known Issues/Limitations

Review the [Known Issues/Limitations](#) to learn about potential compatibility issues with specific database versions.



Free tier

The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GiB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#).

Only enable options eligible for RDS Free Usage Tier [Info](#)

DB instance class [Info](#)

db.r4.xlarge — 4 vCPU, 30.5 GiB RAM

Multi-AZ deployment [Info](#)

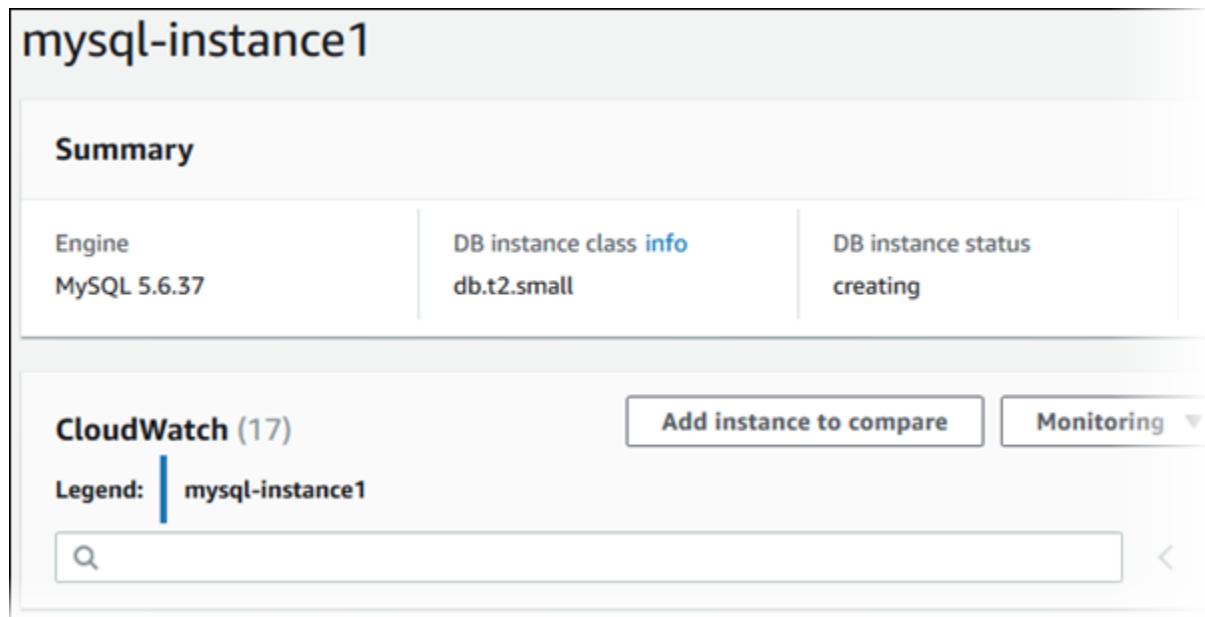
8. Choose **Next**.
9. On the **Configure advanced settings** page, provide additional information that RDS needs to launch the MySQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then choose **Create database**.

For This Parameter	Do This
Virtual Private Cloud (VPC)	Choose Create new VPC .
Subnet group	Choose Create new DB subnet group .
Public accessibility	Choose Yes .
Availability zone	Choose No Preference .

For This Parameter	Do This
VPC security groups	Choose Create new VPC security group .
Database name	<p>Enter a name for your default database that is 1 to 64 alpha-numeric characters. If you don't provide a name, Amazon RDS doesn't automatically create a database on the DB instance you are creating.</p> <p>To create additional databases, connect to the DB instance and use the SQL command <code>CREATE DATABASE</code>. For more information about connecting to the DB instance, see Connecting to a DB Instance Running the MySQL Database Engine (p. 606).</p>
Database port	Leave the default value of 3306 unless you have a specific port you want to access the database through. MySQL installations default to port 3306.
DB parameter group	Leave the default value unless you created your own DB parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 169) .
Option group	Choose the default value because this option group is used with the MySQL version you chose on the previous page.
Copy tags To snapshots	Choose this option to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .
IAM DB authentication	Choose No . For more information, see Authentication and Access Control (p. 342) .
Encryption	Choose Enable encryption to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 389) .
Backup retention period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to 1 .
Backup window	Unless you have a specific time that you want to have your database backup, use the default of No Preference .
Enhanced monitoring	Unless you want to enable gathering metrics in real time for the operating system that your DB instance runs on, use the default of Disable enhanced monitoring .
Log exports	Select General log . For more information, see MySQL Database Log Files (p. 320) .
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available.
Maintenance window	Choose No preference .

10. Choose **Create database**.
11. Choose **View DB instance details**.

On the RDS console, the details for new DB instance appear. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.



Connecting to a Database on a DB Instance Running the MySQL Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to a database on the DB instance. In this example, you connect to a database on a MySQL DB instance using MySQL monitor commands. One GUI-based application you can use to connect is MySQL Workbench; for more information, go to the [Download MySQL Workbench](#) page. For more information on using MySQL, go to the [MySQL documentation](#).

To connect to a database on a DB instance using MySQL monitor

1. Find the endpoint (DNS name) and port number for your DB Instance.
 - a. Open the RDS console and then choose **Databases** to display a list of your DB instances.
 - b. Choose the MySQL DB instance name to display its details.
 - c. On the **Connectivity** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.

mysql-instance1

Summary

DB Name	mysql-instance1	CPU
Role	Master	Current

Connectivity | Monitoring | Logs & events | Configuration | Maintenance

Connectivity

Endpoint & port

Endpoint	mysql-instance1 [REDACTED] rds.amazonaws.com
Port	3306

2. Enter the following command at a command prompt on a client computer to connect to a database on a MySQL DB instance using the MySQL monitor. Substitute the DNS name for your DB instance for `<endpoint>`, the master user name you used for `<mymasteruser>`, and provide the master password you used when prompted for a password.

```
PROMPT> mysql -h <endpoint> -P 3306 -u <mymasteruser> -p
```

After you enter the password for the user, you should see output similar to the following.

```
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 350  
Server version: 5.6.40-log MySQL Community Server (GPL)  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql>
```

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
3. Choose the DB instance you wish to delete.
4. Choose **Actions**, and then choose **Delete**.
5. For **Create final snapshot?**, choose **No**, and select the acknowledgment.
6. Choose **Delete**.

Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance

The basic building block of Amazon RDS is the DB instance. Your Amazon RDS DB instance is similar to your on-premises Oracle database.

Important

You must have an AWS account before you can create a DB instance. If you don't have an AWS account, open <https://aws.amazon.com/>, and then choose **Create an AWS Account**.

In this topic you create a sample Oracle DB instance. You then connect to the DB instance and run a simple query. Finally you delete the sample DB instance.

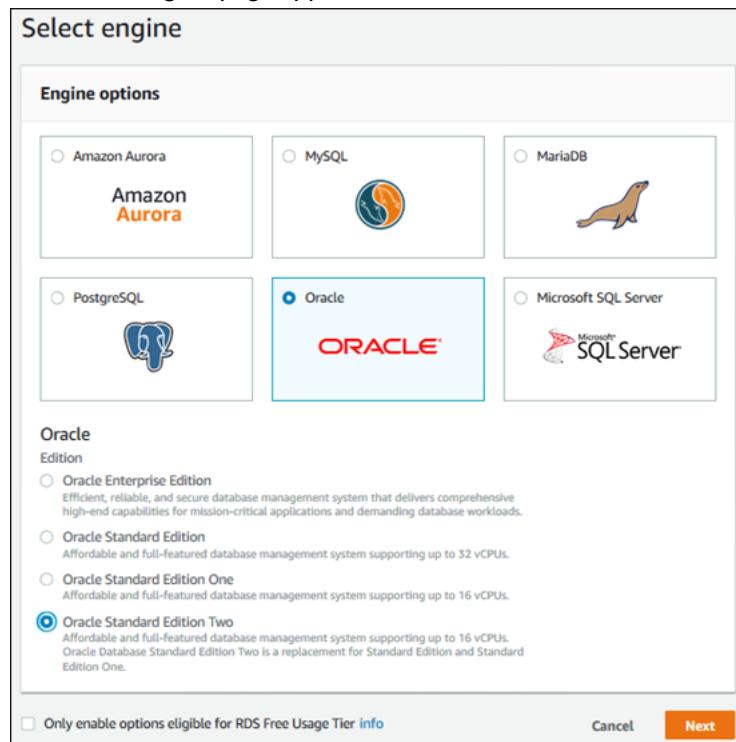
Creating a Sample Oracle DB Instance

In this procedure you use the AWS Management Console to create a sample DB instance. Since you are only creating a sample DB instance, each setting is not fully explained. For a full explanation of each setting, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).

To create a DB instance running the Oracle database engine

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the Amazon RDS console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
4. Choose **Create database**.

The **Select engine** page appears.



5. Choose the Oracle icon, and then choose **Select** for the **Oracle Standard Edition Two** edition.
6. The **Choose use case** page asks if you are planning to use the DB instance you are creating for production. Choose **Dev/Test** and then choose **Next**.

The **Specify DB details** page appears.

Specify DB details

Instance specifications
Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#).

DB engine
Oracle Database Standard Edition Two

License model [info](#)

DB engine version [info](#)

Free tier
The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#).
 Only enable options eligible for RDS Free Usage Tier [info](#)

DB instance class [info](#)

Multi-AZ deployment [info](#)
 Create replica in different zone
Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.
 No

Storage type [info](#)

Allocated storage
20 GB
(Minimum: 20 GB, Maximum: 6144 GB) Higher allocated storage may improve IOPS performance.

Provisioning less than 100 GB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. Click here for more details.

7. On the **Specify DB details** page, provide the information for your DB instance as shown in the following table.

For This Parameter	Do This
License model	Choose license-included to use the general license agreement for Oracle.
DB engine version	Choose the most recent version of Oracle available in the list.
DB instance class	Choose db.t2.small . This instance class is appropriate for testing.
Multi-AZ deployment	For development and testing, choose No .
Storage type	Choose the storage type General Purpose (SSD) .

For This Parameter	Do This
Allocated storage	Enter 20 to allocate 20 GiB of storage for your database.
DB instance identifier	Enter oracle-instance1 .
Master username	Enter a name to use as the master user name to log on to your DB instance with all database privileges. The master user name is a SQL Server Authentication login.
Master password and confirm Password	Enter a password for your master user password. It must contain 8–128 printable ASCII characters (excluding /," and @).

- Choose **Next** to continue.

The **Configure Advanced Settings** page appears.

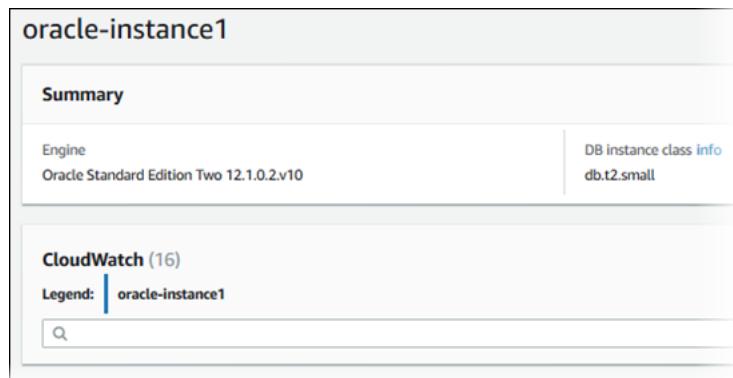
- On the **Configure advanced settings** page, provide the information for your DB instance as shown in the following table.

For This Parameter	Do This
Virtual Private Cloud (VPC)	Choose Create new VPC .
Subnet group	Choose Create new DB subnet group .
Public accessibility	Choose Yes .
Availability zone	Choose No Preference .
VPC security groups	Choose Create new VPC security group .
Database name	Enter ORCL .
Database port	Keep the default value of 1521 unless you have a specific port you want to access the database through. Oracle installations default to port 1521, but in some cases a firewall might block this port. If in doubt, ask your network administrator what port you should use.
DB parameter group	Keep the default value.
Option group	Keep the default value.
Copy tags to snapshots	Keep this setting unselected.
Character set name	Choose the default value of AL32UTF8 for the Unicode 5.0 UTF-8 Universal character set.
Enable encryption	Choose No to enable encryption at rest for this DB instance.
Backup retention period	Choose 7 days .
Backup window	Choose No preference .
Enhanced monitoring	Choose Disable enhanced monitoring .
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine

For This Parameter	Do This
	version upgrades automatically when they become available.
Maintenance window	Choose No preference .

10. Choose **Create database**.
11. Choose **View DB instance details**.

On the RDS console, the details for new DB instance appear. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.



Connecting to Your Sample Oracle DB Instance

After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. In this procedure you connect to your sample DB instance by using the Oracle *sqlplus* command line utility. To download a stand-alone version of this utility, see [SQL*Plus User's Guide and Reference](#).

To connect to a DB Instance using SQL*Plus

1. Find the endpoint (DNS name) and port number for your DB Instance.
 - a. Open the RDS console and then choose **Databases** to display a list of your DB instances.
 - b. Choose the Oracle DB instance name to display its details.
 - c. On the **Connectivity** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.

oracle-instance1

Summary

DB Name
oracle-instance1

Role
Instance

Connectivity Monitoring Logs & events Configuration

Connectivity

Endpoint & port

Endpoint
oracle-instance1.**.rds.amazonaws.com**

Port
1521

2. Enter the following command on one line at a command prompt to connect to your DB instance by using the sqlplus utility. The value for Host is the endpoint for your DB instance, and the value for Port is the port you assigned the DB instance. The value for the Oracle SID is the name of the DB instance's database that you specified when you created the DB instance, not the name of the DB instance.

```
PROMPT>sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=endpoint)  
(PORT=1521))(CONNECT_DATA=(SID=ORCL)))'
```

You should see output similar to the following.

```
SQL*Plus: Release 11.1.0.7.0 - Production on Wed May 25 15:13:59 2011
```

SQL>

Deleting Your Sample DB Instance

Once you are done exploring the sample DB instance that you created, you should delete the DB instance so that you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to delete.
4. For **Actions**, choose **Delete**.
5. For **Create final snapshot?**, choose **No**, and choose the acknowledgment.
6. Choose **Delete**.

Creating a PostgreSQL DB Instance and Connecting to a Database on a PostgreSQL DB Instance

The easiest way to create a DB instance is to use the RDS console. Once you have created the DB instance, you can use standard SQL client utilities to connect to the DB instance such as the pgAdmin utility. In this example, you create a DB instance running the PostgreSQL database engine called west2-*postgres1*, with a db.m1.small DB instance class, 10 GiB of storage, and automated backups enabled with a retention period of one day.

Important

You must complete the tasks in the [Setting Up for Amazon RDS \(p. 5\)](#) section before you can create or connect to a DB instance.

Topics

- [Creating a PostgreSQL DB Instance \(p. 41\)](#)
- [Connecting to a PostgreSQL DB Instance \(p. 46\)](#)
- [Deleting a DB Instance \(p. 49\)](#)

Creating a PostgreSQL DB Instance

To create a DB Instance Running the PostgreSQL DB Engine

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the AWS Management Console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.

If the navigation pane is closed, choose the menu icon at the top left to open it.

4. Choose **Create database** to open the **Select engine** page.

Select engine

Engine options

<input type="radio"/> Amazon Aurora Amazon Aurora	<input type="radio"/> MySQL 	<input type="radio"/> MariaDB 
<input checked="" type="radio"/> PostgreSQL 	<input type="radio"/> Oracle 	<input type="radio"/> Microsoft SQL Server 

PostgreSQL

PostgreSQL is a powerful, open-source object-relational database system with a strong reputation of reliability, stability, and correctness.

- High reliability and stability in a variety of workloads.
- Advanced features to perform in high-volume environments.
- Vibrant open-source community that releases new features multiple times per year.
- Supports multiple extensions that add even more functionality to the database.
- Supports up to 5 Read Replicas per instance, within a single Region or cross-region.
- The most Oracle-compatible open-source database.

Info If you want to create PostgreSQL 11 in the Preview environment, click [here](#)

Info **Aurora global database feature is now available.**
This feature is now available in our new database creation flow.
[Try it now](#)

Only enable options eligible for RDS Free Usage Tier [Info](#)

[Cancel](#) [Next](#)

5. On the **Select engine** page, choose the PostgreSQL icon, and then choose **Next**.
6. Next, the **Use case** page asks if you are planning to use the DB instance you are creating for production. If you are, choose **Production**. If you choose this option, the failover option **Multi-AZ**

and the **Provisioned IOPS** storage options are preselected in the following step. Choose **Next** when you are finished.

7. On the **Specify DB Details** page, specify your DB instance information. Choose **Next** when you are finished.

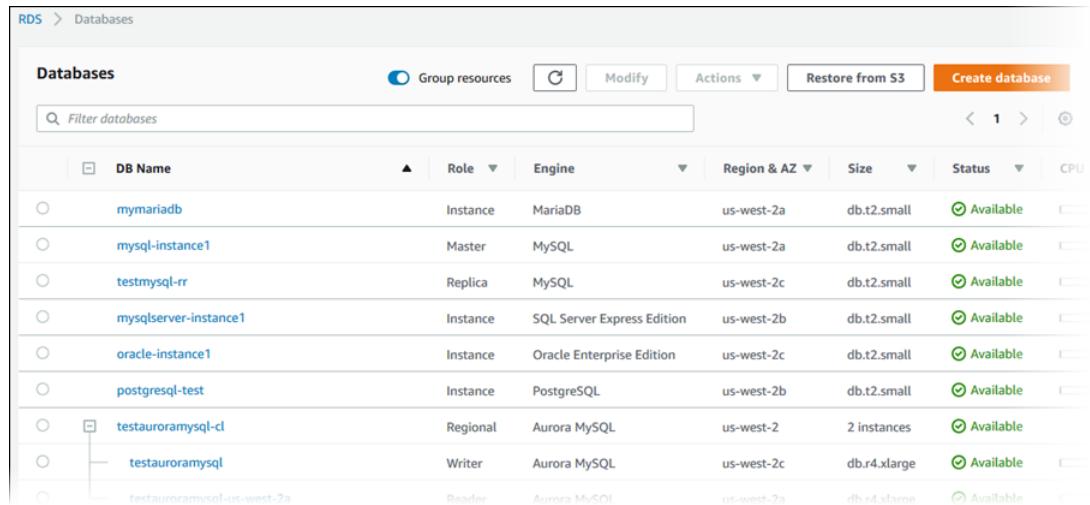
For This Parameter	Do This
License Model	PostgreSQL has only one license model. Choose postgresql-license to use the general license agreement for PostgreSQL.
DB Engine Version	Choose the version of PostgreSQL you want to use.
DB Instance Class	Choose db.t2.small for a configuration that equates to 2 GiB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity. For more information about all the DB instance class options, see DB Instance Class (p. 82) .
Multi-AZ Deployment	Choose Yes to have a standby replica of your DB instance created in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose No . For more information, see High Availability (Multi-AZ) for Amazon RDS (p. 109) .
Storage Type	Choose the storage type General Purpose (SSD) . For more information about storage, see DB instance storage (p. 103) .
Allocated Storage	Enter 20 to allocate 20 GiB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon RDS Features .
DB Instance Identifier	Enter a name for the DB instance that is unique for your account in the AWS Region you chose. You can add some intelligence to the name, such as including the AWS Region and DB engine you chose, for example postgresSQL-test .
Master Username	Enter a name using alphanumeric characters to use as the master user name to log on to your DB instance. For information on the default privileges granted to the master user name, see Amazon RDS for PostgreSQL Versions and Extensions (p. 1029)
Master Password and Confirm Password	Enter a password that contains from 8 to 128 printable ASCII characters (excluding /, ", and @) for your master password, then type the password again in the Confirm Password box.

8. On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the PostgreSQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then choose **Create database**.

For This Parameter	Do This
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, choose Not in VPC . For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 412) .
Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, choose default , which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC. For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 412) .
Publicly Accessible	Choose Yes to give the DB instance a public IP address, meaning that it is accessible outside the VPC; otherwise, choose No , so the DB instance is only accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB Instance in a VPC from the Internet (p. 422) .
Availability Zone	Use the default value of No Preference unless you want to specify an Availability Zone.
VPC Security Group	If you are a new customer to AWS, choose the default VPC. If you created a VPC security group, choose the VPC security group you previously created.
Database Name	Enter a name for your database of up to 63 alpha-numeric characters. If you do not provide a name, the default "postgres" database is created. To create additional databases, connect to the DB instance and use the SQL command <code>CREATE DATABASE</code> . For more information about connecting to the DB instance, see Connecting to a DB Instance Running the PostgreSQL Database Engine (p. 976) .
Database Port	Specify a port you want to use to access the database. PostgreSQL installations default to port 5432.
DB Parameter Group	Use the default value unless you have created your own parameter group.
Option Group	Use the default value unless you have created your own option group.
Copy Tags To Snapshots	Choose this option to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .

For This Parameter	Do This
Enable Encryption	Choose Yes to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 389) .
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to 1 .
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of No Preference .
Enable Enhanced Monitoring	Choose Yes to enable real-time OS monitoring. Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You are only charged for Enhanced Monitoring that exceeds the free tier provided by Amazon CloudWatch Logs.
Monitoring Role	Choose Default to use the default IAM role.
Granularity	Choose 60 to monitor the instance every minute.
Auto Minor Version Upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available.
Maintenance Window	Choose the 30-minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose No Preference .

9. On the final page, choose **Create database**.
10. On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.



The screenshot shows the 'Databases' section of the Amazon RDS console. At the top, there are buttons for 'Group resources', 'Actions', 'Restore from S3', and a prominent orange 'Create database' button. Below this is a search bar labeled 'Filter databases'. The main area displays a table with columns: DB Name, Role, Engine, Region & AZ, Size, Status, and CPU. The table lists several DB instances, including 'mymariadb', 'mysql-instance1', 'testmysql-rr', 'mysqlserver-instance1', 'oracle-instance1', 'postgresql-test', 'testaurorysql-cl', and 'testaurorysql'. The 'testaurorysql-cl' instance is expanded to show its child 'testaurorysql' instance. All instances are currently listed as 'Available'.

Connecting to a PostgreSQL DB Instance

After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. It is important to note that the security group you assigned to the DB instance when you created it must allow access to the DB instance. If you have difficulty connecting to the DB instance, the problem is most often with the access rules you set up in the security group you assigned to the DB instance.

This section shows two ways to connect to a PostgreSQL DB instance. The first example uses *pgAdmin*, a popular Open Source administration and development tool for PostgreSQL. You can download and use *pgAdmin* without having a local instance of PostgreSQL on your client computer. The second example uses *psql*, a command line utility that is part of a PostgreSQL installation. To use *psql*, you must have a PostgreSQL installed on your client computer or have installed the *psql* client on your machine.

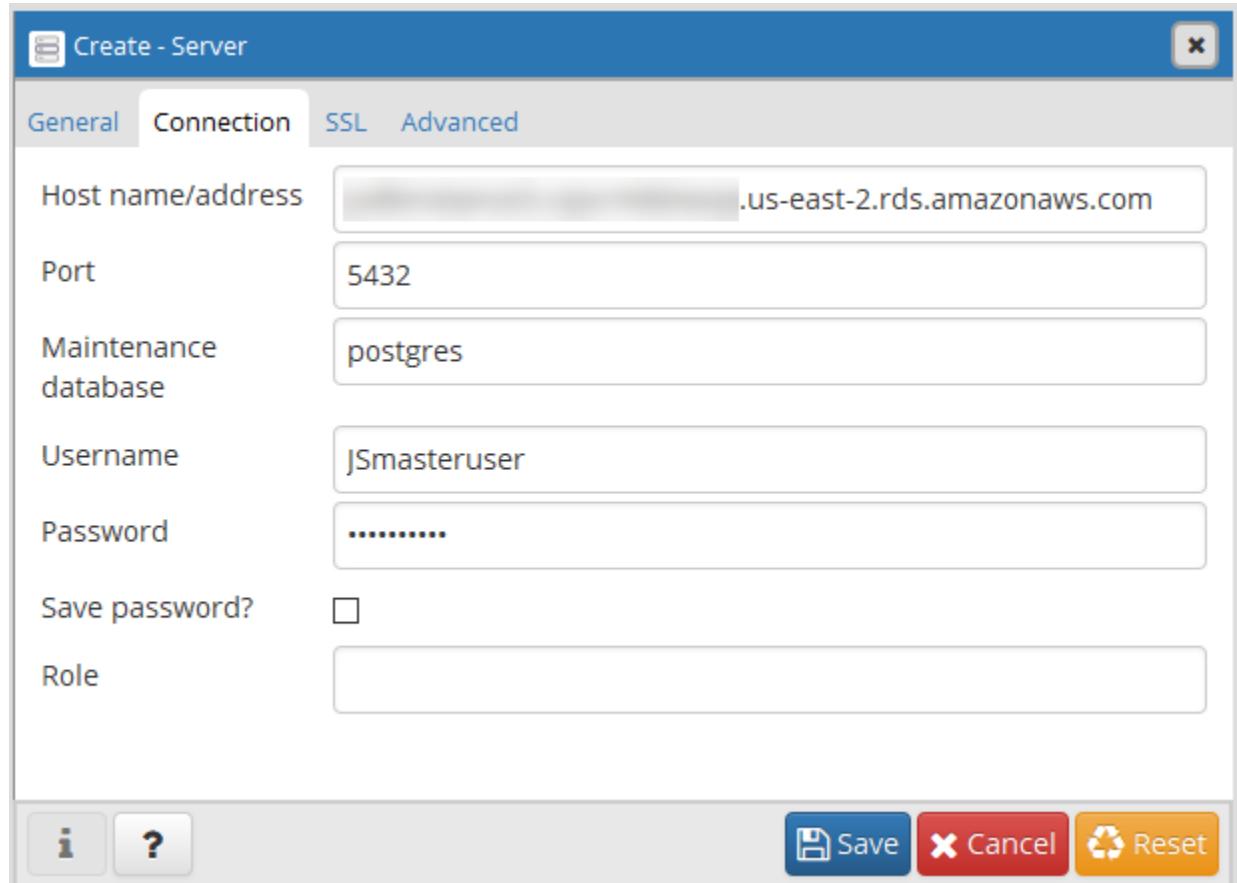
In this example, you connect to a PostgreSQL DB instance using *pgAdmin*.

Using pgAdmin to Connect to a PostgreSQL DB Instance

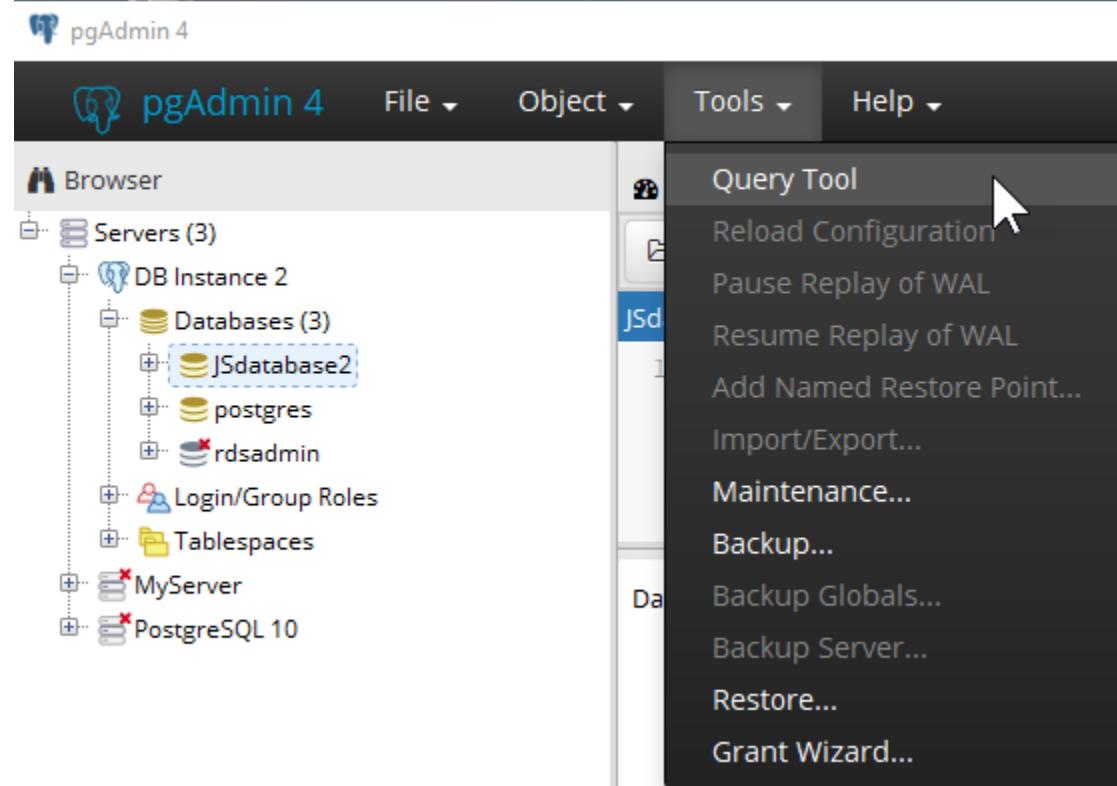
To connect to a PostgreSQL DB instance using pgAdmin

1. Launch the *pgAdmin* application on your client computer. You can install *pgAdmin* from <http://www.pgadmin.org/>.
2. Choose **Add Server** from the **File** menu.
3. In the **New Server Registration** dialog box, enter the DB instance endpoint (for example, mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com) in the **Host** box. Do not include the colon or port number as shown on the Amazon RDS console (mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com:5432).

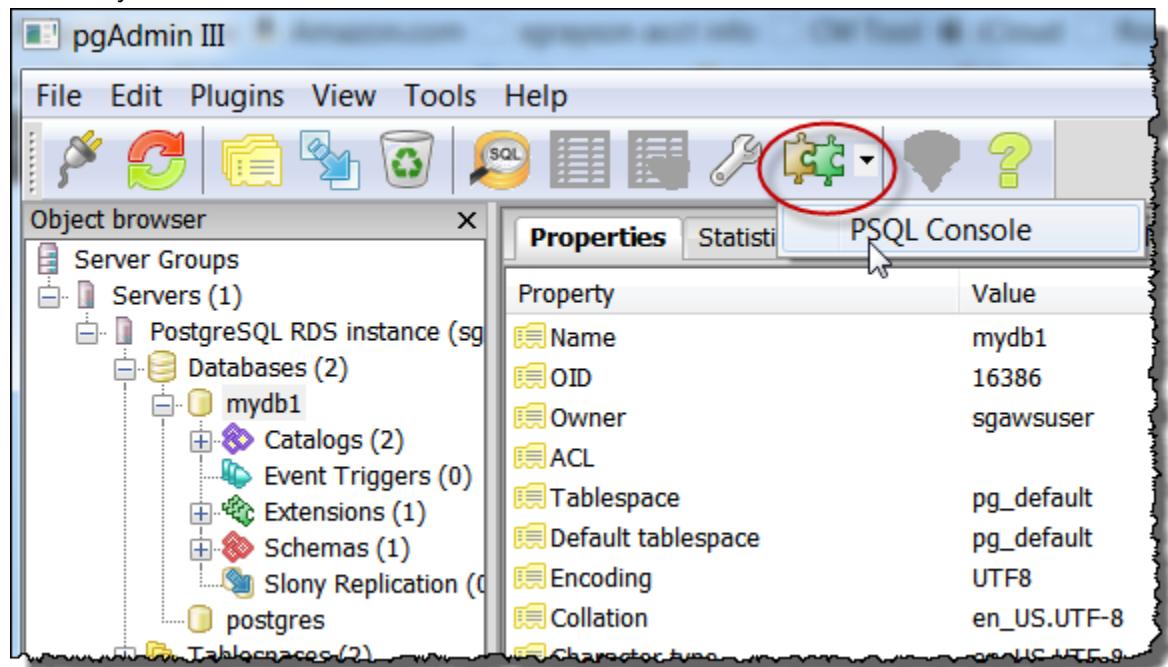
Enter the port you assigned to the DB instance into the **Port** box. Enter the user name and user password you entered when you created the DB instance into the **Username** and **Password** boxes, respectively.



4. Choose **OK**.
5. In the **Object browser**, expand the **Server Groups**. Choose the Server (the DB instance) you created, and then choose the database name.



6. Choose the plugin icon and choose **PSQL Console**. The *psql* command window opens for the default database you created.



7. Use the command window to enter SQL or *psql* commands. Enter \q to close the window.

Using *psql* to Connect to a PostgreSQL DB Instance

If your client computer has PostgreSQL installed, you can use a local instance of *psql* to connect to a PostgreSQL DB instance. To connect to your PostgreSQL DB instance using *psql*, you need to provide host information and access credentials.

The following format is used to connect to a PostgreSQL DB instance on Amazon RDS:

```
psql --host=<DB instance endpoint> --port=<port> --username=<master user name> --password  
--dbname=<database name>
```

For example, the following command connects to a database called `mypgdb` on a PostgreSQL DB instance called `mypostgresql` using fictitious credentials:

```
psql --host=mypostgresql.c6c8mwvfdgv0.us-west-2.rds.amazonaws.com --port=5432 --  
username=awsuser --password --dbname=mypgdb
```

Troubleshooting Connection Issues

By far the most common problem that occurs when attempting to connect to a database on a DB instance is the access rules in the security group assigned to the DB instance. If you used the default DB security group when you created the DB instance, chances are good that the security group did not have the rules that allow you to access the instance. For more information about Amazon RDS security groups, see [Controlling Access with Security Groups \(p. 394\)](#).

The most common error is *could not connect to server: Connection timed out*. If you receive this error, check that the host name is the DB instance endpoint and that the port number is correct. Check that the security group assigned to the DB instance has the necessary rules to allow access through any firewall your connection may be going through.

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
3. Choose the DB instance you wish to delete.
4. Choose **Actions**, and then choose **Delete**.
5. For **Create final snapshot?**, choose **No**, and select the acknowledgment.
6. Choose **Delete**.

Tutorial: Create a Web Server and an Amazon RDS Database

This tutorial helps you install an Apache web server with PHP, and create a MySQL database. The web server runs on an Amazon EC2 instance using Amazon Linux, and the MySQL database is an Amazon RDS MySQL DB instance. Both the Amazon EC2 instance and the Amazon RDS DB instance run in a VPC based in Amazon Virtual Private Cloud service (Amazon VPC).

Note

This tutorial works with Amazon Linux and might not work for other versions of Linux such as Ubuntu.

Before you begin this tutorial, you must have a VPC with both public and private subnets, and corresponding security groups. If you don't have these, complete the following tasks in [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#):

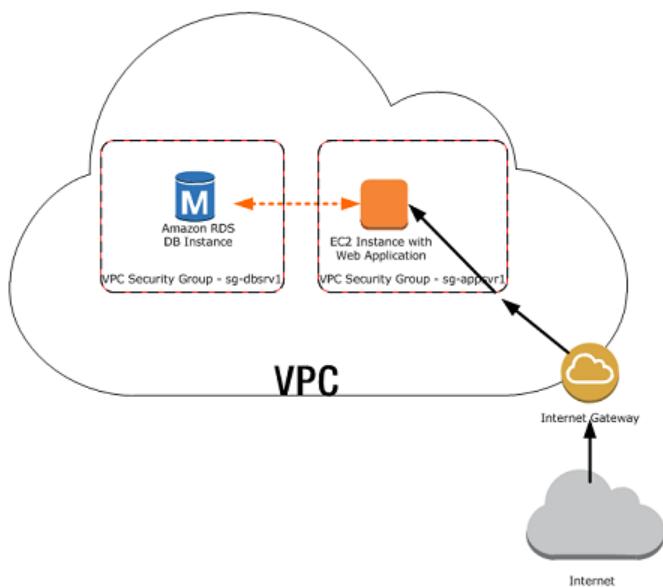
1. [Create a VPC with Private and Public Subnets \(p. 427\)](#)
2. [Create Additional Subnets \(p. 428\)](#)
3. [Create a VPC Security Group for a Public Web Server \(p. 429\)](#)
4. [Create a VPC Security Group for a Private Amazon RDS DB Instance \(p. 430\)](#)
5. [Create a DB Subnet Group \(p. 430\)](#)

In the tutorial that follows, you specify the VPC, subnets, and security groups when you create the DB instance. You also specify them when you create the EC2 instance that will host your web server. The VPC, subnets, and security groups are required for the DB instance and the web server to communicate. After the VPC is set up, this tutorial shows you how to you create the DB instance and install the web server. You connect your web server to your RDS DB instance in the VPC using the DB instance endpoint.

In this tutorial, you perform the following procedures:

- [Step 1: Create an RDS DB Instance \(p. 51\)](#)
- [Step 2: Create an EC2 Instance and Install a Web Server \(p. 56\)](#)

The following diagram shows the configuration when the tutorial is complete.



Step 1: Create an RDS DB Instance

In this step you create an Amazon RDS MySQL DB instance that maintains the data used by a web application.

Important

Before you begin this step, you must have a VPC with both public and private subnets, and corresponding security groups. If you don't have these, see [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#). Complete the steps in [Create a VPC with Private and Public Subnets \(p. 427\)](#), [Create Additional Subnets \(p. 428\)](#), [Create a VPC Security Group for a Public Web Server \(p. 429\)](#), and [Create a VPC Security Group for a Private Amazon RDS DB Instance \(p. 430\)](#).

To launch a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top-right corner of the AWS Management Console, choose the AWS Region in which you want to create the DB instance. This example uses the US West (Oregon) region.
3. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
4. Choose **Create database** to open the **Select engine** page.
5. On the **Select engine** page, shown following, choose **MySQL**, and then choose **Next**.

Select engine

Engine options

- Amazon Aurora
Amazon Aurora
- MySQL

- MariaDB

- PostgreSQL

- Oracle

- Microsoft SQL Server


MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 16 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.

Only enable options eligible for RDS Free Usage Tier [info](#)

[Cancel](#) **Next**

6. On the **Choose use case** page, choose **Dev/Test – MySQL**, and then choose **Next**.
7. On the **Specify DB details** page, shown following, set these values:
 - **License model:** Use the default value.
 - **DB engine version:** Use the default value.
 - **DB instance class:** db.t2.small
 - **Multi-AZ deployment:** No
 - **Storage type:** General Purpose (SSD)
 - **Allocated storage:** 20 GiB
 - **DB instance identifier:** tutorial-db-instance
 - **Master username:** tutorial_user
 - **Master password:** Choose a password.
 - **Confirm password:** Retype the password.

Specify DB details

Instance specifications
Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator.

DB engine: MySQL Community Edition

License model info: general-public-license

DB engine version info: mysql 5.6.37

Known Issues/Limitations
Review the Known Issues/Limitations to learn about potential compatibility issues with specific database versions.

Free tier
The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#).

Only enable options eligible for RDS Free Usage Tier info

DB instance class info: db.t2.small — 1 vCPU, 2 GiB RAM

Multi-AZ deployment info:
 Create replica in different zone: Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.
 No

Storage type info: General Purpose (SSD)

Allocated storage: 20 GB (Minimum: 20 GB, Maximum: 16384 GB) Higher allocated storage may improve IOPS performance.

Settings

DB instance identifier info: Specify a name that is unique for all DB instances owned by your AWS account in the current region.
tutorial-db-instance

DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance".

Master username info: Specify an alphanumeric string that defines the login ID for the master user.
tutorial_user

Master Username must start with a letter.

Master password info: Confirm password info:
Master Password must be at least eight characters long, as in "mypassword".

Cancel **Previous** **Next**

8. Choose **Next** and set the following values in the **Configure advanced settings** page:

- **Virtual Private Cloud (VPC):** Choose an existing VPC with both public and private subnets, such as the tutorial-vpc (`vpc-identifier`) created in [Create a VPC with Private and Public Subnets \(p. 427\)](#)

Note

The VPC must have subnets in different Availability Zones.

- **Subnet group:** The DB subnet group for the VPC, such as the `tutorial-db-subnet-group` created in [Create a DB Subnet Group \(p. 430\)](#)
- **Public accessibility:** No
- **Availability zone:** No Preference
- **VPC security groups:** Choose an existing VPC security group that is configured for private access, such as the `tutorial-db-securitygroup` created in [Create a VPC Security Group for a Private Amazon RDS DB Instance \(p. 430\)](#).

Remove other security groups, such as the default security group, by choosing the **X** associated with each.

- **Database name:** sample

Leave the default settings for the other options.

Configure advanced settings

Network & Security

Virtual Private Cloud (VPC) [Info](#)

VPC defines the virtual networking environment for this DB instance.

tutorial-vpc ([REDACTED]) ▾



Only VPCs with a corresponding DB subnet group are listed.

Subnet group [Info](#)

DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

tutorial-db-subnet-group ▾



Public accessibility [Info](#)

Yes

EC2 instances and devices outside of the VPC hosting the DB instance will connect to the DB instances. You must also select one or more VPC security groups that specify which EC2 instances and devices can connect to the DB instance.

No

DB instance will not have a public IP address assigned. No EC2 instance or devices outside of the VPC will be able to connect.

Availability zone [Info](#)

No preference ▾



VPC security groups

Security groups have rules authorizing connections from all the EC2 instances and devices that need to access the DB instance.

Create new VPC security group

Choose existing VPC security groups

Choose VPC security groups ▾



tutorial-db-securitygroup X

Database options

Database name

sample

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

9. To create your Amazon RDS MySQL DB instance, choose **Create database**.
10. On the next page, choose **View DB instances details** to view your RDS MySQL DB instance.
11. Wait for the **DB instance status** of your new DB instance to show as **available**. Then scroll to the **Connect** section, shown following.

The screenshot shows the 'Connect' section of the AWS RDS console. It displays the endpoint (tutorial-db-instance.rds.amazonaws.com) and port (3306) for a MySQL DB instance. A note indicates that the instance is not publicly accessible. Below this, the 'Security group rules (2)' section is shown, listing two inbound rules from the 'tutorial-db-securitygroup'. The first rule is a Security Group - Inbound rule, and the second is a CIDR/IP - Outbound rule allowing 0.0.0.0/0.

Make note of the endpoint and port for your DB instance. You will use this information to connect your web server to your RDS DB instance.

To make sure your RDS MySQL DB instance is as secure as possible, verify that sources outside of the VPC cannot connect to your RDS MySQL DB instance.

Next Step

[Step 2: Create an EC2 Instance and Install a Web Server \(p. 56\)](#)

Step 2: Create an EC2 Instance and Install a Web Server

In this step you create a web server to connect to the Amazon RDS DB instance that you created in [Step 1: Create an RDS DB Instance \(p. 51\)](#).

Launch an EC2 Instance

First you create an Amazon EC2 instance in the public subnet of your VPC.

To launch an EC2 instance

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Choose **EC2 Dashboard**, and then choose **Launch Instance**, as shown following.

Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

2 Running Instances	1 Elastic IPs
0 Dedicated Hosts	0 Snapshots
2 Volumes	0 Load Balancers
2 Key Pairs	24 Security Groups
0 Placement Groups	

EC2 Spot. Save up to 90% off On-Demand Prices. Turbo Boost your Workloads. [Get started with Amazon EC2 Spot Instances.](#)

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

[Launch Instance](#)

Note: Your instances will launch in the US East (N. Virginia) region

Service Health

Scheduled Events

3. Choose the **Amazon Linux** Amazon Machine Image (AMI), as shown following.

Step 1: Choose an Amazon Machine Image (AMI) [Cancel and Exit](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start	AMIs	Sort by:
<input type="checkbox"/> My AMIs	 Amazon Linux AMI 2017.09.1 (HVM), SSD Volume Type - ami-97785bed Select	1 to 36 of 36 AMIs
<input type="checkbox"/> AWS Marketplace	 Amazon Linux 2 LTS Candidate AMI 2017.12.0 (HVM), SSD Volume Type - ami-428aa938 Select	64-bit
<input type="checkbox"/> Community AMIs		
<input type="checkbox"/> Free tier only <small>(1)</small>		

The screenshot shows the 'Amazon Linux AMI 2017.09.1 (HVM), SSD Volume Type - ami-97785bed' entry highlighted with a red box. This entry includes a 'select' button and a '64-bit' link. Below it, another AMI entry for 'Amazon Linux 2 LTS Candidate AMI 2017.12.0 (HVM), SSD Volume Type - ami-428aa938' is partially visible.

4. Choose the **t2.small** instance type, as shown following, and then choose **Next: Configure Instance Details**.

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Currently selected: t2.small (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 2 GiB memory, EBS only)								
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.micro	1	1	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	m5.large	2	8	EBS only	Yes	Up to 10 Gigabit	Yes

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Instance Details](#)

5. On the **Configure Instance Details** page, shown following, set these values and leave the other values as their defaults:

- **Network:** Choose the VPC with both public and private subnets that you chose for the DB instance, such as the `tutorial-vpc` (`vpc-identifier`) created in [Create a VPC with Private and Public Subnets \(p. 427\)](#).
- **Subnet:** Choose an existing public subnet, such as `subnet-identifier` | Tutorial public | `us-west-2a` created in [Create a VPC Security Group for a Public Web Server \(p. 429\)](#).
- **Auto-assign Public IP:** Choose **Enable**.

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 1

Purchasing option: Request Spot Instances

Network: vpc-971c12ee | tutorial-vpc

Subnet: subnet-0ccde220 | Tutorial public | us-east-1a
249 IP Addresses available

Auto-assign Public IP: Enable

IAM role: None

Shutdown behavior: Stop

Enable termination protection: Protect against accidental termination

Monitoring: Enable CloudWatch detailed monitoring
Additional charges apply.

Tenancy: Shared - Run a shared hardware instance
Additional charges will apply for dedicated tenancy.

T2 Unlimited: Enable
Additional charges may apply

6. Choose **Next: Add Storage**.
7. On the **Add Storage** page, keep the default values and choose **Next: Add Tags**.
8. On the **Add Tags** page, shown following, choose **Add Tag**, then enter **Name** for **Key** and enter **tutorial-web-server** for **Value**.

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key	(127 characters maximum)	Value	(255 characters maximum)	Instances	Volumes
Name	tutorial-web-server	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="button" value="Add another tag"/> (Up to 50 tags maximum)					

9. Choose **Next: Configure Security Group**.
10. On the **Configure Security Group** page, shown following, choose **Select an existing security group**, and then choose an existing security group, such as the **tutorial-securitygroup** created in [Create a VPC Security Group for a Public Web Server \(p. 429\)](#). The security group must include inbound rules for SSH and HTTP access.

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

Security Group ID	Name	Description	Actions
sg-[REDACTED]	default	default VPC security group	Copy to new
sg-[REDACTED]	tutorial-db-securitygroup	Tutorial DB Instance Security Group	Copy to new
<input checked="" type="checkbox"/> sg-[REDACTED]	tutorial-securitygroup	Tutorial Security Group	Copy to new

Inbound rules for sg-0b56af78 (Selected security groups: sg-0b56af78)

Type <i>(i)</i>	Protocol <i>(i)</i>	Port Range <i>(i)</i>	Source <i>(i)</i>	Description <i>(i)</i>
HTTP	TCP	80	203.0.113.25/32	
SSH	TCP	22	203.0.113.25/32	

[Cancel](#) [Previous](#) [Review and Launch](#)

11. Choose **Review and Launch**.
12. On the **Review Instance Launch** page, shown following, verify your settings and then choose **Launch**.

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

⚠ Your instance configuration is not eligible for the free usage tier

To launch an instance that's eligible for the free usage tier, check your AMI selection, instance type, configuration options, or storage devices. Learn more about [free usage tier](#) eligibility and usage restrictions.

[Don't show me this again](#)

[Edit AMI](#)

AMI Details



Amazon Linux AMI 2017.09.1 (HVM), SSD Volume Type - ami-877785bed

Free tier eligible

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root Device Type: ebs Virtualization type: hvm

Instance Type

[Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.small	Variable	1	2	EBS only	-	Low to Moderate

Security Groups

[Edit security groups](#)

Security Group ID	Name	Description
sg-0b56af78	tutorial-securitygroup	Tutorial Security Group

All selected security groups inbound rules

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	203.0.113.25/32	
SSH	TCP	22	203.0.113.25/32	

Instance Details

[Edit instance details](#)

[Cancel](#) [Previous](#) [Launch](#)

13. On the **Select an existing key pair or create a new key pair** page, shown following, choose **Create a new key pair** and set **Key pair name** to **tutorial-key-pair**. Choose **Download Key Pair**, and then save the key pair file on your local machine. You use this key pair file to connect to your EC2 instance.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair name

tutorial-key-pair

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue.
Store it in a secure and accessible location. You will not be able to download the file again after it's created.

Cancel Launch Instances

14. To launch your EC2 instance, choose **Launch Instances**. On the **Launch Status** page, shown following, note the identifier for your new EC2 instance, for example: i-0288d65fd4470b6a9.

Launch Status

Your instances are now launching
The following instance launches have been initiated: **i-0288d65fd4470b6a9** [View launch log](#)

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click [View Instances](#) to monitor your instances' status. Once your instances are in the **running** state, you can [connect](#) to them from the Instances screen. [Find out](#) how to connect to your instances.

▼ Here are some helpful resources to get you started

- [How to connect to your Linux instance](#)
- [Learn about AWS Free Usage Tier](#)
- [Amazon EC2: User Guide](#)
- [Amazon EC2: Discussion Forum](#)

While your instances are launching you can also

- [Create status check alarms](#) to be notified when these instances fail status checks. (Additional charges may apply)
- [Create and attach additional EBS volumes](#) (Additional charges may apply)
- [Manage security groups](#)

[View Instances](#)

15. To find your instance, choose **[View Instances](#)**.
16. Wait until **Instance Status** for your instance reads as **running** before continuing.

Install an Apache Web Server with PHP

Next you connect to your EC2 instance and install the web server.

To connect to your EC2 instance and install the Apache web server with PHP

1. To connect to the EC2 instance that you created earlier, follow the steps in [Connect to Your Instance](#).
2. To get the latest bug fixes and security updates, update the software on your EC2 instance by using the following command:

Note

The `-y` option installs the updates without asking for confirmation. To examine updates before installing, omit this option.

```
[ec2-user ~]$ sudo yum update -y
```

3. After the updates complete, install the Apache web server with the PHP software package using the **yum install** command, which installs multiple software packages and related dependencies at the same time.

```
[ec2-user ~]$ sudo yum install -y httpd24 php56 php56-mysqlnd
```

For more information, see [Updating Instance Software](#).

4. Start the web server with the command shown following.

```
[ec2-user ~]$ sudo service httpd start
```

You can test that your web server is properly installed and started by entering the public DNS name of your EC2 instance in the address bar of a web browser, for example: `http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com`. If your web server is running, then you see the Apache test page. If you don't see the Apache test page, then verify that your inbound rules for the VPC security group that you created in [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#) include a rule allowing HTTP (port 80) access for the IP address you use to connect to the web server.

Note

The Apache test page appears only when there is no content in the document root directory, `/var/www/html`. After you add content to the document root directory, your content appears at the public DNS address of your EC2 instance instead of the Apache test page.

5. Configure the web server to start with each system boot using the `chkconfig` command.

```
[ec2-user ~]$ sudo chkconfig httpd on
```

To allow `ec2-user` to manage files in the default root directory for your Apache web server, you need to modify the ownership and permissions of the `/var/www` directory. In this tutorial, you add a group named `www` to your EC2 instance, and then you give that group ownership of the `/var/www` directory and add write permissions for the group. Any members of that group can then add, delete, and modify files for the web server.

To set file permissions for the Apache web server

1. Add the `www` group to your EC2 instance with the following command.

```
[ec2-user ~]$ sudo groupadd www
```

2. Add the `ec2-user` user to the `www` group.

```
[ec2-user ~]$ sudo usermod -a -G www ec2-user
```

3. To refresh your permissions and include the new `www` group, log out.

```
[ec2-user ~]$ exit
```

4. Log back in again and verify that the `www` group exists with the `groups` command.

```
[ec2-user ~]$ groups
ec2-user wheel www
```

5. Change the group ownership of the /var/www directory and its contents to the www group.

```
[ec2-user ~]$ sudo chown -R root:www /var/www
```

6. Change the directory permissions of /var/www and its subdirectories to add group write permissions and set the group ID on subdirectories created in the future.

```
[ec2-user ~]$ sudo chmod 2775 /var/www
[ec2-user ~]$ find /var/www -type d -exec sudo chmod 2775 {} +
```

7. Recursively change the permissions for files in the /var/www directory and its subdirectories to add group write permissions.

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0664 {} +
```

Connect your Apache web server to your RDS DB instance

Next, you add content to your Apache web server that connects to your Amazon RDS DB instance.

To add content to the Apache web server that connects to your RDS DB instance

1. While still connected to your EC2 instance, change the directory to /var/www and create a new subdirectory named inc.

```
[ec2-user ~]$ cd /var/www
[ec2-user ~]$ mkdir inc
[ec2-user ~]$ cd inc
```

2. Create a new file in the inc directory named dbinfo.inc, and then edit the file by calling nano (or the editor of your choice).

```
[ec2-user ~]$ >dbinfo.inc
[ec2-user ~]$ nano dbinfo.inc
```

3. Add the following contents to the dbinfo.inc file, where *endpoint* is the endpoint of your RDS MySQL DB instance, without the port, and *master password* is the master password for your RDS MySQL DB instance.

Note

Placing the user name and password information in a folder that is not part of the document root for your web server reduces the possibility of your security information being exposed.

```
<?php

define('DB_SERVER', 'endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');

?>
```

4. Save and close the dbinfo.inc file.
5. Change the directory to /var/www/html.

```
[ec2-user ~]$ cd /var/www/html
```

6. Create a new file in the html directory named SamplePage.php, and then edit the file by calling nano (or the editor of your choice).

```
[ec2-user ~]$ >SamplePage.php
[ec2-user ~]$ nano SamplePage.php
```

7. Add the following contents to the SamplePage.php file:

Note

Placing the user name and password information in a folder that is not part of the document root for your web server reduces the possibility of your security information being exposed.

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Sample page</h1>
<?php

/* Connect to MySQL and select the database. */
$connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
mysqli_connect_error();

$dbname = mysqli_select_db($connection, DB_DATABASE);

/* Ensure that the Employees table exists. */
VerifyEmployeesTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the Employees table. */
$employee_name = htmlentities($_POST['Name']);
$employee_address = htmlentities($_POST['Address']);

if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
}
?>

<!-- Input form --&gt;
&lt;form action=&lt;?PHP echo $_SERVER['SCRIPT_NAME'] ?&gt;" method="POST"&gt;
    &lt;table border="0"&gt;
        &lt;tr&gt;
            &lt;td&gt;Name&lt;/td&gt;
            &lt;td&gt;Address&lt;/td&gt;
        &lt;/tr&gt;
        &lt;tr&gt;
            &lt;td&gt;
                &lt;input type="text" name="Name" maxlength="45" size="30" /&gt;
            &lt;/td&gt;
            &lt;td&gt;
                &lt;input type="text" name="Address" maxlength="90" size="60" /&gt;
            &lt;/td&gt;
        &lt;td&gt;</pre>
```

```

        <input type="submit" value="Add Data" />
    </td>
</tr>
</table>
</form>

<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
    <tr>
        <td>ID</td>
        <td>Name</td>
        <td>Address</td>
    </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM Employees");

while($query_data = mysqli_fetch_row($result)) {
    echo "<tr>";
    echo "<td>", $query_data[0], "</td>",
        "<td>", $query_data[1], "</td>",
        "<td>", $query_data[2], "</td>";
    echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

    mysqli_free_result($result);
    mysqli_close($connection);

?>

</body>
</html>

<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
    $n = mysqli_real_escape_string($connection, $name);
    $a = mysqli_real_escape_string($connection, $address);

    $query = "INSERT INTO `Employees` (`Name`, `Address`) VALUES ('$n', '$a');";

    if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
    if(!TableExists("Employees", $connection, $dbName))
    {
        $query = "CREATE TABLE `Employees` (
            `ID` int(11) NOT NULL AUTO_INCREMENT,
            `Name` varchar(45) DEFAULT NULL,
            `Address` varchar(90) DEFAULT NULL,
            PRIMARY KEY (`ID`),
            UNIQUE KEY `ID_UNIQUE` (`ID`)
        ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1";

        if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</p>");
```

```
    }

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
    $t = mysqli_real_escape_string($connection, $tableName);
    $d = mysqli_real_escape_string($connection, $dbName);

    $checktable = mysqli_query($connection,
        "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t' AND
TABLE_SCHEMA = '$d'");

    if(mysqli_num_rows($checktable) > 0) return true;

    return false;
}
?>
```

8. Save and close the SamplePage.php file.
9. Verify that your web server successfully connects to your RDS MySQL DB instance by opening a web browser and browsing to <http://EC2 instance endpoint>/SamplePage.php, for example: <http://ec2-55-122-41-31.us-west-2.compute.amazonaws.com/SamplePage.php>.

You can use SamplePage.php to add data to your RDS MySQL DB instance. The data that you add is then displayed on the page.

To make sure your RDS MySQL DB instance is as secure as possible, verify that sources outside of the VPC cannot connect to your RDS MySQL DB instance.

Tutorials

The following tutorials show you how to perform common tasks that use Amazon RDS:

- [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#)
- [Tutorial: Create a Web Server and an Amazon RDS Database \(p. 50\)](#)
- [Tutorial: Restore a DB Instance from a DB Snapshot \(p. 239\)](#)

For videos, see [AWS Instructional Videos and Labs](#).

Best Practices for Amazon RDS

Learn best practices for working with Amazon RDS. As new best practices are identified, we will keep this section up to date.

Topics

- [Amazon RDS Basic Operational Guidelines \(p. 70\)](#)
- [DB Instance RAM Recommendations \(p. 71\)](#)
- [Amazon RDS Security Best Practices \(p. 71\)](#)
- [Using Enhanced Monitoring to Identify Operating System Issues \(p. 71\)](#)
- [Using Metrics to Identify Performance Issues \(p. 72\)](#)
- [Best Practices for Working with MySQL Storage Engines \(p. 75\)](#)
- [Best Practices for Working with MariaDB Storage Engines \(p. 76\)](#)
- [Best Practices for Working with Oracle \(p. 76\)](#)
- [Best Practices for Working with PostgreSQL \(p. 77\)](#)
- [Best Practices for Working with SQL Server \(p. 78\)](#)
- [Working with DB Parameter Groups \(p. 79\)](#)
- [Amazon RDS Best Practices Presentation Video \(p. 79\)](#)

Amazon RDS Basic Operational Guidelines

The following are basic operational guidelines that everyone should follow when working with Amazon RDS. Note that the Amazon RDS Service Level Agreement requires that you follow these guidelines:

- Monitor your memory, CPU, and storage usage. Amazon CloudWatch can be set up to notify you when usage patterns change or when you approach the capacity of your deployment, so that you can maintain system performance and availability.
- Scale up your DB instance when you are approaching storage capacity limits. You should have some buffer in storage and memory to accommodate unforeseen increases in demand from your applications.
- Enable automatic backups and set the backup window to occur during the daily low in write IOPS.
- If your database workload requires more I/O than you have provisioned, recovery after a failover or database failure will be slow. To increase the I/O capacity of a DB instance, do any or all of the following:
 - Migrate to a DB instance class with High I/O capacity.
 - Convert from standard storage to either General Purpose or Provisioned IOPS storage, depending on how much of an increase you need. For information on available storage types, see [Amazon RDS Storage Types \(p. 103\)](#).

If you convert to Provisioned IOPS storage, make sure you also use a DB instance class that is optimized for Provisioned IOPS. For information on Provisioned IOPS, see [Provisioned IOPS SSD Storage \(p. 105\)](#).

- If you are already using Provisioned IOPS storage, provision additional throughput capacity.
- If your client application is caching the Domain Name Service (DNS) data of your DB instances, set a time-to-live (TTL) value of less than 30 seconds. Because the underlying IP address of a DB instance can change after a failover, caching the DNS data for an extended time can lead to connection failures if your application tries to connect to an IP address that no longer is in service.

- Test failover for your DB instance to understand how long the process takes for your use case and to ensure that the application that accesses your DB instance can automatically connect to the new DB instance after failover.

DB Instance RAM Recommendations

An Amazon RDS performance best practice is to allocate enough RAM so that your working set resides almost completely in memory. To tell if your working set is almost all in memory, check the ReadIOPS metric (using Amazon CloudWatch) while the DB instance is under load. The value of ReadIOPS should be small and stable. If scaling up the DB instance class—to a class with more RAM—results in a dramatic drop in ReadIOPS, your working set was not almost completely in memory. Continue to scale up until ReadIOPS no longer drops dramatically after a scaling operation, or ReadIOPS is reduced to a very small amount. For information on monitoring a DB instance's metrics, see [Viewing DB Instance Metrics \(p. 254\)](#).

Amazon RDS Security Best Practices

Use AWS IAM accounts to control access to Amazon RDS API actions, especially actions that create, modify, or delete RDS resources such as DB instances, security groups, option groups, or parameter groups, and actions that perform common administrative actions such as backing up and restoring DB instances, or configuring Provisioned IOPS storage.

- Assign an individual IAM account to each person who manages RDS resources. Do not use AWS root credentials to manage Amazon RDS resources; you should create an IAM user for everyone, including yourself.
- Grant each user the minimum set of permissions required to perform his or her duties.
- Use IAM groups to effectively manage permissions for multiple users.
- Rotate your IAM credentials regularly.

For more information about IAM, go to [AWS Identity and Access Management](#). For information on IAM best practices, go to [IAM Best Practices](#).

Use the AWS Management Console, the AWS CLI, or the Amazon RDS API to change the password for your master user. If you use another tool, such as a SQL client, to change the master user password, it might result in privileges being revoked for the user unintentionally.

Using Enhanced Monitoring to Identify Operating System Issues

Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You can view the metrics for your DB instance using the console, or consume the Enhanced Monitoring JSON output from Amazon CloudWatch Logs in a monitoring system of your choice. For more information about Enhanced Monitoring, see [Enhanced Monitoring \(p. 256\)](#)

Enhanced Monitoring is available for the following database engines:

- MariaDB
- Microsoft SQL Server
- MySQL version 5.5 or later

- Oracle
- PostgreSQL

Enhanced monitoring is available for all DB instance classes except for db.m1.small. Enhanced Monitoring is available in all regions except for AWS GovCloud (US-West).

Using Metrics to Identify Performance Issues

To identify performance issues caused by insufficient resources and other common bottlenecks, you can monitor the metrics available for your Amazon RDS DB instance.

Viewing Performance Metrics

You should monitor performance metrics on a regular basis to see the average, maximum, and minimum values for a variety of time ranges. If you do so, you can identify when performance is degraded. You can also set Amazon CloudWatch alarms for particular metric thresholds so you are alerted if they are reached.

In order to troubleshoot performance issues, it's important to understand the baseline performance of the system. When you set up a new DB instance and get it running with a typical workload, you should capture the average, maximum, and minimum values of all of the performance metrics at a number of different intervals (for example, one hour, 24 hours, one week, two weeks) to get an idea of what is normal. It helps to get comparisons for both peak and off-peak hours of operation. You can then use this information to identify when performance is dropping below standard levels.

To view performance metrics

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose a DB instance.
3. Choose **Show Monitoring**. The first eight performance metrics display. The metrics default to showing information for the current day.
4. Use the numbered buttons at top right to page through the additional metrics, or choose **Show All** to see all metrics.
5. Choose a performance metric to adjust the time range in order to see data for other than the current day. You can change the **Statistic**, **Time Range**, and **Period** values to adjust the information displayed. For example, to see the peak values for a metric for each day of the last two weeks, set **Statistic** to **Maximum**, **Time Range** to **Last 2 Weeks**, and **Period** to **Day**.

Note

Changing the **Statistic**, **Time Range**, and **Period** values changes them for all metrics. The updated values persist for the remainder of your session or until you change them again.

You can also view performance metrics using the CLI or API. For more information, see [Viewing DB Instance Metrics \(p. 254\)](#).

To set a CloudWatch alarm

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose a DB instance.
3. Choose **Show Monitoring**, and then choose a performance metric to bring up the expanded view.

4. Choose **Create Alarm**.
5. On the **Create Alarm** page, identify what email address should receive the alert by choosing a value for **Send a notification to**. Choose **create topic** to the right of that box to create a new alarm recipient if necessary.
6. For **Whenever**, choose the alarm statistic to set.
7. For **of**, choose the alarm metric.
8. For **Is** and the unlabeled box to the right of it, set the alarm threshold, as shown following.

The screenshot shows the 'Create Alarm' wizard. The 'Whenever' section is highlighted with a red box, specifically the 'Is' dropdown and the '80' value. The 'Name of alarm' field contains 'awsrds-sql-carpc-High-CPU-Utilization'.

Create Alarm

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a threshold. To edit an alarm, first choose whom to notify and then define when the notification should be sent.

Send a notification to: carpccluster-default-alarms [create topic](#)

Whenever: Average of CPU Utilization

Is: > 80 Percent

For at least: 1 consecutive period(s) of 5 Minutes

Name of alarm: awsrds-sql-carpc-High-CPU-Utilization

9. For **For at least**, enter the number of times that the specified threshold must be reached in order to trigger the alarm.
10. For **consecutive period(s) of**, choose the period during which the threshold must have been reached in order to trigger the alarm.
11. For **Name of alarm**, enter a name for the alarm.
12. Choose **Create Alarm**.

The performance metrics page appears, and you can see the new alarm in the **CloudWatch Alarms** status bar. If you don't see the status bar, refresh your page.

Evaluating Performance Metrics

A DB instance has a number of different categories of metrics, and how to determine acceptable values depends on the metric.

CPU

- CPU Utilization – Percentage of computer processing capacity used.

Memory

- Freeable Memory – How much RAM is available on the DB instance, in megabytes. The red line in the Monitoring tab metrics is marked at 75% for CPU, Memory and Storage Metrics. If instance memory consumption frequently crosses that line, then this indicates that you should check your workload or upgrade your instance.
- Swap Usage – How much swap space is used by the DB instance, in megabytes.

Disk space

- Free Storage Space – How much disk space is not currently being used by the DB instance, in megabytes.

Input/output operations

- Read IOPS, Write IOPS – The average number of disk read or write operations per second.
- Read Latency, Write Latency – The average time for a read or write operation in milliseconds.
- Read Throughput, Write Throughput – The average number of megabytes read from or written to disk per second.
- Queue Depth – The number of I/O operations that are waiting to be written to or read from disk.

Network traffic

- Network Receive Throughput, Network Transmit Throughput – The rate of network traffic to and from the DB instance in megabytes per second.

Database connections

- DB Connections – The number of client sessions that are connected to the DB instance.

For more detailed individual descriptions of the performance metrics available, see [Amazon RDS Dimensions and Metrics](#).

Generally speaking, acceptable values for performance metrics depend on what your baseline looks like and what your application is doing. Investigate consistent or trending variances from your baseline. Advice about specific types of metrics follows:

- **High CPU or RAM consumption** – High values for CPU or RAM consumption might be appropriate, provided that they are in keeping with your goals for your application (like throughput or concurrency) and are expected.
- **Disk space consumption** – Investigate disk space consumption if space used is consistently at or above 85 percent of the total disk space. See if it is possible to delete data from the instance or archive data to a different system to free up space.
- **Network traffic** – For network traffic, talk with your system administrator to understand what expected throughput is for your domain network and Internet connection. Investigate network traffic if throughput is consistently lower than expected.
- **Database connections** – Consider constraining database connections if you see high numbers of user connections in conjunction with decreases in instance performance and response time. The best number of user connections for your DB instance will vary based on your instance class and the complexity of the operations being performed. You can determine the number of database connections by associating your DB instance with a parameter group where the *User Connections* parameter is set to other than 0 (unlimited). You can either use an existing parameter group or create a new one. For more information, see [Working with DB Parameter Groups \(p. 169\)](#).
- **IOPS metrics** – The expected values for IOPS metrics depend on disk specification and server configuration, so use your baseline to know what is typical. Investigate if values are consistently different than your baseline. For best IOPS performance, make sure your typical working set will fit into memory to minimize read and write operations.

For issues with any performance metrics, one of the first things you can do to improve performance is tune the most used and most expensive queries to see if that lowers the pressure on system resources. For more information, see [Tuning Queries \(p. 75\)](#)

If your queries are tuned and an issue persists, consider upgrading your Amazon RDS [DB Instance Class \(p. 82\)](#) to one with more of the resource (CPU, RAM, disk space, network bandwidth, I/O capacity) that is related to the issue you are experiencing.

Tuning Queries

One of the best ways to improve DB instance performance is to tune your most commonly used and most resource-intensive queries to make them less expensive to run.

MySQL Query Tuning

Go to [Optimizing SELECT Statements](#) in the MySQL documentation for more information on writing queries for better performance. You can also go to [MySQL Performance Tuning and Optimization Resources](#) for additional query tuning resources.

Oracle Query Tuning

Go to the [Database SQL Tuning Guide](#) in the Oracle documentation for more information on writing and analyzing queries for better performance.

SQL Server Query Tuning

Go to [Analyzing a Query](#) in the SQL Server documentation to improve queries for SQL Server DB instances. You can also use the execution-, index- and I/O-related data management views (DMVs) described in the [Dynamic Management Views and Functions](#) documentation to troubleshoot SQL Server query issues.

A common aspect of query tuning is creating effective indexes. You can use the [Database Engine Tuning Advisor](#) to get potential index improvements for your DB instance. For more information, see [Analyzing Your Database Workload on an Amazon RDS DB Instance with SQL Server Tuning Advisor \(p. 567\)](#).

PostgreSQL Query Tuning

Go to [Using EXPLAIN](#) in the PostgreSQL documentation to learn how to analyze a query plan. You can use this information to modify a query or underlying tables in order to improve query performance. You can also go to [Controlling the Planner with Explicit JOIN Clauses](#) to get tips about how to specify joins in your query for the best performance.

MariaDB Query Tuning

Go to [Query Optimizations](#) in the MariaDB documentation for more information on writing queries for better performance.

Best Practices for Working with MySQL Storage Engines

On a MySQL DB instance, observe the following table creation limits:

- You're limited to 10,000 tables if you are either using Provisioned IOPS storage, or using General Purpose storage and the DB instance is 200 GiB or larger in size.
- You're limited to 1000 tables if you are either using standard storage, or using General Purpose storage and the DB instance is less than 200 GiB in size.

We recommend these limits because having large numbers of tables significantly increases database recovery time after a failover or database crash. If you need to create more tables than recommended,

set the `innodb_file_per_table` parameter to 0. For more information, see [Working with InnoDB Tablespaces to Improve Crash Recovery Times \(p. 686\)](#) and [Working with DB Parameter Groups \(p. 169\)](#).

For MySQL DB instances that use version 5.7 or later, you can exceed these table creation limits due to improvements in InnoDB crash recovery. However, we still recommend that you take caution due to the potential performance impact of creating very large numbers of tables.

On a MySQL DB instance, avoid tables in your database growing too large. Provisioned storage limits restrict the maximum size of a MySQL table file to 16 TB. Instead, partition your large tables so that file sizes are well under the 16 TB limit. This approach can also improve performance and recovery time. For more information, see [MySQL File Size Limits \(p. 690\)](#).

The Point-In-Time Restore and snapshot restore features of Amazon RDS for MySQL require a crash-recoverable storage engine and are supported for the InnoDB storage engine only. Although MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for crash recovery and data durability. For example, the MyISAM storage engine does not support reliable crash recovery and might prevent a Point-In-Time Restore or snapshot restore from working as intended. This might result in lost or corrupt data when MySQL is restarted after a crash.

InnoDB is the recommended and supported storage engine for MySQL DB instances on Amazon RDS. InnoDB instances can also be migrated to Aurora, while MyISAM instances can't be migrated. However, MyISAM performs better than InnoDB if you require intense, full-text search capability. If you still choose to use MyISAM with Amazon RDS, following the steps outlined in [Automated Backups with Unsupported MySQL Storage Engines \(p. 214\)](#) can be helpful in certain scenarios for snapshot restore functionality.

If you want to convert existing MyISAM tables to InnoDB tables, you can use the process outlined in the [MySQL documentation](#). MyISAM and InnoDB have different strengths and weaknesses, so you should fully evaluate the impact of making this switch on your applications before doing so.

In addition, Federated Storage Engine is currently not supported by Amazon RDS for MySQL.

Best Practices for Working with MariaDB Storage Engines

The point-in-time restore and snapshot restore features of Amazon RDS for MariaDB require a crash-recoverable storage engine. Although MariaDB supports multiple storage engines with varying capabilities, not all of them are optimized for crash recovery and data durability. For example, although Aria is a crash-safe replacement for MyISAM, it might still prevent a point-in-time restore or snapshot restore from working as intended. This might result in lost or corrupt data when MariaDB is restarted after a crash. InnoDB (for version 10.2 and higher) and XtraDB (for version 10.0 and 10.1) are the recommended and supported storage engines for MariaDB DB instances on Amazon RDS. If you still choose to use Aria with Amazon RDS, following the steps outlined in [Automated Backups with Unsupported MariaDB Storage Engines \(p. 215\)](#) can be helpful in certain scenarios for snapshot restore functionality.

Best Practices for Working with Oracle

For information about best practices for working with Amazon RDS for Oracle, see [Best Practices for Running Oracle Database on Amazon Web Services](#) and the video [Running Oracle Databases on Amazon RDS](#).

Best Practices for Working with PostgreSQL

Two important areas where you can improve performance with PostgreSQL on Amazon RDS are when loading data into a DB instance and when using the PostgreSQL autovacuum feature. The following sections cover some of the practices we recommend for these areas.

Loading Data into a PostgreSQL DB Instance

When loading data into an Amazon RDS PostgreSQL DB instance, you should modify your DB instance settings and your DB parameter group values to allow for the most efficient importing of data into your DB instance.

Modify your DB instance settings to the following:

- Disable DB instance backups (set backup_retention to 0)
- Disable Multi-AZ

Modify your DB parameter group to include the following settings. You should test the parameter settings to find the most efficient settings for your DB instance:

- Increase the value of the `maintenance_work_mem` parameter. For more information about PostgreSQL resource consumption parameters, see the [PostgreSQL documentation](#).
- Increase the value of the `checkpoint_segments` and `checkpoint_timeout` parameters to reduce the number of writes to the wal log.
- Disable the `synchronous_commit` parameter (do not turn off FSYNC).
- Disable the PostgreSQL autovacuum parameter.
- Make sure none of the tables you are importing are unlogged. Data stored in unlogged tables can be lost during a failover. For more information see, [CREATE TABLE UNLOGGED](#)

Use the `pg_dump -Fc` (compressed) or `pg_restore -j` (parallel) commands with these settings.

Working with the `fsync` and `full_page_writes` database parameters

In PostgreSQL 9.4.1 on Amazon RDS, the `fsync` and `full_page_writes` database parameters are not modifiable. Disabling the `fsync` and `full_page_writes` database parameters can lead to data corruption, so we have enabled them for you. We recommend that customers with other 9.3 DB engine versions of PostgreSQL not disable the `fsync` and `full_page_writes` parameters.

Working with the PostgreSQL Autovacuum Feature

The autovacuum feature for PostgreSQL databases is a feature that we strongly recommend you use to maintain the health of your PostgreSQL DB instance. Autovacuum automates the execution of the `VACUUM` and `ANALYZE` command; using autovacuum is required by PostgreSQL, not imposed by Amazon RDS, and its use is critical to good performance. The feature is enabled by default for all new Amazon RDS PostgreSQL DB instances, and the related configuration parameters are appropriately set by default.

Your database administrator needs to know and understand this maintenance operation. For the PostgreSQL documentation on autovacuum, see <http://www.postgresql.org/docs/current/static/routine-vacuuming.html#AUTOVACUUM>.

Autovacuum is not a “resource free” operation, but it works in the background and yields to user operations as much as possible. When enabled, autovacuum checks for tables that have had a large number of updated or deleted tuples. It also protects against loss of very old data due to [transaction ID wraparound](#).

Autovacuum should not be thought of as a high-overhead operation that can be reduced to gain better performance. On the contrary, tables that have a high velocity of updates and deletes will quickly deteriorate over time if autovacuum is not run.

Important

Not running autovacuum can result in an eventual required outage to perform a much more intrusive vacuum operation. When an Amazon RDS PostgreSQL DB instance becomes unavailable because of an over conservative use of autovacuum, the PostgreSQL database will shut down to protect itself. At that point, Amazon RDS must perform a single-user-mode full vacuum directly on the DB instance , which can result in a multi-hour outage. Thus, we strongly recommend that you do not turn off autovacuum, which is enabled by default.

The autovacuum parameters determine when and how hard autovacuum works. The `autovacuum_vacuum_threshold` and `autovacuum_vacuum_scale_factor` parameters determine when autovacuum is run. The `autovacuum_max_workers`, `autovacuum_nap_time`, `autovacuum_cost_limit`, and `autovacuum_cost_delay` parameters determine how hard autovacuum works. For more information about autovacuum, when it runs, and what parameters are required, see the [PostgreSQL documentation](#).

The following query shows the number of "dead" tuples in a table named `table1` :

```
PROMPT> select relname, n_dead_tup, last_vacuum, last_autovacuum from pg_catalog.pg_stat_all_tables where n_dead_tup > 0 and relname = 'table1' order by n_dead_tup desc;
```

The results of the query will resemble the following:

relname	n_dead_tup	last_vacuum	last_autovacuum
tasks	81430522		

(1 row)

Best Practices for Working with SQL Server

Best practices for a Multi-AZ deployment with a SQL Server DB instance include the following:

- Use Amazon RDS DB events to monitor failovers. For example, you can be notified by text message or email when a DB instance fails over. For more information about Amazon RDS events, see [Using Amazon RDS Event Notification \(p. 287\)](#).
- If your application caches DNS values, set time to live (TTL) to less than 30 seconds. Setting TTL as so is a good practice in case there is a failover, where the IP address might change and the cached value might no longer be in service.
- We recommend that you *do not* enable the following modes because they turn off transaction logging, which is required for Multi-AZ:
 - Simple recover mode
 - Offline mode
 - Read-only mode
- Test to determine how long it takes for your DB instance to failover. Failover time can vary due to the type of database, the instance class, and the storage type you use. You should also test your application's ability to continue working if a failover occurs.

- To shorten failover time, you should do the following:
 - Ensure that you have sufficient Provisioned IOPS allocated for your workload. Inadequate I/O can lengthen failover times. Database recovery requires I/O.
 - Use smaller transactions. Database recovery relies on transactions, so if you can break up large transactions into multiple smaller transactions, your failover time should be shorter.
- Take into consideration that during a failover, there will be elevated latencies. As part of the failover process, Amazon RDS automatically replicates your data to a new standby instance. This replication means that new data is being committed to two different DB instances, so there might be some latency until the standby DB instance has caught up to the new primary DB instance.
- Deploy your applications in all Availability Zones. If an Availability Zone does go down, your applications in the other Availability Zones will still be available.

When working with a Multi-AZ deployment of SQL Server, remember that Amazon RDS creates replicas for all SQL Server databases on your instance. If you don't want specific databases to have secondary replicas, set up a separate DB instance that doesn't use Multi-AZ for those databases.

Working with DB Parameter Groups

We recommend that you try out DB parameter group changes on a test DB instance before applying parameter group changes to your production DB instances. Improperly setting DB engine parameters in a DB parameter group can have unintended adverse effects, including degraded performance and system instability. Always exercise caution when modifying DB engine parameters and back up your DB instance before modifying a DB parameter group.

For information about backing up your DB instance, see [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#).

Amazon RDS Best Practices Presentation Video

The 2016 AWS Summit conference in Chicago included a presentation on best practices for creating and configuring a secure, highly available database instance using Amazon RDS. A video of the presentation is available [here](#).

Amazon RDS DB Instances

A *DB instance* is an isolated database environment running in the cloud. It is the basic building block of Amazon RDS. A DB instance can contain multiple user-created databases, and can be accessed using the same client tools and applications you might use to access a standalone database instance. DB instances are simple to create and modify with the Amazon AWS command line tools, Amazon RDS API actions, or the AWS Management Console.

Note

Amazon RDS supports access to databases using any standard SQL client application. Amazon RDS does not allow direct host access.

You can have up to 40 Amazon RDS DB instances. Of these 40, up to 10 can be Oracle or SQL Server DB instances under the "License Included" model. All 40 DB instances can be used for MySQL, MariaDB, or PostgreSQL. You can also have 40 DB instances for SQL Server or Oracle under the "BYOL" licensing model. If your application requires more DB instances, you can request additional DB instances using the form at <https://console.aws.amazon.com/support/home#/case/create?issueType=service-limit-increase&limitType=service-code-rds-instances>.

Each DB instance has a DB instance identifier. This customer-supplied name uniquely identifies the DB instance when interacting with the Amazon RDS API and AWS CLI commands. The DB instance identifier must be unique for that customer in an AWS Region.

Each DB instance supports a database engine. Amazon RDS currently supports MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, and Amazon Aurora database engines.

When creating a DB instance, some database engines require that a database name be specified. A DB instance can host multiple databases, or a single Oracle database with multiple schemas. The database name value depends on the database engine:

- For the MySQL and MariaDB database engines, the database name is the name of a database hosted in your DB instance. Databases hosted by the same DB instance must have a unique name within that instance.
- For the Oracle database engine, database name is used to set the value of ORACLE_SID, which must be supplied when connecting to the Oracle RDS instance.
- For the Microsoft SQL Server database engine, database name is not a supported parameter.
- For the PostgreSQL database engine, the database name is the name of a database hosted in your DB instance. A database name is not required when creating a DB instance. Databases hosted by the same DB instance must have a unique name within that instance.

Amazon RDS creates a master user account for your DB instance as part of the creation process. This master user has permissions to create databases and to perform create, delete, select, update, and insert operations on tables the master user creates. You must set the master user password when you create a DB instance, but you can change it at any time using the Amazon AWS command line tools, Amazon RDS API actions, or the AWS Management Console. You can also change the master user password and manage users using standard SQL commands.

Note

This guide covers non-Aurora Amazon RDS database engines. For information about using Amazon Aurora, see the [Amazon Aurora User Guide](#).

Topics

- [DB Instance Class \(p. 82\)](#)
- [DB Instance Status \(p. 98\)](#)

- [Regions and Availability Zones \(p. 101\)](#)
- [DB instance storage \(p. 103\)](#)
- [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#)
- [Amazon RDS DB Instance Lifecycle \(p. 112\)](#)
- [Tagging Amazon RDS Resources \(p. 138\)](#)
- [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#)
- [Working with Option Groups \(p. 156\)](#)
- [Working with DB Parameter Groups \(p. 169\)](#)
- [Working with Amazon Resource Names \(ARNs\) in Amazon RDS \(p. 181\)](#)
- [Working with Storage \(p. 188\)](#)
- [DB Instance Billing for Amazon RDS \(p. 193\)](#)

DB Instance Class

The DB instance class determines the computation and memory capacity of an Amazon RDS DB instance. The DB instance class you need depends on your processing power and memory requirements.

For more information about instance class pricing, see [Amazon RDS Pricing](#).

DB Instance Class Types

Amazon RDS supports three types of instance classes: Standard, Memory Optimized, and Burstable Performance. For more information about Amazon EC2 instance types, see [Instance Type](#) in the Amazon EC2 documentation.

The following are the Standard DB instance classes available:

- **db.m5** – Latest-generation general-purpose instance classes that provide a balance of compute, memory, and network resources, and are a good choice for many applications. The db.m5 instance classes provide more computing capacity than the previous db.m4 instance classes.
- **db.m4** – Current-generation general-purpose instance classes that provide more computing capacity than the previous db.m3 instance classes.
- **db.m3** – Previous-generation general-purpose instance classes that provide more computing capacity than the previous db.m1 instance classes.
- **db.m1** – Previous-generation general-purpose instance classes.

The following are the Memory Optimized DB instance classes available:

- **db.x1e** – Latest-generation instance classes optimized for memory-intensive applications. These offer one of the lowest price per GiB of RAM among the DB instance classes and up to 3,904 GiB of DRAM-based instance memory. The db.x1e instance classes are available only in the following regions: US East (N. Virginia), US West (Oregon), EU (Ireland), Asia Pacific (Tokyo), and Asia Pacific (Sydney).
- **db.x1** – Current-generation instance classes optimized for memory-intensive applications. These offer one of the lowest price per GiB of RAM among the DB instance classes and up to 1,952 GiB of DRAM-based instance memory.
- **db.r5** – Latest-generation instance classes optimized for memory-intensive applications. These offer improved networking and Amazon Elastic Block Store (Amazon EBS) performance. They are powered by the AWS Nitro System, a combination of dedicated hardware and lightweight hypervisor.
- **db.r4** – Current-generation instance classes optimized for memory-intensive applications. These offer improved networking and Amazon EBS performance.
- **db.r3** – Previous-generation instance classes that provide memory optimization and more computing capacity than the db.m2 instance classes. The db.r3 instances classes are not available in the EU (Paris) region and the South America (São Paulo) region.
- **db.m2** – Previous-generation memory-optimized instance classes.

The following are the Burstable Performance DB instance classes available:

- **db.t2** – Instance classes that provide a baseline performance level, with the ability to burst to full CPU usage.

Specifications for All Available DB Instance Classes

The following table provides details of the Amazon RDS DB instance classes. The table columns are explained after the table.

Instance Class	vCPU	ECU ²	Memory (GiB)	VPC Only ⁴	EBS Optimized	Max. Bandwidth (Mbps)	Network Performance	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL ¹⁰
db.m5 – Latest Generation Standard Instance Classes												
db.m5.24xlarge	96	345	384	Yes	Yes	14,000	25 Gbps	Yes	No	Yes	Yes ⁹	Yes ¹⁰
db.m5.12xlarge	48	173	192	Yes	Yes	7,000	10 Gbps	Yes	No	Yes	Yes ⁹	Yes ¹⁰
db.m5.4xlarge	16	61	64	Yes	Yes	3,500	Up to 10 Gigabit	Yes	No	Yes	Yes ⁹	Yes ¹⁰
db.m5.2xlarge	8	31	32	Yes	Yes	3,500	Up to 10 Gigabit	Yes	No	Yes	Yes ⁹	Yes ¹⁰
db.m5.xlarge	4	15	16	Yes	Yes	3,500	Up to 10 Gigabit	Yes	No	Yes	Yes ⁹	Yes ¹⁰
db.m5.large	2	10	8	Yes	Yes	3,500	Up to 10 Gigabit	Yes	No	Yes	Yes ⁹	Yes ¹⁰
db.m4 – Current Generation Standard Instance Classes												
db.m4.16xlarge	64	188	256	Yes	Yes	10,000	25 Gbps	Yes	Yes ⁸	MySQL 8.0, 5.7, 5.6	Yes ⁹	Yes
db.m4.10xlarge	40	124.5	160	Yes	Yes	4,000	10 Gbps	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.m4.4xlarge	16	53.5	64	Yes	Yes	2,000	High	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.m4.2xlarge	8	25.5	32	Yes	Yes	1,000	High	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.m4.xlarge	4	13	16	Yes	Yes	750	High	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.m4.large	2	6.5	8	Yes	Yes	450	Moderate	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.m3 – Previous Generation Standard Instance Classes												
db.m3.2xlarge	8	26	30	No	Yes	1,000	High	No	Yes ⁸	Yes	Yes ⁹	Yes
db.m3.xlarge	4	13	15	No	Yes	500	High	No	Yes ⁸	Yes	Yes ⁹	Yes
db.m3.large	2	6.5	7.5	No	No	—	Moderate	No	Yes ⁸	Yes	Yes ⁹	Yes
db.m3.medium	1	3	3.75	No	No	—	Moderate	No	Yes ⁸	Yes	Yes ⁹	Yes
db.m1 – Previous Generation Standard Instance Classes												
db.m1.xlarge	4	4	15	No	Yes	450	High	No	Yes ⁸	MySQL 5.6, 5.5	Deprecated	PostgreSQL 9.4, 9.3

Instance Class	vCPU	ECU ²	Memory (GiB)	VPC Only ⁴	EBS Optimized	Max. Bandwidth (Mbps)	Network Performance	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL ¹⁰
db.m1.large	2	2	7.5	No	Yes	450	Moderate	No	Yes ⁸	MySQL 5.6, 5.5	Deprecated ⁹	PostgreSQL 9.4, 9.3
db.m1.medium	1	1	3.75	No	No	—	Moderate	No	Yes ⁸	MySQL 5.6, 5.5	Deprecated ⁹	PostgreSQL 9.4, 9.3
db.m1.small	1	1	1.7	No	No	—	Very Low	No	Yes ⁸	MySQL 5.6, 5.5	Deprecated ⁹	PostgreSQL 9.4, 9.3
db.x1e – Latest Generation Memory Optimized Instance Classes												
db.x1e.32xlarge	128	340	3,904	Yes	Yes	14,000	25 Gbps	No	No	No	Yes ⁹	No
db.x1e.16xlarge	64	179	1,952	Yes	Yes	7,000	10 Gbps	No	No	No	Yes ⁹	No
db.x1e.8xlarge	32	91	976	Yes	Yes	3,500	Up to 10 Gbps	No	No	No	Yes ⁹	No
db.x1e.4xlarge	16	47	488	Yes	Yes	1,750	Up to 10 Gbps	No	No	No	Yes ⁹	No
db.x1e.2xlarge	8	23	244	Yes	Yes	1,000	Up to 10 Gbps	No	No	No	Yes ⁹	No
db.x1e.xlarge	4	12	122	Yes	Yes	500	Up to 10 Gbps	No	No	No	Yes ⁹	No
db.x1 – Current Generation Memory Optimized Instance Classes												
db.x1.32xlarge	128	349	1,952	Yes	Yes	14,000	25 Gbps	No	No	No	Yes ⁹	No
db.x1.16xlarge	64	349	976	Yes	Yes	7,000	10 Gbps	No	No	No	Yes ⁹	No
db.r5 – Latest Generation Memory Optimized Instance Classes												
db.r5.24xlarge	96	347	768	Yes	Yes	14,000	25 Gbps	No	No	No	Yes ⁹	Yes ¹⁰
db.r5.12xlarge	48	173	384	Yes	Yes	7,000	10 Gbps	No	No	No	Yes ⁹	Yes ¹⁰
db.r5.4xlarge	16	71	128	Yes	Yes	3,500	Up to 10 Gbps	No	No	No	Yes ⁹	Yes ¹⁰

Instance Class	vCPU	ECU ²	Memory (GiB)	VPC Only ⁴	EBS Optimized	Max. Bandwidth (Mbps)	Network Performance	MariaDB	Microsoft SQL Server	MySQL ⁵	Oracle ⁶	PostgreSQL ¹⁰
db.r5.2xlarge	8	38	64	Yes	Yes	Up to 3,500	Up to 10 Gbps	No	No	No	Yes ⁹	Yes ¹⁰
db.r5.xlarge	4	19	32	Yes	Yes	Up to 3,500	Up to 10 Gbps	No	No	No	Yes ⁹	Yes ¹⁰
db.r5.large	2	10	16	Yes	Yes	Up to 3,500	Up to 10 Gbps	No	No	No	Yes ⁹	Yes ¹⁰
db.r4 – Current Generation Memory Optimized Instance Classes												
db.r4.16xlarge	64	195	488	Yes	Yes	14,000	25 Gbps	Yes	Yes ⁸	MySQL Yes ⁸ , 8.0, 5.7, 5.6	Yes ⁹	PostgreSQL 9.6, 9.5, 9.4
db.r4.8xlarge	32	99	244	Yes	Yes	7,000	10 Gbps	Yes	Yes ⁸	MySQL Yes ⁸ , 8.0, 5.7, 5.6	Yes ⁹	PostgreSQL 9.6, 9.5, 9.4
db.r4.4xlarge	16	53	122	Yes	Yes	3,500	Up to 10 Gbps	Yes	Yes ⁸	MySQL Yes ⁸ , 8.0, 5.7, 5.6	Yes ⁹	PostgreSQL 9.6, 9.5, 9.4
db.r4.2xlarge	8	27	61	Yes	Yes	1,750	Up to 10 Gbps	Yes	Yes ⁸	MySQL Yes ⁸ , 8.0, 5.7, 5.6	Yes ⁹	PostgreSQL 9.6, 9.5, 9.4
db.r4.xlarge	4	13.5	30.5	Yes	Yes	875	Up to 10 Gbps	Yes	Yes ⁸	MySQL Yes ⁸ , 8.0, 5.7, 5.6	Yes ⁹	PostgreSQL 9.6, 9.5, 9.4
db.r4.large	2	7	15.25	Yes	Yes	437	Up to 10 Gbps	Yes	Yes ⁸	MySQL Yes ⁸ , 8.0, 5.7, 5.6	Yes ⁹	PostgreSQL 9.6, 9.5, 9.4
db.r3 – Previous Generation Memory Optimized Instance Classes												
db.r3.8xlarge	32	104	244	No	No	—	10 Gbps	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.r3.4xlarge	16	52	122	No	Yes	2,000	High	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.r3.2xlarge	8	26	61	No	Yes	1,000	High	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.r3.xlarge	4	13	30.5	No	Yes	500	Moderate	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.r3.large	2	6.5	15.25	No	No	—	Moderate	Yes	Yes ⁸	Yes	Yes ⁹	Yes

Instance Class	vCPU	ECU ²	Memory (GiB)	VPC Only ¹	EBS Optimized	Max. Bandwidth (Mbps)	Network Performance	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL ¹⁰
db.m2 – Previous Generation Memory Optimized Instance Classes												
db.m2.4xlarge	8	26	68.4	No	Yes	1,000	High	No	Yes ⁸	MySQL 5.6, 5.5	Deprec.	PostgreSQL 9.4, 9.3
db.m2.2xlarge	4	13	34.2	No	Yes	500	Moderate	No	Yes ⁸	MySQL 5.6, 5.5	Deprec.	PostgreSQL 9.4, 9.3
db.m2.xlarge	2	6.5	17.1	No	No	—	Moderate	No	Yes ⁸	MySQL 5.6, 5.5	Deprec.	PostgreSQL 9.4, 9.3
db.t2 – Current Generation Burstable Performance Instance Classes												
db.t2.2xlarge	8	Variable ⁶	162	Yes	No	—	Moderate	Yes	No	MySQL Yes ⁹ 8.0, 5.7, 5.6	Deprec.	PostgreSQL 9.6, 9.5, 9.4
db.t2.xlarge	4	Variable ⁶	66	Yes	No	—	Moderate	Yes	No	MySQL Yes ⁹ 8.0, 5.7, 5.6	Deprec.	PostgreSQL 9.6, 9.5, 9.4
db.t2.large	2	Variable ⁶	33	Yes	No	—	Moderate	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.t2.medium	2	Variable ⁶	16	Yes	No	—	Moderate	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.t2.small	1	Variable ⁶	8	Yes	No	—	Low	Yes	Yes ⁸	Yes	Yes ⁹	Yes
db.t2.micro	1	Variable ⁶	4	Yes	No	—	Low	Yes	Yes ⁸	Yes	Yes ⁹	Yes

1. **vCPU** – The number of virtual central processing units (CPUs). A virtual CPU is a unit of capacity that you can use to compare DB instance classes. Instead of purchasing or leasing a particular processor to use for several months or years, you are renting capacity by the hour. Our goal is to make a consistent and specific amount of CPU capacity available, within the limits of the actual underlying hardware.
2. **ECU** – The relative measure of the integer processing power of an Amazon EC2 instance. To make it easy for developers to compare CPU capacity between different instance classes, we have defined an Amazon EC2 Compute Unit. The amount of CPU that is allocated to a particular instance is expressed in terms of these EC2 Compute Units. One ECU currently provides CPU capacity equivalent to a 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor.
3. **Memory (GiB)** – The RAM memory, in gibibytes, allocated to the DB instance. There is often a consistent ratio between memory and vCPU. For example, the db.m1 instance class has the same memory to vCPU ratio as the db.m3 instance class, but for most use cases the db.m3 instance class provides better, more consistent performance, than the db.m1 instance class.
4. **VPC Only** – The instance class is supported only for DB instances that are in an Amazon Virtual Private Cloud (VPC). If your current DB instance is not in a VPC, and you want to use an instance class that requires a VPC, first move your DB instance into a VPC. For more information, see [Moving a DB Instance Not in a VPC into a VPC \(p. 426\)](#).
5. **EBS-Optimized** – The DB instance uses an optimized configuration stack and provides additional, dedicated capacity for I/O. This optimization provides the best performance by minimizing contention

between I/O and other traffic from your instance. For more information about Amazon EBS-optimized instances, see [Amazon EBS-Optimized Instances](#) in the Amazon EC2 documentation.

6. **Max. Bandwidth (Mbps)** – The maximum bandwidth in megabits per second. Divide by 8 to get the expected throughput in megabytes per second.

Important

General Purpose SSD (gp2) volumes for Amazon RDS DB instances have a throughput limit of 250 MiB/s in most cases. However, the throughput limit can vary depending on volume size. For more information, see [Amazon EBS Volume Types](#) in the Amazon EC2 documentation. For information on estimating bandwidth for gp2 storage, see [General Purpose SSD Storage \(p. 103\)](#).

7. **Network Performance** – The network speed relative to other DB instance classes.
8. **Microsoft SQL Server** – Instance class support varies according to the version and edition of SQL Server. For instance class support by version and edition, see [DB Instance Class Support for Microsoft SQL Server \(p. 491\)](#).
9. **Oracle** – Instance class support varies according to the version and edition of Oracle. For instance class support by version and edition, see [DB Instance Class Support for Oracle \(p. 720\)](#).
10. **PostgreSQL** – PostgreSQL versions 9.6.9 (and above) and 10.4 (and above) are supported.

Changing Your DB Instance Class

You can change the CPU and memory available to a DB instance by changing its DB instance class. To change the DB instance class, modify your DB instance by following the instructions for your specific database engine.

- [Modifying a DB Instance Running the MariaDB Database Engine \(p. 456\)](#)
- [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#)
- [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#)
- [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#)
- [Modifying a DB Instance Running the PostgreSQL Database Engine \(p. 979\)](#)

MySQL DB instances created after April 23, 2014, can change to the db.r3 instance class by modifying the DB instance just as with any other modification. MySQL DB instances running MySQL versions 5.5 and created before April 23, 2014, must first upgrade to MySQL version 5.6. For more information, see [Upgrading the MySQL DB Engine \(p. 618\)](#).

Some instance classes require that your DB instance is in a VPC. If your current DB instance is not in a VPC, and you want to use an instance class that requires a VPC, first move your DB instance into a VPC. For more information, see [Moving a DB Instance Not in a VPC into a VPC \(p. 426\)](#).

Configuring the Processor for a DB Instance Class

Amazon RDS DB instance classes support Intel Hyper-Threading Technology, which enables multiple threads to run concurrently on a single Intel Xeon CPU core. Each thread is represented as a virtual CPU (vCPU) on the DB instance. A DB instance has a default number of CPU cores, which varies according to DB instance type. For example, a db.m4.xlarge DB instance type has two CPU cores and two threads per core by default—four vCPUs in total.

Note

Each vCPU is a hyperthread of an Intel Xeon CPU core.

In most cases, you can find a DB instance class that has a combination of memory and number of vCPUs to suit your workloads. However, you can also specify the following processor features to optimize your DB instance for specific workloads or business needs:

- **Number of CPU cores** – You can customize the number of CPU cores for the DB instance. You might do this to potentially optimize the licensing costs of your software with a DB instance that has sufficient amounts of RAM for memory-intensive workloads but fewer CPU cores.
- **Threads per core** – You can disable Intel Hyper-Threading Technology by specifying a single thread per CPU core. You might do this for certain workloads, such as high-performance computing (HPC) workloads.

You can control the number of CPU cores and threads for each core separately. You can set one or both in a request. After a setting is associated with a DB instance, the setting persists until you change it.

The processor settings for a DB instance are associated with snapshots of the DB instance. When a snapshot is restored, its restored DB instance uses the processor feature settings used when the snapshot was taken.

If you modify the DB instance class for a DB instance with nondefault processor settings, you must either specify default processor settings or explicitly specify processor settings when you modify the DB instance. This requirement ensures that you are aware of the third-party licensing costs that might be incurred when you modify the DB instance.

There is no additional or reduced charge for specifying processor features on an Amazon RDS DB instance. You're charged the same as for DB instances that are launched with default CPU configurations.

You can configure the number of CPU cores and threads per core for the DB instance class when you perform the following operations:

- Creating a DB instance
- Modifying a DB instance
- Restoring a DB instance from a snapshot
- Restoring a DB instance to a point in time

Note

When you modify a DB instance to configure the number of CPU cores or threads per core, there is a brief DB instance outage.

CPU Cores and Threads Per CPU Core Per DB Instance Class

In following table, you can find the DB instance classes that support setting a number of CPU cores and CPU threads per core. You can also find the default value and the valid values for the number of CPU cores and CPU threads per core for each DB instance class.

DB Instance Class	Default vCPUs	Default CPU Cores	Default Threads per Core	Valid Number of CPU Cores	Valid Number of Threads per Core
db.m5.large	2	1	2	1	1, 2
db.m5.xlarge	4	2	2	2	1, 2
db.m5.2xlarge	8	4	2	2, 4	1, 2
db.m5.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.m5.12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2

DB Instance Class	Default vCPUs	Default CPU Cores	Default Threads per Core	Valid Number of CPU Cores	Valid Number of Threads per Core
db.m5.24xlarge	96	48	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2
db.m4.10xlarge	40	20	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20	1, 2
db.m4.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.r3.large	2	1	2	1	1, 2
db.r3.xlarge	4	2	2	1, 2	1, 2
db.r3.2xlarge	8	4	2	1, 2, 3, 4	1, 2
db.r3.4xlarge	16	8	2	1, 2, 3, 4, 5, 6, 7, 8	1, 2
db.r3.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.r5.large	2	1	2	1	1
db.r5.xlarge	4	2	2	2	1, 2
db.r5.2xlarge	8	4	2	2, 4	1, 2
db.r5.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.r5.12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2
db.r5.24xlarge	96	48	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2
db.r4.large	2	1	2	1	1, 2
db.r4.xlarge	4	2	2	1, 2	1, 2
db.r4.2xlarge	8	4	2	1, 2, 3, 4	1, 2
db.r4.4xlarge	16	8	2	1, 2, 3, 4, 5, 6, 7, 8	1, 2

DB Instance Class	Default vCPUs	Default CPU Cores	Default Threads per Core	Valid Number of CPU Cores	Valid Number of Threads per Core
db.r4.8xlarge	32	16	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16	1, 2
db.r4.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.x1.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.x1.32xlarge	128	64	2	4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64	1, 2
db.x1e.xlarge	4	2	2	1, 2	1, 2
db.x1e.2xlarge	8	4	2	1, 2, 3, 4	1, 2
db.x1e.4xlarge	16	8	2	1, 2, 3, 4, 5, 6, 7, 8	1, 2
db.x1e.8xlarge	32	16	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16	1, 2
db.x1e.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.x1e.32xlarge	128	64	2	4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64	1, 2

Note

Currently, you can configure the number of CPU cores and threads per core only for Oracle DB instances. For information about the DB instance classes supported by different Oracle database editions, see [DB Instance Class Support for Oracle \(p. 720\)](#).

For Oracle DB instances, configuring the number of CPU cores and threads per core is only supported with the Bring Your Own License (BYOL) licensing option. For more information about Oracle licensing options, see [Oracle Licensing \(p. 719\)](#).

Setting the CPU Cores and Threads per CPU Core for a DB Instance Class

You can set the CPU cores and the threads per CPU core for a DB instance class using the AWS Management Console, the AWS CLI, or the RDS API.

AWS Management Console

When you are creating, modifying, or restoring a DB instance, you set the DB instance class in the AWS Management Console. The **Instance specifications** section shows options for the processor. The following image shows the processor features options.

Instance specifications

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#)

DB engine
Oracle Database Enterprise Edition

License model [Info](#)

DB engine version [Info](#)

DB instance class [Info](#)

Multi-AZ deployment [Info](#)

Create replica in different zone
Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

No

Storage type [Info](#)

Allocated storage
 GiB
(Minimum: 100 GiB, Maximum: 16384 GiB)

Provisioned IOPS [Info](#)

▼ Additional configuration

Processor features

Override default values
You can change the number of CPU cores and threads per core on the DB instance class.

Core count [Info](#)

Threads per core [Info](#)

Set the following options to the appropriate values for your DB instance class under **Processor features**:

- **Core count** – Set the number of CPU cores using this option. The value must be equal to or less than the maximum number of CPU cores for the DB instance class.
- **Threads per core** – Specify **2** to enable multiple threads per core, or specify **1** to disable multiple threads per core.

When you modify or restore a DB instance, you can also set the CPU cores and the threads per CPU core to the default settings for the selected DB instance class.

When you view the details for a DB instance in the console, you can view the processor information for its DB instance class on the **Configuration** tab. The following image shows a DB instance class with one CPU core and multiple threads per core enabled.

Instance and IOPS	
Instance Class	db.r4.large
Core count	1
Threads per core	2
vCPU enabled	2
Storage Type	Provisioned IOPS (SSD)
IOPS	1000
Storage	100 GiB

For Oracle DB instances, the processor information only appears for Bring Your Own License (BYOL) DB instances.

CLI

You can set the processor features for a DB instance when you run one of the following AWS CLI commands:

- [create-db-instance](#)

- [modify-db-instance](#)
- [restore-db-instance-from-db-snapshot](#)
- [restore-db-instance-from-s3](#)
- [restore-db-instance-to-point-in-time](#)

To configure the processor of a DB instance class for a DB instance by using the AWS CLI, include the `--processor-features` option in the command. Specify the number of CPU cores with the `coreCount` feature name, and specify whether multiple threads per core are enabled with the `threadsPerCore` feature name.

The option has the following syntax.

```
--processor-features "Name=coreCount,Value=<value>" "Name=threadsPerCore,Value=<value>"
```

Example Setting the Number of CPU Cores for a DB Instance

The following example modifies `mydbinstance` by setting the number of CPU cores to 4. The changes are applied immediately by using `--apply-immediately`. If you want to apply the changes during the next scheduled maintenance window, omit the `--apply-immediately` option.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \  
  --processor-features "Name=coreCount,Value=4" \  
  --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^  
  --processor-features "Name=coreCount,Value=4" ^  
  --apply-immediately
```

Example Setting the Number of CPU Cores and Disabling Multiple Threads for a DB Instance

The following example modifies `mydbinstance` by setting the number of CPU cores to 4 and disabling multiple threads per core. The changes are applied immediately by using `--apply-immediately`. If you want to apply the changes during the next scheduled maintenance window, omit the `--apply-immediately` option.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \  
  --processor-features "Name=coreCount,Value=4" "Name=threadsPerCore,Value=1" \  
  --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^  
  --processor-features "Name=coreCount,Value=4" "Name=threadsPerCore,Value=1" ^  
  --apply-immediately
```

Example Viewing the Valid Processor Values for a DB Instance Class

You can view the valid processor values for a particular DB instance class by running the [describe-orderable-db-instance-options](#) command and specifying the instance class for the --db-instance-class option. For example, the output for the following command shows the processor options for the db.r3.large instance class.

```
aws rds describe-orderable-db-instance-options --engine oracle-ee --db-instance-class db.r3.large
```

Following is sample output for the command in JSON format.

```
{  
    "SupportsIops": true,  
    "MaxIopsPerGib": 50.0,  
    "LicenseModel": "bring-your-own-license",  
    "DBInstanceClass": "db.r3.large",  
    "SupportsIAMDatabaseAuthentication": false,  
    "MinStorageSize": 100,  
    "AvailabilityZones": [  
        {  
            "Name": "us-west-2a"  
        },  
        {  
            "Name": "us-west-2b"  
        },  
        {  
            "Name": "us-west-2c"  
        }  
    ],  
    "EngineVersion": "12.1.0.2.v2",  
    "MaxStorageSize": 32768,  
    "MinIopsPerGib": 1.0,  
    "MaxIopsPerDbInstance": 40000,  
    "ReadReplicaCapable": false,  
    "AvailableProcessorFeatures": [  
        {  
            "Name": "coreCount",  
            "DefaultValue": "1",  
            "AllowedValues": "1"  
        },  
        {  
            "Name": "threadsPerCore",  
            "DefaultValue": "2",  
            "AllowedValues": "1,2"  
        }  
    ],  
    "SupportsEnhancedMonitoring": true,  
    "SupportsPerformanceInsights": false,  
    "MinIopsPerDbInstance": 1000,  
    "StorageType": "io1",  
    "Vpc": false,  
    "SupportsStorageEncryption": true,  
    "Engine": "oracle-ee",  
    "MultiAZCapable": true  
}
```

In addition, you can run the following commands for DB instance class processor information:

- **describe-db-instances** – Shows the processor information for the specified DB instance.
- **describe-db-snapshots** – Shows the processor information for the specified DB snapshot.
- **describe-valid-db-instance-modifications** – Shows the valid modifications to the processor for the specified DB instance.

Example Returning to Default Processor Settings for a DB Instance

The following example modifies `mydbinstance` by returning its DB instance class to the default processor values for it. The changes are applied immediately by using `--apply-immediately`. If you want to apply the changes during the next scheduled maintenance window, omit the `--apply-immediately` option.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --use-default-processor-features \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --use-default-processor-features ^
    --apply-immediately
```

Example Returning to the Default Number of CPU Cores for a DB Instance

The following example modifies `mydbinstance` by returning its DB instance class to the default number of CPU cores for it. The threads per core setting isn't changed. The changes are applied immediately by using `--apply-immediately`. If you want to apply the changes during the next scheduled maintenance window, omit the `--apply-immediately` option.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --processor-features "Name=coreCount,Value=DEFAULT" \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --processor-features "Name=coreCount,Value=DEFAULT" ^
    --apply-immediately
```

Example Returning to the Default Number of Threads Per Core for a DB Instance

The following example modifies `mydbinstance` by returning its DB instance class to the default number of threads per core for it. The number of CPU cores setting isn't changed. The changes are applied immediately by using `--apply-immediately`. If you want to apply the changes during the next scheduled maintenance window, omit the `--apply-immediately` option.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --processor-features "Name=threadsPerCore,Value=DEFAULT" \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^  
  --processor-features "Name=threadsPerCore,Value=DEFAULT" ^  
  --apply-immediately
```

API

You can set the processor features for a DB instance when you call one of the following Amazon RDS API actions:

- [CreateDBInstance](#)
- [ModifyDBInstance](#)
- [RestoreDBInstanceFromDBSnapshot](#)
- [RestoreDBInstanceFromS3](#)
- [RestoreDBInstanceToPointInTime](#)

To configure the processor features of a DB instance class for a DB instance by using the Amazon RDS API, include the `ProcessFeatures` parameter in the call.

The parameter has the following syntax.

```
ProcessFeatures "Name=coreCount,Value=<value>" "Name=threadsPerCore,Value=<value>"
```

Specify the number of CPU cores with the `coreCount` feature name, and specify whether multiple threads per core are enabled with the `threadsPerCore` feature name.

You can view the valid processor values for a particular instance class by running the [DescribeOrderableDBInstanceOptions](#) action and specifying the instance class for the `DBInstanceClass` parameter.

In addition, you can use the following actions for DB instance class processor information:

- [DescribeDBInstances](#) – Shows the processor information for the specified DB instance.
- [DescribeDBSnapshots](#) – Shows the processor information for the specified DB snapshot.
- [DescribeValidDBInstanceModifications](#) – Shows the valid modifications to the processor for the specified DB instance.

DB Instance Status

The status of a DB instance indicates the health of the DB instance. You can view the status of a DB instance by using the Amazon RDS console, the AWS CLI command [describe-db-instances](#), or the API action [DescribeDBInstances](#).

Note

Amazon RDS also uses another status called *maintenance status*, which is shown in the **Maintenance** column of the Amazon RDS console. This value indicates the status of any maintenance patches that need to be applied to a DB instance. Maintenance status is independent of DB instance status. For more information on *maintenance status*, see [Applying Updates for a DB Instance \(p. 118\)](#).

Find the possible status values for DB instances in the following table, which also shows how you are billed for each status. It shows if you will be billed for the DB instance and storage, billed only for storage, or not billed. For all DB instance statuses, you are always billed for backup usage.

DB Instance Status	Billed	Description
available	Billed	The DB instance is healthy and available.
backing-up	Billed	The DB instance is currently being backed up.
backtracking	Billed	The DB instance is currently being backtracked. This status only applies to Aurora MySQL.
configuring-enhanced-monitoring	Billed	Enhanced Monitoring is being enabled or disabled for this DB instance.
configuring-iam-database-auth	Billed	AWS Identity and Access Management (IAM) database authentication is being enabled or disabled for this DB instance.
configuring-log-exports	Billed	Publishing log files to Amazon CloudWatch Logs is being enabled or disabled for this DB instance.
converting-to-vpc	Billed	The DB instance is being converted from a DB instance that is not in an Amazon Virtual Private Cloud (Amazon VPC) to a DB instance that is in an Amazon VPC.
creating	Not billed	The DB instance is being created. The DB instance is inaccessible while it is being created.
deleting	Not billed	The DB instance is being deleted.
failed	Not billed	The DB instance has failed and Amazon RDS can't recover it. Perform a point-in-time restore to the latest restorable time of the DB instance to recover the data.
inaccessible-encryption-credentials	Not billed	The AWS KMS key used to encrypt or decrypt the DB instance can't be accessed.
incompatible-credentials	Billed	The supplied CloudHSM Classic user name or password is incorrect. Update the CloudHSM Classic credentials for the DB instance.
incompatible-network	Not billed	Amazon RDS is attempting to perform a recovery action on a DB instance but can't do so because the VPC is in a state that prevents the action from being completed. This status can occur

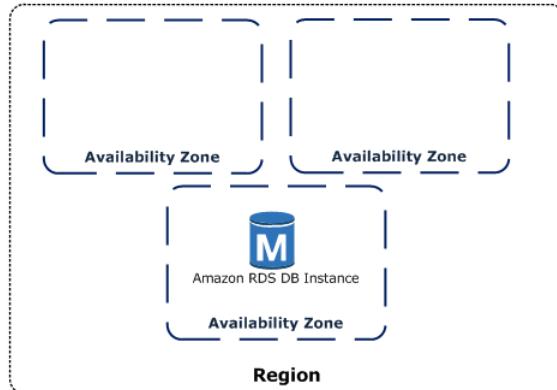
DB Instance Status	Billed	Description
		if, for example, all available IP addresses in a subnet are in use and Amazon RDS can't get an IP address for the DB instance.
incompatible-option-group	Billed	Amazon RDS attempted to apply an option group change but can't do so, and Amazon RDS can't roll back to the previous option group state. For more information, check the Recent Events list for the DB instance. This status can occur if, for example, the option group contains an option such as TDE and the DB instance doesn't contain encrypted information.
incompatible-parameters	Billed	Amazon RDS can't start the DB instance because the parameters specified in the DB instance's DB parameter group aren't compatible with the DB instance. Revert the parameter changes or make them compatible with the DB instance to regain access to your DB instance. For more information about the incompatible parameters, check the Recent Events list for the DB instance.
incompatible-restore	Not billed	Amazon RDS can't do a point-in-time restore. Common causes for this status include using temp tables, using MyISAM tables with MySQL, or using Aria tables with MariaDB.
maintenance	Billed	Amazon RDS is applying a maintenance update to the DB instance. This status is used for instance-level maintenance that RDS schedules well in advance.
modifying	Billed	The DB instance is being modified because of a customer request to modify the DB instance.
moving-to-vpc	Billed	The DB instance is being moved to a new Amazon Virtual Private Cloud (Amazon VPC).
rebooting	Billed	The DB instance is being rebooted because of a customer request or an Amazon RDS process that requires the rebooting of the DB instance.
renaming	Billed	The DB instance is being renamed because of a customer request to rename it.
resetting-master-credentials	Billed	The master credentials for the DB instance are being reset because of a customer request to reset them.
restore-error	Billed	The DB instance encountered an error attempting to restore to a point-in-time or from a snapshot.
starting	Billed for storage	The DB instance is starting.
stopped	Billed for storage	The DB instance is stopped.
stopping	Billed for storage	The DB instance is being stopped.

DB Instance Status	Billed	Description
storage-full	Billed	The DB instance has reached its storage capacity allocation. This is a critical status, and we recommend that you fix this issue immediately. To do so, scale up your storage by modifying the DB instance. To avoid this situation, set Amazon CloudWatch alarms to warn you when storage space is getting low.
storage-optimization	Billed	Your DB instance is being modified to change the storage size or type. The DB instance is fully operational. However, while the status of your DB instance is storage-optimization, you can't request any changes to the storage of your DB instance. The storage optimization process is usually short, but can sometimes take up to and even beyond 24 hours.
upgrading	Billed	The database engine version is being upgraded.

Regions and Availability Zones

Amazon cloud computing resources are hosted in multiple locations world-wide. These locations are composed of AWS Regions and Availability Zones. Each *AWS Region* is a separate geographic area. Each AWS Region has multiple, isolated locations known as *Availability Zones*. Amazon RDS provides you the ability to place resources, such as instances, and data in multiple locations. Resources aren't replicated across AWS Regions unless you do so specifically.

Amazon operates state-of-the-art, highly-available data centers. Although rare, failures can occur that affect the availability of instances that are in the same location. If you host all your instances in a single location that is affected by such a failure, none of your instances would be available.



It is important to remember that each AWS Region is completely independent. Any Amazon RDS activity you initiate (for example, creating database instances or listing available database instances) runs only in your current default AWS Region. The default AWS Region can be changed in the console, by setting the `EC2_REGION` environment variable, or it can be overridden by using the `--region` parameter with the AWS Command Line Interface. See [Configuring the AWS Command Line Interface](#), specifically, the sections on Environment Variables and Command Line Options for more information.

Amazon RDS supports a special AWS Region called AWS GovCloud (US-West) that is designed to allow US government agencies and customers to move more sensitive workloads into the cloud. AWS GovCloud (US-West) addresses the US government's specific regulatory and compliance requirements. For more information about AWS GovCloud (US-West), see [What Is AWS GovCloud \(US-West\)?](#)

To create or work with an Amazon RDS DB instance in a specific AWS Region, use the corresponding regional service endpoint.

Amazon RDS supports the endpoints listed in the following table.

Region Name	Region	Endpoint	Protocol
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS
US East (N. Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS
US West (N. California)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS

Region Name	Region	Endpoint	Protocol
Asia Pacific (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS
Asia Pacific (Osaka-Local)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS
Asia Pacific (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS
Asia Pacific (Singapore)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS
Asia Pacific (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS
Asia Pacific (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
Canada (Central)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
China (Beijing)	cn-north-1	rds.cn-north-1.amazonaws.com.cn	HTTPS
China (Ningxia)	cn-northwest-1	rds.cn-northwest-1.amazonaws.com.cn	HTTPS
EU (Frankfurt)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
EU (Ireland)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS
EU (London)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS
EU (Paris)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
EU (Stockholm)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS
South America (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (US-East)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (US)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS

If you do not explicitly specify an endpoint, the US West (Oregon) endpoint is the default.

DB instance storage

DB instances for Amazon RDS for MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server use Amazon Elastic Block Store (Amazon EBS) volumes for database and log storage. Depending on the amount of storage requested, Amazon RDS automatically stripes across multiple Amazon EBS volumes to enhance performance.

Amazon RDS Storage Types

Amazon RDS provides three storage types: General Purpose SSD (also known as gp2), Provisioned IOPS SSD (also known as io1), and magnetic. They differ in performance characteristics and price, which means that you can tailor your storage performance and cost to the needs of your database workload. You can create MySQL, MariaDB, Oracle, and PostgreSQL RDS DB instances with up to 32 TiB of storage. You can create SQL Server RDS DB instances with up to 16 TiB of storage. For this amount of storage, use the Provisioned IOPS SSD and General Purpose SSD storage types.

The following list briefly describes the three storage types:

- **General Purpose SSD** – General Purpose SSD , also called gp2, volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods of time. Baseline performance for these volumes is determined by the volume's size.

For more information about General Purpose SSD storage, including the storage size ranges, see [General Purpose SSD Storage \(p. 103\)](#).

- **Provisioned IOPS** – Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that require low I/O latency and consistent I/O throughput.

For more information about provisioned IOPS storage, including the storage size ranges, see [Provisioned IOPS SSD Storage \(p. 105\)](#).

- **Magnetic** – Amazon RDS also supports magnetic storage for backward compatibility. We recommend that you use General Purpose SSD or Provisioned IOPS for any new storage needs. The maximum amount of storage allowed for DB instances on magnetic storage is less than that of the other storage types. For more information, see [Magnetic storage \(p. 106\)](#).

Several factors can affect the performance of Amazon EBS volumes, such as instance configuration, I/O characteristics, and workload demand. For more information about getting the most out of your Provisioned IOPS volumes, see [Amazon EBS Volume Performance](#).

General Purpose SSD Storage

General Purpose SSD storage offers cost-effective storage that is acceptable for most database workloads. The following are the storage size ranges for General Purpose SSD DB instances:

- MySQL, MariaDB, Oracle, and PostgreSQL DB instances: 20 GiB–32 TiB
- SQL Server for Enterprise, Standard, Web, and Express editions: 20 GiB–16 TiB

Baseline I/O performance for General Purpose SSD storage is 3 IOPS for each GiB, which means that larger volumes have better performance. For example, baseline performance for a 100-GiB volume is 300 IOPS, and 3,000 IOPS for a 1-TiB volume. Volumes of 3.34 TiB and greater have a baseline performance of 10,000 IOPS.

Volumes below 1 TiB in size also have ability to burst to 3,000 IOPS for extended periods of time (burst is not relevant for volumes above 1 TiB). Instance I/O credit balance determines burst performance. For more information about instance I/O credits see, [I/O Credits and Burst Performance \(p. 104\)](#).

Many workloads never deplete the burst balance, making General Purpose SSD an ideal storage choice for many workloads. However, some workloads can exhaust the 3000 IOPS burst storage credit balance, so you should plan your storage capacity to meet the needs of your workloads.

I/O Credits and Burst Performance

General Purpose SSD storage performance is governed by volume size, which dictates the base performance level of the volume and how quickly it accumulates I/O credits. Larger volumes have higher base performance levels and accumulate I/O credits faster. *I/O credits* represent the available bandwidth that your General Purpose SSD storage can use to burst large amounts of I/O when more than the base level of performance is needed. The more I/O credits your storage has for I/O, the more time it can burst beyond its base performance level and the better it performs when your workload requires more performance.

When using General Purpose SSD storage, your DB instance receives an initial I/O credit balance of 5.4 million I/O credits. This initial credit balance is enough to sustain a burst performance of 3,000 IOPS for 30 minutes. This balance is designed to provide a fast initial boot cycle for boot volumes and to provide a good bootstrapping experience for other applications. Volumes earn I/O credits at the baseline performance rate of 3 IOPS for each GiB of volume size. For example, a 100-GiB SSD volume has a baseline performance of 300 IOPS.

When your storage requires more than the base performance I/O level, it uses I/O credits in the I/O credit balance to burst to the required performance level. Such a burst goes to a maximum of 3,000 IOPS. Storage larger than 1,000 GiB has a base performance that is equal or greater than the maximum burst performance. When your storage uses fewer I/O credits than it earns in a second, unused I/O credits are added to the I/O credit balance. The maximum I/O credit balance for a DB instance using General Purpose SSD storage is equal to the initial I/O credit balance (5.4 million I/O credits).

Suppose that your storage uses all of its I/O credit balance. If so, its maximum performance remains at the base performance level until I/O demand drops below the base level and unused I/O credits are added to the I/O credit balance. (The *base performance level* is the rate at which your storage earns I/O credits.) The more storage, the greater the base performance is and the faster it replenishes the I/O credit balance.

Note

Storage conversions between magnetic storage and General Purpose SSD storage can potentially deplete your I/O credit balance, resulting in longer conversion times. For more information about scaling storage, see [Working with Storage \(p. 188\)](#).

The following table lists several storage sizes. For each storage size, it lists the associated base performance of the storage, which is also the rate at which it accumulates I/O credits. The table also lists the burst duration at the 3,000 IOPS maximum, when starting with a full I/O credit balance. In addition, the table lists the time in seconds that the storage takes to refill an empty I/O credit balance.

Storage size (GiB)	Base Performance (IOPS)	Maximum Burst Duration at 3,000 IOPS (Seconds)	Seconds to Fill Empty I/O Credit Balance
1	100	1,862	54,000
100	300	2,000	18,000
250	750	2,400	7,200
500	1,500	3,600	3,600

Storage size (GiB)	Base Performance (IOPS)	Maximum Burst Duration at 3,000 IOPS (Seconds)	Seconds to Fill Empty I/O Credit Balance
750	2,250	7,200	2,400
1,000	3,000	Infinite	N/A
3,333	10,000	Infinite	N/A
10,000	10,000	Infinite	N/A

The burst duration of your storage depends on the size of the storage, the burst IOPS required, and the I/O credit balance when the burst begins. This relationship is shown in the equation following.

$$\text{Burst duration} = \frac{(\text{Credit balance})}{(\text{Burst IOPS}) - 3(\text{Storage size in GiB})}$$

You might notice that your storage performance is frequently limited to the base level due to an empty I/O credit balance. If so, consider allocating more General Purpose SSD storage with a higher base performance level. Alternatively, you can switch to Provisioned IOPS storage for workloads that require sustained IOPS performance.

For workloads with steady state I/O requirements, provisioning less than 100 GiB of General Purpose SSD storage might result in higher latencies if you exhaust your I/O credit balance.

Note

In general, most workloads never exceed the I/O credit balance.

For a more detailed description of how baseline performance and I/O credit balance affect performance see [Understanding Burst vs. Baseline Performance with Amazon RDS and GP2](#).

Provisioned IOPS SSD Storage

For production application that requires fast and consistent I/O performance, we recommend Provisioned IOPS (input/output operations per second) storage. Provisioned IOPS storage is a storage type that delivers predictable performance, and consistently low latency. Provisioned IOPS storage is optimized for online transaction processing (OLTP) workloads that have consistent performance requirements. Provisioned IOPS helps performance tuning of these workloads.

When you create a DB instance, you specify an IOPS rate and the size of the volume. Amazon RDS provides that IOPS rate for the DB instance until you change it.

Note

Your database workload might not be able to achieve 100 percent of the IOPS that you have provisioned.

The following table shows the range of Provisioned IOPS and storage size range for each database engine.

Database Engine	Range of Provisioned IOPS	Range of Storage
MariaDB	1,000–40,000 IOPS	100 GiB–32 TiB
SQL Server, Enterprise and Standard editions	1000–32,000 IOPS	20 GiB–16 TiB

Database Engine	Range of Provisioned IOPS	Range of Storage
SQL Server, Web and Express editions	1000–32,000 IOPS	100 GiB–16 TiB
MySQL	1,000–40,000 IOPS	100 GiB–32 TiB
Oracle	1,000–40,000 IOPS	100 GiB–32 TiB
PostgreSQL	1,000–40,000 IOPS	100 GiB–32 TiB

Combining Provisioned IOPS Storage with Multi-AZ deployments, or Read Replicas

For production OLTP use cases, we recommend that you use Multi-AZ deployments for enhanced fault tolerance with Provisioned IOPS storage for fast and predictable performance.

You can also use Provisioned IOPS SSD storage with Read Replicas for MySQL, MariaDB or PostgreSQL. The type of storage for a Read Replica is independent of that on the master DB instance. For example, you might use General Purpose SSD for Read Replicas with a master DB instance that uses Provisioned IOPS SSD storage to reduce costs. However, your Read Replicas performance in this case might differ from that of a configuration where both the master DB instance and the Read Replicas use Provisioned IOPS SSD storage.

Provisioned IOPS Storage Costs

With Provisioned IOPS storage, you are charged for the provisioned resources whether or not you use them in a given month.

For more information about pricing, see [Amazon RDS Pricing](#).

Getting the most out of Amazon RDS Provisioned IOPS SSD storage

If your workload is I/O constrained, using Provisioned IOPS SSD storage can increase the number of I/O requests that the system can process concurrently. Increased concurrency allows for decreased latency because I/O requests spend less time in a queue. Decreased latency allows for faster database commits, which improves response time and allows for higher database throughput.

Provisioned IOPS SSD storage provides a way to reserve I/O capacity by specifying IOPS. However, as with any other system capacity attribute, its maximum throughput under load is constrained by the resource that is consumed first. That resource might be network bandwidth, CPU, memory, or database internal resources.

Magnetic storage

Amazon RDS also supports magnetic storage for backward compatibility. We recommend that you use General Purpose SSD or Provisioned IOPS SSD for any new storage needs. The following are some limitations for magnetic storage:

- Doesn't allow you to scale storage when using the SQL Server database engine.
- Doesn't support elastic volumes.
- Limited to a maximum size of 4 TiB.
- Limited to a maximum of 1,000 IOPS.

Monitoring storage performance

Amazon RDS provides several metrics that you can use to determine how your DB instance is performing. You can view the metrics on the summary page for your instance in Amazon RDS Management Console. You can also use Amazon CloudWatch to monitor these metrics. For more information, see [Viewing DB Instance Metrics \(p. 254\)](#). Enhanced Monitoring provides more detailed I/O metrics; for more information, see [Enhanced Monitoring \(p. 256\)](#).

The following metrics are useful for monitoring storage for your DB instance:

- **IOPS** – The number of I/O operations completed each second. This metric is reported as the average IOPS for a given time interval. Amazon RDS reports read and write IOPS separately on 1-minute intervals. Total IOPS is the sum of the read and write IOPS. Typical values for IOPS range from zero to tens of thousands per second.
- **Latency** – The elapsed time between the submission of an I/O request and its completion. This metric is reported as the average latency for a given time interval. Amazon RDS reports read and write latency separately on 1-minute intervals in units of seconds. Typical values for latency are in the millisecond (ms). For example, Amazon RDS reports 2 ms as 0.002 seconds.
- **Throughput** – The number of bytes each second that are transferred to or from disk. This metric is reported as the average throughput for a given time interval. Amazon RDS reports read and write throughput separately on 1-minute intervals using units of megabytes per second (MB/s). Typical values for throughput range from zero to the I/O channel's maximum bandwidth.
- **Queue Depth** – The number of I/O requests in the queue waiting to be serviced. These are I/O requests that have been submitted by the application but have not been sent to the device because the device is busy servicing other I/O requests. Time spent waiting in the queue is a component of latency and service time (not available as a metric). This metric is reported as the average queue depth for a given time interval. Amazon RDS reports queue depth in 1-minute intervals. Typical values for queue depth range from zero to several hundred.

Measured IOPS values are independent of the size of the individual I/O operation. This means that when you measure I/O performance, you should look at the throughput of the instance, not simply the number of I/O operations.

Factors That Affect Storage Performance

Both system activities and database workload can affect storage performance.

System activities

The following system-related activities consume I/O capacity and might reduce database instance performance while in progress:

- Multi-AZ standby creation
- Read replica creation
- Changing storage types

Database workload

In some cases your database or application design results in concurrency issues, locking, or other forms of database contention. In these cases, you might not be able to use all the provisioned bandwidth directly. In addition, you may encounter the following workload-related situations:

- The throughput limit of the underlying instance type is reached.
- Queue depth is consistently less than 1 because your application is not driving enough I/O operations.

- You experience query contention in the database even though some I/O capacity is unused.

If there isn't at least one system resource that is at or near a limit, and adding threads doesn't increase the database transaction rate, the bottleneck is most likely contention in the database. The most common forms are row lock and index page lock contention, but there are many other possibilities. If this is your situation, you should seek the advice of a database performance tuning expert.

DB instance class

To get the most performance out of your Amazon RDS database instance, choose a current generation instance type with enough bandwidth to support your storage type. For example, you can choose EBS-optimized instances and instances with 10-gigabit network connectivity.

For the full list of Amazon EC2 instance types that support EBS optimization, see [Instance types that support EBS optimization](#).

We encourage you to use the latest generation of instances to get the best performance. Previous generation DB instances have a lower instance storage limit. Scaling higher than 6 TiB is not supported on the following previous generation instances.

- db.m1.small
- db.m1.medium
- db.m1.large
- db.m1.xlarge
- db.m2.xlarge
- db.m2.2xlarge
- db.m2.4xlarge
- db.m3.large
- db.m3.xlarge
- db.m3.2xlarge

For more information, see [Previous Generation DB Instances](#).

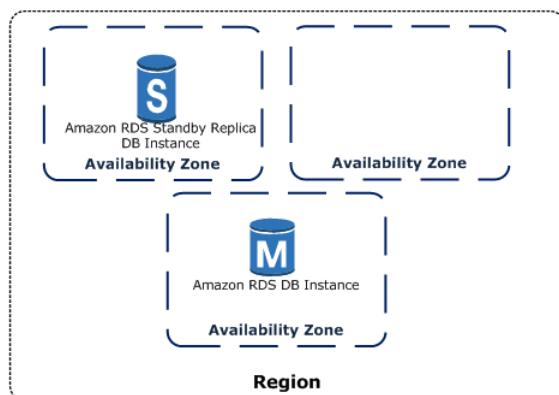
High Availability (Multi-AZ) for Amazon RDS

Amazon RDS provides high availability and failover support for DB instances using Multi-AZ deployments. Amazon RDS uses several different technologies to provide failover support. Multi-AZ deployments for Oracle, PostgreSQL, MySQL, and MariaDB DB instances use Amazon's failover technology. SQL Server DB instances use SQL Server Mirroring.

In a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to a standby replica to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups. Running a DB instance with high availability can enhance availability during planned system maintenance, and help protect your databases against DB instance failure and Availability Zone disruption. For more information on Availability Zones, see [Regions and Availability Zones \(p. 101\)](#).

Note

The high-availability feature is not a scaling solution for read-only scenarios; you cannot use a standby replica to serve read traffic. To service read-only traffic, you should use a Read Replica. For more information, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).



Using the RDS console, you can create a Multi-AZ deployment by simply specifying Multi-AZ when creating a DB instance. You can also use the console to convert existing DB instances to Multi-AZ deployments by modifying the DB instance and specifying the Multi-AZ option. The RDS console shows the Availability Zone of the standby replica, called the secondary AZ.

You can specify a Multi-AZ deployment using the CLI as well. Use the AWS CLI [describe-db-instances](#) command, or the Amazon RDS API [DescribeDBInstances](#) action to show the Availability Zone of the standby replica (called the secondary AZ).

The RDS console shows the Availability Zone of the standby replica (called the secondary AZ), or you can use the AWS CLI [describe-db-instances](#) command, or the Amazon RDS API [DescribeDBInstances](#) action to find the secondary AZ.

DB instances using Multi-AZ deployments may have increased write and commit latency compared to a Single-AZ deployment, due to the synchronous data replication that occurs. You may have a change in latency if your deployment fails over to the standby replica, although AWS is engineered with low-latency network connectivity between Availability Zones. For production workloads, we recommend that you use Provisioned IOPS and DB instance classes (m1.large and larger) that are optimized for Provisioned IOPS for fast, consistent performance.

Modifying a DB Instance to Be a Multi-AZ Deployment

If you have a DB instance in a Single-AZ deployment and you modify it to be a Multi-AZ deployment (for engines other than SQL Server or Amazon Aurora), Amazon RDS takes several steps. First, Amazon RDS takes a snapshot of the primary DB instance from your deployment and then restores the snapshot into another Availability Zone. Amazon RDS then sets up synchronous replication between your primary DB instance and the new instance. This action avoids downtime when you convert from Single-AZ to Multi-AZ, but you can experience a significant performance impact when first converting to Multi-AZ. This impact is more noticeable for large and write-intensive DB instances.

Once the modification is complete, Amazon RDS triggers an event (RDS-EVENT-0025) that indicates the process is complete. You can monitor Amazon RDS events; for more information about events, see [Using Amazon RDS Event Notification \(p. 287\)](#).

Failover Process for Amazon RDS

In the event of a planned or unplanned outage of your DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone if you have enabled Multi-AZ. The time it takes for the failover to complete depends on the database activity and other conditions at the time the primary DB instance became unavailable. Failover times are typically 60-120 seconds. However, large transactions or a lengthy recovery process can increase failover time. When the failover is complete, it can take additional time for the RDS console UI to reflect the new Availability Zone.

The failover mechanism automatically changes the DNS record of the DB instance to point to the standby DB instance. As a result, you need to re-establish any existing connections to your DB instance. Due to how the Java DNS caching mechanism works, you may need to reconfigure your JVM environment. For more information on how to manage a Java application that caches DNS values in the case of a failover, see the [AWS SDK for Java](#).

Amazon RDS handles failovers automatically so you can resume database operations as quickly as possible without administrative intervention. The primary DB instance switches over automatically to the standby replica if any of the following conditions occur:

- An Availability Zone outage
- The primary DB instance fails
- The DB instance's server type is changed
- The operating system of the DB instance is undergoing software patching
- A manual failover of the DB instance was initiated using **Reboot with failover**

There are several ways to determine if your Multi-AZ DB instance has failed over:

- DB event subscriptions can be setup to notify you via email or SMS that a failover has been initiated. For more information about events, see [Using Amazon RDS Event Notification \(p. 287\)](#)
- You can view your DB events by using the Amazon RDS console or API actions.
- You can view the current state of your Multi-AZ deployment by using the Amazon RDS console and API actions.

For information on how you can respond to failovers, reduce recovery time, and other best practices for Amazon RDS, see [Best Practices for Amazon RDS \(p. 70\)](#).

Related Topics

- [Multi-AZ Deployments for Microsoft SQL Server \(p. 551\)](#)
- [Licensing Oracle Multi-AZ Deployments \(p. 720\)](#)

Amazon RDS DB Instance Lifecycle

The lifecycle of an Amazon RDS DB instance includes creating, modifying, maintaining and upgrading, performing backups and restores, rebooting, and deleting the instance. This section provides information on and links to more about these processes.

Topics

- [Creating an Amazon RDS DB Instance \(p. 113\)](#)
- [Connecting to an Amazon RDS DB Instance \(p. 114\)](#)
- [Modifying an Amazon RDS DB Instance \(p. 115\)](#)
- [Maintaining a DB Instance \(p. 117\)](#)
- [Upgrading a DB Instance Engine Version \(p. 123\)](#)
- [Renaming a DB Instance \(p. 126\)](#)
- [Rebooting a DB Instance \(p. 129\)](#)
- [Stopping an Amazon RDS DB Instance Temporarily \(p. 131\)](#)
- [Starting an Amazon RDS DB Instance That Was Previously Stopped \(p. 133\)](#)
- [Deleting a DB Instance \(p. 135\)](#)

Creating an Amazon RDS DB Instance

The basic building block of Amazon RDS is the DB instance. To create an Amazon RDS DB instance, follow the instructions for your specific database engine.

- [Creating a DB Instance Running the MariaDB Database Engine \(p. 443\)](#)
- [Creating a DB Instance Running the Microsoft SQL Server Database Engine \(p. 504\)](#)
- [Creating a DB Instance Running the MySQL Database Engine \(p. 597\)](#)
- [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#)
- [Creating a DB Instance Running the PostgreSQL Database Engine \(p. 969\)](#)

Connecting to an Amazon RDS DB Instance

After you create an Amazon RDS DB instance, you can use any standard SQL client application to connect to the DB instance. To connect to an Amazon RDS DB instance, follow the instructions for your specific database engine.

- [Connecting to a DB Instance Running the MariaDB Database Engine \(p. 452\)](#)
- [Connecting to a DB Instance Running the Microsoft SQL Server Database Engine \(p. 514\)](#)
- [Connecting to a DB Instance Running the MySQL Database Engine \(p. 606\)](#)
- [Connecting to a DB Instance Running the Oracle Database Engine \(p. 751\)](#)
- [Connecting to a DB Instance Running the PostgreSQL Database Engine \(p. 976\)](#)

Modifying an Amazon RDS DB Instance

Most modifications to a DB instance can be applied immediately or deferred until the next maintenance window. Some modifications, such as parameter group changes, require that you manually reboot your DB instance for the change to take effect.

Important

Some modifications result in an outage because Amazon RDS must reboot your DB instance for the change to take effect. Review the impact to your database and applications before modifying your DB instance settings.

To modify an Amazon RDS DB instance, follow the instructions for your specific database engine.

- [Modifying a DB Instance Running the MariaDB Database Engine \(p. 456\)](#)
- [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#)
- [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#)
- [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#)
- [Modifying a DB Instance Running the PostgreSQL Database Engine \(p. 979\)](#)

Using the Apply Immediately Parameter

When you modify a DB instance, you can apply the changes immediately. To apply changes immediately, you select the **Apply Immediately** option in the AWS Management Console, you use the `--apply-immediately` parameter when calling the AWS CLI, or you set the `ApplyImmediately` parameter to `true` when using the Amazon RDS API.

If you don't choose to apply changes immediately, the changes are put into the pending modifications queue. During the next maintenance window, any pending changes in the queue are applied. If you choose to apply changes immediately, your new changes and any changes in the pending modifications queue are applied.

Important

If any of the pending modifications require downtime, choosing apply immediately can cause unexpected downtime.

When you choose to apply a change immediately, any pending modifications are also applied immediately, instead of during the next maintenance window.

Changes to some database settings are applied immediately, even if you choose to defer your changes. To see how the different database settings interact with the apply immediately setting, see the settings for your specific database engine.

- [Settings for MariaDB DB Instances \(p. 457\)](#)
- [Settings for Microsoft SQL Server DB Instances \(p. 522\)](#)
- [Settings for MySQL DB Instances \(p. 611\)](#)
- [Settings for Oracle DB Instances \(p. 759\)](#)
- [Settings for PostgreSQL DB Instances \(p. 980\)](#)

Related Topics

- [Renaming a DB Instance \(p. 126\)](#)
- [Rebooting a DB Instance \(p. 129\)](#)
- [Stopping an Amazon RDS DB Instance Temporarily \(p. 131\)](#)
- [modify-db-instance](#)

- [ModifyDBInstance](#)

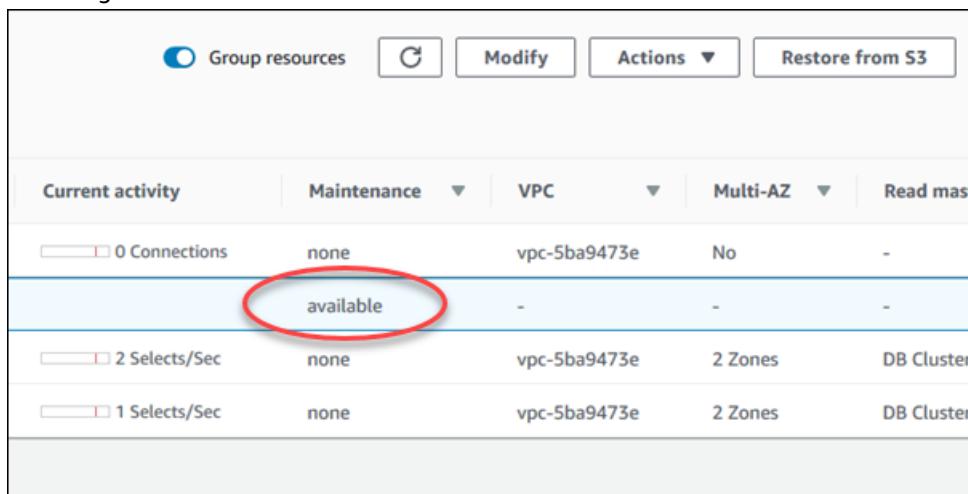
Maintaining a DB Instance

Periodically, Amazon RDS performs maintenance on Amazon RDS resources. Maintenance most often involves updates to the DB instance's underlying operating system (OS) or database engine version. Updates to the operating system most often occur for security issues and should be done as soon as possible.

Some maintenance items require that Amazon RDS take your DB instance offline for a short time. Maintenance items that require a resource to be offline include required operating system or database patching. Required patching is automatically scheduled only for patches that are related to security and instance reliability. Such patching occurs infrequently (typically once every few months) and seldom requires more than a fraction of your maintenance window.

Deferred DB instance modifications that you have chosen not to apply immediately are applied during the maintenance window. For example, you may choose to change the DB instance class or parameter group during the maintenance window. For information about modifying a DB instance, see [Modifying an Amazon RDS DB Instance \(p. 115\)](#).

You can view whether a maintenance update is available for your DB instance by using the RDS console, the AWS CLI, or the Amazon RDS API. If an update is available, it is indicated by the word **available** or **required** in the **Maintenance** column for the DB instance on the Amazon RDS console, as shown following.



Current activity	Maintenance	VPC	Multi-AZ	Read mas
0 Connections	none	vpc-5ba9473e	No	-
	available	-	-	-
2 Selects/Sec	none	vpc-5ba9473e	2 Zones	DB Cluster
1 Selects/Sec	none	vpc-5ba9473e	2 Zones	DB Cluster

If an update is available, you can take one of the actions:

- Defer the maintenance items.
- Apply the maintenance items immediately.
- Schedule the maintenance items to start during your next maintenance window.
- Take no action.

Note

Certain OS updates are marked as **required**. If you defer a required update, you get a notice from Amazon RDS indicating when the update will be performed. Other updates are marked as **available**, and these you can defer indefinitely.

The maintenance window determines when pending operations start, but doesn't limit the total execution time of these operations. Maintenance operations aren't guaranteed to finish before the maintenance window ends, and can continue beyond the specified end time. For more information, see [The Amazon RDS Maintenance Window \(p. 120\)](#).

Applying Updates for a DB Instance

With Amazon RDS, you can choose when to apply maintenance operations. You can decide when Amazon RDS applies updates by using the RDS console, AWS Command Line Interface (AWS CLI), or RDS API.

AWS Management Console

To manage an update for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that has a required update.
4. For **Actions**, choose one of the following:
 - **Upgrade now**
 - **Upgrade at next window**

Note

If you choose **Upgrade at next window** and later want to delay the update, you can choose **Defer upgrade**.

CLI

To apply a pending update to a DB instance, use the [apply-pending-maintenance-action](#) AWS CLI command.

Example

For Linux, OS X, or Unix:

```
aws rds apply-pending-maintenance-action \
--resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db \
--apply-action system-update \
--opt-in-type immediate
```

For Windows:

```
aws rds apply-pending-maintenance-action ^
--resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db ^
--apply-action system-update ^
--opt-in-type immediate
```

To return a list of resources that have at least one pending update, use the [describe-pending-maintenance-actions](#) AWS CLI command.

Example

For Linux, OS X, or Unix:

```
aws rds describe-pending-maintenance-actions \
--resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

For Windows:

```
aws rds describe-pending-maintenance-actions ^
--resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

You can also return a list of resources for a DB instance by specifying the `--filters` parameter of the `describe-pending-maintenance-actions` AWS CLI command. The format for the `--filters` command is `Name=filter-name,Value=resource-id,...`.

The following are the accepted values for the `Name` parameter of a filter:

- `db-instance-id` – Accepts a list of DB instance identifiers or Amazon Resource Names (ARNs). The returned list only includes pending maintenance actions for the DB instances identified by these identifiers or ARNs.
- `db-cluster-id` – Accepts a list of DB cluster identifiers or ARNs for Amazon Aurora. The returned list only includes pending maintenance actions for the DB clusters identified by these identifiers or ARNs.

For example, the following example returns the pending maintenance actions for the `sample-instance1` and `sample-instance2` DB instances.

Example

For Linux, OS X, or Unix:

```
aws rds describe-pending-maintenance-actions \
--filters Name=db-instance-id,Values=sample-instance1,sample-instance2
```

For Windows:

```
aws rds describe-pending-maintenance-actions ^
--filters Name=db-instance-id,Values=sample-instance1,sample-instance2
```

API

To apply an update to a DB instance, call the Amazon RDS API [ApplyPendingMaintenanceAction](#) action.

To return a list of resources that have at least one pending update, call the Amazon RDS API [DescribePendingMaintenanceActions](#) action.

Maintenance for Multi-AZ Deployments

Running a DB instance as a Multi-AZ deployment can further reduce the impact of a maintenance event, because Amazon RDS applies operating system updates by following these steps:

1. Perform maintenance on the standby.
2. Promote the standby to primary.
3. Perform maintenance on the old primary, which becomes the new standby.

When you modify the database engine for your DB instance in a Multi-AZ deployment, then Amazon RDS upgrades both the primary and secondary DB instances at the same time. In this case, the database engine for the entire Multi-AZ deployment is shut down during the upgrade.

For more information on Multi-AZ deployments, see [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#).

The Amazon RDS Maintenance Window

Every DB instance has a weekly maintenance window during which any system changes are applied. You can think of the maintenance window as an opportunity to control when modifications and software patching occur, in the event either are requested or required. If a maintenance event is scheduled for a given week, it is initiated during the 30-minute maintenance window you identify. Most maintenance events also complete during the 30-minute maintenance window, although larger maintenance events may take more than 30 minutes to complete.

The 30-minute maintenance window is selected at random from an 8-hour block of time per region. If you don't specify a preferred maintenance window when you create the DB instance, then Amazon RDS assigns a 30-minute maintenance window on a randomly selected day of the week.

RDS will consume some of the resources on your DB instance while maintenance is being applied. You might observe a minimal effect on performance. For a DB instance, on rare occasions, a Multi-AZ failover might be required for a maintenance update to complete.

Following, you can find the time blocks for each region from which default maintenance windows are assigned.

Region	Time Block
US West (Oregon) Region	06:00–14:00 UTC
US West (N. California) Region	06:00–14:00 UTC
US East (Ohio) Region	03:00–11:00 UTC
US East (N. Virginia) Region	03:00–11:00 UTC
Asia Pacific (Mumbai) Region	17:30–01:30 UTC
Asia Pacific (Seoul) Region	13:00–21:00 UTC
Asia Pacific (Singapore) Region	14:00–22:00 UTC
Asia Pacific (Sydney) Region	12:00–20:00 UTC
Asia Pacific (Tokyo) Region	13:00–21:00 UTC
Canada (Central) Region	03:00–11:00 UTC
EU (Frankfurt) Region	23:00–07:00 UTC
EU (Ireland) Region	22:00–06:00 UTC
EU (London) Region	22:00–06:00 UTC
South America (São Paulo) Region	00:00–08:00 UTC
AWS GovCloud (US-West)	06:00–14:00 UTC

Adjusting the Preferred DB Instance Maintenance Window

The maintenance window should fall at the time of lowest usage and thus might need modification from time to time. Your DB instance will only be unavailable during this time if the system changes, such as a

change in DB instance class, are being applied and require an outage, and only for the minimum amount of time required to make the necessary changes.

In the following example, you adjust the preferred maintenance window for a DB instance.

For the purpose of this example, we assume that the DB instance named *mydbinstance* exists and has a preferred maintenance window of "Sun:05:00-Sun:06:00" UTC.

AWS Management Console

To adjust the preferred maintenance window

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then select the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. In the **Maintenance** section, update the maintenance window.

Note

The maintenance window and the backup window for the DB instance cannot overlap. If you enter a value for the maintenance window that overlaps the backup window, an error message appears.

5. Choose **Continue**.

On the confirmation page, review your changes.

6. To apply the changes to the maintenance window immediately, select **Apply immediately**.
7. Choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To adjust the preferred maintenance window, use the AWS CLI `modify-db-instance` command with the following parameters:

- `--db-instance-identifier`
- `--preferred-maintenance-window`

Example

The following code example sets the maintenance window to Tuesdays from 4:00-4:30AM UTC.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

API

To adjust the preferred maintenance window, use the Amazon RDS API [ModifyDBInstance](#) action with the following parameters:

- `DBInstanceIdentifier = mydbinstance`
- `PreferredMaintenanceWindow = Tue:04:00-Tue:04:30`

Example

The following code example sets the maintenance window to Tuesdays from 4:00-4:30AM UTC.

```
https://rds.us-west-2.amazonaws.com/
?Action=ModifyDBInstance
&DBInstanceIdentifier=mydbinstance
&PreferredMaintenanceWindow=Tue:04:00-Tue:04:30
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/rds/aws4_request
&X-Amz-Date=20140425T192732Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

Upgrading a DB Instance Engine Version

Amazon RDS keeps your DB instance up-to-date by providing newer versions of each supported database engine. Newer versions can include bug fixes, security enhancements, and other improvements for the database engine. When Amazon RDS supports a new version of a database engine, you can choose how and when to upgrade your database DB instances.

There are two kinds of upgrades: *major version upgrades* and *minor version upgrades*. In general, a major engine version upgrade can introduce changes that are not compatible with existing applications. In contrast, a minor version upgrade includes only changes that are backward-compatible with existing applications.

The version numbering sequence is specific for each database engine. For example, Amazon RDS MySQL 5.7 and 8.0 are major engine versions and upgrading from any 5.7 version to any 8.0 version is a major version upgrade. Amazon RDS MySQL version 5.7.22 and 5.7.23 are minor versions and upgrading from 5.7.22 to 5.7.23 is a minor version upgrade.

For more information about major and minor version upgrades for a specific DB engine, see the following documentation for your DB engine:

- [Upgrading the MariaDB DB Engine \(p. 464\)](#)
- [Upgrading the Microsoft SQL Server DB Engine \(p. 529\)](#)
- [Upgrading the MySQL DB Engine \(p. 618\)](#)
- [Upgrading the Oracle DB Engine \(p. 770\)](#)
- [Upgrading the PostgreSQL DB Engine \(p. 988\)](#)

For major version upgrades, you must manually modify the DB engine version through the AWS Management Console, AWS CLI, or RDS API. For minor version upgrades, you can manually modify the engine version, or you can choose to enable auto minor version upgrades.

Topics

- [Manually Upgrading the Engine Version \(p. 123\)](#)
- [Automatically Upgrading the Minor Engine Version \(p. 124\)](#)

Manually Upgrading the Engine Version

To manually upgrade the engine version of a DB instance, you can use the AWS Management Console, the AWS CLI, or the RDS API.

Upgrading the Engine Version of a DB Instance Using the Console

To upgrade the engine version of a DB instance by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to upgrade.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. For **DB engine version**, choose the new version.
5. Choose **Continue** and check the summary of modifications.
6. To apply the changes immediately, choose **Apply immediately**. Choosing this option can cause an outage in some cases. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).

7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

Upgrading the Engine Version of a DB Instance Using the AWS CLI

To upgrade the engine version of a DB instance, use the CLI [modify-db-instance](#) command. Specify the following parameters:

- **--db-instance-identifier** – the name of the DB instance.
- **--engine-version** – the version number of the database engine to upgrade to.
For information about valid engine versions, use the AWS CLI [describe-db-engine-versions](#) command.
- **--allow-major-version-upgrade** – to upgrade the major version.
- **--no-apply-immediately** – to apply changes during the next maintenance window. To apply changes immediately, use **--apply-immediately**.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier <mydbinstance> \
  --engine-version <new_version> \
  --allow-major-version-upgrade \
  --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
  --db-instance-identifier <mydbinstance> ^
  --engine-version <new_version> ^
  --allow-major-version-upgrade ^
  --no-apply-immediately
```

Upgrading the Engine Version of a DB Instance Using the RDS API

To upgrade the engine version of a DB instance, use the [ModifyDBInstance](#) action. Specify the following parameters:

- **DBInstanceIdentifier** – the name of the DB instance, for example *mydbinstance*.
- **EngineVersion** – the version number of the database engine to upgrade to. For information about valid engine versions, use the [DescribeDBEngineVersions](#) operation.
- **AllowMajorVersionUpgrade** – whether to allow a major version upgrade. To do so, set the value to **true**.
- **ApplyImmediately** – whether to apply changes immediately or during the next maintenance window. To apply changes immediately, set the value to **true**. To apply changes during the next maintenance window, set the value to **false**.

Automatically Upgrading the Minor Engine Version

A *minor engine version* is an update to a DB engine version within a major engine version. For example, a major engine version might be 5.7 with the minor engine versions 5.7.4 and 5.7.5 within it.

If you want Amazon RDS to upgrade the DB engine version of a database automatically, you can enable auto minor version upgrades for the database. When a minor engine version is designated as the preferred minor engine version, each database that meets both of the following conditions is upgraded to the minor engine version automatically:

- The database is running a minor version of the DB engine that is lower than the preferred minor engine version.
- The database has auto minor version upgrade enabled.

You can control whether auto minor version upgrade is enabled for a DB instance when you perform the following tasks:

- [Creating a DB instance \(p. 113\)](#)
- [Modifying a DB instance \(p. 115\)](#)
- [Creating a Read Replica \(p. 145\)](#)
- [Restoring a DB instance from a snapshot \(p. 218\)](#)
- [Restoring a DB instance to a specific time \(p. 237\)](#)
- [Importing a DB instance from Amazon S3 \(p. 631\) \(for a MySQL backup on Amazon S3\)](#)

When you perform these tasks, you can control whether auto minor version upgrade is enabled for the DB instance in the following ways:

- Using the console, set the **Auto minor version upgrade** option.
- Using the AWS CLI, set the `--auto-minor-version-upgrade | --no-auto-minor-version-upgrade` option.
- Using the RDS API, set the `AutoMinorVersionUpgrade` parameter.

You can get an Amazon RDS event notification when a new minor engine version upgrade is available for one of your databases. To get notifications, subscribe to Amazon RDS event notification through the Amazon Simple Notification Service (Amazon SNS). For more information, see [Using Amazon RDS Event Notification \(p. 287\)](#).

To determine whether a maintenance update, such as a DB engine version upgrade, is available for your DB instance, you can use the console, AWS CLI, or RDS API. You can also upgrade the DB engine version manually and adjust the maintenance window. For more information, see [Maintaining a DB Instance \(p. 117\)](#).

Renaming a DB Instance

You can rename a DB instance by using the AWS Management Console, the AWS CLI `modify-db-instance` command, or the Amazon RDS API `ModifyDBInstance` action. Renaming a DB instance can have far-reaching effects; the following is a list of things you should know before you rename a DB instance.

- When you rename a DB instance, the endpoint for the DB instance changes, because the URL includes the name you assigned to the DB instance. You should always redirect traffic from the old URL to the new one.
- When you rename a DB instance, the old DNS name that was used by the DB instance is immediately deleted, although it could remain cached for a few minutes. The new DNS name for the renamed DB instance becomes effective in about 10 minutes. The renamed DB instance is not available until the new name becomes effective.
- You cannot use an existing DB instance name when renaming an instance.
- All read replicas associated with a DB instance remain associated with that instance after it is renamed. For example, suppose you have a DB instance that serves your production database and the instance has several associated read replicas. If you rename the DB instance and then replace it in the production environment with a DB snapshot, the DB instance that you renamed will still have the read replicas associated with it.
- Metrics and events associated with the name of a DB instance are maintained if you reuse a DB instance name. For example, if you promote a Read Replica and rename it to be the name of the previous master, the events and metrics associated with the master are associated with the renamed instance.
- DB instance tags remain with the DB instance, regardless of renaming.
- DB snapshots are retained for a renamed DB instance.

Renaming to Replace an Existing DB Instance

The most common reasons for renaming a DB instance are that you are promoting a Read Replica or you are restoring data from a DB snapshot or PITR. By renaming the database, you can replace the DB instance without having to change any application code that references the DB instance. In these cases, you would do the following:

1. Stop all traffic going to the master DB instance. This can involve redirecting traffic from accessing the databases on the DB instance or some other way you want to use to prevent traffic from accessing your databases on the DB instance.
2. Rename the master DB instance to a name that indicates it is no longer the master as described later in this topic.
3. Create a new master DB instance by restoring from a DB snapshot or by promoting a read replica, and then give the new instance the name of the previous master DB instance.
4. Associate any read replicas with the new master DB instance.

If you delete the old master DB instance, you are responsible for deleting any unwanted DB snapshots of the old master instance.

For information about promoting a Read Replica, see [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

AWS Management Console

To rename a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to rename.
4. Choose **Modify**.
5. In **Settings**, enter a new name for **DB instance identifier**.
6. Choose **Continue**.
7. To apply the changes immediately, choose **Apply immediately**. Choosing this option can cause an outage in some cases. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).
8. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To rename a DB instance, use the AWS CLI command `modify-db-instance`. Provide the current `--db-instance-identifier` value and `--new-db-instance-identifier` parameter with the new name of the DB instance.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier DBInstanceIdentifier \
  --new-db-instance-identifier NewDBInstanceIdentifier
```

For Windows:

```
aws rds modify-db-instance ^
  --db-instance-identifier DBInstanceIdentifier ^
  --new-db-instance-identifier NewDBInstanceIdentifier
```

API

To rename a DB instance, call Amazon RDS API function `ModifyDBInstance` with the following parameters:

- `DBInstanceIdentifier` = existing name for the instance
- `NewDBInstanceIdentifier` = new name for the instance

```
https://rds.amazonaws.com/
?Action=ModifyDBInstance
&DBInstanceIdentifier=mydbinstance
&NewDBInstanceIdentifier=mynewdbinstanceidentifier
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
```

```
&Timestamp=2012-01-20T22%3A06%3A23.624Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Rebooting a DB Instance

You might need to reboot your DB instance, usually for maintenance reasons. For example, if you make certain modifications, or if you change the DB parameter group associated with the DB instance , you must reboot the instance for the changes to take effect.

Rebooting a DB instance restarts the database engine service. Rebooting a DB instance results in a momentary outage, during which the DB instance status is set to *rebooting*.

If the Amazon RDS instance is configured for Multi-AZ, you can perform the reboot with a failover. An Amazon RDS event is created when the reboot is completed. If your DB instance is a Multi-AZ deployment, you can force a failover from one Availability Zone (AZ) to another when you reboot. When you force a failover of your DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone, and updates the DNS record for the DB instance to point to the standby DB instance. As a result, you need to clean up and re-establish any existing connections to your DB instance. Rebooting with failover is beneficial when you want to simulate a failure of a DB instance for testing, or restore operations to the original AZ after a failover occurs. For more information, see [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#).

You can't reboot your DB instance if it is not in the available state. Your database can be unavailable for several reasons, such as an in-progress backup, a previously requested modification, or a maintenance-window action.

The time required to reboot your DB instance depends on the crash recovery process of your specific database engine. To improve the reboot time, we recommend that you reduce database activity as much as possible during the reboot process. Reducing database activity reduces rollback activity for in-transit transactions.

AWS Management Console

To reboot a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to reboot.
3. For **Actions**, choose **Reboot**.

The **Reboot DB Instance** page appears.

4. (Optional) Choose **Reboot with failover?** to force a failover from one AZ to another.
5. Choose **Reboot** to reboot your DB instance.

Alternatively, choose **Cancel**.

CLI

To reboot a DB instance by using the AWS CLI, call the `reboot-db-instance` command.

Example Simple Reboot

For Linux, OS X, or Unix:

```
aws rds reboot-db-instance \
--db-instance-identifier mydbinstance
```

For Windows:

```
aws rds reboot-db-instance ^
--db-instance-identifier mydbinstance
```

Example Reboot with Failover

To force a failover from one AZ to the other, use the `--force-failover` parameter.

For Linux, OS X, or Unix:

```
aws rds reboot-db-instance \
--db-instance-identifier mydbinstance \
--force-failover
```

For Windows:

```
aws rds reboot-db-instance ^
--db-instance-identifier mydbinstance ^
--force-failover
```

API

To reboot a DB instance by using the Amazon RDS API, call the `RebootDBInstance` action.

Example Simple Reboot

```
https://rds.amazonaws.com/
?Action=RebootDBInstance
&DBInstanceIdentifier=mydbinstance
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
&X-Amz-Date=20131016T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Example Reboot with Failover

To force a failover from one AZ to the other, set the `ForceFailover` parameter to `true`.

```
https://rds.amazonaws.com/
?Action=RebootDBInstance
&DBInstanceIdentifier=mydbinstance
&ForceFailover=true
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
&X-Amz-Date=20131016T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Stopping an Amazon RDS DB Instance Temporarily

If you use a DB instance intermittently, for temporary testing, or for a daily development activity, you can stop your Amazon RDS DB instance temporarily to save money. While your DB instance is stopped, you are charged for provisioned storage (including Provisioned IOPS) and backup storage (including manual snapshots and automated backups within your specified retention window), but not for DB instance hours. For more information, see [Billing FAQs](#).

You can stop and start DB instances that are running the following engines:

- MariaDB
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL

Stopping and starting a DB instance is supported for all DB instance classes, and in all AWS Regions.

You can stop and start a DB instance whether it is configured for a single Availability Zone or for Multi-AZ, for database engines that support Multi-AZ deployments. You can't stop an Amazon RDS for SQL Server DB instance in a Multi-AZ configuration.

When you stop a DB instance, the DB instance performs a normal shutdown and stops running. The status of the DB instance changes to `stopping` and then `stopped`. Any storage volumes remain attached to the DB instance, and their data is kept. Any data stored in the RAM of the DB instance is deleted.

You can stop a DB instance for up to seven days. If you don't manually start your DB instance after seven days, your DB instance is automatically started.

Benefits

Stopping and starting a DB instance is faster than creating a DB snapshot, and then restoring the snapshot.

When you stop a DB instance it retains its ID, Domain Name Server (DNS) endpoint, parameter group, security group, and option group. When you start a DB instance, it has the same configuration as when you stopped it. In addition, if you stop a DB instance, Amazon RDS retains the Amazon Simple Storage Service (Amazon S3) transaction logs so you can do a point-in-time restore if necessary.

Limitations

The following are some limitations to stopping and starting a DB instance:

- You can't stop a DB instance that has a Read Replica, or that is a Read Replica.
- You can't stop an Amazon RDS for SQL Server DB instance in a Multi-AZ configuration.
- You can't modify a stopped DB instance.
- You can't delete an option group that is associated with a stopped DB instance.
- You can't delete a DB parameter group that is associated with a stopped DB instance.

Option and Parameter Group Considerations

You can't remove persistent options (including permanent options) from an option group if there are DB instances associated with that option group. This functionality is also true of any DB instance with a state of `stopping`, `stopped`, or `starting`.

You can change the option group or DB parameter group that is associated with a stopped DB instance, but the change does not occur until the next time you start the DB instance. If you chose to apply changes immediately, the change occurs when you start the DB instance. Otherwise the changes occurs during the next maintenance window after you start the DB instance.

VPC Considerations

When you stop a DB instance it retains its DNS endpoint. If you stop a DB instance that is not in an Amazon Virtual Private Cloud (Amazon VPC), Amazon RDS releases the IP addresses of the DB instance. If you stop a DB instance that is in a VPC, the DB instance retains its IP addresses.

Note

You should always connect to a DB instance using the DNS endpoint, not the IP address.

AWS Management Console

To stop a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to stop.
3. Choose **Actions**, and then choose **Stop**.
4. (Optional) In the **Stop DB Instance** window, choose **Yes** for **Create Snapshot?** and enter the snapshot name for **Snapshot name**. Choose **Yes** if you want to create a snapshot of the DB instance before stopping it.
5. Choose **Yes, Stop Now** to stop the DB instance, or choose **Cancel** to cancel the operation.

CLI

To stop a DB instance by using the AWS CLI, call the `stop-db-instance` command with the following parameters:

- `--db-instance-identifier` – the name of the DB instance.

Example

```
stop-db-instance --db-instance-identifier mydbinstance
```

API

To stop a DB instance by using the Amazon RDS API, call the `StopDBInstance` action with the following parameter:

- `DBInstanceIdentifier` – the name of the DB instance.

Starting an Amazon RDS DB Instance That Was Previously Stopped

You can stop your Amazon RDS DB instance temporarily to save money. After you stop your DB instance, you can restart it to begin using it again. For more details about stopping and starting DB instances, see [Stopping an Amazon RDS DB Instance Temporarily \(p. 131\)](#).

When you start a DB instance that you previously stopped, the DB instance retains the ID, Domain Name Server (DNS) endpoint, parameter group, security group, and option group. When you start a stopped instance, you are charged a full instance hour.

AWS Management Console

To start a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to start.
3. Choose **Actions**, and then choose **Start**.

CLI

To start a DB instance by using the AWS CLI, call the `start-db-instance` command with the following parameters:

- `--db-instance-identifier` – the name of the DB instance.

Example

```
start-db-instance --db-instance-identifier mydbinstance
```

API

To start a DB instance by using the Amazon RDS API, call the `StartDBInstance` action with the following parameters:

- `DBInstanceIdentifier` – the name of the DB instance.

Example

```
https://rds.amazonaws.com/
    ?Action=StartDBInstance
    &DBInstanceIdentifier=mydbinstance
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- [Deleting a DB Instance \(p. 135\)](#)
- [Rebooting a DB Instance \(p. 129\)](#)

Deleting a DB Instance

To delete a DB instance, you must do the following:

- Provide the name of the instance
- Enable or disable the option to take a final DB snapshot of the instance
- Enable or disable the option to retain automated backups

You can only delete instances that don't have deletion protection enabled. When you create or modify a DB instance, you have the option to enable deletion protection so that users can't delete the DB instance. Deletion protection is disabled by default for you when you use AWS CLI and API commands. Deletion protection is enabled for you when you use the AWS Management Console to create a production DB instance. However, Amazon RDS enforces deletion protection when you use the console, the CLI, or the API to delete a DB instance. To delete a DB instance that has deletion protection enabled, first modify the instance and disable deletion protection.

If the DB instance that you want to delete has a Read Replica, you should either promote the Read Replica or delete it. For more information, see [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

Creating a Final Snapshot and Retaining Automated Backups

When you delete a DB instance, you can choose whether to create a final snapshot of the DB instance. You can also choose to retain automated backups after the DB instance is deleted. To be able to restore the DB instance at a later time, create a final snapshot or retain automated backups.

	With Final Snapshot	Without Final Snapshot	Retain Automated Backups
How to choose	To be able to restore your deleted DB instance at a later time, create a final DB snapshot.	To delete a DB instance quickly, you can skip creating a final DB snapshot. Important If you skip the snapshot, to restore your DB instance you need one of the following: <ul style="list-style-type: none">• You have to use an earlier manual snapshot of the DB instance to restore the DB instance to that snapshot's point in time.• You have to choose to retain automated backups; you can use those to restore it to any point in time within your retention period.	Instead of creating a snapshot, you can choose to enable Retain automated backups when you delete a DB instance. These backups are still subject to the retention period of the DB instance and age out the same way systems snapshots do.
Automated backups	All automated backups are deleted and can't be recovered, unless you enable Retain automated backups .	All automated backups are deleted and can't be recovered, unless you choose to retain automated backups when you delete the DB instance.	Automated backups are retained for a set period of time, regardless of whether you chose to create a final snapshot. They are retained for retention period that was set on the DB instance at the time you deleted it.

	With Final Snapshot	Without Final Snapshot	Retain Automated Backups
Manual snapshots	Earlier manual snapshots aren't deleted.	Earlier manual snapshots aren't deleted.	No snapshots are deleted.

You can't create a final snapshot of your DB instance if it has the status `creating`, `failed`, `incompatible-restore`, or `incompatible-network`. For more information about DB instance statuses, see [DB Instance Status \(p. 98\)](#).

Deleting a DB Instance by Using the Console, CLI, and API

You can delete a DB instance using the AWS Management Console, the AWS CLI, or the RDS API.

Console

To delete a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to delete.
3. For **Actions**, choose **Delete**.
4. For **Create final Snapshot?**, choose **Yes** or **No**.
5. If you chose **Yes** in the previous step, for **Final snapshot name** enter the name of your final DB snapshot.
6. To retain automated backups, choose **Retain automated backups**.
7. Enter **delete me** in the box.
8. Choose **Delete**.

AWS CLI

To delete a DB instance by using the AWS CLI, call the `delete-db-instance` command with the following options:

- `--db-instance-identifier`
- `--final-db-snapshot-identifier` or `--skip-final-snapshot`

Example With a final snapshot and no retained automated backups

For Linux, OS X, or Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance \
    --final-db-snapshot-identifier mydbinstancefinalsnapshot \
    --delete-automated-backups
```

For Windows:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance ^
    --final-db-snapshot-identifier mydbinstancefinalsnapshot ^
    --delete-automated-backups
```

Example With retained automated backups and no final snapshot

For Linux, OS X, or Unix:

```
aws rds delete-db-instance \
--db-instance-identifier mydbinstance \
--skip-final-snapshot \
--no-delete-automated-backups
```

For Windows:

```
aws rds delete-db-instance ^
--db-instance-identifier mydbinstance ^
--skip-final-snapshot ^
--no-delete-automated-backups
```

RDS API

To delete a DB instance by using the Amazon RDS API, call the [DeleteDBInstance](#) action with the following parameters:

- `DBInstanceIdentifier`
- `FinalDBSnapshotIdentifier` or `SkipFinalSnapshot`

Example With a final snapshot and no retained automated backups

```
https://rds.amazonaws.com/
?Action=DeleteDBInstance
&DBInstanceIdentifier=mydbinstance
&FinalDBSnapshotIdentifier=mydbinstancefinalsnapshot
&DeleteAutomatedBackups=true
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-1/rds/aws4_request
&X-Amz-Date=20140305T185838Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=b441901545441d3c7a48f63b5b1522c5b2b37c137500c93c45e209d4b3a064a3
```

Example With retained automated backups and no final snapshot

```
https://rds.amazonaws.com/
?Action=DeleteDBInstance
&DBInstanceIdentifier=mydbinstance
&SkipFinalSnapshot=true
&DeleteAutomatedBackups=false
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-1/rds/aws4_request
&X-Amz-Date=20140305T185838Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=b441901545441d3c7a48f63b5b1522c5b2b37c137500c93c45e209d4b3a064a3
```

Tagging Amazon RDS Resources

You can use Amazon RDS tags to add metadata to your Amazon RDS resources. In addition, these tags can be used with IAM policies to manage access to Amazon RDS resources and to control what actions can be applied to the Amazon RDS resources. Finally, these tags can be used to track costs by grouping expenses for similarly tagged resources.

All Amazon RDS resources can be tagged

- DB instances
- DB clusters
- Read Replicas
- DB snapshots
- DB cluster snapshots
- Reserved DB instances
- Event subscriptions
- DB option groups
- DB parameter groups
- DB cluster parameter groups
- DB security groups
- DB subnet groups

For information on managing access to tagged resources with IAM policies, see [Authentication and Access Control \(p. 342\)](#).

Overview of Amazon RDS Resource Tags

An Amazon RDS tag is a name-value pair that you define and associate with an Amazon RDS resource. The name is referred to as the key. Supplying a value for the key is optional. You can use tags to assign arbitrary information to an Amazon RDS resource. You can use a tag key, for example, to define a category, and the tag value might be an item in that category. For example, you might define a tag key of "project" and a tag value of "Salix," indicating that the Amazon RDS resource is assigned to the Salix project. You can also use tags to designate Amazon RDS resources as being used for test or production by using a key such as environment=test or environment=production. We recommend that you use a consistent set of tag keys to make it easier to track metadata associated with Amazon RDS resources.

Use tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of combined resources, organize your billing information according to resources with the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see [Cost Allocation and Tagging in About AWS Billing and Cost Management](#).

Each Amazon RDS resource has a tag set, which contains all the tags that are assigned to that Amazon RDS resource. A tag set can contain as many as 10 tags, or it can be empty. If you add a tag to an Amazon RDS resource that has the same key as an existing tag on resource, the new value overwrites the old value.

AWS does not apply any semantic meaning to your tags; tags are interpreted strictly as character strings. Amazon RDS can set tags on a DB instance or other Amazon RDS resources, depending on the settings that you use when you create the resource. For example, Amazon RDS might add a tag indicating that a DB instance is for production or for testing.

- The tag key is the required name of the tag. The string value can be from 1 to 128 Unicode characters in length and cannot be prefixed with "aws:" or "rds:". The string can contain only the set of Unicode letters, digits, white-space, '_', '!', '/', '=', '+', '-' (Java regex: " $^([\backslash p\{L\} \backslash p\{Z\} \backslash p\{N\}_:/=+\backslash -]*$$ ").
- The tag value is an optional string value of the tag. The string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with "aws:". The string can contain only the set of Unicode letters, digits, white-space, '_', '!', '/', '=', '+', '-' (Java regex: " $^([\backslash p\{L\} \backslash p\{Z\} \backslash p\{N\}_:/=+\backslash -]*$$ ").

Values do not have to be unique in a tag set and can be null. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity.

Note

You can add a tag to a snapshot, however, your bill will not reflect this grouping.

You can use the AWS Management Console, the command line interface, or the Amazon RDS API to add, list, and delete tags on Amazon RDS resources. When using the command line interface or the Amazon RDS API, you must provide the Amazon Resource Name (ARN) for the Amazon RDS resource you want to work with. For more information about constructing an ARN, see [Constructing an ARN for Amazon RDS \(p. 181\)](#).

Tags are cached for authorization purposes. Because of this, additions and updates to tags on Amazon RDS resources can take several minutes before they are available.

Copying Tags

When you create or restore a DB instance, you can specify that the tags from the DB instance are copied to snapshots of the DB instance. Copying tags ensures that the metadata for the DB snapshots matches that of the source DB instance and any access policies for the DB snapshot also match those of the source DB instance. Tags are not copied by default.

You can specify that tags are copied to DB snapshots for the following actions:

- Creating a DB instance.
- Restoring a DB instance.
- Creating a Read Replica.
- Copying a DB snapshot.

Note

If you include a value for the `--tag-key` parameter of the `create-db-snapshot` AWS CLI command (or supply at least one tag to the `CreateDBSnapshot` API action) then RDS doesn't copy tags from the source DB instance to the new DB snapshot. This functionality applies even if the source DB instance has the `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) option enabled. If you take this approach, you can create a copy of a DB instance from a DB snapshot and avoid adding tags that don't apply to the new DB instance. Once you have created your DB snapshot using the AWS CLI `create-db-snapshot` command (or the `CreateDBSnapshot` Amazon RDS API action) you can then add tags as described later in this topic.

AWS Management Console

The process to tag an Amazon RDS resource is similar for all resources. The following procedure shows how to tag an Amazon RDS DB instance.

To add a tag to a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.

Note

To filter the list of DB instances in the **Databases** pane, enter a text string for **Filter databases**. Only DB instances that contain the string appear.

3. Choose the name of the DB instance that you want to tag to show its details.
4. In the details section, scroll down to the **Tags** section.
5. Choose **Add**. The **Add tags** window appears.

Tag key	Value
<input type="text"/>	<input type="text"/>

6. Enter a value for **Tag key** and **Value**.
7. To add another tag, you can choose **Add another Tag** and enter a value for its **Tag key** and **Value**.
Repeat this step as many times as necessary.
8. Choose **Add**.

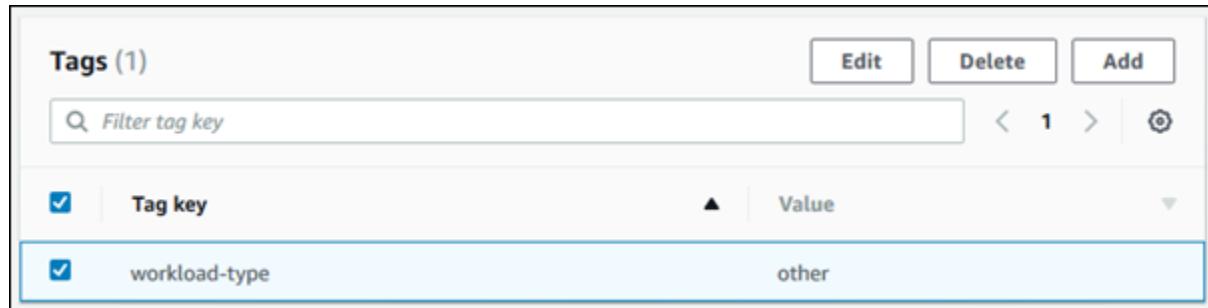
To delete a tag from a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.

Note

To filter the list of DB instances in the **Databases** pane, enter a text string in the **Filter databases** box. Only DB instances that contain the string appear.

3. Choose the name of the DB instance to show its details.
4. In the details section, scroll down to the **Tags** section.
5. Choose the tag you want to delete.



6. Choose **Delete**, and then choose **Delete** in the **Delete tags** window.

CLI

You can add, list, or remove tags for a DB instance using the AWS CLI.

- To add one or more tags to an Amazon RDS resource, use the AWS CLI command [add-tags-to-resource](#).
- To list the tags on an Amazon RDS resource, use the AWS CLI command [list-tags-for-resource](#).
- To remove one or more tags from an Amazon RDS resource, use the AWS CLI command [remove-tags-from-resource](#).

To learn more about how to construct the required ARN, see [Constructing an ARN for Amazon RDS \(p. 181\)](#).

API

You can add, list, or remove tags for a DB instance using the Amazon RDS API.

- To add a tag to an Amazon RDS resource, use the [AddTagsToResource](#) operation.
- To list tags that are assigned to an Amazon RDS resource, use the [ListTagsForResource](#).
- To remove tags from an Amazon RDS resource, use the [RemoveTagsFromResource](#) operation.

To learn more about how to construct the required ARN, see [Constructing an ARN for Amazon RDS \(p. 181\)](#).

When working with XML using the Amazon RDS API, tags use the following schema:

```
<Tagging>
  <TagSet>
    <Tag>
      <Key>Project</Key>
      <Value>Trinity</Value>
    </Tag>
    <Tag>
      <Key>User</Key>
      <Value>Jones</Value>
    </Tag>
  </TagSet>
</Tagging>
```

The following table provides a list of the allowed XML tags and their characteristics. Values for Key and Value are case-dependent. For example, project=Trinity and PROJECT=Trinity are two distinct tags.

Tagging Element	Description
TagSet	A tag set is a container for all tags assigned to an Amazon RDS resource. There can be only one tag set per resource. You work with a TagSet only through the Amazon RDS API.
Tag	A tag is a user-defined key-value pair. There can be from 1 to 50 tags in a tag set.
Key	<p>A key is the required name of the tag. The string value can be from 1 to 128 Unicode characters in length and cannot be prefixed with "rds:" or "aws:". The string can only contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "<code>^([\\p{L}\\p{Z}\\p{N}_.:/=-\\-]*\$</code>").</p> <p>Keys must be unique to a tag set. For example, you cannot have a key-pair in a tag set with the key the same but with different values, such as project/Trinity and project/Xanadu.</p>
Value	<p>A value is the optional value of the tag. The string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with "rds:" or "aws:". The string can only contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "<code>^([\\p{L}\\p{Z}\\p{N}_.:/=-\\-]*\$</code>").</p> <p>Values do not have to be unique in a tag set and can be null. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity.</p>

Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances

Amazon RDS uses the MariaDB, MySQL, and PostgreSQL DB engines' built-in replication functionality to create a special type of DB instance called a Read Replica from a source DB instance. Updates made to the source DB instance are asynchronously copied to the Read Replica. You can reduce the load on your source DB instance by routing read queries from your applications to the Read Replica. Using Read Replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.

Note

The information following applies to creating Amazon RDS Read Replicas either in the same AWS Region as the source DB instance, or in a separate AWS Region. The information following doesn't apply to setting up replication with an instance that is running on an Amazon EC2 instance or that is on-premises.

When you create a Read Replica, you first specify an existing DB instance as the source. Then Amazon RDS takes a snapshot of the source instance and creates a read-only instance from the snapshot. Amazon RDS then uses the asynchronous replication method for the DB engine to update the Read Replica whenever there is a change to the source DB instance. The Read Replica operates as a DB instance that allows only read-only connections. Applications connect to a Read Replica the same way they do to any DB instance. Amazon RDS replicates all databases in the source DB instance.

Amazon RDS sets up a secure communications channel between the source DB instance and a Read Replica if that Read Replica is in a different AWS Region from the DB instance. Amazon RDS establishes any AWS security configurations needed to enable the secure channel, such as adding security group entries.

Read Replicas are supported by the MariaDB, MySQL, and PostgreSQL engines. This section provides general information about using Read Replicas with all three of these engines. For information about using Read Replicas with a specific engine, see the following sections:

- [Working with MySQL Read Replicas \(p. 657\)](#)
- [Working with MariaDB Read Replicas \(p. 470\)](#)
- [Working with PostgreSQL Read Replicas \(p. 992\)](#)

Overview of Amazon RDS Read Replicas

Deploying one or more Read Replicas for a given source DB instance might make sense in a variety of scenarios, including the following:

- Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads. You can direct this excess read traffic to one or more Read Replicas.
- Serving read traffic while the source DB instance is unavailable. If your source DB instance can't take I/O requests (for example, due to I/O suspension for backups or scheduled maintenance), you can direct read traffic to your Read Replicas. For this use case, keep in mind that the data on the Read Replica might be "stale" because the source DB instance is unavailable.
- Business reporting or data warehousing scenarios where you might want business reporting queries to run against a Read Replica, rather than your primary, production DB instance.
- Implementing disaster recovery. You can use promote a Read Replica to a standalone instances as a disaster recovery solution if the source DB instance fails.

By default, a Read Replica is created with the same storage type as the source DB instance. However, you can create a Read Replica that has a different storage type from the source DB instance based on the options listed in the following table.

Source DB Instance Storage Type	Source DB Instance Storage Allocation	Read Replica Storage Type Options
PIOPS	100 GiB – 32 TiB	PIOPS, GP2, Standard
GP2	100 GiB – 32 TiB	PIOPS, GP2, Standard
GP2	Less than 100 GiB	GP2, Standard
Standard	100 GiB – 32 TiB	PIOPS, GP2, Standard
Standard	Less than 100 GiB	GP2, Standard

Amazon RDS doesn't support circular replication. You can't configure a DB instance to serve as a replication source for an existing DB instance; you can only create a new Read Replica from an existing DB instance. For example, if MyDBInstance replicates to ReadReplica1, you can't configure ReadReplica1 to replicate back to MyDBInstance. From ReadReplica1, you can only create a new Read Replica, such as ReadReplica2.

Differences Between PostgreSQL and MySQL or MariaDB Read Replicas

Because the PostgreSQL DB engine implements replication differently than the MySQL and MariaDB DB engines, there are several significant differences you should know about, as shown in the following table.

Feature or Behavior	PostgreSQL	MySQL and MariaDB
What is the replication method?	Physical replication.	Logical replication.
How are transaction logs purged?	PostgreSQL has a parameter, <code>wal_keep_segments</code> , that dictates how many write ahead log (WAL) files are kept to provide data to the Read Replicas. The parameter value specifies the number of logs to keep.	Amazon RDS keeps any binary logs that haven't been applied.
Can a replica be made writable?	No. A PostgreSQL Read Replica is a physical copy, and PostgreSQL doesn't allow for a Read Replica to be made writable.	Yes. You can enable the MySQL or MariaDB Read Replica to be writable.
Can backups be performed on the replica?	Yes, you can create a manual snapshot of a PostgreSQL Read Replica, but you can't enable automatic backups.	Yes. You can enable automatic backups on a MySQL or MariaDB Read Replica.
Can you use parallel replication?	No. PostgreSQL has a single process handling replication.	Yes. MySQL version 5.6 and later and all supported MariaDB versions allow for parallel replication threads.

Creating a Read Replica

You can create a Read Replica from an existing MySQL, MariaDB, or PostgreSQL DB instance using the AWS Management Console, AWS CLI, or AWS API. You create a Read Replica by specifying the `SourceDBInstanceIdentifier`, which is the DB instance identifier of the source DB instance from which you wish to replicate.

When you create a Read Replica, Amazon RDS takes a DB snapshot of your source DB instance and begins replication. As a result, you experience a brief I/O suspension on your source DB instance while the DB snapshot occurs. The I/O suspension typically lasts about one minute. You can avoid the I/O suspension if the source DB instance is a Multi-AZ deployment, because in that case the snapshot is taken from the secondary DB instance. An active, long-running transaction can slow the process of creating the Read Replica. We recommend that you wait for long-running transactions to complete before creating a Read Replica. If you create multiple Read Replicas in parallel from the same source DB instance, Amazon RDS takes only one snapshot at the start of the first create action.

When creating a Read Replica, there are a few things to consider. First, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0. This requirement also applies to a Read Replica that is the source DB instance for another Read Replica. For MySQL DB instances, automatic backups are supported only for Read Replicas running MySQL 5.6 and later, but not for MySQL versions 5.5. To enable automatic backups on an Amazon RDS MySQL version 5.6 and later Read Replica, first create the Read Replica, then modify the Read Replica to enable automatic backups.

AWS Management Console

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the MySQL, MariaDB, or PostgreSQL DB instance that you want to use as the source for a Read Replica.
4. For **Actions**, choose **Create read replica**.
5. Choose the instance specifications that you want to use. We recommend that you use the same DB instance class and storage type as the source DB instance for the Read Replica. For **Multi-AZ deployment**, choose **Yes** to create a standby of your replica in another Availability Zone for failover support for the replica. Creating your Read Replica as a Multi-AZ DB instance is independent of whether the source database is a Multi-AZ DB instance.
6. Choose the settings that you want to use. For **DB instance identifier**, enter a name for the Read Replica. Adjust other settings as needed.
7. Choose the other settings that you want to use.
8. Choose **Create read replica**.

CLI

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance, use the AWS CLI command `create-db-instance-read-replica`.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance-read-replica \
```

```
--db-instance-identifier myreadreplica \
--source-db-instance-identifier mydbinstance
```

For Windows:

```
aws rds create-db-instance-read-replica ^
--db-instance-identifier myreadreplica ^
--source-db-instance-identifier mydbinstance
```

API

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance, call the Amazon RDS API function [CreateDBInstanceReadReplica](#).

```
https://rds.amazonaws.com/
?Action=CreateDBInstanceReadReplica
&DBInstanceIdentifier=myreadreplica
&SourceDBInstanceIdentifier=mydbinstance
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2012-01-20T22%3A06%3A23.624Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Promoting a Read Replica to Be a Standalone DB Instance

You can promote a MySQL, MariaDB, or PostgreSQL Read Replica into a standalone DB instance. When you promote a Read Replica, the DB instance is rebooted before it becomes available.

There are several reasons you might want to promote a Read Replica to a standalone DB instance:

- **Performing DDL operations (MySQL and MariaDB only)** – DDL operations, such as creating or rebuilding indexes, can take time and impose a significant performance penalty on your DB instance. You can perform these operations on a MySQL or MariaDB Read Replica once the Read Replica is in sync with its source DB instance. Then you can promote the Read Replica and direct your applications to use the promoted instance.
- **Sharding** – Sharding embodies the "share-nothing" architecture and essentially involves breaking a large database into several smaller databases. One common way to split a database is splitting tables that are not joined in the same query onto different hosts. Another method is duplicating a table across multiple hosts and then using a hashing algorithm to determine which host receives a given update. You can create Read Replicas corresponding to each of your shards (smaller databases) and promote them when you decide to convert them into standalone shards. You can then carve out the key space (if you are splitting rows) or distribution of tables for each of the shards depending on your requirements.
- **Implementing failure recovery** – You can use Read Replica promotion as a data recovery scheme if the source DB instance fails. This approach complements synchronous replication, automatic failure detection, and failover.

If you are aware of the ramifications and limitations of asynchronous replication and you still want to use Read Replica promotion for data recovery, you can do so. To do this, first create a Read Replica and then monitor the source DB instance for failures. In the event of a failure, do the following:

1. Promote the Read Replica.

2. Direct database traffic to the promoted DB instance.
3. Create a replacement Read Replica with the promoted DB instance as its source.

When you promote a Read Replica, the new DB instance that is created retains the backup retention period, the backup window, and the parameter group of the former Read Replica source. The promotion process can take several minutes or longer to complete, depending on the size of the Read Replica. Once you promote the Read Replica to a new DB instance, it's just like any other DB instance. For example, you can convert the new DB instance into a Multi-AZ DB instance, create Read Replicas from it, and perform point-in-time restore operations. Because the promoted DB instance is no longer a Read Replica, you can't use it as a replication target. If a source DB instance has several Read Replicas, promoting one of the Read Replicas to a DB instance has no effect on the other replicas.

Backup duration is a function of the amount of changes to the database since the previous backup. If you plan to promote a Read Replica to a standalone instance, we recommend that you enable backups and complete at least one backup prior to promotion. In addition, a Read Replica cannot be promoted to a standalone instance when it is in the *backing-up* status. If you have enabled backups on your Read Replica, configure the automated backup window so that daily backups do not interfere with Read Replica promotion.

The following steps show the general process for promoting a Read Replica to a DB instance:

1. Stop any transactions from being written to the Read Replica source DB instance, and then wait for all updates to be made to the Read Replica. Database updates occur on the Read Replica after they have occurred on the source DB instance, and this replication lag can vary significantly. Use the [Replica Lag](#) metric to determine when all updates have been made to the Read Replica.
2. For MySQL and MariaDB only: If you need to make changes to the MySQL or MariaDB Read Replica, you must set the `read_only` parameter to 0 in the DB parameter group for the Read Replica. You can then perform all needed DDL operations, such as creating indexes, on the Read Replica. Actions taken on the Read Replica don't affect the performance of the source DB instance.
3. Promote the Read Replica by using the **Promote Read Replica** option on the Amazon RDS console, the AWS CLI command [promote-read-replica](#), or the [PromoteReadReplica](#) Amazon RDS API operation.

Note

The promotion process takes a few minutes to complete. When you promote a Read Replica, replication is stopped and the Read Replica is rebooted. When the reboot is complete, the Read Replica is available as a new DB instance.

4. (Optional) Modify the new DB instance to be a Multi-AZ deployment. For more information, see [Modifying an Amazon RDS DB Instance \(p. 115\)](#) and [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#).

AWS Management Console

To promote a Read Replica to a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the Amazon RDS console, choose **Databases**.

The **Databases** pane appears. Each Read Replica shows **Replica** in the **Role** column.

3. Choose the Read Replica that you want to promote.
4. For **Actions**, choose **Promote read replica**.
5. On the **Promote Read Replica** page, enter the backup retention period and the backup window for the new promoted DB instance.

6. When the settings are as you want them, choose **Continue**.
7. On the acknowledgment page, choose **Promote Read Replica**.

CLI

To promote a Read Replica to a DB instance, use the AWS CLI [promote-read-replica](#) command.

Example

For Linux, OS X, or Unix:

```
aws rds promote-read-replica \
--db-instance-identifier myreadreplica
```

For Windows:

```
aws rds promote-read-replica ^
--db-instance-identifier myreadreplica
```

API

To promote a Read Replica to a DB instance, call [PromoteReadReplica](#).

```
https://rds.amazonaws.com/
?Action=PromoteReadReplica
&DBInstanceIdentifier=myreadreplica
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2012-01-20T22%3A06%3A23.624Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Creating a Read Replica in a Different AWS Region

With Amazon RDS, you can create a MySQL, PostgreSQL, or MariaDB Read Replica in a different AWS Region than the source DB instance. You create a Read Replica to do the following:

- Improve your disaster recovery capabilities.
- Scale read operations into an AWS Region closer to your users.
- Make it easier to migrate from a data center in one AWS Region to a data center in another AWS Region.

Creating a MySQL, PostgreSQL, or MariaDB Read Replica in a different AWS Region than the source instance is similar to creating a replica in the same AWS Region. To create a Read Replica across regions, you can use the AWS Management Console, run the [create-db-instance-read-replica](#) command, or call the [CreateDBInstanceReadReplica](#) API action.

To create an encrypted Read Replica in a different AWS Region than the source DB instance, the source DB instance must be encrypted.

Following, you can find information on how to create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance in a different AWS Region.

AWS Management Console

You can create a Read Replica across regions using the AWS Management Console.

To create a Read Replica across regions with the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the MySQL, MariaDB, or PostgreSQL DB instance that you want to use as the source for a Read Replica, and then choose **Create read replica** for **Actions**. To create an encrypted Read Replica, the source DB instance must be encrypted. To learn more about encrypting the source DB instance, see [Encrypting Amazon RDS Resources \(p. 389\)](#).
4. Choose the instance specifications you want to use. We recommend that you use the same DB instance class and storage type for the Read Replica.
5. Choose the other settings you want to use:
 - For **DB instance identifier**, enter a name for the Read Replica.
 - In the **Network & Security** section, choose a value for **Designation region** and **Designation DB subnet group**.
 - To create an encrypted Read Replica in another AWS Region, choose **Enable Encryption**, and then choose the **Master key**. For the **Master key**, choose the AWS Key Management Service (AWS KMS) key identifier of the destination AWS Region.
 - Choose the other settings that you want to use.
6. Choose **Create read replica**.

CLI

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance in a different AWS Region, you can use the `create-db-instance-read-replica` command. In this case, you use `create-db-instance-read-replica` from the AWS Region where you want the Read Replica and specify the Amazon Resource Name (ARN) for the source DB instance. An ARN uniquely identifies a resource created in Amazon Web Services.

For example, if your source DB instance is in the US East (N. Virginia) region, the ARN looks similar to the following.

```
arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
```

For information about ARNs, see [Working with Amazon Resource Names \(ARNs\) in Amazon RDS \(p. 181\)](#).

To create an encrypted Read Replica in a different AWS Region than the source DB instance, you can use the AWS CLI `create-db-instance-read-replica` command from the destination AWS Region. The following parameters are used to create an encrypted Read Replica in another AWS Region:

- `--source-region` — The AWS Region that the encrypted Read Replica is created in. If `source-region` is not specified, you must specify a `pre-signed-url`. A `pre-signed-url` is a URL that contains a Signature Version 4 signed request for the `CreateDBInstanceReadReplica` action that is called in the source AWS Region where the Read Replica is created from. To learn more about the `pre-signed-url`, see [CreateDBInstanceReadReplica](#).
- `--source-db-instance-identifier` — The DB instance identifier for the encrypted Read Replica that is created. This identifier must be in the ARN format for the source AWS Region. The AWS Region specified in `source-db-instance-identifier` must match the AWS Region specified as the `source-region`.

- **--db-instance-identifier** — The identifier for the encrypted Read Replica in the destination AWS Region.
- **--kms-key-id** — The AWS KMS key identifier for the key to use to encrypt the Read Replica in the destination AWS Region.

The following code creates a Read Replica in the `us-west-2` region.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier DBInstanceIdentifier \
    --region us-west-2 \
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
```

For Windows:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier DBInstanceIdentifier ^
    --region us-west-2 ^
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
```

The following code creates a Read Replica in a different AWS Region than the source DB instance. The AWS Region where you call the `create-db-instance-read-replica` command is the destination AWS Region for the encrypted Read Replica.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier DBInstanceIdentifier \
    --region us-west-2 \
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
    \
    --source-region us-east-1 \
    --kms-key-id my-us-east-1-key
```

For Windows:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier DBInstanceIdentifier ^
    --region us-west-2 ^
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
    ^
    --source-region us-east-1 ^
    --kms-key-id my-us-east-1-key
```

API

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance in a different AWS Region, you can call the Amazon RDS API function [CreateDBInstanceReadReplica](#). In this case, you call [CreateDBInstanceReadReplica](#) from the AWS Region where you want the Read Replica and specify the

Amazon Resource Name (ARN) for the source DB instance. An ARN uniquely identifies a resource created in Amazon Web Services.

To create an encrypted Read Replica in a different AWS Region than the source DB instance, you can use the Amazon RDS API [CreateDBInstanceReadReplica](#) action from the destination AWS Region. To create an encrypted Read Replica in another AWS Region, you must specify a value for `PreSignedURL`. `PreSignedURL` should contain a request for the [CreateDBInstanceReadReplica](#) action to call in the source AWS Region where the Read Replica is created in. To learn more about `PreSignedUrl`, see [CreateDBInstanceReadReplica](#).

For example, if your source DB instance is in the US East (N. Virginia) region, the ARN looks similar to the following.

```
arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
```

For information about ARNs, see [Working with Amazon Resource Names \(ARNs\) in Amazon RDS \(p. 181\)](#).

Example

```
https://us-west-2.rds.amazonaws.com/
    ?Action=CreateDBInstanceReadReplica
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
        %253FAction%253D CreateDBInstanceReadReplica
        %2526DestinationRegion%253Dus-east-1
        %2526KmsKeyId%253Dmy-us-east-1-key
        %2526SourceDBInstanceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253Adb%25253Amy-mysql-instance
        %2526SignatureMethod%253DHmacSHA256
        %2526SignatureVersion%253D4%2526SourceDBInstanceIdentifier%253Darn%25253Aaws
%25253Ards%25253Aus-west-2%25253A123456789012%25253Ainstance%25253Amysql-instance1-
instance-20161115
        %2526Version%253D2014-10-31
        %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
        %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
        %2526X-Amz-Date%253D20161117T215409Z
        %2526X-Amz-Expires%253D3600
        %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
        %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &DBInstanceIdentifier=myreadreplica
    &SourceDBInstanceIdentifier=arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
    &Version=2012-01-15
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2012-01-20T22%3A06%3A23.624Z
    &AWSAccessKeyId=<AWS Access Key ID>
    &Signature=<Signature>
```

Cross-Region Replication Considerations

All of the considerations for performing replication within an AWS Region apply to cross-region replication. The following extra considerations apply when replicating between regions:

- You can only replicate between regions when using Amazon RDS DB instances of MariaDB, PostgreSQL (versions 9.4.7 and 9.5.2 and later), or MySQL 5.6 and later.
- A source DB instance can have cross-region Read Replicas in multiple regions.

- You can only create a cross-region Amazon RDS Read Replica from a source Amazon RDS DB instance that is not a Read Replica of another Amazon RDS DB instance.
- You can't set up a replication channel into or out of the AWS GovCloud (US-West) region.
- You can expect to see a higher level of lag time for any Read Replica that is in a different AWS Region than the source instance, due to the longer network channels between regional data centers.
- Within an AWS Region, all cross-region Read Replicas created from the same source DB instance must either be in the same Amazon VPC or be outside of a VPC. For cross-region Read Replicas, any of the create Read Replica commands that specify the `--db-subnet-group-name` parameter must specify a DB subnet group from the same VPC.
- You can create a cross-region Read Replica in a VPC from a source DB instance that is in a VPC in another region. You can also create a cross-region Read Replica in a VPC from a source DB instance that is not in a VPC. You can also create a cross-region Read Replica that is not in a VPC from a source DB instance that is in a VPC.
- Due to the limit on the number of access control list (ACL) entries for a VPC, we can't guarantee more than five cross-region Read Replica instances.

Cross-Region Replication Costs

The data transferred for cross-region replication incurs Amazon RDS data transfer charges. These cross-region replication actions generate charges for the data transferred out of the source AWS Region:

- When you create a Read Replica, Amazon RDS takes a snapshot of the source instance and transfers the snapshot to the Read Replica region.
- For each data modification made in the source databases, Amazon RDS transfers data from the source AWS Region to the Read Replica region.

For more information about data transfer pricing, see [Amazon RDS Pricing](#).

For MySQL and MariaDB instances, you can reduce your data transfer costs by reducing the number of cross-region Read Replicas that you create. For example, suppose that you have a source DB instance in one AWS Region and want to have three Read Replicas in another AWS Region. In this case, you create only one of the Read Replicas from the source DB instance. You create the other two replicas from the first Read Replica instead of the source DB instance.

For example, if you have `source-instance-1` in one AWS Region, you can do the following:

- Create `read-replica-1` in the new AWS Region, specifying `source-instance-1` as the source.
- Create `read-replica-2` from `read-replica-1`.
- Create `read-replica-3` from `read-replica-1`.

In this example, you are only charged for the data transferred from `source-instance-1` to `read-replica-1`. You are not charged for the data transferred from `read-replica-1` to the other two replicas because they are all in the same AWS Region. If you create all three replicas directly from `source-instance-1`, you are charged for the data transfers to all three replicas.

How Amazon RDS Does Cross-Region Replication

Amazon RDS uses the following process to create a cross-region Read Replica. Depending on the regions involved and the amount of data in the databases, this process can take hours to complete. You can use this information to determine how far the process has proceeded when you create a cross-region Read Replica:

1. Amazon RDS begins configuring the source DB instance as a replication source and sets the status to *modifying*.

2. Amazon RDS begins setting up the specified Read Replica in the destination AWS Region and sets the status to *creating*.
3. Amazon RDS creates an automated DB snapshot of the source DB instance in the source AWS Region. The format of the DB snapshot name is `rds:<InstanceID>-<timestamp>`, where `<InstanceID>` is the identifier of the source instance, and `<timestamp>` is the date and time the copy started. For example, `rds:mysourceinstance-2013-11-14-09-24` was created from the instance `mysourceinstance` at `2013-11-14-09-24`. During the creation of an automated DB snapshot, the source DB instance status remains *modifying*, the Read Replica status remains *creating*, and the DB snapshot status is *creating*. The progress column of the DB snapshot page in the console reports how far the DB snapshot creation has progressed. When the DB snapshot is complete, the status of both the DB snapshot and source DB instance are set to *available*.
4. Amazon RDS begins a cross-region snapshot copy for the initial data transfer. The snapshot copy is listed as an automated snapshot in the destination AWS Region with a status of *creating*. It has the same name as the source DB snapshot. The progress column of the DB snapshot display indicates how far the copy has progressed. When the copy is complete, the status of the DB snapshot copy is set to *available*.
5. Amazon RDS then uses the copied DB snapshot for the initial data load on the Read Replica. During this phase, the Read Replica is in the list of DB instances in the destination, with a status of *creating*. When the load is complete, the Read Replica status is set to *available*, and the DB snapshot copy is deleted.
6. When the Read Replica reaches the available status, Amazon RDS starts by replicating the changes made to the source instance since the start of the create Read Replica operation. During this phase, the replication lag time for the Read Replica is greater than 0.

For MySQL, MariaDB, and PostgreSQL Read Replicas, you can monitor replication lag in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric. For MySQL and MariaDB, the `ReplicaLag` metric reports the value of the `Seconds_Behind_Master` field of the `SHOW SLAVE STATUS` command. For PostgreSQL, the `ReplicaLag` metric reports the value of `SELECT extract(epoch from now() - pg_last_xact_replay_timestamp()) AS slave_lag`.

Common causes for replication lag for MySQL and MariaDB are the following:

- A network outage.
- Writing to tables with indexes on a Read Replica. If the `read_only` parameter is not set to 0 on the Read Replica, it can break replication.
- Using a non-transactional storage engine such as MyISAM. Replication is only supported for the InnoDB storage engine on MySQL and the XtraDB storage engine on MariaDB.

When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance. If the `ReplicaLag` metric returns -1, then replication is currently not active. `ReplicaLag = -1` is equivalent to `Seconds_Behind_Master = NULL`.

PostgreSQL (versions 9.4.7 and 9.5.2 and later) uses physical replication slots to manage Write Ahead Log (WAL) retention on the source instance. For each cross-region Read Replica instance, Amazon RDS creates a physical replication slot and associates it with the instance. Two Amazon CloudWatch metrics, `Oldest Replication Slot Lag` and `Transaction Logs Disk Usage`, show how far behind the most lagging replica is in terms of WAL data received and how much storage is being used for WAL data. The `Transaction Logs Disk Usage` value can substantially increase when a cross-region Read Replica is lagging significantly.

Cross-Region Replication Examples

Example Create a Cross-Region Read Replica Outside of Any VPC

The following example creates a Read Replica in us-west-2 from a source DB instance in us-east-1. The Read Replica is created outside of a VPC:

For Linux, OS X, or Unix:

```
aws rds create-db-instance-read-replica \
--db-instance-identifier SimCoProd01Replica01 \
--region us-west-2
--source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:SimcoProd01
```

For Windows:

```
aws rds create-db-instance-read-replica ^
--db-instance-identifier SimCoProd01Replica01 ^
--region us-west-2
--source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:SimcoProd01
```

Example Create Cross-Region Read Replica in a VPC

This example creates a Read Replica in us-west-2 from a source DB instance in us-east-1. The Read Replica is created in the VPC associated with the specified DB subnet group:

For Linux, OS X, or Unix:

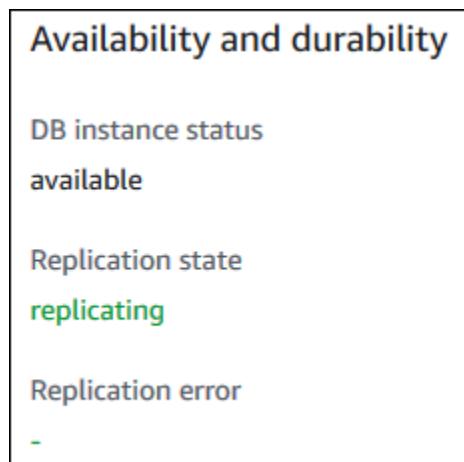
```
aws rds create-db-instance-read-replica \
--db-instance-identifier SimCoProd01Replica01 \
--region us-west-2
--db-subnet-group-name my-us-west-2-subnet
--source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:SimcoProd01
```

For Windows:

```
aws rds create-db-instance-read-replica ^
--db-instance-identifier SimCoProd01Replica01 ^
--region us-west-2
--db-subnet-group-name my-us-west-2-subnet
--source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:SimcoProd01
```

Monitoring Read Replication

You can monitor the status of a Read Replica in several ways. The Amazon RDS console shows the status of a Read Replica in the **Availability and durability** section of the Read Replica details. To view the details for a Read Replica, click the name of the Read Replica in the list of instances in the Amazon RDS console.



You can also see the status of a Read Replica using the AWS CLI `describe-db-instances` command or the Amazon RDS API `DescribeDBInstances` action.

The status of a Read Replica can be one of the following:

- **replicating**—The Read Replica is replicating successfully.
- **error**—An error has occurred with the replication. Check the **Replication Error** field in the Amazon RDS console or the event log to determine the exact error. For more information about troubleshooting a replication error, see [Troubleshooting a MySQL Read Replica Problem \(p. 661\)](#).
- **terminated**—Replication is terminated. This occurs if replication is stopped for more than thirty consecutive days, either manually or due to a replication error. In this case, Amazon RDS terminates replication between the master DB instance and all Read Replicas in order to prevent increased storage requirements on the master DB instance and long failover times.

Broken replication can affect storage because the logs can grow in size and number due to the high volume of errors messages being written to the log. Broken replication can also affect failure recovery due to the time Amazon RDS requires to maintain and process the large number of logs during recovery.

- **stopped (MySQL or MariaDB only)**—Replication has stopped because of a customer initiated request.
- **replication stop point set (MySQL only)**—A customer initiated stop point was set using the [mysql.rds_start_replication_until \(p. 704\)](#) stored procedure and the replication is in progress.
- **replication stop point reached (MySQL only)**—A customer initiated stop point was set using the [mysql.rds_start_replication_until \(p. 704\)](#) stored procedure and replication is stopped because the stop point was reached.

Working with Option Groups

Some DB engines offer additional features that make it easier to manage data and databases, and to provide additional security for your database. Amazon RDS uses option groups to enable and configure these features. An *option group* can specify features, called options, that are available for a particular Amazon RDS DB instance. Options can have settings that specify how the option works. When you associate a DB instance with an option group, the specified options and option settings are enabled for that DB instance.

Amazon RDS supports options for the following database engines:

Database Engine	Relevant Documentation
MariaDB	Options for MariaDB Database Engine (p. 477)
Microsoft SQL Server	Options for the Microsoft SQL Server Database Engine (p. 559)
MySQL	Options for MySQL DB Instances (p. 677)
Oracle	Options for Oracle DB Instances (p. 787)

Option Groups Overview

Amazon RDS provides an empty default option group for each new DB instance. You cannot modify this default option group, but any new option group that you create derives its settings from the default option group. To apply an option to a DB instance, you must do the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add one or more options to the option group.
3. Associate the option group with the DB instance.

Both DB instances and DB snapshots can be associated with an option group. When you restore from a DB snapshot or perform a point-in-time restore for a DB instance, the option group associated with the DB snapshot or DB instance will, by default, be associated with the restored DB instance. You can associate a different option group with a restored DB instance. However, the new option group must contain any persistent or permanent options that were included in the original option group. Persistent and permanent options are described following.

Options require additional memory to run on a DB instance, so you might need to launch a larger instance to use them, depending on your current use of your DB instance. For example, Oracle Enterprise Manager Database Control uses about 300 MB of RAM; if you enable this option for a small DB instance, you might encounter performance problems or out-of-memory errors.

Persistent and Permanent Options

Two types of options, persistent and permanent, require special consideration when you add them to an option group.

Persistent options, such as the TDE option for Microsoft SQL Server transparent data encryption (TDE), cannot be removed from an option group while DB instances are associated with the option group. You must disassociate all DB instances from the option group before a persistent option can be removed from the option group. When you restore or perform a point-in-time restore from a DB snapshot, if the option group associated with that DB snapshot contains a persistent option, you can only associate the restored DB instance with that option group.

Permanent options, such as the TDE option for Oracle Advanced Security TDE, can never be removed from an option group, and the option group cannot be disassociated from the DB instance. When you restore or perform a point-in-time restore from a DB snapshot, if the option group associated with that DB snapshot contains a permanent option, you can only associate the restored DB instance with an option group with that permanent option.

VPC and Platform Considerations

When an option group is assigned to a DB instance, it is linked to the platform that the DB instance is on. That platform can either be a VPC supported by the Amazon Virtual Private Cloud (Amazon VPC) service, or EC2-Classic (non-VPC) supported by the Amazon Elastic Compute Cloud (Amazon EC2) service. For details on these two platforms, see [Amazon EC2 and Amazon Virtual Private Cloud](#).

If a DB instance is in a VPC, the option group associated with the instance is linked to that VPC. This means that you cannot use the option group assigned to a DB instance if you attempt to restore the instance into a different VPC or onto a different platform. If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the DB instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. Note that with persistent or permanent options, such as Oracle TDE, you must create a new option group that includes the persistent or permanent option when restoring a DB instance into a different VPC.

Option settings control the behavior of an option. For example, the Oracle Advanced Security option `NATIVE_NETWORK_ENCRYPTION` has a setting that you can use to specify the encryption algorithm for network traffic to and from the DB instance. Some options settings are optimized for use with Amazon RDS and cannot be changed.

Mutually Exclusive Options

Some options are mutually exclusive. You can use one or the other, but not both at the same time. The following options are mutually exclusive:

- [Oracle Enterprise Manager Database Express \(p. 796\)](#) and [Oracle Management Agent for Enterprise Manager Cloud Control \(p. 799\)](#).
- [Oracle Native Network Encryption \(p. 815\)](#) and [Oracle Secure Sockets Layer \(p. 817\)](#).
- [Oracle Transparent Data Encryption \(p. 836\)](#) and [Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys \(p. 886\)](#).

Creating an Option Group

You can create a new option group that derives its settings from the default option group, and then add one or more options to the new option group. Alternatively, if you already have an existing option group, you can copy that option group with all of its options to a new option group. For more information, see [Making a Copy of an Option Group \(p. 159\)](#).

After you create a new option group, it has no options. To learn how to add options to the option group, see [Adding an Option to an Option Group \(p. 160\)](#). After you have added the options you want, you can then associate the option group with a DB instance so that the options become available on the DB instance. For information about associating an option group with a DB instance, see the documentation for your specific engine listed at [Working with Option Groups \(p. 156\)](#).

AWS Management Console

One way of creating an option group is by using the AWS Management Console.

To create a new option group by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Option groups**.
3. Choose **Create group**.
4. In the **Create option group** window, do the following:
 - a. For **Name**, type a name for the option group that is unique within your AWS account. The name can contain only letters, digits, and hyphens.
 - b. For **Description**, type a brief description of the option group. The description is used for display purposes.
 - c. For **Engine**, choose the DB engine that you want.
 - d. For **Major engine version**, choose the major version of the DB engine that you want.
5. To continue, choose **Create**. To cancel the operation instead, choose **Cancel**.

CLI

To create an option group, use the AWS CLI `create-option-group` command with the following required parameters.

- `--option-group-name`
- `--engine-name`
- `--major-engine-version`
- `--option-group-description`

Example

The following example creates an option group named `testoptiongroup`, which is associated with the Oracle Enterprise Edition DB engine. The description is enclosed in quotation marks.

For Linux, OS X, or Unix:

```
aws rds create-option-group \
--option-group-name testoptiongroup \
--engine-name oracle-ee \
--major-engine-version 12.1 \
--option-group-description "Test option group"
```

For Windows:

```
aws rds create-option-group ^
--option-group-name testoptiongroup ^
--engine-name oracle-ee ^
--major-engine-version 12.1 ^
--option-group-description "Test option group"
```

API

To create an option group, call the Amazon RDS API [CreateOptionGroup](#) action. Include the following parameters:

- `OptionGroupName`
- `EngineName`
- `MajorEngineVersion`
- `OptionGroupDescription`

Making a Copy of an Option Group

You can use the AWS CLI or the Amazon RDS API to make a copy of an option group. Copying an option group is a convenient solution when you have already created an option group and you want to include most of the custom parameters and values from that group in a new option group. You can also make a copy of an option group that you use in production and then modify the copy to test other option settings.

CLI

To copy an option group, use the AWS CLI [copy-option-group](#) command. Include the following required parameters:

- `--source-option-group-identifier`
- `--target-option-group-identifier`
- `--target-option-group-description`

Example

The following example creates an option group named `new-local-option-group`, which is a local copy of the option group `my-remote-option-group`.

For Linux, OS X, or Unix:

```
aws rds copy-option-group \
  --source-option-group-identifier arn:aws:rds:us-west-2:123456789012:og:my-remote-
option-group \
  --target-option-group-identifier new-local-option-group \
  --target-option-group-description "Option group 2"
```

For Windows:

```
aws rds copy-option-group ^
  --source-option-group-identifier arn:aws:rds:us-west-2:123456789012:og:my-remote-
option-group ^
  --target-option-group-identifier new-local-option-group ^
  --target-option-group-description "Option group 2"
```

API

To copy an option group, call the Amazon RDS API [CopyOptionGroup](#) action. Include the following required parameters.

- `SourceOptionGroupIdentifier`

- TargetOptionGroupIdentifier
- TargetOptionGroupDescription

Adding an Option to an Option Group

You can add an option to an existing option group. After you have added the options you want, you can then associate the option group with a DB instance so that the options become available on the DB instance. For information about associating an option group with a DB instance, see the documentation for your specific DB engine listed at [Working with Option Groups \(p. 156\)](#).

Option group changes must be applied immediately in two cases:

- When you add an option that adds or updates a port value, such as the OEM option.
- When you add or remove an option group with an option that includes a port value.

In these cases, you must select the **Apply Immediately** option in the console, or include the `Apply-Immediately` option when using the AWS CLI or set the `Apply-Immediately` parameter to `true` when using the Amazon RDS API. Options that don't include port values can be applied immediately, or can be applied during the next maintenance window for the DB instance.

AWS Management Console

You can use the AWS Management Console to add an option to an option group.

To add an option to an option group by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Option groups**.
3. Select the option group that you want to modify, and then choose **Add Option**.

Option groups (9)		
<input type="text"/> Filter subnet groups		
	Name	Description
<input type="checkbox"/>	carpcmysql	carpcmysql
<input checked="" type="checkbox"/>	carporacle	carporacle
<input type="checkbox"/>	default:mysql-5-5	Default option group for mysql 5.5
<input type="checkbox"/>	default:mysql-5-6	Default option group for mysql 5.6
<input type="checkbox"/>	default:mysql-5-7	Default option group for mysql 5.7

4. In the **Add option** window, do the following:
 - a. Choose the option that you want to add. You might need to provide additional values, depending on the option that you select. For example, when you choose the OEM option, you must also type a port value and specify a DB security group.
 - b. To enable the option on all associated DB instances as soon as you add it, for **Apply Immediately**, choose **Yes**. If you choose **No** (the default), the option is enabled for each associated DB instance during its next maintenance window.

Add Option

Option details

Option group name
carpcoracle

Option
Name of Option you want to add to this group

Port
The port number, if applicable, to use when connecting to the Option

Security Groups
A list of VPC or DB Security Groups for which this Option is enabled

default

Apply Immediately [info](#)
 Yes
 No

- When the settings are as you want them, choose **Add Option**.

CLI

To add an option to an option group, run the AWS CLI [add-option-to-option-group](#) command with the option that you want to add. To enable the new option immediately on all associated DB instances, include the `--apply-immediately` parameter. By default, the option is enabled for each associated DB instance during its next maintenance window. Include the following required parameter:

- `--option-group-name`

Example

The following example adds the Oracle Enterprise Manager Database Control (OEM) option to an option group named `testoptiongroup` and immediately enables it. Note that even if you use the `default` security group, you must specify that security group.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
```

```
--option-group-name testoptiongroup \
--options OptionName=OEM,Port=5500,DBSecurityGroupMemberships=default \
--apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group ^
--option-group-name testoptiongroup ^
--options OptionName=OEM,Port=5500,DBSecurityGroupMemberships=default ^
--apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP  False  oracle-ee  12.1  arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
Test Option Group  testoptiongroup default
OPTIONS Oracle 12c EM Express  OEM      False    False   5500
DBSECURITYGROUPEMEMBERSHIPS  default authorized
```

Example

The following example adds the Oracle OEM option to an option group, specifies a custom port, and specifies a pair of Amazon EC2 VPC security groups to use for that port.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
--option-group-name testoptiongroup \
--options OptionName=OEM,Port=5500,VpcSecurityGroupMemberships="sg-test1,sg-test2" \
--apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group ^
--option-group-name testoptiongroup ^
--options OptionName=OEM,Port=5500,VpcSecurityGroupMemberships="sg-test1,sg-test2" ^
--apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP  False  oracle-ee  12.1  arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
Test Option Group  testoptiongroup vpc-test
OPTIONS Oracle 12c EM Express  OEM      False    False   5500
VPCSECURITYGROUPEMEMBERSHIPS  active  sg-test1
VPCSECURITYGROUPEMEMBERSHIPS  active  sg-test2
```

Example

The following example adds the Oracle option NATIVE_NETWORK_ENCRYPTION to an option group and specifies the option settings. If no option settings are specified, default values are used.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
--option-group-name testoptiongroup \
--options '[{"OptionSettings": [{"Name": "SQLNET.ENCRYPTION_SERVER", "Value": "REQUIRED"}, {"Name": "SQLNET.ENCRYPTION_TYPES_SERVER", "Value": "AES256,AES192,DES"}]}, {"OptionName": "NATIVE_NETWORK_ENCRYPTION", "Value": "AES256,AES192,DES"}]' \
--apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group ^
--option-group-name testoptiongroup ^
--options "OptionSettings=[{"Name": "SQLNET.ENCRYPTION_SERVER", "Value": "REQUIRED"}, {"Name": "SQLNET.ENCRYPTION_TYPES_SERVER", "Value": "AES256\,AES192\,DES"}], "OptionName": "NATIVE_NETWORK_ENCRYPTION", "Value": "AES256\,AES192\,DES"}]" ^
--apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP False oracle-ee 12.1 arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
Test Option Group testoptiongroup
OPTIONS Oracle Advanced Security - Native Network Encryption      NATIVE_NETWORK_ENCRYPTION
      False False
OPTIONSETTINGS
RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40
      STATIC STRING
RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40      Specifies
list of encryption algorithms in order of intended use
      True True SQLNET.ENCRYPTION_TYPES_SERVER AES256,AES192,DES
OPTIONSETTINGS ACCEPTED,REJECTED,REQUESTED,REQUIRED      STATIC STRING REQUESTED
      Specifies the desired encryption behavior False True SQLNET.ENCRYPTION_SERVER
REQUIRED
OPTIONSETTINGS SHA1,MD5      STATIC STRING SHA1,MD5      Specifies list of checksumming
algorithms in order of intended use True True SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER
SHA1,MD5
```

API

To add an option to an option group using the Amazon RDS API, call the [ModifyOptionGroup](#) action with the option that you want to add. To enable the new option immediately on all associated DB instances, include the `ApplyImmediately` parameter and set it to `true`. By default, the option is enabled for each associated DB instance during its next maintenance window. Include the following required parameter:

- `OptionGroupName`

Listing the Options and Option Settings for an Option Group

You can list all the options and option settings for an option group.

AWS Management Console

You can use the AWS Management Console to list all of the options and option settings for an option group.

To list the options and option settings for an option group

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Option groups**. The **Options** column in the table shows the options and option settings in the option group.

CLI

To list the options and option settings for an option group, use the AWS CLI [describe-option-groups](#) command. Specify the name of the option group whose options and settings you want to view. If you don't specify an option group name, all option groups are described.

Example

The following example lists the options and option settings for all option groups.

```
aws rds describe-option-groups
```

Example

The following example lists the options and option settings for an option group named `testoptiongroup`.

```
aws rds describe-option-groups --option-group-name testoptiongroup
```

API

To list the options and option settings for an option group, use the Amazon RDS API [DescribeOptionGroups](#) action. Specify the name of the option group whose options and settings you want to view. If you don't specify an option group name, all option groups are described.

Modifying an Option Setting

After you have added an option that has modifiable option settings, you can modify the settings at any time. If you change options or option settings in an option group, those changes are applied to all DB instances that are associated with that option group. For more information on what settings are available for the various options, see the documentation for your specific engine listed at [Working with Option Groups \(p. 156\)](#).

Option group changes must be applied immediately in two cases:

- When you add an option that adds or updates a port value, such as the `OEM` option.
- When you add or remove an option group with an option that includes a port value.

In these cases, you must select the **Apply Immediately** option in the console, or include the `Apply-Immediately` option when using the AWS CLI or set the `Apply-Immediately` parameter to `true` when using the Amazon RDS API. Options that don't include port values can be applied immediately, or can be applied during the next maintenance window for the DB instance.

AWS Management Console

You can use the AWS Management Console to modify an option setting.

To modify an option setting by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Option groups**.
3. Select the option group whose option that you want to modify, and then choose **Modify option**.
4. In the **Modify option** window, from **Installed Options**, choose the option whose setting you want to modify. Make the changes that you want.
5. To enable the option as soon as you add it, for **Apply Immediately**, choose **Yes**. If you choose **No** (the default), the option is enabled for each associated DB instance during its next maintenance window.
6. When the settings are as you want them, choose **Modify Option**.

CLI

To modify an option setting, use the AWS CLI `add-option-to-option-group` command with the option group and option that you want to modify. By default, the option is enabled for each associated DB instance during its next maintenance window. To apply the change immediately to all associated DB instances, include the `--apply-immediately` parameter. To modify an option setting, use the `--settings` argument.

Example

The following example modifies the port that the Oracle Enterprise Manager Database Control (OEM) uses in an option group named `testoptiongroup` and immediately applies the change.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
--option-group-name testoptiongroup \
--options OptionName=OEM,Port=5432,DBSecurityGroupMemberships=default \
--apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group ^
--option-group-name testoptiongroup ^
--options OptionName=OEM,Port=5432,DBSecurityGroupMemberships=default ^
--apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP  False  oracle-ee  12.1  arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
Test Option Group      testoptiongroup
OPTIONS Oracle 12c EM Express OEM      False    False   5432
DBSECURITYGROUPMEMBERSHIPS  default  authorized
```

Example

The following example modifies the Oracle option NATIVE_NETWORK_ENCRYPTION and changes the option settings.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
--option-group-name testoptiongroup \
--options '[{"OptionSettings": [{"Name": "SQLNET.ENCRYPTION_SERVER", "Value": "REQUIRED"}, {"Name": "SQLNET.ENCRYPTION_TYPES_SERVER", "Value": "AES256,AES192,DES,RC4_256"}]}, {"OptionName": "NATIVE_NETWORK_ENCRYPTION", "Value": "AES256,AES192,DES,RC4_256"}]' --apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group ^
--option-group-name testoptiongroup ^
--options "OptionSettings=[{"Name": "SQLNET.ENCRYPTION_SERVER", "Value": "REQUIRED"}, {"Name": "SQLNET.ENCRYPTION_TYPES_SERVER", "Value": "AES256\,AES192\,DES\,RC4_256"}], "OptionName": "NATIVE_NETWORK_ENCRYPTION" ^
--apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP  False  oracle-ee 12.1  arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
Test Option Group      testoptiongroup
OPTIONS Oracle Advanced Security - Native Network Encryption      NATIVE_NETWORK_ENCRYPTION
      False  False
OPTIONSETTINGS
      RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40  STATIC
      STRING
      RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40
      Specifies list of encryption algorithms in order of intended use
      True  True  SQLNET.ENCRYPTION_TYPES_SERVER  AES256,AES192,DES,RC4_256
OPTIONSETTINGS ACCEPTED,REJECTED,REQUESTED,REQUIRED  STATIC  STRING  REQUESTED
      Specifies the desired encryption behavior  False  True  SQLNET.ENCRYPTION_SERVER
      REQUIRED
OPTIONSETTINGS SHA1,MD5  STATIC  STRING  SHA1,MD5  Specifies list of
      checksumming algorithms in order of intended use  True  True
      SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER  SHA1,MD5
OPTIONSETTINGS ACCEPTED,REJECTED,REQUESTED,REQUIRED  STATIC  STRING
      REQUESTED  Specifies the desired data integrity behavior  False  True
      SQLNET.CRYPTO_CHECKSUM_SERVER  REQUESTED
```

API

To modify an option setting, use the Amazon RDS API [ModifyOptionGroup](#) command with the option group and option that you want to modify. By default, the option is enabled for each associated DB instance during its next maintenance window. To apply the change immediately to all associated DB instances, include the `ApplyImmediately` parameter and set it to `true`.

Removing an Option from an Option Group

Some options can be removed from an option group, and some cannot. A persistent option cannot be removed from an option group until all DB instances associated with that option group are disassociated. A permanent option can never be removed from an option group. For more information about what options are removable, see the documentation for your specific engine listed at [Working with Option Groups \(p. 156\)](#).

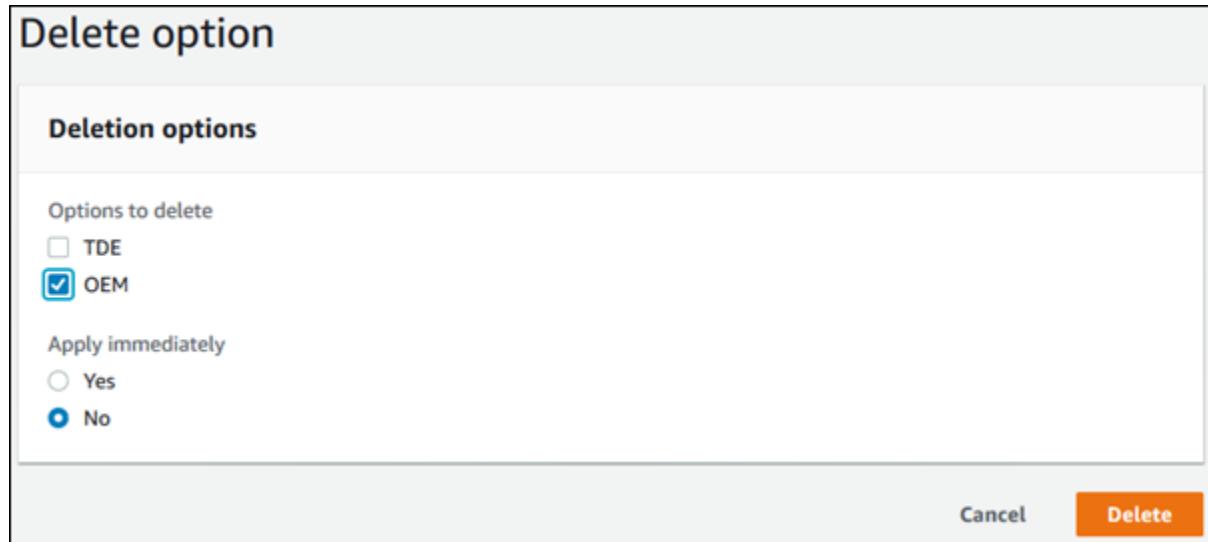
If you remove all options from an option group, Amazon RDS doesn't delete the option group. DB instances that are associated with the empty option group continue to be associated with it; they just won't have any active options. Alternatively, to remove all options from a DB instance, you can associate the DB instance with the default (empty) option group.

AWS Management Console

You can use the AWS Management Console to remove an option from an option group.

To remove an option from an option group by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Option groups**.
3. Select the option group whose option you want to remove, and then choose **Delete option**.
4. In the **Delete option** window, do the following:
 - Select the check box for the option that you want to delete.
 - For the deletion to take effect as soon as you make it, for **Apply immediately**, choose **Yes**. If you choose **No** (the default), the option is deleted for each associated DB instance during its next maintenance window.



5. When the settings are as you want them, choose **Yes, Delete**.

CLI

To remove an option from an option group, use the AWS CLI `remove-option-from-option-group` command with the option that you want to delete. By default, the option is removed from each

associated DB instance during its next maintenance window. To apply the change immediately, include the --apply-immediately parameter.

Example

The following example removes the Oracle Enterprise Manager Database Control (OEM) option from an option group named testoptiongroup and immediately applies the change.

For Linux, OS X, or Unix:

```
aws rds remove-option-from-option-group \
--option-group-name testoptiongroup \
--options OEM \
--apply-immediately
```

For Windows:

```
aws rds remove-option-from-option-group ^
--option-group-name testoptiongroup ^
--options OEM ^
--apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP      testoptiongroup oracle-ee    12.1      Test option group
```

API

To remove an option from an option group, use the Amazon RDS API [ModifyOptionGroup](#) action. By default, the option is removed from each associated DB instance during its next maintenance window. To apply the change immediately, include the `ApplyImmediately` parameter and set it to `true`.

Include the following parameters:

- `OptionGroupName`
- `OptionsToRemove.OptionName`

Working with DB Parameter Groups

You manage your DB engine configuration through the use of parameters in a DB parameter group . DB parameter groups act as a container for engine configuration values that are applied to one or more DB instances.

A default DB parameter group is created if you create a DB instance without specifying a customer-created DB parameter group. Each default DB parameter group contains database engine defaults and Amazon RDS system defaults based on the engine, compute class, and allocated storage of the instance. You cannot modify the parameter settings of a default DB parameter group; you must create your own DB parameter group to change parameter settings from their default value. Note that not all DB engine parameters can be changed in a customer-created DB parameter group.

If you want to use your own DB parameter group, you simply create a new DB parameter group, modify the desired parameters, and modify your DB instance to use the new DB parameter group. All DB instances that are associated with a particular DB parameter group get all parameter updates to that DB parameter group.

You can copy an existing DB parameter group with the AWS CLI [copy-db-parameter-group](#) command. Copying a parameter group is a convenient solution when you have already created a DB parameter group and you want to include most of the custom parameters and values from that group in a new DB parameter group.

Here are some important points you should know about working with parameters in a DB parameter group:

- When you change a dynamic parameter and save the DB parameter group, the change is applied immediately regardless of the **Apply Immediately** setting. When you change a static parameter and save the DB parameter group, the parameter change will take effect after you manually reboot the DB instance. You can reboot a DB instance using the RDS console or explicitly calling the `RebootDBInstance` API action (without failover, if the DB instance is in a Multi-AZ deployment). The requirement to reboot the associated DB instance after a static parameter change helps mitigate the risk of a parameter misconfiguration affecting an API call, such as calling `ModifyDBInstance` to change DB instance class or scale storage.
- When you change the DB parameter group associated with a DB instance, you must manually reboot the instance before the new DB parameter group is used by the DB instance.
- The value for a DB parameter can be specified as an integer or as an integer expression built from formulas, variables, functions, and operators. Functions can include a mathematical log expression. For more information, see [DB Parameter Values \(p. 177\)](#).
- Set any parameters that relate to the character set or collation of your database in your parameter group prior to creating the DB instance and before you create a database in your DB instance. This ensures that the default database and new databases in your DB instance use the character set and collation values that you specify. If you change character set or collation parameters for your DB instance, the parameter changes are not applied to existing databases.

You can change character set or collation values for an existing database using the `ALTER DATABASE` command, for example:

```
ALTER DATABASE database_name CHARACTER SET character_set_name COLLATE collation;
```

- Improperly setting parameters in a DB parameter group can have unintended adverse effects, including degraded performance and system instability. Always exercise caution when modifying database parameters and back up your data before modifying a DB parameter group. You should try out parameter group setting changes on a test DB instance before applying those parameter group changes to a production DB instance.

Topics

- [Creating a DB Parameter Group \(p. 170\)](#)
- [Modifying Parameters in a DB Parameter Group \(p. 171\)](#)
- [Copying a DB Parameter Group \(p. 174\)](#)
- [Listing DB Parameter Groups \(p. 175\)](#)
- [Viewing Parameter Values for a DB Parameter Group \(p. 176\)](#)
- [Comparing DB Parameter Groups \(p. 177\)](#)
- [DB Parameter Values \(p. 177\)](#)

Creating a DB Parameter Group

You can create a new DB parameter group using the AWS Management Console, the AWS CLI, or the RDS API.

AWS Management Console

To create a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Parameter groups**.
3. Choose **Create parameter group**.

The **Create parameter group** window appears.

4. In the **Parameter group family** list, select a DB parameter group family.
5. In the **Type** list, select **DB Parameter Group**.
6. In the **Group name** box, type the name of the new DB parameter group.
7. In the **Description** box, type a description for the new DB parameter group.
8. Choose **Create**.

CLI

To create a DB parameter group, use the AWS CLI `create-db-parameter-group` command. The following example creates a DB parameter group named *mydbparametergroup* for MySQL version 5.6 with a description of "My new parameter group."

Include the following required parameters:

- `--db-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

To list all of the available parameter group families, use the following command:

```
aws rds describe-db-engine-versions --query "DBEngineVersions[ ].DBParameterGroupFamily"
```

Note

The output contains duplicates.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-parameter-group \
--db-parameter-group-name mydbparametergroup \
--db-parameter-group-family MySQL5.6 \
--description "My new parameter group"
```

For Windows:

```
aws rds create-db-parameter-group ^
--db-parameter-group-name mydbparametergroup ^
--db-parameter-group-family MySQL5.6 ^
--description "My new parameter group"
```

This command produces output similar to the following:

```
DBPARAMETERGROUP mydbparametergroup mysql5.6 My new parameter group
```

API

To create a DB parameter group, use the Amazon RDS API [CreateDBParameterGroup](#) action.

Include the following required parameters:

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Modifying Parameters in a DB Parameter Group

You can modify parameter values in a customer-created DB parameter group. You can't change the parameter values in a default DB parameter group. Changes to parameters in a customer-created DB parameter group are applied to all DB instances that are associated with the DB parameter group.

If you change a parameter value, when the change is applied is determined by the type of parameter. Changes to dynamic parameters are applied immediately. Changes to static parameters require that the DB instance associated with DB parameter group be rebooted before the change takes effect. To determine the type of a parameter, list the parameters in a parameter group using one of the procedures shown in the section [Listing DB Parameter Groups \(p. 175\)](#).

The RDS console shows the status of the DB parameter group associated with a DB instance on the **Configuration** tab. For example, if the DB instance is not using the latest changes to its associated DB parameter group, the RDS console shows the DB parameter group with a status of **pending-reboot**. You need to manually reboot the DB instance for the latest parameter changes to take effect for that DB instance.

Connectivity Monitoring Logs & events Configuration

Instance

Configuration

DB instance id
oracle-instance1

Engine version
12.1.0.2.v14

Storage type
General Purpose (SSD)

IOPS
-

Storage
20 GiB

DB name
ORCL

License model
Bring Your Own License

Character set
AL32UTF8

Option groups
default:oracle-ee-12-1

ARN
[REDACTED]

Resource id
[REDACTED]

Created time
Fri Nov 30 2018 15:41:20 GMT-0800 (Pacific Standard Time)

Parameter group
[testsqnet \(pending-reboot\)](#)

Deletion protection
Disabled

AWS Management Console

To modify a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Parameter groups**.
3. In the list, choose the parameter group that you want to modify.
4. For **Parameter group actions**, choose **Edit**.
5. Change the values of the parameters that you want to modify. You can scroll through the parameters using the arrow keys at the top right of the dialog box.

You can't change values in a default parameter group.
6. Choose **Save changes**.

CLI

To modify a DB parameter group, use the AWS CLI `modify-db-parameter-group` command with the following required parameters:

- `--db-parameter-group-name`
- `--parameters`

The following example modifies the `max_connections` and `max_allowed_packet` values in the DB parameter group named `mydbparametergroup`.

Note

Amazon RDS does not support passing multiple comma-delimited parameter values for a single parameter.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name mydbparametergroup \
  --parameters "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" \
  "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

For Windows:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name mydbparametergroup ^
  --parameters "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" ^
  "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

The command produces output like the following:

```
DBPARAMETERGROUP  mydbparametergroup
```

API

To modify a DB parameter group, use the Amazon RDS API `ModifyDBParameterGroup` command with the following required parameters:

- `DBParameterGroupName`
- `Parameters`

Copying a DB Parameter Group

You can copy custom DB parameter groups that you create. Copying a parameter group is a convenient solution when you have already created a DB parameter group and you want to include most of the custom parameters and values from that group in a new DB parameter group. You can copy a DB parameter group by using the AWS CLI [copy-db-parameter-group](#) command or the Amazon RDS API [CopyDBParameterGroup](#) action.

After you copy a DB parameter group, you should wait at least 5 minutes before creating your first DB instance that uses that DB parameter group as the default parameter group. This allows Amazon RDS to fully complete the copy action before the parameter group is used as the default for a new DB instance. This is especially important for parameters that are critical when creating the default database for a DB instance, such as the character set for the default database defined by the `character_set_database` parameter. You can use the **Parameter Groups** option of the [Amazon RDS console](#) or the [describe-db-parameters](#) command to verify that your DB parameter group has been created.

Note

You can't copy a default parameter group. However, you can create a new parameter group that is based on a default parameter group.

AWS Management Console

To copy a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Parameter groups**.
3. In the list, choose the custom parameter group that you want to copy.
4. For **Parameter group actions**, choose **Copy**.
5. In **New DB parameter group identifier**, enter a name for the new parameter group.
6. In **Description**, enter a description for the new parameter group.
7. Choose **Copy**.

CLI

To copy a DB parameter group, use the AWS CLI [copy-db-parameter-group](#) command with the following required parameters:

- `--source-db-parameter-group-identifier`
- `--target-db-parameter-group-identifier`
- `--target-db-parameter-group-description`

The following example creates a new DB parameter group named `mygroup2` that is a copy of the DB parameter group `mygroup1`.

Example

For Linux, OS X, or Unix:

```
aws rds copy-db-parameter-group \
```

```
--source-db-parameter-group-identifier mygroup1 \
--target-db-parameter-group-identifier mygroup2 \
--target-db-parameter-group-description "DB parameter group 2"
```

For Windows:

```
aws rds copy-db-parameter-group ^
--source-db-parameter-group-identifier mygroup1 ^
--target-db-parameter-group-identifier mygroup2 ^
--target-db-parameter-group-description "DB parameter group 2"
```

API

To copy a DB parameter group, use the RDS API [CopyDBParameterGroup](#) action with the following required parameters:

- `SourceDBParameterGroupIdentifier`
- `TargetDBParameterGroupIdentifier`
- `TargetDBParameterGroupDescription`

Listing DB Parameter Groups

You can list the DB parameter groups you've created for your AWS account.

Note

Default parameter groups are automatically created from a default parameter template when you create a DB instance for a particular DB engine and version. These default parameter groups contain preferred parameter settings and cannot be modified. When you create a custom parameter group, you can modify parameter settings.

AWS Management Console

To list all DB parameter groups for an AWS account

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Parameter groups**.

The DB parameter groups appear in a list.

CLI

To list all DB parameter groups for an AWS account, use the AWS CLI [describe-db-parameter-groups](#) command.

Example

The following example lists all available DB parameter groups for an AWS account.

```
aws rds describe-db-parameter-groups
```

The command returns a response like the following:

```
DBPARAMETERGROUP  default.mysql5.5      mysql5.5  Default parameter group for MySQL5.5
```

```
DBPARAMETERGROUP default.mysql5.6      mysql5.6  Default parameter group for MySQL5.6
DBPARAMETERGROUP mydbparametergroup     mysql5.6  My new parameter group
```

The following example describes the *mydbparamgroup1* parameter group.

For Linux, OS X, or Unix:

```
aws rds describe-db-parameter-groups \
--db-parameter-group-name mydbparamgroup1
```

For Windows:

```
aws rds describe-db-parameter-groups ^
--db-parameter-group-name mydbparamgroup1
```

The command returns a response like the following:

```
DBPARAMETERGROUP mydbparametergroup1 mysql5.5  My new parameter group
```

API

To list all DB parameter groups for an AWS account, use the RDS API [DescribeDBParameterGroups](#) action.

Viewing Parameter Values for a DB Parameter Group

You can get a list of all parameters in a DB parameter group and their values.

AWS Management Console

To view the parameter values for a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Parameter groups**.
The DB parameter groups appear in a list.
3. Choose the name of the parameter group to see its list of parameters.

CLI

To view the parameter values for a DB parameter group, use the AWS CLI [describe-db-parameters](#) command with the following required parameter.

- `--db-parameter-group-name`

Example

The following example lists the parameters and parameter values for a DB parameter group named *mydbparametergroup*.

```
aws rds describe-db-parameters --db-parameter-group-name mydbparametergroup
```

The command returns a response like the following:

DBPARAMETER	Parameter Name	Parameter Value	Source	Data Type	Apply
Type	Is Modifiable				
DBPARAMETER	allow-suspicious-udfs		engine-default	boolean	static
	false				
DBPARAMETER	auto_increment_increment		engine-default	integer	dynamic
	true				
DBPARAMETER	auto_increment_offset		engine-default	integer	dynamic
	true				
DBPARAMETER	binlog_cache_size	32768	system	integer	dynamic
	true				
DBPARAMETER	socket	/tmp/mysql.sock	system	string	static
	false				

API

To view the parameter values for a DB parameter group, use the Amazon RDS API [DescribeDBParameters](#) command with the following required parameter.

- `DBParameterGroupName`

Comparing DB Parameter Groups

You can use the AWS Management Console to view the differences between two parameter groups for the same DB engine and version.

To compare two parameter groups

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Parameter groups**.
3. In the list, choose the two parameter groups that you want to compare.
4. For **Parameter group actions**, choose **Compare**.

DB Parameter Values

The value for a DB parameter can be specified as:

- An integer constant
- A DB parameter formula
- A DB parameter function
- A character string constant
- A log expression (the log function represents log base 2), such as
`value={log(DBInstanceClassMemory/8187281418)*1000}`

DB Parameter Formulas

A DB parameter formula is an expression that resolves to an integer value or a Boolean value, and is enclosed in braces: `{}`. You can specify formulas for either a DB parameter value or as an argument to a DB parameter function.

Syntax

```
{FormulaVariable}
```

```
{FormulaVariable*Integer}
```

```
{FormulaVariable*Integer/Integer}
```

```
{FormulaVariable/Integer}
```

DB Parameter Formula Variables

Each formula variable returns integer or a Boolean value. The names of the variables are case sensitive.

AllocatedStorage

Returns the size, in bytes, of the data volume.

DBInstanceClassMemory

Returns the number of bytes of memory allocated to the DB instance class associated with the current DB instance, less the memory used by the Amazon RDS processes that manage the instance.

EndPointPort

Returns the number of the port used when connecting to the DB instance.

DBInstanceClassHugePagesDefault

Returns a Boolean value. Currently, it is only supported for Oracle engines.

For more information, see [Using Huge Pages with an Oracle DB Instance \(p. 736\)](#).

DB Parameter Formula Operators

DB parameter formulas support two operators: division and multiplication.

Division Operator: /

Divides the dividend by the divisor, returning an integer quotient. Decimals in the quotient are truncated, not rounded.

Syntax

```
dividend / divisor
```

The dividend and divisor arguments must be integer expressions.

*Multiplication Operator: **

Multiplies the expressions, returning the product of the expressions. Decimals in the expressions are truncated, not rounded.

Syntax

```
expression * expression
```

Both expressions must be integers.

DB Parameter Functions

The parameter arguments can be specified as either integers or formulas. Each function must have at least one argument. Multiple arguments can be specified as a comma-separated list. The list cannot have any empty members, such as *argument1,,argument3*. Function names are case insensitive.

Note

DB Parameter functions are not currently supported in CLI.

IF()

Returns an argument.

Currently, it is only supported for Oracle engines, and the only supported first argument is `{DBInstanceClassHugePagesDefault}`. For more information, see [Using Huge Pages with an Oracle DB Instance \(p. 736\)](#).

Syntax

```
IF(argument1, argument2, argument3)
```

Returns the second argument if the first argument evaluates to true. Returns the third argument otherwise.

GREATEST()

Returns the largest value from a list of integers or parameter formulas.

Syntax

```
GREATEST(argument1, argument2,...argumentn)
```

Returns an integer.

LEAST()

Returns the smallest value from a list of integers or parameter formulas.

Syntax

```
LEAST(argument1, argument2,...argumentn)
```

Returns an integer.

SUM()

Adds the values of the specified integers or parameter formulas.

Syntax

```
SUM(argument1, argument2,...argumentn)
```

Returns an integer.

DB Parameter Value Examples

These examples show using formulas and functions in the values for DB parameters.

Warning

Improperly setting parameters in a DB parameter group can have unintended adverse effects, including degraded performance and system instability. Always exercise caution when modifying database parameters and back up your data before modifying your DB parameter group. You should try out parameter group changes on a test DB instances, created using point-in-time-restores, before applying those parameter group changes to your production DB instances.

You can specify the `GREATEST` function in an Oracle processes parameter to set the number of user processes to the larger of either 80 or `DBInstanceClassMemory` divided by 9,868,951.

```
GREATEST({DBInstanceClassMemory/9868951},80)
```

You can specify the `LEAST()` function in a MySQL `max_binlog_cache_size` parameter value to set the maximum cache size a transaction can use in a MySQL instance to the lesser of 1 MB or `DBInstanceClass/256`.

```
LEAST({DBInstanceClassMemory/256},10485760)
```

Working with Amazon Resource Names (ARNs) in Amazon RDS

Resources created in Amazon Web Services are each uniquely identified with an Amazon Resource Name (ARN). For certain Amazon RDS operations, you must uniquely identify an Amazon RDS resource by specifying its ARN. For example, when you create an RDS DB instance Read Replica, you must supply the ARN for the source DB instance.

Constructing an ARN for Amazon RDS

Resources created in Amazon Web Services are each uniquely identified with an Amazon Resource Name (ARN). You can construct an ARN for an Amazon RDS resource using the following syntax.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Region Name	Region	Endpoint	Protocol
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS
US East (N. Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS
US West (N. California)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS
Asia Pacific (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS
Asia Pacific (Osaka-Local)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS
Asia Pacific (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS
Asia Pacific (Singapore)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS
Asia Pacific (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS
Asia Pacific (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
Canada (Central)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
China (Beijing)	cn-north-1	rds.cn-north-1.amazonaws.com.cn	HTTPS
China (Ningxia)	cn-northwest-1	rds.cn-northwest-1.amazonaws.com.cn	HTTPS

Region Name	Region	Endpoint	Protocol
EU (Frankfurt)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
EU (Ireland)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS
EU (London)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS
EU (Paris)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
EU (Stockholm)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS
South America (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (US-East)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (US)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS

The following table shows the format that you should use when constructing an ARN for a particular Amazon RDS resource type.

Resource Type	ARN Format
DB instance	arn:aws:rds:<region>:<account>:db:<name> For example: <div style="border: 1px solid #ccc; padding: 2px;">arn:aws:rds:us-east-2:123456789012:db:my-mysql-instance-1</div>
DB cluster	arn:aws:rds:<region>:<account>:cluster:<name> For example: <div style="border: 1px solid #ccc; padding: 2px;">arn:aws:rds:us-east-2:123456789012:cluster:my-aurora-cluster-1</div>
Event subscription	arn:aws:rds:<region>:<account>:es:<name> For example: <div style="border: 1px solid #ccc; padding: 2px;">arn:aws:rds:us-east-2:123456789012:es:my-subscription</div>
DB option group	arn:aws:rds:<region>:<account>:og:<name> For example:

Resource Type	ARN Format
	arn:aws:rds:<region>:<account>:og:<name>
DB parameter group	<p>arn:aws:rds:<region>:<account>:pg:<name></p> <p>For example:</p> <pre>arn:aws:rds:us-east-2:123456789012:pg:my-param-enable-logs</pre>
DB cluster parameter group	<p>arn:aws:rds:<region>:<account>:cluster-pg:<name></p> <p>For example:</p> <pre>arn:aws:rds:us-east-2:123456789012:cluster-pg:my-cluster-param-timezone</pre>
Reserved DB instance	<p>arn:aws:rds:<region>:<account>:ri:<name></p> <p>For example:</p> <pre>arn:aws:rds:us-east-2:123456789012:ri:my-reserved-postgresql</pre>
DB security group	<p>arn:aws:rds:<region>:<account>:secgrp:<name></p> <p>For example:</p> <pre>arn:aws:rds:us-east-2:123456789012:secgrp:my-public</pre>
DB snapshot	<p>arn:aws:rds:<region>:<account>:snapshot:<name></p> <p>For example:</p> <pre>arn:aws:rds:us-east-2:123456789012:snapshot:my-mysql-snap-20130507</pre>
DB cluster snapshot	<p>arn:aws:rds:<region>:<account>:cluster-snapshot:<name></p> <p>For example:</p> <pre>arn:aws:rds:us-east-2:123456789012:cluster-snapshot:my-aurora-snap-20160809</pre>
DB subnet group	<p>arn:aws:rds:<region>:<account>:subgrp:<name></p> <p>For example:</p> <pre>arn:aws:rds:us-east-2:123456789012:subgrp:my-subnet-10</pre>

Getting an Existing ARN

You can get the ARN of an RDS resource by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or RDS API.

AWS Management Console

To get an ARN from the AWS Management Console, navigate to the resource you want an ARN for, and view the details for that resource. For example, you can get the ARN for a DB instance from the **Configuration** tab of the DB instance details, as shown following.

Connectivity Monitoring Logs & events Configuration

Instance

Configuration

DB instance id
oracle-instance1

Engine version
12.1.0.2.v14

Storage type
General Purpose (SSD)

IOPS
-

Storage
20 GiB

DB name
ORCL

License model
Bring Your Own License

Character set
AL32UTF8

Option groups
default:oracle-ee-12-1

ARN
arn:aws:rds:us-west-2:██████████:db:oracle-instance1

Resource id

API Version 2014-10-31
185

AWS CLI

To get an ARN from the AWS CLI for a particular RDS resource, you use the `describe` command for that resource. The following table shows each AWS CLI command, and the ARN property used with the command to get an ARN.

AWS CLI Command	ARN Property
<code>describe-event-subscriptions</code>	<code>EventSubscriptionArn</code>
<code>describe-certificates</code>	<code>CertificateArn</code>
<code>describe-db-parameter-groups</code>	<code>DBParameterGroupArn</code>
<code>describe-db-cluster-parameter-groups</code>	<code>DBClusterParameterGroupArn</code>
<code>describe-db-instances</code>	<code>DBInstanceArn</code>
<code>describe-db-security-groups</code>	<code>DBSecurityGroupArn</code>
<code>describe-db-snapshots</code>	<code>DBSnapshotArn</code>
<code>describe-events</code>	<code>SourceArn</code>
<code>describe-reserved-db-instances</code>	<code>ReservedDBInstanceArn</code>
<code>describe-db-subnet-groups</code>	<code>DBSubnetGroupArn</code>
<code>describe-option-groups</code>	<code>OptionGroupArn</code>
<code>describe-db-clusters</code>	<code>DBClusterArn</code>
<code>describe-db-cluster-snapshots</code>	<code>DBClusterSnapshotArn</code>

For example, the following AWS CLI command gets the ARN for a DB instance.

Example

For Linux, OS X, or Unix:

```
aws rds describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2
```

For Windows:

```
aws rds describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2
```

API

To get an ARN for a particular RDS resource, you can call the following RDS API actions and use the ARN properties shown following.

RDS API Action	ARN Property
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
DescribeDBParameterGroups	DBParameterGroupArn
DescribeDBClusterParameterGroups	DBClusterParameterGroupArn
DescribeDBInstances	DBInstanceArn
DescribeDBSecurityGroups	DBSecurityGroupArn
DescribeDBSchemas	DBSnapshotArn
DescribeEvents	SourceArn
DescribeReservedDBInstances	ReservedDBInstanceArn
DescribeDBSubnetGroups	DBSubnetGroupArn
DescribeOptionGroups	OptionGroupArn
DescribeDBClusters	DBClusterArn
DescribeDBClusterSnapshots	DBClusterSnapshotArn

Working with Storage

To specify how you want your data stored in Amazon RDS, you select a storage type and provide a storage size when you create or modify a DB instance. Later, you can increase the amount or change the type of storage by modifying the DB instance. For more information about which storage type to use for your workload, see [Amazon RDS Storage Types \(p. 103\)](#).

Topics

- [Increasing DB Instance Storage Capacity \(p. 188\)](#)
- [Changing Your Storage Type \(p. 189\)](#)
- [Modifying Provisioned IOPS SSD storage settings \(p. 191\)](#)

Increasing DB Instance Storage Capacity

If you need space for additional data, you can scale up the storage of an existing DB instance. To do so, you can use the Amazon RDS Management Console, the Amazon RDS API, or the AWS Command Line Interface (AWS CLI). If you are using General Purpose SSD or Provisioned IOPS SSD storage, you can increase your storage to a maximum of 16 TiB. Scaling storage for Amazon RDS for SQL Server database instance, is supported only for General Purpose SSD or Provisioned IOPS SSD storage types.

We recommend that you create an Amazon CloudWatch alarm to monitor the amount of free storage for your DB instance so you can respond when necessary. For more information on setting CloudWatch alarms, see [Using Amazon RDS Event Notification \(p. 287\)](#).

In most cases, scaling storage doesn't require any outage and doesn't degrade performance of the server. After you modify the storage size for a DB instance, the status of the DB instance is **storage-optimization**. The DB instance is fully operational after a storage modification. However, you can't make further storage modifications for either six (6) hours or while the DB instance status is **storage-optimization**, whichever is longer.

If you have a SQL Server DB instance and haven't modified the storage configuration since November 2017, you might experience a short outage of a few minutes when you modify your DB instance to increase the allocated storage. After the outage, the DB instance is online but in the **storage-optimization** state. Performance might be degraded during storage optimization.

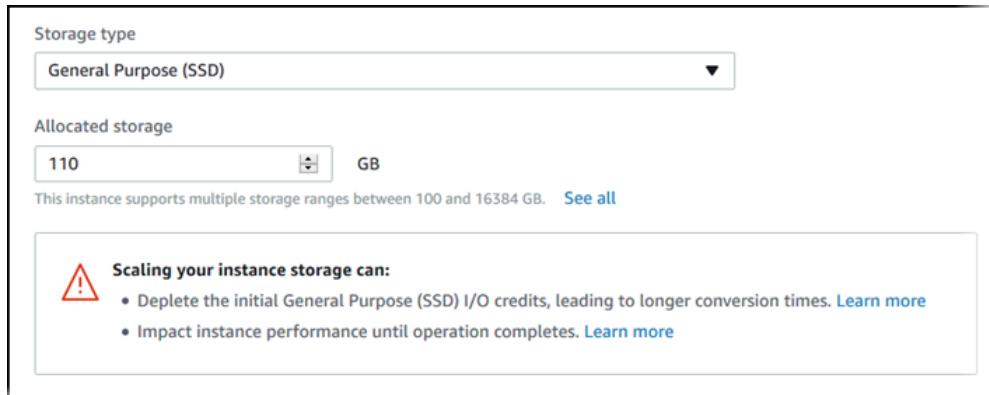
Note

You can't reduce the amount of storage for a DB instance after it has been allocated.

AWS Management Console

To increase storage for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to modify.
4. Choose **Modify**.
5. Enter a new value for **Allocated Storage**. It must be greater than the current value.



Note

When you increase **Allocated Storage**, it must be by at least 10 percent. If you try to increase by less than 10 percent, you see an error.

6. Choose **Continue** to move to the next screen.
7. To immediately start conversion of the DB instance to use the new storage type, choose **Apply immediately** in the **Scheduling of modifications** section. Alternatively, you can choose **Apply during the next scheduled maintenance window**.
8. When the settings are as you want them, choose **Modify DB instance**.

CLI

To increase the storage for a DB instance, use the AWS CLI `modify-db-instance` command. Set the following parameters:

- `--allocated-storage` – Amount of storage to be allocated for the DB instance, in gibibytes.
- `--apply-immediately` – Use `--apply-immediately` to initiate conversion immediately, or `--no-apply-immediately` (the default) to apply the conversion during the next maintenance window. An immediate outage occurs when the conversion is applied. For more information about storage, see [DB instance storage \(p. 103\)](#).

API

To increase storage for a DB instance, use the Amazon RDS API `ModifyDBInstance` action. Set the following parameters:

- `AllocatedStorage` – Amount of storage to be allocated for the DB instance, in gibibytes.
- `ApplyImmediately` – Set this option to `True` if you want to initiate conversion immediately. If this option is `False` (the default), the scaling is applied during the next maintenance window. An immediate outage occurs when the conversion is applied.

For more information about storage, see [DB instance storage \(p. 103\)](#).

Changing Your Storage Type

You can change the type of storage for your DB instance by using the AWS Management Console, the Amazon RDS API, or the AWS Command Line Interface (AWS CLI).

When you convert from one storage type to another, an outage occurs while the data for that DB instance is migrated to a new volume. The duration of the migration depends on several factors such as

database load, storage size, storage type, and amount of IOPS provisioned (if any). The typical migration time is a few minutes. The DB instance is available for use during the migration.

However, when you are migrating to or from magnetic storage, the migration time can take up to several days in some cases. During the migration to or from magnetic storage, the DB instance is available for use, but might experience performance degradation.

Storage conversions from Provisioned IOPS SSD or magnetic storage to General Purpose SSD storage can potentially deplete the I/O credits allocated for General Purpose SSD storage. This is especially true on smaller volumes. After the initial I/O burst credits for the volume are depleted, the remaining data is converted at the base performance rate of 3 IOPS per GiB of allocated General Purpose SSD storage. This approach can result in significantly longer conversion times.

AWS Management Console

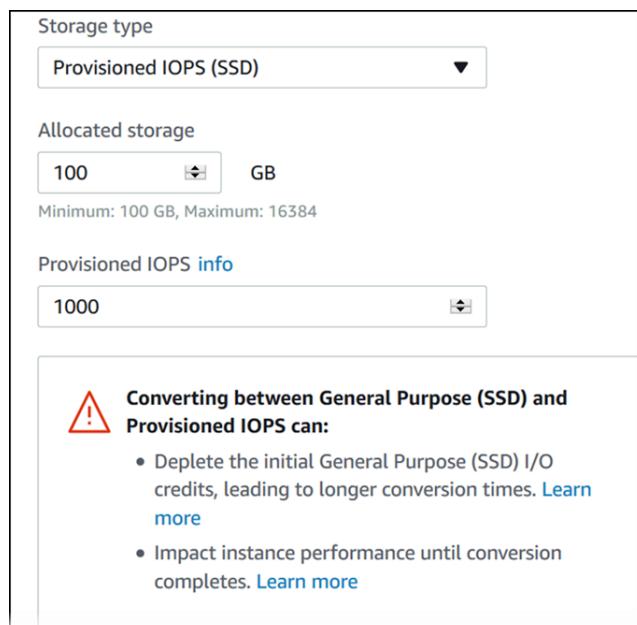
To change the storage type for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.

Note

To filter the list of DB instances, for **Filter databases** enter a text string for Amazon RDS to use to filter the results. Only DB instances whose names contain the string appear.

3. Choose the DB instance that you want to modify.
4. Choose **Modify**.
5. On the **Modify DB Instance page**, choose the type of storage from the **Storage type** list. If you are modifying your DB instance to use Provisioned IOPS SSD storage type, then also provide a Provisioned IOPS value.



6. Choose **Continue**.
7. To apply the changes to the DB instance immediately, choose **Apply immediately** in the **Scheduling of modifications** section. Alternatively, you can choose **Apply during the next scheduled maintenance window**.

An immediate outage occurs when the storage type changes. For more information about storage, see [DB instance storage \(p. 103\)](#).

8. Review the parameters to be changed, and choose **Modify DB instance** to complete the modification.

CLI

To change the type of storage for a DB instance, use the AWS CLI `modify-db-instance` command. Set the following parameters:

- `--storage-type` – Set to `io1` for Provisioned IOPS.
- `--apply-immediately` – Use `--apply-immediately` to initiate conversion immediately. Use `--no-apply-immediately` (the default) to apply the conversion during the next maintenance window.

API

To change the type of storage for a DB instance, use the Amazon RDS API `ModifyDBInstance` action. Set the following parameters:

- `StorageType` – Set to `io1` for Provisioned IOPS.
- `ApplyImmediately` – Set this option to `True` if you want to initiate conversion immediately. If this option is `False` (the default), the conversion is applied during the next maintenance window.

Modifying Provisioned IOPS SSD storage settings

You can modify the settings for a DB instance that uses Provisioned IOPS SSD Storage by using the AWS Management Console, the Amazon RDS API, or the AWS CLI. Specify the storage type, allocated storage, and the amount of Provisioned IOPS that you require. You can choose between 1,000 IOPS and 100 GiB of storage up to 40,000 IOPS and 32 TiB (32768 GiB) of storage, depending on your database engine.

Although you can reduce the amount of IOPS provisioned for your instance, you can't reduce the amount of General Purpose SSD or magnetic storage allocated.

AWS Management Console

To change the Provisioned IOPS settings for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.

Note

To filter the list of DB instances, for **Filter databases** enter a text string for Amazon RDS to use to filter the results. Only DB instances whose names contain the string appear.

3. Choose the DB instance with Provisioned IOPS that you want to modify.
4. Choose **Modify**.
5. On the **Modify DB Instance page**, choose Provisioned IOPS for **Storage type** and then provide a Provisioned IOPS value.

The screenshot shows the 'Modify DB instance' configuration page. Under 'Storage type info', 'Provisioned IOPS (SSD)' is selected. In the 'Allocated storage' section, a dropdown menu shows '100' with 'GB' next to it, and a note below says 'Minimum: 100 GB, Maximum: 16384'. In the 'Provisioned IOPS info' section, a dropdown menu shows '1000'.

If the value you specify for either **Allocated storage** or **Provisioned IOPS** is outside the limits supported by the other parameter, a warning message is displayed. This message gives the range of values required for the other parameter.

6. Choose **Continue**.
7. To apply the changes to the DB instance immediately, choose **Apply immediately** in the **Scheduling of modifications** section. Alternatively, you can choose **Apply during the next scheduled maintenance window**.

An immediate outage occurs when the storage type changes. For more information about storage, see [DB instance storage \(p. 103\)](#).

8. Review the parameters to be changed, and choose **Modify DB instance** to complete the modification.

The new value for allocated storage or for Provisioned IOPS appears in the **Status** column.

CLI

To change the Provisioned IOPS setting for a DB instance, use the AWS CLI `modify-db-instance` command. Set the following parameters:

- `--storage-type` – Set to `io1` for Provisioned IOPS.
- `--allocated-storage` – Amount of storage to be allocated for the DB instance, in gibibytes.
- `--iops` – The new amount of Provisioned IOPS for the DB instance, expressed in I/O operations per second.
- `--apply-immediately` – Use `--apply-immediately` to initiate conversion immediately. Use `--no-apply-immediately` (the default) to apply the conversion during the next maintenance window.

API

To change the Provisioned IOPS settings for a DB instance, use the Amazon RDS API `ModifyDBInstance` action. Set the following parameters:

- `StorageType` – Set to `io1` for Provisioned IOPS.
- `AllocatedStorage` – Amount of storage to be allocated for the DB instance, in gibibytes.
- `Iops` – The new IOPS rate for the DB instance, expressed in I/O operations per second.
- `ApplyImmediately` – Set this option to `True` if you want to initiate conversion immediately. If this option is `False` (the default), the modification is applied during the next maintenance window.

DB Instance Billing for Amazon RDS

Amazon RDS instances are billed based on the following components:

- DB instance hours (per hour) – Based on the DB instance class of the DB instance (for example, db.t2.small or db.m4.large). Partial DB instance hours consumed are billed as full hours. For more information, see [DB Instance Class \(p. 82\)](#).
- Storage (per GiB per month) – Storage capacity that you have provisioned to your DB instance. If you scale your provisioned storage capacity within the month, your bill is pro-rated. For more information, see [DB instance storage \(p. 103\)](#).
- I/O requests (per 1 million requests per month) – Total number of storage I/O requests that you have made in a billing cycle, for Amazon RDS magnetic storage only.
- Provisioned IOPS (per IOPS per month) – Provisioned IOPS rate, regardless of IOPS consumed, for Amazon RDS Provisioned IOPS (SSD) Storage only.
- Backup storage (per GiB per month) – *Backup storage* is the storage that is associated with automated database backups and any active database snapshots that you have taken. Increasing your backup retention period or taking additional database snapshots increases the backup storage consumed by your database.

For more information, see [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#).

- Data transfer (per GB) – Data transfer in and out of your DB instance from or to the internet and other AWS Regions.

Amazon RDS provides the following purchasing options to enable you to optimize your costs based on your needs:

- **On-Demand Instances** – Pay by the hour for the DB instance hours that you use.
- **Reserved Instances** – Reserve a DB instance for a one-year or three-year term and receive a significant discount compared to the on-demand DB instance pricing.

For Amazon RDS pricing information, see the [Amazon RDS product page](#).

Topics

- [On-Demand DB Instances \(p. 194\)](#)
- [Reserved DB Instances \(p. 195\)](#)

On-Demand DB Instances

Amazon RDS on-demand DB instances are billed based on the class of the DB instance (for example, db.t2.small or db.m4.large). Partial DB instance hours consumed are billed as full hours. For Amazon RDS pricing information, see the [Amazon RDS product page](#).

Billing starts for a DB instance as soon as the DB instance is available. DB instance hours are billed for each hour that your DB instance is running in an available state. Billing continues until the DB instance terminates, which occurs when you delete the DB instance or if the DB instance fails.

If you no longer want to be charged for your DB instance, you must stop or delete it to avoid being billed for additional DB instance hours. For more information about the DB instance states for which you are billed, see [DB Instance Status \(p. 98\)](#).

Stopped DB Instances

While your DB instance is stopped, you are charged for provisioned storage, including Provisioned IOPS. You are also charged for backup storage, including storage for manual snapshots and automated backups within your specified retention window. You are not charged for DB instance hours.

Multi-AZ DB Instances

If you specify that your DB instance should be a Multi-AZ deployment, you are billed according to the Multi-AZ pricing posted on the Amazon RDS pricing page.

Reserved DB Instances

Using reserved DB instances, you can reserve a DB instance for a one- or three-year term. Reserved DB instances provide you with a significant discount compared to on-demand DB instance pricing. Reserved DB instances are not physical instances, but rather a billing discount applied to the use of certain on-demand DB instances in your account. Discounts for reserved DB instances are tied to instance type and AWS Region.

The general process for working with reserved DB instances is: First get information about available reserved DB instance offerings, then purchase a reserved DB instance offering, and finally get information about your existing reserved DB instances.

Overview of Reserved DB Instances

When you purchase a reserved DB instance in Amazon RDS, you purchase a commitment to getting a discounted rate, on a specific DB instance type, for the duration of the reserved DB instance. To use an Amazon RDS reserved DB instance, you create a new DB instance just like you do for an on-demand instance. The new DB instance that you create must match the specifications of the reserved DB instance. If the specifications of the new DB instance match an existing reserved DB instance for your account, you are billed at the discounted rate offered for the reserved DB instance. Otherwise, the DB instance is billed at an on-demand rate.

For more information about reserved DB instances, including pricing, see [Amazon RDS Reserved Instances](#).

Offering Types

Reserved DB instances are available in three varieties—No Upfront, Partial Upfront, and All Upfront—that let you optimize your Amazon RDS costs based on your expected usage.

No Upfront

This option provides access to a reserved DB instance without requiring an upfront payment. Your No Upfront reserved DB instance bills a discounted hourly rate for every hour within the term, regardless of usage, and no upfront payment is required. This option is only available as a one-year reservation.

Partial Upfront

This option requires a part of the reserved DB instance to be paid upfront. The remaining hours in the term are billed at a discounted hourly rate, regardless of usage. This option is the replacement for the previous Heavy Utilization option.

All Upfront

Full payment is made at the start of the term, with no other costs incurred for the remainder of the term regardless of the number of hours used.

If you are using consolidated billing, all the accounts in the organization are treated as one account. This means that all accounts in the organization can receive the hourly cost benefit of reserved DB instances that are purchased by any other account. For more information about consolidated billing, see [Amazon RDS Reserved DB Instances](#) in the *AWS Billing and Cost Management User Guide*.

Size-Flexible Reserved DB Instances

When you purchase a reserved DB instance, one thing that you specify is the instance class, for example db.m4.large. For more information about instance classes, see [DB Instance Class \(p. 82\)](#).

If you have a DB instance, and you need to scale it to larger capacity, your reserved DB instance is automatically applied to your scaled DB instance. That is, your reserved DB instances are automatically applied across all DB instance class sizes. Size-flexible reserved DB instances are available for DB instances with the same AWS Region and database engine. Size-flexible reserved DB instances can only scale in their instance class type. For example, a reserved DB instance for a db.m4.large can apply to a db.m4.xlarge, but not to a db.m5.large, because db.m4 and db.m5 are different instance class types. Reserved DB instance benefits also apply for both Multi-AZ and Single-AZ configurations.

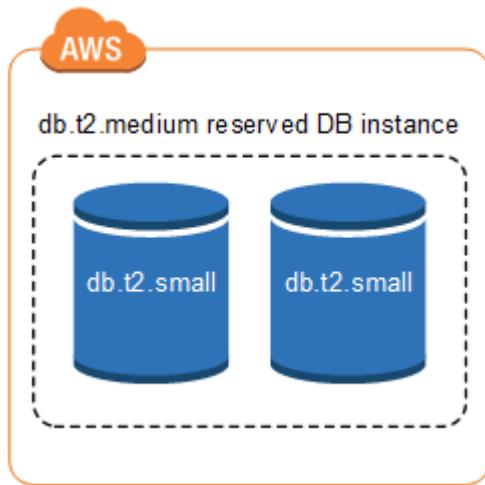
Size-flexible reserved DB instances are available for the following database engines:

- MariaDB
- MySQL
- Oracle, Bring Your Own License
- PostgreSQL

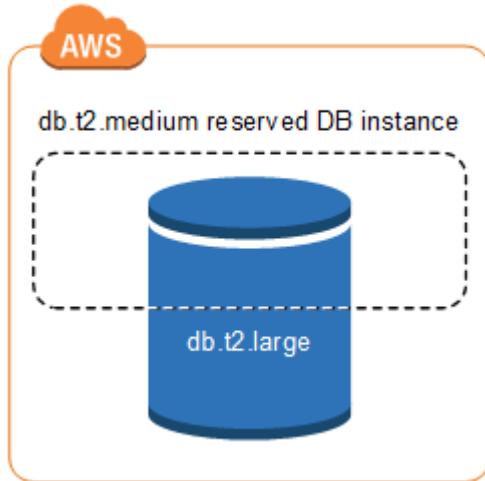
You can compare usage for different reserved DB instance sizes by using normalized units. For example, one unit of usage on two db.m3.large DB instances is equivalent to eight normalized units of usage on one db.m3.small. The following table shows the number of normalized units for each DB instance size.

Instance Size	Single-AZ Normalized Units	Multi-AZ Normalized Units
micro	0.5	1
small	1	2
medium	2	4
large	4	8
xlarge	8	16
2xlarge	16	32
4xlarge	32	64
8xlarge	64	128
10xlarge	80	160
16xlarge	128	256

For example, suppose that you purchase a db.t2.medium reserved DB instance, and you have two running db.t2.small DB instances in your account in the same AWS Region. In this case, the billing benefit is applied in full to both instances.



Alternatively, if you have one db.t2.large instance running in your account in the same AWS Region, the billing benefit is applied to 50 percent of the usage of the DB instance.



Reserved DB Instance Billing Example

The price for a reserved DB instance doesn't include regular costs associated with storage, backups, and I/O. The following example illustrates the total cost per month for a reserved DB instance:

- An Amazon RDS MySQL reserved Single-AZ db.r4.large DB instance class in US East (N. Virginia) with the No Upfront option at a cost of \$0.12 for the instance, or \$90 per month
- 400 GiB of General Purpose SSD (gp2) storage at a cost of 0.115 per GiB per month, or \$45.60 per month
- 600 GiB of backup storage at \$0.095, or \$19 per month (400 GiB free)

Add all of these options (\$90 + \$45.60 + \$19) with the reserved DB instance, and the total cost per month is \$154.60.

If you chose to use an on-demand DB instance instead of a reserved DB instance, an Amazon RDS MySQL Single-AZ db.r4.large DB instance class in US East (N. Virginia) costs \$0.1386 per hour, or \$101.18 per month. So, for an on-demand DB instance, add all of these options (\$101.18 + \$45.60 + \$19), and the total cost per month is \$165.78.

Note

The prices in this example are sample prices and might not match actual prices.
For Amazon RDS pricing information, see the [Amazon RDS product page](#).

Deleting a Reserved DB Instance

The terms for a reserved DB instance involve a one-year or three-year commitment. You can't cancel a reserved DB instance. However, you can delete a DB instance that is covered by a reserved DB instance discount. The process for deleting a DB instance that is covered by a reserved DB instance discount is the same as for any other DB instance.

Your upfront payment for a reserved DB instance reserves the resources for your use. Because these resources are reserved for you, you are billed for the resources regardless of whether you use them.

If you delete a DB instance that is covered by a reserved DB instance discount, you can launch another DB instance with compatible specifications. In this case, you continue to get the discounted rate during the reservation term (one or three years).

Console

You can use the AWS Management Console to work with reserved DB instances as shown in the following procedures.

To get pricing and information about available reserved DB instance offerings

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Reserved instances**.
3. Choose **Purchase Reserved DB Instance**.
4. For **Product description**, choose the DB engine and licensing type.
5. For **DB instance class**, choose the DB instance class.
6. For **Multi-AZ deployment**, choose whether you want a Multi-AZ deployment.
7. For **Term**, choose the length of time you want the DB instance reserved.
8. For **Offering type**, choose the offering type.

After you select the offering type, you can see the pricing information.

Important

Choose **Cancel** to avoid purchasing the reserved DB instance and incurring any charges.

After you have information about the available reserved DB instance offerings, you can use the information to purchase an offering as shown in the following procedure.

To purchase a reserved DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Reserved instances**.
3. Choose **Purchase Reserved DB Instance**.
4. For **Product description**, choose the DB engine and licensing type.
5. For **DB instance class**, choose the DB instance class.
6. For **Multi-AZ deployment**, choose whether you want a Multi-AZ deployment.
7. For **Term**, choose the length of time you want the DB instance reserved.
8. For **Offering type**, choose the offering type.

After you choose the offering type, you can see the pricing information, as shown following.

Purchase Reserved DB Instances

Choose from the options below, then enter the number of DB instances you wish to reserve with this order. When you are done, click the Continue button.

Options

Product description

aurora-mysql

DB instance class

db.r4.4xlarge — 16 vCPU, 122 GiB RAM

Multi AZ deployment

Multi-AZ deployment model is not applicable for this database engine and edition

- Yes
 No

Term

1 year

Offering type

All Upfront

Reserved Id (optional)

Optional tag to track your reservation

Number of DB instances

1

Pricing details

One-time payment (per instance)

Total one-time payment*

*Additional taxes may apply

Normalized units per hour [info](#)

32

Usage charges*

USD (hourly)

*Additional taxes may apply

This hourly rate is charged for every hour for each instance in the Reserved Instance term you purchase, regardless of instance usage

Charges for your usage will appear on your monthly bill.

[Cancel](#)

[Continue](#)

9. (Optional) You can assign your own identifier to the reserved DB instances that you purchase to help you track them. For **Reserved Id**, type an identifier for your reserved DB instance.
10. Choose **Continue**.

The **Purchase Reserved DB Instance** dialog box appears, with a summary of the reserved DB instance attributes that you've selected and the payment due, as shown following.

Purchase Reserved DB Instances

Summary of Purchase

You are about to purchase a Reserved DB Instance with the following information.

Region	US East (N. Virginia)
Product Description	aurora-mysql
DB Instance Class	db.r4.4xlarge
Offering Type	All Upfront
Multi AZ Deployment	No
Term	1 year
Reserved DB Instance	default
Quantity	1
Price Per Instance	[REDACTED]
Total Payment Due Now	[REDACTED]

⚠️ Purchasing this Reserved DB Instance will charge [REDACTED] to the payment method associated with this Amazon Web Services account. Are you sure you would like to proceed?

Cancel **Back** **Purchase**

11. On the confirmation page, review your reserved DB instance. If the information is correct, choose **Purchase** to purchase the reserved DB instance.

Alternatively, choose **Back** to edit your reserved DB instance.

After you have purchased reserved DB instances, you can get information about your reserved DB instances as shown in the following procedure.

To get information about reserved DB instances for your AWS account

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the **Navigation** pane, choose **Reserved instances**.

The reserved DB instances for your account appear. To see detailed information about a particular reserved DB instance, choose that instance in the list. You can then see detailed information about that instance in the detail pane at the bottom of the console.

AWS CLI

You can use the AWS CLI to work with reserved DB instances as shown in the following examples.

Example Get Available Reserved DB Instance Offerings

To get information about available reserved DB instance offerings, call the AWS CLI command `describe-reserved-db-instances-offerings`.

```
aws rds describe-reserved-db-instances-offerings
```

This call returns output similar to the following:

OFFERING	OfferingId	Class	Multi-AZ	Duration	Fixed
Price	Usage Price	Description	Offering Type		
OFFERING	438012d3-4052-4cc7-b2e3-8d3372e0e706	db.m1.large	y	1y	1820.00
USD	0.368	mysql	Partial Upfront		
OFFERING	649fd0c8-cf6d-47a0-bfa6-060f8e75e95f	db.m1.small	n	1y	227.50
USD	0.046	mysql	Partial Upfront		
OFFERING	123456cd-ab1c-47a0-bfa6-12345667232f	db.m1.small	n	1y	162.00
USD	0.00	mysql	All Upfront		
Recurring Charges:	Amount	Currency	Frequency		
Recurring Charges:	0.123	USD	Hourly		
OFFERING	123456cd-ab1c-37a0-bfa6-12345667232d	db.m1.large	y	1y	700.00
USD	0.00	mysql	All Upfront		
Recurring Charges:	Amount	Currency	Frequency		
Recurring Charges:	1.25	USD	Hourly		
OFFERING	123456cd-ab1c-17d0-bfa6-12345667234e	db.m1.xlarge	n	1y	4242.00
USD	2.42	mysql	No Upfront		

After you have information about the available reserved DB instance offerings, you can use the information to purchase an offering as shown in the following example.

Example Purchase a Reserved DB Instance

To purchase a reserved DB instance, use the AWS CLI command `purchase-reserved-db-instances-offering` with the following parameters:

- **--reserved-db-instances-offering-id** – the id of the offering that you want to purchase. See the preceding example to get the offering ID.
- **--reserved-db-instance-id** – you can assign your own identifier to the reserved DB instances that you purchase to help you track them.

The following example purchases the reserved DB instance offering with ID **649fd0c8-cf6d-47a0-bfa6-060f8e75e95f**, and assigns the identifier of **MyReservation**.

For Linux, OS X, or Unix:

```
aws rds purchase-reserved-db-instances-offering \
--reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
--reserved-db-instance-id MyReservation
```

For Windows:

```
aws rds purchase-reserved-db-instances-offering ^
--reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
--reserved-db-instance-id MyReservation
```

The command returns output similar to the following:

RESERVATION	ReservationId	Class	Multi-AZ	Start Time	Duration
Fixed Price	Usage Price	Count	State	Description	Offering Type
RESERVATION	MyReservation	db.m1.small	y	2011-12-19T00:30:23.247Z	1y
455.00 USD	0.092 USD	1	payment-pending	mysql	Partial Upfront

After you have purchased reserved DB instances, you can get information about your reserved DB instances as shown in the following example.

Example Get Your Reserved DB Instances

To get information about reserved DB instances for your AWS account, call the AWS CLI command [describe-reserved-db-instances](#).

```
aws rds describe-reserved-db-instances
```

The command returns output similar to the following:

RESERVATION	ReservationId	Class	Multi-AZ	Start Time	Duration
Fixed Price	Usage Price	Count	State	Description	Offering Type
RESERVATION	MyReservation	db.m1.small	y	2011-12-09T23:37:44.720Z	1y
455.00 USD	0.092 USD	1	retired	mysql	Partial Upfront

RDS API

You can use the RDS API to work with reserved DB instances as shown in the following examples.

Example Get Available Reserved DB Instance Offerings

To get information about available reserved DB instance offerings, call the Amazon RDS API function [DescribeReservedDBInstancesOfferings](#).

```
https://rds.us-east-1.amazonaws.com/
?Action=DescribeReservedDBInstancesOfferings
&SignatureMethod=HmacSHA256
&SignatureVersion=4
```

```
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140411/us-east-1/rds/aws4_request
&X-Amz-Date=20140411T203327Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=545f04acffeb4b80d2e778526b1c9da79d0b3097151c24f28e83e851d65422e2
```

This call returns output similar to the following:

```
<DescribeReservedDBInstancesOfferingsResponse xmlns="http://rds.amazonaws.com/doc/2014-10-31/">
  <DescribeReservedDBInstancesOfferingsResult>
    <ReservedDBInstancesOfferings>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <FixedPrice>1820.0</FixedPrice>
        <ProductDescription>mysql</ProductDescription>
        <UsagePrice>0.368</UsagePrice>
        <MultiAZ>true</MultiAZ>
        <ReservedDBInstancesOfferingId>438012d3-4052-4cc7-b2e3-8d3372e0e706</
      ReservedDBInstancesOfferingId>
        <DBInstanceClass>db.m1.large</DBInstanceClass>
      </ReservedDBInstancesOffering>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <FixedPrice>227.5</FixedPrice>
        <ProductDescription>mysql</ProductDescription>
        <UsagePrice>0.046</UsagePrice>
        <MultiAZ>false</MultiAZ>
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</
      ReservedDBInstancesOfferingId>
        <DBInstanceClass>db.m1.small</DBInstanceClass>
      </ReservedDBInstancesOffering>
    </ReservedDBInstancesOfferings>
  </DescribeReservedDBInstancesOfferingsResult>
  <ResponseMetadata>
    <RequestId>5e4ec40b-2978-11e1-9e6d-771388d6ed6b</RequestId>
  </ResponseMetadata>
</DescribeReservedDBInstancesOfferingsResponse>
```

After you have information about the available reserved DB instance offerings, you can use the information to purchase an offering as shown in the following example.

Example Purchase a Reserved DB Instance

To purchase a reserved DB instance, call the Amazon RDS API action [PurchaseReservedDBInstancesOffering](#) with the following parameters:

- `--reserved-db-instances-offering-id` – the id of the offering that you want to purchase. See the preceding example to get the offering ID.
- `--reserved-db-instance-id` – you can assign your own identifier to the reserved DB instances that you purchase to help you track them.

The following example purchases the reserved DB instance offering with ID `649fd0c8-cf6d-47a0-bfa6-060f8e75e95f`, and assigns the identifier of `MyReservation`.

```
https://rds.us-east-1.amazonaws.com/
?Action=PurchaseReservedDBInstancesOffering
&ReservedDBInstanceId=MyReservation
&ReservedDBInstancesOfferingId=438012d3-4052-4cc7-b2e3-8d3372e0e706
&DBInstanceCount=10
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140415/us-east-1/rds/aws4_request
&X-Amz-Date=20140415T232655Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=c2ac761e8c8f54a8c0727f5a87ad0a766fbb0024510b9aa34ea6d1f7df52fb11
```

This call returns output similar to the following:

```
<PurchaseReservedDBInstancesOfferingResponse xmlns="http://rds.amazonaws.com/
doc/2014-10-31/">
  <PurchaseReservedDBInstancesOfferingResult>
    <ReservedDBInstance>
      <OfferingType>Partial Upfront</OfferingType>
      <CurrencyCode>USD</CurrencyCode>
      <RecurringCharges/>
      <ProductDescription>mysql</ProductDescription>
      <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</
      ReservedDBInstancesOfferingId>
      <MultiAZ>true</MultiAZ>
      <State>payment-pending</State>
      <ReservedDBInstanceId>MyReservation</ReservedDBInstanceId>
      <DBInstanceCount>10</DBInstanceCount>
      <StartTime>2011-12-18T23:24:56.577Z</StartTime>
      <Duration>31536000</Duration>
      <FixedPrice>123.0</FixedPrice>
      <UsagePrice>0.123</UsagePrice>
      <DBInstanceClass>db.m1.small</DBInstanceClass>
    </ReservedDBInstance>
  </PurchaseReservedDBInstancesOfferingResult>
  <ResponseMetadata>
    <RequestId>7f099901-29cf-11e1-bd06-6fe008f046c3</RequestId>
  </ResponseMetadata>
</PurchaseReservedDBInstancesOfferingResponse>
```

After you have purchased reserved DB instances, you can get information about your reserved DB instances as shown in the following example.

Example Get Your Reserved DB Instances

To get information about reserved DB instances for your AWS account, call the Amazon RDS API action [DescribeReservedDBInstances](#).

```
https://rds.us-west-2.amazonaws.com/
?Action=DescribeReservedDBInstances
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140420/us-west-2/rds/aws4_request
&X-Amz-Date=20140420T162211Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=3312d17a4c43bcd209bc22a0778dd23e73f8434254abbd7ac53b89ade3dae88e
```

The API returns output similar to the following:

```
<DescribeReservedDBInstancesResponse xmlns="http://rds.amazonaws.com/doc/2014-10-31/">
<DescribeReservedDBInstancesResult>
  <ReservedDBInstances>
    <ReservedDBInstance>
      <OfferingType>Partial Upfront</OfferingType>
      <CurrencyCode>USD</CurrencyCode>
      <RecurringCharges/>
      <ProductDescription>mysql</ProductDescription>
      <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</
      ReservedDBInstancesOfferingId>
      <MultiAZ>false</MultiAZ>
      <State>payment-failed</State>
      <ReservedDBInstanceId>MyReservation</ReservedDBInstanceId>
      <DBInstanceCount>1</DBInstanceCount>
      <StartTime>2010-12-15T00:25:14.131Z</StartTime>
      <Duration>31536000</Duration>
      <FixedPrice>227.5</FixedPrice>
      <UsagePrice>0.046</UsagePrice>
      <DBInstanceClass>db.m1.small</DBInstanceClass>
    </ReservedDBInstance>
    <ReservedDBInstance>
      <OfferingType>Partial Upfront</OfferingType>
      <CurrencyCode>USD</CurrencyCode>
      <RecurringCharges/>
      <ProductDescription>mysql</ProductDescription>
      <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</
      ReservedDBInstancesOfferingId>
      <MultiAZ>false</MultiAZ>
      <State>payment-failed</State>
      <ReservedDBInstanceId>MyReservation</ReservedDBInstanceId>
      <DBInstanceCount>1</DBInstanceCount>
      <StartTime>2010-12-15T01:07:22.275Z</StartTime>
      <Duration>31536000</Duration>
      <FixedPrice>227.5</FixedPrice>
      <UsagePrice>0.046</UsagePrice>
      <DBInstanceClass>db.m1.small</DBInstanceClass>
    </ReservedDBInstance>
  </ReservedDBInstances>
</DescribeReservedDBInstancesResult>
<ResponseMetadata>
  <RequestId>23400d50-2978-11e1-9e6d-771388d6ed6b</RequestId>
</ResponseMetadata>
</DescribeReservedDBInstancesResponse>
```

Backing Up and Restoring Amazon RDS DB Instances

This section shows how to back up and restore a DB instance.

Topics

- [Working With Backups \(p. 208\)](#)
- [Creating a DB Snapshot \(p. 216\)](#)
- [Restoring from a DB Snapshot \(p. 218\)](#)
- [Copying a Snapshot \(p. 221\)](#)
- [Sharing a DB Snapshot \(p. 230\)](#)
- [Restoring a DB Instance to a Specified Time \(p. 237\)](#)
- [Tutorial: Restore a DB Instance from a DB Snapshot \(p. 239\)](#)

Working With Backups

Amazon RDS creates and saves automated backups of your DB instance. Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases.

Amazon RDS creates automated backups of your DB instance during the backup window of your DB instance. Amazon RDS saves the automated backups of your DB instance according to the backup retention period that you specify. If necessary, you can recover your database to any point in time during the backup retention period.

Automated backups follow these rules:

- Your DB instance must be in the `ACTIVE` state for automated backups to occur. Automated backups don't occur while your DB instance is in a state other than `ACTIVE`, for example `STORAGE_FULL`.
- Automated backups and automated snapshots don't occur while a copy is executing in the same region for the same DB instance.

You can also back up your DB instance manually, by manually creating a DB snapshot. For more information about creating a DB snapshot, see [Creating a DB Snapshot \(p. 216\)](#).

The first snapshot of a DB instance contains the data for the full DB instance. Subsequent snapshots of the same DB instance are incremental, which means that only the data that has changed after your most recent snapshot is saved.

You can copy both automatic and manual DB snapshots, and share manual DB snapshots. For more information about copying a DB snapshot, see [Copying a Snapshot \(p. 221\)](#). For more information about sharing a DB snapshot, see [Sharing a DB Snapshot \(p. 230\)](#).

Backup Storage

Your Amazon RDS backup storage for each region is composed of the automated backups and manual DB snapshots for that region. Your backup storage is equivalent to the sum of the database storage for all instances in that region. Moving a DB snapshot to another region increases the backup storage in the destination region.

For more information about backup storage costs, see [Amazon RDS Pricing](#).

If you chose to retain automated backups when you delete a DB instance, the automated backups are saved for the full retention period. If you don't choose **Retain automated backups** when you delete a DB instance, all automated backups are deleted with the DB instance. After they are deleted, the automated backups can't be recovered. If you choose to have Amazon RDS create a final DB snapshot before it deletes your DB instance, you can use that to recover your DB instance. Or you can use a previously created manual snapshot. Manual snapshots are not deleted.

Backup Window

Automated backups occur daily during the preferred backup window. If the backup requires more time than allotted to the backup window, the backup continues after the window ends, until it finishes. The backup window can't overlap with the weekly maintenance window for the DB instance.

During the automatic backup window, storage I/O might be suspended briefly while the backup process initializes (typically under a few seconds). You might experience elevated latencies for a few minutes during backups for Multi-AZ deployments. For MariaDB, MySQL, Oracle, and PostgreSQL, I/O activity is not suspended on your primary during backup for Multi-AZ deployments, because the backup is taken from the standby. For SQL Server, I/O activity is suspended briefly during backup for Multi-AZ deployments.

If you don't specify a preferred backup window when you create the DB instance, Amazon RDS assigns a default 30-minute backup window. This window is selected at random from an 8-hour block of time for each AWS Region. The following table lists the time blocks for each region from which the default backups windows are assigned.

Region	Time Block
US West (Oregon) Region	06:00–14:00 UTC
US West (N. California) Region	06:00–14:00 UTC
US East (Ohio) Region	03:00–11:00 UTC
US East (N. Virginia) Region	03:00–11:00 UTC
Asia Pacific (Mumbai) Region	16:30–00:30 UTC
Asia Pacific (Seoul) Region	13:00–21:00 UTC
Asia Pacific (Singapore) Region	14:00–22:00 UTC
Asia Pacific (Sydney) Region	12:00–20:00 UTC
Asia Pacific (Tokyo) Region	13:00–21:00 UTC
Canada (Central) Region	06:29–14:29 UTC
EU (Frankfurt) Region	20:00–04:00 UTC
EU (Ireland) Region	22:00–06:00 UTC
EU (London) Region	06:00–14:00 UTC
South America (São Paulo) Region	23:00–07:00 UTC
AWS GovCloud (US-West)	03:00–11:00 UTC

Backup Retention Period

You can set the backup retention period when you create a DB instance. If you don't set the backup retention period, the default backup retention period is one day if you create the DB instance using the Amazon RDS API or the AWS CLI. The default backup retention period is seven days if you create the DB instance using the console. After you create a DB instance, you can modify the backup retention period. You can set the backup retention period to between 0 and 35 days. Setting the backup retention period to 0 disables automated backups. Manual snapshot limits (100 per region) do not apply to automated backups.

Important

An outage occurs if you change the backup retention period from 0 to a non-zero value or from a non-zero value to 0.

Disabling Automated Backups

You might want to temporarily disable automated backups in certain situations; for example, while loading large amounts of data.

Important

We highly discourage disabling automated backups because it disables point-in-time recovery. Disabling automatic backups for a DB instance deletes all existing automated backups for the instance. If you disable and then re-enable automated backups, you are only able to restore starting from the time you re-enabled automated backups.

In this example, you disable automated backups for a DB instance named *mydbinstance* by setting the backup retention parameter to 0.

Console

To disable automated backups immediately

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. For **Backup Retention Period**, choose **0 days**.
5. Choose **Continue**.
6. Choose **Apply Immediately**.
7. On the confirmation page, choose **Modify DB Instance** to save your changes and disable automated backups.

AWS CLI

To disable automated backups immediately, use the `modify-db-instance` command and set the backup retention period to 0 with `--apply-immediately`.

Example

The following example immediately disabled automatic backups.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--backup-retention-period 0 \
--apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--backup-retention-period 0 ^
--apply-immediately
```

To know when the modification is in effect, call `describe-db-instances` for the DB instance until the value for backup retention period is 0 and *mydbinstance* status is available.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

RDS API

To disable automated backups immediately, call the `ModifyDBInstance` action with the following parameters:

- `DBInstanceIdentifier = mydbinstance`
- `BackupRetentionPeriod = 0`

Example

```
https://rds.amazonaws.com/  
    ?Action=ModifyDBInstance  
    &DBInstanceIdentifier=mydbinstance  
    &BackupRetentionPeriod=0  
    &SignatureVersion=2  
    &SignatureMethod=HmacSHA256  
    &Timestamp=2009-10-14T17%3A48%3A21.746Z  
    &AWSAccessKeyId=<AWS Access Key ID>  
    &Signature=<Signature>
```

Enabling Automated Backups

If your DB instance doesn't have automated backups enabled, you can enable them at any time. You enable automated backups by setting the backup retention period to a positive non-zero value. When automated backups are enabled, your RDS instance and database is taken offline and a backup is immediately created.

In this example, you enable automated backups for a DB instance named *mydbinstance* by setting the backup retention period to a positive non-zero value (in this case, 3).

Console

To enable automated backups immediately

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. For **Backup Retention Period**, choose a positive nonzero value, for example 3 days.
5. Choose **Continue**.
6. Choose **Apply Immediately**.
7. On the confirmation page, choose **Modify DB Instance** to save your changes and enable automated backups.

AWS CLI

To enable automated backups immediately, use the AWS CLI `modify-db-instance` command.

In this example, we enable automated backups by setting the backup retention period to three days.

Include the following parameters:

- `--db-instance-identifier`
- `--backup-retention-period`
- `--apply-immediately` or `--no-apply-immediately`

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--backup-retention-period 3 \
--apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--backup-retention-period 3 ^
--apply-immediately
```

RDS API

To enable automated backups immediately, use the RDS API [ModifyDBInstance](#) operation.

In this example, we enable automated backups by setting the backup retention period to three days.

Include the following parameters:

- `DBInstanceIdentifier`
- `BackupRetentionPeriod`
- `ApplyImmediately = true`

Example

```
https://rds.amazonaws.com/
?Action=ModifyDBInstance
&DBInstanceIdentifier=mydbinstance
&BackupRetentionPeriod=3
&ApplyImmediately=true
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-14T17%3A48%3A21.746Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Retaining Automated Backups

When you delete a DB instance, you can retain automated backups.

Retained automated backups contain system snapshots and transaction logs from a DB instance. They also include your DB instance properties like allocated storage and DB instance class, which are required to restore it to an active instance.

You can retain automated backups for RDS instances running MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server engines.

You can restore or remove retained automated backups using the AWS Management Console, RDS API, and AWS CLI.

Retention Period

The system snapshots and transaction logs in a retained automated backup expire the same way that they expire for the source DB instance. Because there are no new snapshots or logs created for this

instance, the retained automated backups eventually expire completely. Effectively, they live as long their last system snapshot would have done, based on the settings for retention period the source instance had when you deleted it. Retained automated backups are removed by the system after their last system snapshot expires.

You can remove a retained automated backup in the same way that you can delete a DB instance. You can remove retained automated backups using the console or the RDS API operation [DeleteDBInstanceAutomatedBackup](#).

Final snapshots are independent of retained automated backups. We strongly suggest that you take a final snapshot even if you retain automated backups, because the retained automated backups eventually expire. The final snapshot doesn't expire.

Restoration

To view your retained automated backups, switch to the automated backups page. You can view individual snapshots associated with a retained automated backup on the database snapshots page in the console. Alternatively, you can describe individual snapshots associated with a retained automated backup. From there, you can restore a DB instance directly from one of those snapshots.

Restored DB instances are automatically associated with the default parameter and option groups. However, you can apply a custom parameter group and option group by specifying them during a restore.

In this example, you restore a DB instance to a point in time using the retained automated backup. First, you describe your retained automated backups, so you can see which of them to restore.

To describe your retained automated backups using the RDS API, call the [DescribeDBInstanceAutomatedBackups](#) action with one of the following parameters:

- `DBInstanceIdentifier`
- `DbiResourceId`

```
aws rds describe-db-instance-automated-backups --db-instance-  
identifier DBInstanceIdentifier  
OR  
aws rds describe-db-instance-automated-backups --dbi-resource-idDbiResourceId
```

Next, to restore your retained automated backup to a point in time, using the RDS API, call the [RestoreDBInstanceToPointInTime](#) action with the following parameters:

- `SourceDbiResourceId`
- `TargetDBInstanceIdentifier`

```
aws rds restore-db-instance-to-point-in-time --source-dbi-resource-id SourceDbiResourceId  
--target-db-instance-identifier TargetDBInstanceIdentifier --use-latest-restorable-time
```

Retention Costs

The cost of a retained automated backup is the cost of total storage of the system snapshots that are associated with it. There is no additional charge for transaction logs or instance metadata. All other pricing rules for backups apply to restorable instances.

For example, suppose that your total allocated storage of running instances is 100 GB. Suppose also that you have 50 GB of manual snapshots plus 75 GB of system snapshots associated with a retained automated backup. In this case, you are charged only for the additional 25 GB of backup storage, like this: (50 GB + 75 GB) – 100 GB = 25 GB.

Limitations and Recommendations

The following limitations apply to retained automated backups:

- The maximum number of retained automated backups in one region is 20. It's not included in the DB instances limit. You can have 20 running DB instances and an additional 20 retained automated backups at the same time.
- Retained automated backups don't contain information about parameters or option groups.
- You can restore a deleted instance to a point in time that is within the retention period at the time of delete.
- A retained automated backup can't be modified because it consists of system backups, transaction logs, and the DB instance properties that existed at the time you deleted the source instance.

Automated Backups with Unsupported MySQL Storage Engines

For the MySQL DB engine, automated backups are only supported for the InnoDB storage engine. Use of these features with other MySQL storage engines, including MyISAM, can lead to unreliable behavior while restoring from backups. Specifically, since storage engines like MyISAM don't support reliable crash recovery, your tables can be corrupted in the event of a crash. For this reason, we encourage you to use the InnoDB storage engine.

- To convert existing MyISAM tables to InnoDB tables, you can use the `ALTER TABLE` command, for example: `ALTER TABLE table_name ENGINE=innodb, ALGORITHM=COPY;`
- If you choose to use MyISAM, you can attempt to manually repair tables that become damaged after a crash by using the `REPAIR` command. For more information, see [REPAIR TABLE Syntax](#) in the MySQL documentation. However, as noted in the MySQL documentation, there is a good chance that you might not be able to recover all your data.
- If you want to take a snapshot of your MyISAM tables before restoring, follow these steps:
 1. Stop all activity to your MyISAM tables (that is, close all sessions).

You can close all sessions by calling the `mysql.rds_kill` command for each process that is returned from the `SHOW FULL PROCESSLIST` command.

2. Lock and flush each of your MyISAM tables. For example, the following commands lock and flush two tables named `myisam_table1` and `myisam_table2`:

```
mysql> FLUSH TABLES myisam_table, myisam_table2 WITH READ LOCK;
```

3. Create a snapshot of your DB instance. When the snapshot has completed, release the locks and resume activity on the MyISAM tables. You can release the locks on your tables using the following command:

```
mysql> UNLOCK TABLES;
```

These steps force MyISAM to flush data stored in memory to disk, which ensures a clean start when you restore from a DB snapshot. For more information on creating a DB snapshot, see [Creating a DB Snapshot \(p. 216\)](#).

Automated Backups with Unsupported MariaDB Storage Engines

For the MariaDB DB engine, automated backups are only supported with the InnoDB storage engine (version 10.2 and later) and XtraDB storage engine (versions 10.0 and 10.1). Use of these features with other MariaDB storage engines, including Aria, might lead to unreliable behavior while restoring from backups. Even though Aria is a crash-resistant alternative to MyISAM, your tables can still be corrupted in the event of a crash. For this reason, we encourage you to use the XtraDB storage engine.

- To convert existing Aria tables to InnoDB tables, you can use the `ALTER TABLE` command. For example: `ALTER TABLE table_name ENGINE=innodb, ALGORITHM=COPY;`
- To convert existing Aria tables to XtraDB tables, you can use the `ALTER TABLE` command. For example: `ALTER TABLE table_name ENGINE=xtradb, ALGORITHM=COPY;`
- If you choose to use Aria, you can attempt to manually repair tables that become damaged after a crash by using the `REPAIR TABLE` command. For more information, see <http://mariadb.com/kb/en/mariadb/repair-table/>.
- If you want to take a snapshot of your Aria tables before restoring, follow these steps:
 1. Stop all activity to your Aria tables (that is, close all sessions).
 2. Lock and flush each of your Aria tables.
 3. Create a snapshot of your DB instance. When the snapshot has completed, release the locks and resume activity on the Aria tables. These steps force Aria to flush data stored in memory to disk, thereby ensuring a clean start when you restore from a DB snapshot.

Creating a DB Snapshot

Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. Creating this DB snapshot on a Single-AZ DB instance results in a brief I/O suspension that can last from a few seconds to a few minutes, depending on the size and class of your DB instance. Multi-AZ DB instances are not affected by this I/O suspension since the backup is taken on the standby.

When you create a DB snapshot, you need to identify which DB instance you are going to back up, and then give your DB snapshot a name so you can restore from it later. The amount of time it takes to create a snapshot varies with the size your databases. Since the snapshot includes the entire storage volume, the size of files, such as temporary files, also affects the amount of time it takes to create the snapshot.

You can create a DB snapshot using the AWS Management Console, the AWS CLI, or the RDS API.

AWS Management Console

To create a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. In the list of DB instances, choose the DB instance for which you want to take a snapshot.
4. For **Actions**, choose **Take snapshot**.

The **Take DB Snapshot** window appears.

5. Type the name of the snapshot in the **Snapshot Name** box.

Take DB Snapshot

This feature is currently supported for InnoDB storage engine only. If you are using MyISAM, refer to details [here](#). 

Settings

To take a snapshot of this DB instance you must provide a name for the snapshot.

DB instance

The unique key that identifies a DB instance. This parameter isn't case-sensitive.

mydbinstance3

Snapshot name

The Identifier for the DB Snapshot.

Cancel

Take Snapshot

6. Choose **Take Snapshot**.

CLI

When you create a DB snapshot using the AWS CLI, you need to identify which DB instance you are going to back up, and then give your DB snapshot a name so you can restore from it later. You can do this by using the AWS CLI [create-db-snapshot](#) command with the following parameters:

- `--db-instance-identifier`
- `--db-snapshot-identifier`

In this example, you create a DB snapshot called `mydbsnapshot` for a DB instance called `mydbinstance`.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-snapshot /  
  --db-instance-identifier mydbinstance /  
  --db-snapshot-identifier mydbsnapshot
```

For Windows:

```
aws rds create-db-snapshot ^  
  --db-instance-identifier mydbinstance ^  
  --db-snapshot-identifier mydbsnapshot
```

API

When you create a DB snapshot using the Amazon RDS API, you need to identify which DB instance you are going to back up, and then give your DB snapshot a name so you can restore from it later. You can do this by using the Amazon RDS API [CreateDBSnapshot](#) command with the following parameters:

- `DBInstanceIdentifier`
- `DBSnapshotIdentifier`

Restoring from a DB Snapshot

Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. You can create a DB instance by restoring from this DB snapshot. When you restore the DB instance, you provide the name of the DB snapshot to restore from, and then provide a name for the new DB instance that is created from the restore. You can't restore from a DB snapshot to an existing DB instance; a new DB instance is created when you restore.

You can restore a DB instance and use a different storage type than the source DB snapshot. In this case, the restoration process is slower because of the additional work required to migrate the data to the new storage type. If you restore to or from Magnetic (Standard) storage, the migration process is the slowest. That's because Magnetic storage doesn't have the IOPS capability of Provisioned IOPS or General Purpose (SSD) storage.

Note

You can't restore a DB instance from a DB snapshot that is both shared and encrypted. Instead, you can make a copy of the DB snapshot and restore the DB instance from the copy.

Parameter Group Considerations

We recommend that you retain the parameter group for any DB snapshots you create, so that you can associate your restored DB instance with the correct parameter group. You can specify the parameter group when you restore the DB instance.

Security Group Considerations

When you restore a DB instance, the default security group is associated with the restored instance. As soon as the restore is complete and your new DB instance is available, you must associate any custom security groups used by the instance you restored from. You must apply these changes by using the RDS console's *Modify* command, the `ModifyDBInstance` Amazon RDS API, or the AWS CLI `modify-db-instance` command.

Option Group Considerations

When you restore a DB instance, the option group associated with the DB snapshot is associated with the restored DB instance after it is created. For example, if the DB snapshot you are restoring from uses Oracle Transparent Data Encryption, the restored DB instance will use the same option group.

When you assign an option group to a DB instance, the option group is also linked to the supported platform the DB instance is on, either VPC or EC2-Classic (non-VPC). If a DB instance is in a VPC, the option group associated with the DB instance is linked to that VPC. This means that you can't use the option group assigned to a DB instance if you attempt to restore the instance into a different VPC or onto a different platform. If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. For persistent or permanent options, when restoring a DB instance into a different VPC you must create a new option group that includes the persistent or permanent option.

Microsoft SQL Server Considerations

When you restore a Microsoft SQL Server DB snapshot to a new instance, you can always restore to the same edition as your snapshot. In some cases, you can also change the edition of the DB instance. The following are the limitations when you change editions:

- The DB snapshot must have enough storage allocated for the new edition.

- Only the following edition changes are supported:
 - From Standard Edition to Enterprise Edition
 - From Web Edition to Standard Edition or Enterprise Edition
 - From Express Edition to Web Edition, Standard Edition or Enterprise Edition

If you want to change from one edition to a new edition that is not supported by restoring a snapshot, you can try using the native backup and restore feature. SQL Server verifies whether or not your database is compatible with the new edition based on what SQL Server features you have enabled on the database. For more information, see [Importing and Exporting SQL Server Databases \(p. 533\)](#).

Oracle Considerations

If you use Oracle GoldenGate, always retain the parameter group with the `compatible` parameter. When you restore a DB instance from a DB snapshot, you must specify the parameter group that has a matching or greater `compatible` parameter value.

You can upgrade a DB snapshot while it is still a DB snapshot, before you restore it. For more information, see [Upgrading an Oracle DB Snapshot \(p. 774\)](#).

Restoring from a Snapshot

You can restore a DB instance from a DB snapshot using the AWS Management Console, the AWS CLI, or the RDS API.

AWS Management Console

To restore a DB instance from a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Snapshots**.
3. Choose the DB snapshot that you want to restore from.
4. For **Actions**, choose **Restore Snapshot**.
5. On the **Restore DB Instance** page, for **DB Instance Identifier**, enter the name for your restored DB instance.
6. Choose **Restore DB Instance**.
7. If you want to restore the functionality of the DB instance to that of the DB instance that the snapshot was created from, you must modify the DB instance to use the security group. The next steps assume that your DB instance is in a VPC. If your DB instance is not in a VPC, use the EC2 Management Console to locate the security group you need for the DB instance.
 - a. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
 - b. In the navigation pane, choose **Security Groups**.
 - c. Select the security group that you want to use for your DB instances. If necessary, add rules to link the security group to a security group for an EC2 instance. For more information, see [A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC \(p. 414\)](#).

CLI

To restore a DB instance from a DB snapshot, use the AWS CLI command `restore-db-instance-from-db-snapshot`.

In this example, you restore from a previously created DB snapshot named *mydbsnapshot*. You restore to a new DB instance named *mynewdbinstance*.

Example

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-from-db-snapshot \
--db-instance-identifier mynewdbinstance \
--db-snapshot-identifier mydbsnapshot
```

For Windows:

```
aws rds restore-db-instance-from-db-snapshot ^
--db-instance-identifier mynewdbinstance ^
--db-snapshot-identifier mydbsnapshot
```

This command returns output similar to the following:

```
DBINSTANCE mynewdbinstance db.m3.large MySQL      50          sa           creating  3  n
5.6.40 general-public-license
```

After the DB instance has been restored, you must add the DB instance to the security group used by the DB instance used to create the DB snapshot if you want the same functionality as that of the previous DB instance.

API

To restore a DB instance from a DB snapshot, call the Amazon RDS API function [RestoreDBInstanceFromDBSnapshot](#) with the following parameters:

- **DBInstanceIdentifier**
- **DBSnapshotIdentifier**

Copying a Snapshot

With Amazon RDS, you can copy automated or manual DB snapshots. After you copy a snapshot, the copy is a manual snapshot.

You can copy a snapshot within the same AWS Region, you can copy a snapshot across AWS Regions, and you can copy a snapshot across AWS accounts.

Copying an automated snapshot to another AWS account is a two-step process: You first create a manual snapshot from the automated snapshot, and then you copy the manual snapshot to the other account.

Limitations

The following are some limitations when you copy snapshots:

- You can't copy a snapshot to or from the following AWS Regions: China (Beijing) or China (Ningxia).
- You can copy a snapshot between AWS GovCloud (US-East) and AWS GovCloud (US-West), but you can't copy a snapshot between these AWS GovCloud (US) regions and other AWS Regions.
- If you delete a source snapshot before the target snapshot becomes available, the snapshot copy may fail. Verify that the target snapshot has a status of **AVAILABLE** before you delete a source snapshot.
- You can have up to five snapshot copy requests in progress to a single destination region per account.
- You can't copy a DB snapshot across regions if it was created from an Oracle DB instance that is using AWS CloudHSM Classic to store TDE keys.
- Depending on the regions involved and the amount of data to be copied, a cross-region snapshot copy can take hours to complete. If there is a large number of cross-region snapshot copy requests from a given source AWS Region, Amazon RDS might put new cross-region copy requests from that source AWS Region into a queue until some in-progress copies complete. No progress information is displayed about copy requests while they are in the queue. Progress information is displayed when the copy starts.

Snapshot Retention

Amazon RDS deletes automated snapshots at the end of their retention period, when you disable automated snapshots for a DB instance, or when you delete a DB instance. If you want to keep an automated snapshot for a longer period, copy it to create a manual snapshot, which is retained until you delete it. Amazon RDS storage costs might apply to manual snapshots if they exceed your default storage space.

For more information about backup storage costs, see [Amazon RDS Pricing](#).

Copying Shared Snapshots

You can copy snapshots shared to you by other AWS accounts. If you are copying an encrypted snapshot that has been shared from another AWS account, you must have access to the KMS encryption key that was used to encrypt the snapshot.

You can copy a shared DB snapshot across regions, provided that the snapshot is unencrypted. However, if the shared DB snapshot is encrypted, you can only copy it in the same AWS Region.

Handling Encryption

You can copy a snapshot that has been encrypted using an AWS KMS encryption key. If you copy an encrypted snapshot, the copy of the snapshot must also be encrypted. If you copy an encrypted

snapshot within the same AWS Region, you can encrypt the copy with the same KMS encryption key as the original snapshot, or you can specify a different KMS encryption key. If you copy an encrypted snapshot across regions, you can't use the same KMS encryption key for the copy as used for the source snapshot, because KMS keys are region-specific. Instead, you must specify a KMS key valid in the destination AWS Region.

You can also encrypt a copy of an unencrypted snapshot. This way, you can quickly add encryption to a previously unencrypted DB instance. That is, you can create a snapshot of your DB instance when you are ready to encrypt it, and then create a copy of that snapshot and specify a KMS encryption key to encrypt that snapshot copy. You can then restore an encrypted DB instance from the encrypted snapshot.

Copying Snapshots Across AWS Regions

When you copy a snapshot to an AWS Region that is different from the source snapshot's AWS Region, the first copy is a full snapshot copy, even if you copy an incremental snapshot. A full snapshot copy contains all of the data and metadata required to restore the DB instance. After the first snapshot copy, you can copy incremental snapshots of the same DB instance to the same destination region.

An incremental snapshot contains only the data that has changed after the most recent snapshot of the same DB instance. Incremental snapshot copying is faster and results in lower storage costs than full snapshot copying. Incremental snapshot copying across AWS Regions is supported for both unencrypted and encrypted snapshots.

Depending on the AWS Regions involved and the amount of data to be copied, a cross-region snapshot copy can take hours to complete. In some cases, there might be a large number of cross-region snapshot copy requests from a given source AWS Region. In these cases, Amazon RDS might put new cross-region copy requests from that source AWS Region into a queue until some in-progress copies complete. No progress information is displayed about copy requests while they are in the queue. Progress information is displayed when the copy starts.

Note

When you copy a source snapshot that is a snapshot copy, the copy isn't incremental because the snapshot copy doesn't include the required metadata for incremental copies.

Option Group Considerations

Option groups are specific to the AWS Region that they are created in, and you can't use an option group from one AWS Region in another AWS Region.

When you copy a snapshot across regions, you can specify a new option group for the snapshot. We recommend that you prepare the new option group before you copy the snapshot. In the destination AWS Region, create an option group with the same settings as the original DB instance . If one already exists in the new AWS Region, you can use that one.

If you copy a snapshot and you don't specify a new option group for the snapshot, when you restore it the DB instance gets the default option group. To give the new DB instance the same options as the original, you must do the following:

1. In the destination AWS Region, create an option group with the same settings as the original DB instance . If one already exists in the new AWS Region, you can use that one.
2. After you restore the snapshot in the destination AWS Region, modify the new DB instance and add the new or existing option group from the previous step.

Parameter Group Considerations

When you copy a snapshot across regions, the copy doesn't include the parameter group used by the original DB instance . When you restore a snapshot to create a new DB instance , that DB instance gets

the default parameter group for the AWS Region it is created in. To give the new DB instance the same parameters as the original, you must do the following:

1. In the destination AWS Region, create a DB parameter group with the same settings as the original DB instance . If one already exists in the new AWS Region, you can use that one.
2. After you restore the snapshot in the destination AWS Region, modify the new DB instance and add the new or existing parameter group from the previous step.

Copying a DB Snapshot

Use the procedures in this topic to copy a DB snapshot. For an overview of copying a snapshot, see [Copying a Snapshot \(p. 221\)](#)

For each AWS account, you can copy up to five DB snapshots at a time from one AWS Region to another. If you copy a DB snapshot to another AWS Region, you create a manual DB snapshot that is retained in that AWS Region. Copying a DB snapshot out of the source AWS Region incurs Amazon RDS data transfer charges.

For more information about data transfer pricing, see [Amazon RDS Pricing](#).

After the DB snapshot copy has been created in the new AWS Region, the DB snapshot copy behaves the same as all other DB snapshots in that AWS Region.

You can copy a DB snapshot using the AWS Management Console, the AWS CLI, or the RDS API.

AWS Management Console

This procedure copies an encrypted or unencrypted DB snapshot, in the same AWS Region or across regions, by using the AWS Management Console.

To copy a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Snapshots**.
3. Select the DB snapshot that you want to copy.
4. Choose **Actions**, and then choose **Copy Snapshot**. The **Make Copy of DB Snapshot** page appears.

Make Copy of DB Snapshot?

Settings

Source DB Snapshot

DB Snapshot Identifier for the automated snapshot being copied.
mydbinstancesnapshot

Destination Region [Info](#)

US East (N. Virginia) ▾

New DB Snapshot Identifier

DB Snapshot Identifier for the new snapshot

Target Option Group (Optional) [Info](#)

No preference ▾

Copy Tags [Info](#)

i Please note that depending on the amount of data to be copied and the Region you choose, this operation could take several hours to complete and the display on the progress bar could be delayed until setup is complete.

Encryption

Encryption [Info](#)

Enable encryption [Learn more](#)

Select to encrypt the given instance. Master key ids and aliases appear in the list after they have been created using the Key Management Service(KMS) console.

Disable encryption

[Cancel](#)

[Copy Snapshot](#)

5. (Optional) To copy the DB snapshot to a different AWS Region, for **Destination Region**, choose the new AWS Region.

Note

The destination AWS Region must have the same database engine version available as the source AWS Region.

6. For **New DB Snapshot Identifier**, type the name of the DB snapshot copy.
7. (Optional) For **Target Option Group**, choose a new option group.

Specify this option if you are copying a snapshot from one AWS Region to another, and your DB instance uses a non-default option group.

If your source DB instance uses Transparent Data Encryption for Oracle or Microsoft SQL Server, you must specify this option when copying across regions. For more information, see [Option Group Considerations \(p. 222\)](#).

8. (Optional) Select **Copy Tags** to copy tags and values from the snapshot to the copy of the snapshot.
9. (Optional) For **Enable Encryption**, choose one of the following options:
 - Choose **Disable encryption** if the DB snapshot isn't encrypted and you don't want to encrypt the copy.
 - Choose **Enable encryption** if the DB snapshot isn't encrypted but you want to encrypt the copy. In this case, for **Master Key**, specify the KMS key identifier to use to encrypt the DB snapshot copy.
 - Choose **Enable encryption** if the DB snapshot is encrypted. In this case, you must encrypt the copy, so **Yes** is already selected. For **Master Key**, specify the KMS key identifier to use to encrypt the DB snapshot copy.
10. Choose **Copy Snapshot**.

CLI

You can copy a DB snapshot by using the AWS CLI command `copy-db-snapshot`. If you are copying the snapshot to a new AWS Region, run the command in the new AWS Region.

The following options are used to copy a DB snapshot. Not all options are required for all scenarios. Use the descriptions and the examples that follow to determine which options to use.

- `--source-db-snapshot-identifier` – The identifier for the source DB snapshot.
 - If the source snapshot is in the same AWS Region as the copy, specify a valid DB snapshot identifier. For example, `rds:mysql-instance1-snapshot-20130805`.
 - If the source snapshot is in a different AWS Region than the copy, specify a valid DB snapshot ARN. For example, `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`.
 - If you are copying from a shared manual DB snapshot, this parameter must be the Amazon Resource Name (ARN) of the shared DB snapshot.
 - If you are copying an encrypted snapshot this parameter must be in the ARN format for the source AWS Region, and must match the `SourceDBSnapshotIdentifier` in the `PreSignedUrl` parameter.
- `--target-db-snapshot-identifier` – The identifier for the new copy of the encrypted DB snapshot.
- `--copy-tags` – Include the copy tags option to copy tags and values from the snapshot to the copy of the snapshot.
- `--option-group-name` – The option group to associate with the copy of the snapshot.

Specify this option if you are copying a snapshot from one AWS Region to another, and your DB instance uses a non-default option group.

If your source DB instance uses Transparent Data Encryption for Oracle or Microsoft SQL Server, you must specify this option when copying across regions. For more information, see [Option Group Considerations \(p. 222\)](#).

- `--kms-key-id` – The AWS KMS key ID for an encrypted DB snapshot. The KMS key ID is the Amazon Resource Name (ARN), KMS key identifier, or the KMS key alias for the KMS encryption key.
 - If you copy an encrypted DB snapshot from your AWS account, you can specify a value for this parameter to encrypt the copy with a new KMS encryption key. If you don't specify a value for this parameter, then the copy of the DB snapshot is encrypted with the same KMS key as the source DB snapshot.

- If you copy an encrypted DB snapshot that is shared from another AWS account, then you must specify a value for this parameter.
- If you specify this parameter when you copy an unencrypted snapshot, the copy is encrypted.
- If you copy an encrypted snapshot to a different AWS Region, then you must specify a KMS key for the destination AWS Region. KMS encryption keys are specific to the AWS Region that they are created in, and you cannot use encryption keys from one AWS Region in another AWS Region.
- `--source-region` – The ID of the AWS Region of the source DB snapshot. If you copy an encrypted snapshot to a different AWS Region, then you must specify this option.

Example From Unencrypted, To Same Region

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the same AWS Region as the source snapshot. When the copy is made, all tags on the original snapshot are copied to the snapshot copy.

For Linux, OS X, or Unix:

```
aws rds copy-db-snapshot \  
  --source-db-snapshot-identifier mysql-instance1-snapshot-20130805 \  
  --target-db-snapshot-identifier mydbsnapshotcopy \  
  --copy-tags
```

For Windows:

```
aws rds copy-db-snapshot ^  
  --source-db-snapshot-identifier mysql-instance1-snapshot-20130805 ^  
  --target-db-snapshot-identifier mydbsnapshotcopy ^  
  --copy-tags
```

Example From Unencrypted, Across Regions

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the AWS Region in which the command is run.

For Linux, OS X, or Unix:

```
aws rds copy-db-snapshot \  
  --source-db-snapshot-identifier arn:aws:rds:us-east-1:123456789012:snapshot:mysql-  
instance1-snapshot-20130805 \  
  --target-db-snapshot-identifier mydbsnapshotcopy
```

For Windows:

```
aws rds copy-db-snapshot ^  
  --source-db-snapshot-identifier arn:aws:rds:us-east-1:123456789012:snapshot:mysql-  
instance1-snapshot-20130805 ^  
  --target-db-snapshot-identifier mydbsnapshotcopy
```

Example From Encrypted, Across Regions

The following code example copies an encrypted DB snapshot from the us-west-2 region in the us-east-1 region. Run the command in the us-east-1 region.

For Linux, OS X, or Unix:

```
aws rds copy-db-snapshot \  
  --source-region us-west-2 \  
  --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-  
instance1-snapshot-20130805 \  
  --target-db-snapshot-identifier mydbsnapshotcopy \  
  --source-key-id alias/AmazonRDS \  
  --target-key-id alias/AmazonRDS
```

```
--source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-
instance1-snapshot-20161115 \
--target-db-snapshot-identifier mydbsnapshotcopy \
--source-region us-west-2 \
--kms-key-id my-us-east-1-key \
--option-group-name custom-option-group-name
```

For Windows:

```
aws rds copy-db-snapshot ^
--source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-
instance1-snapshot-20161115 ^
--target-db-snapshot-identifier mydbsnapshotcopy ^
--source-region us-west-2 ^
--kms-key-id my-us-east-1-key ^
--option-group-name custom-option-group-name
```

API

You can copy a DB snapshot by using the Amazon RDS API action [CopyDBSnapshot](#). If you are copying the snapshot to a new AWS Region, perform the action in the new AWS Region.

The following parameters are used to copy a DB snapshot. Not all parameters are required for all scenarios. Use the descriptions and the examples that follow to determine which parameters to use.

- **SourceDBSnapshotIdentifier** – The identifier for the source DB snapshot.
 - If the source snapshot is in the same AWS Region as the copy, specify a valid DB snapshot identifier. For example, `rds:mysql-instance1-snapshot-20130805`.
 - If the source snapshot is in a different AWS Region than the copy, specify a valid DB snapshot ARN. For example, `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`.
 - If you are copying from a shared manual DB snapshot, this parameter must be the Amazon Resource Name (ARN) of the shared DB snapshot.
 - If you are copying an encrypted snapshot this parameter must be in the ARN format for the source AWS Region, and must match the `SourceDBSnapshotIdentifier` in the `PreSignedUrl` parameter.
- **TargetDBSnapshotIdentifier** – The identifier for the new copy of the encrypted DB snapshot.
- **CopyTags** – Set this parameter to `true` to copy tags and values from the snapshot to the copy of the snapshot. The default is `false`.
- **OptionGroupName** – The option group to associate with the copy of the snapshot.

Specify this parameter if you are copying a snapshot from one AWS Region to another, and your DB instance uses a non-default option group.

If your source DB instance uses Transparent Data Encryption for Oracle or Microsoft SQL Server, you must specify this parameter when copying across regions. For more information, see [Option Group Considerations \(p. 222\)](#).

- **KmsKeyId** – The AWS KMS key ID for an encrypted DB snapshot. The KMS key ID is the Amazon Resource Name (ARN), KMS key identifier, or the KMS key alias for the KMS encryption key.
 - If you copy an encrypted DB snapshot from your AWS account, you can specify a value for this parameter to encrypt the copy with a new KMS encryption key. If you don't specify a value for this parameter, then the copy of the DB snapshot is encrypted with the same KMS key as the source DB snapshot.
 - If you copy an encrypted DB snapshot that is shared from another AWS account, then you must specify a value for this parameter.

- If you specify this parameter when you copy an unencrypted snapshot, the copy is encrypted.
- If you copy an encrypted snapshot to a different AWS Region, then you must specify a KMS key for the destination AWS Region. KMS encryption keys are specific to the AWS Region that they are created in, and you cannot use encryption keys from one AWS Region in another AWS Region.
- **PreSignedUrl** – The URL that contains a Signature Version 4 signed request for the `CopyDBSnapshot` API action in the source AWS Region that contains the source DB snapshot to copy.

You must specify this parameter when you copy an encrypted DB snapshot from another AWS Region by using the Amazon RDS API. You can specify the source region option instead of this parameter when you copy an encrypted DB snapshot from another AWS Region by using the AWS CLI.

The presigned URL must be a valid request for the `CopyDBSnapshot` API action that can be executed in the source AWS Region that contains the encrypted DB snapshot to be copied. The presigned URL request must contain the following parameter values:

- **DestinationRegion** – The AWS Region that the encrypted DB snapshot will be copied to. This AWS Region is the same one where the `CopyDBSnapshot` action is called that contains this presigned URL.

For example, if you copy an encrypted DB snapshot from the us-west-2 region to the us-east-1 region, then you call the `CopyDBSnapshot` action in the us-east-1 region and provide a presigned URL that contains a call to the `CopyDBSnapshot` action in the us-west-2 region. For this example, the `DestinationRegion` in the presigned URL must be set to the us-east-1 region.

- **KmsKeyId** – The KMS key identifier for the key to use to encrypt the copy of the DB snapshot in the destination AWS Region. This is the same identifier for both the `CopyDBSnapshot` action that is called in the destination AWS Region, and the action contained in the presigned URL.
- **SourceDBSnapshotIdentifier** – The DB snapshot identifier for the encrypted snapshot to be copied. This identifier must be in the Amazon Resource Name (ARN) format for the source AWS Region. For example, if you are copying an encrypted DB snapshot from the us-west-2 region, then your `SourceDBSnapshotIdentifier` looks like the following example: `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20161115`.

For more information on Signature Version 4 signed requests, see the following:

- [Authenticating Requests: Using Query Parameters \(AWS Signature Version 4\)](#) in the Amazon Simple Storage Service API Reference
- [Signature Version 4 Signing Process](#) in the AWS General Reference

Example From Unencrypted, To Same Region

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the same AWS Region as the source snapshot. When the copy is made, all tags on the original snapshot are copied to the snapshot copy.

```
https://rds.us-west-1.amazonaws.com/
?Action=CopyDBSnapshot
&CopyTags=true
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&SourceDBSnapshotIdentifier=mysql-instance1-snapshot-20130805
&TargetDBSnapshotIdentifier=mydbsnapshotcopy
&Version=2013-09-09
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
&X-Amz-Date=20140429T175351Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Example From Unencrypted, Across Regions

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the `us-west-1` region.

```
https://rds.us-west-1.amazonaws.com/
?Action=CopyDBSnapshot
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Asnapshot%3Amysql-
instance1-snapshot-20130805
&TargetDBSnapshotIdentifier=mydbsnapshotcopy
&Version=2013-09-09
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
&X-Amz-Date=20140429T175351Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Example From Encrypted, Across Regions

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the `us-east-1` region.

```
https://rds.us-east-1.amazonaws.com/
?Action=CopyDBSnapshot
&KmsKeyId=my-us-east-1-key
&OptionGroupName=custom-option-group-name
&PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
%25253FAction%253DCopyDBSnapshot
%2526DestinationRegion%253Dus-east-1
%2526KmsKeyId%253Dmy-us-east-1-key
%2526SourceDBSnapshotIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Asnapshot%25253Amysql-instance1-snapshot-20161115
%2526SignatureMethod%253DHmacSHA256
%2526SignatureVersion%253D4
%2526Version%253D2014-10-31
%2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
%2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
%2526X-Amz-Date%253D20161117T215409Z
%2526X-Amz-Expires%253D3600
%2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
%2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fcfa613
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-west-2%3A123456789012%3Asnapshot
%3Amysql-instance1-snapshot-20161115
&TargetDBSnapshotIdentifier=mydbsnapshotcopy
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
&X-Amz-Date=20161117T221704Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

Sharing a DB Snapshot

Using Amazon RDS, you can share a manual DB snapshot in the following ways:

- Sharing a manual DB snapshot, whether encrypted or unencrypted, enables authorized AWS accounts to copy the snapshot.
- Sharing an unencrypted manual DB snapshot enables authorized AWS accounts to directly restore a DB instance from the snapshot instead of taking a copy of it and restoring from that. However, you can't restore a DB instance from a DB snapshot that is both shared and encrypted. Instead, you can make a copy of the DB snapshot and restore the DB instance from the copy.

Note

To share an automated DB snapshot, create a manual DB snapshot by copying the automated snapshot, and then share that copy.

For more information on copying a snapshot, see [Copying a Snapshot \(p. 221\)](#). For more information on restoring a DB instance from a DB snapshot, see [Restoring from a DB Snapshot \(p. 218\)](#).

You can share a manual snapshot with up to 20 other AWS accounts. You can also share an unencrypted manual snapshot as public, which makes the snapshot available to all AWS accounts. Take care when sharing a snapshot as public so that none of your private information is included in any of your public snapshots.

The following limitations apply when sharing manual snapshots with other AWS accounts:

- When you restore a DB instance from a shared snapshot using the AWS Command Line Interface (AWS CLI) or Amazon RDS API, you must specify the Amazon Resource Name (ARN) of the shared snapshot as the snapshot identifier.
- You cannot share a DB snapshot that uses an option group with permanent or persistent options.

A *permanent option* cannot be removed from an option group. Option groups with persistent options cannot be removed from a DB instance once the option group has been assigned to the DB instance.

The following table lists permanent and persistent options and their related DB engines.

Option Name	Persistent	Permanent	DB Engine
TDE	Yes	No	Microsoft SQL Server Enterprise Edition
TDE	Yes	Yes	Oracle Enterprise Edition
TDE_HSM	Yes	Yes	Oracle Enterprise Edition
Timezone	Yes	Yes	Oracle Enterprise Edition Oracle Standard Edition Oracle Standard Edition One Oracle Standard Edition Two

Sharing an Encrypted Snapshot

You can share DB snapshots that have been encrypted "at rest" using the AES-256 encryption algorithm, as described in [Encrypting Amazon RDS Resources \(p. 389\)](#). To do this, you must take the following steps:

1. Share the AWS Key Management Service (AWS KMS) encryption key that was used to encrypt the snapshot with any accounts that you want to be able to access the snapshot.

You can share AWS KMS encryption keys with another AWS account by adding the other account to the KMS key policy. For details on updating a key policy, see [Key Policies](#) in the *AWS KMS Developer Guide*. For an example of creating a key policy, see [Allowing Access to an AWS KMS Encryption Key \(p. 231\)](#) later in this topic.

2. Use the AWS Management Console, AWS CLI, or Amazon RDS API to share the encrypted snapshot with the other accounts.

These restrictions apply to sharing encrypted snapshots:

- You can't share encrypted snapshots as public.
- You can't share Oracle or Microsoft SQL Server snapshots that are encrypted using Transparent Data Encryption (TDE).
- You can't share a snapshot that has been encrypted using the default AWS KMS encryption key of the AWS account that shared the snapshot.

Allowing Access to an AWS KMS Encryption Key

For another AWS account to copy an encrypted DB snapshot shared from your account, the account that you share your snapshot with must have access to the KMS key that encrypted the snapshot. To allow another AWS account access to an AWS KMS key, update the key policy for the KMS key with the ARN of the AWS account that you are sharing to as a `Principal` in the KMS key policy, and then allow the `kms:CreateGrant` action.

After you have given an AWS account access to your KMS encryption key, to copy your encrypted snapshot, that AWS account must create an AWS Identity and Access Management (IAM) user if it doesn't already have one. In addition, that AWS account must also attach an IAM policy to that IAM user that allows the IAM user to copy an encrypted DB snapshot using your KMS key. The account must be an IAM user and cannot be a root AWS account identity due to KMS security restrictions.

In the following key policy example, user `111122223333` is the owner of the KMS encryption key, and user `444455556666` is the account that the key is being shared with. This updated key policy gives the AWS account access to the KMS key by including the ARN for the root AWS account identity for user `444455556666` as a `Principal` for the policy, and by allowing the `kms:CreateGrant` action.

```
{  
    "Id": "key-policy-1",  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "Allow use of the key",  
            "Effect": "Allow",  
            "Principal": {"AWS": [  
                "arn:aws:iam::111122223333:user/KeyUser",  
                "arn:aws:iam::444455556666:root"  
            ]},  
            "Action": [  
                "kms:CreateGrant",  
                "kms:Encrypt",  
                "kms:Decrypt",  
                "kms:ReEncrypt",  
                "kms:GenerateDataKey",  
                "kms:DescribeKey",  
                "kms:ListAliases",  
                "kms:ListGrants",  
                "kms:ListKeys",  
                "kms:PutGrant",  
                "kms:RevokeGrant"  
            ]  
        }  
    ]  
}
```

```
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/KeyUser",
        "arn:aws:iam::444455556666:root"
    ]},
    "Action": [
        "kms>CreateGrant",
        "kms>ListGrants",
        "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
]
```

Creating an IAM Policy to Enable Copying of the Encrypted Snapshot

Once the external AWS account has access to your KMS key, the owner of that AWS account can create a policy that allows an IAM user created for that account to copy an encrypted snapshot encrypted with that KMS key.

The following example shows a policy that can be attached to an IAM user for AWS account 444455556666 that enables the IAM user to copy a shared snapshot from AWS account 111122223333 that has been encrypted with the KMS key c989c1dd-a3f2-4a5d-8d96-e793d082ab26 in the us-west-2 region.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms>CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms>CreateGrant",
                "kms>ListGrants",
                "kms:RevokeGrant"
            ],
        }
    ]
}
```

```
        "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-a3f2-4a5d-8d96-e793d082ab26"],  
        "Condition": {  
            "Bool": {  
                "kms:GrantIsForAWSResource": true  
            }  
        }  
    }  
}
```

For details on updating a key policy, see [Key Policies in the AWS KMS Developer Guide](#).

Sharing a Snapshot

You can share a DB snapshot using the AWS Management Console, the AWS CLI, or the RDS API.

AWS Management Console

Using the Amazon RDS console, you can share a manual DB snapshot with up to 20 AWS accounts. You can also use the console to stop sharing a manual snapshot with one or more accounts.

To share a manual DB snapshot by using the Amazon RDS console

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Snapshots**.
3. Select the manual snapshot that you want to share.
4. Choose **Actions**, and then choose **Share Snapshot**.
5. Choose one of the following options for **DB snapshot visibility**.
 - If the source is unencrypted, choose **Public** to permit all AWS accounts to restore a DB instance from your manual DB snapshot, or choose **Private** to permit only AWS accounts that you specify to restore a DB instance from your manual DB snapshot.

Warning

If you set **DB snapshot visibility** to **Public**, all AWS accounts can restore a DB instance from your manual DB snapshot and have access to your data. Do not share any manual DB snapshots that contain private information as **Public**.

- If the source is encrypted, **DB snapshot visibility** is set as **Private** because encrypted snapshots can't be shared as public.
6. For **AWS Account ID**, type the AWS account identifier for an account that you want to permit to restore a DB instance from your manual snapshot, and then choose **Add**. Repeat to include additional AWS account identifiers, up to 20 AWS accounts.

If you make an error when adding an AWS account identifier to the list of permitted accounts, you can delete it from the list by choosing **Delete** at the right of the incorrect AWS account identifier.

Snapshot permissions

Preferences

You are sharing an unencrypted DB snapshot. When you share an unencrypted DB snapshot, you give the other account permission to make a copy of the DB snapshot and to restore a database from your DB snapshot.

DB snapshot

testoracletags-snap

DB snapshot visibility

- Private
 Public

AWS account ID

Add

AWS account ID

Delete

Please add AWS account ID

Cancel

Save

- After you have added identifiers for all of the AWS accounts that you want to permit to restore the manual snapshot, choose **Save** to save your changes.

To stop sharing a manual DB snapshot with an AWS account

- Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
- In the navigation pane, choose **Snapshots**.
- Select the manual snapshot that you want to stop sharing.
- Choose **Actions**, and then choose **Share Snapshot**.
- To remove permission for an AWS account, choose **Delete** for the AWS account identifier for that account from the list of authorized accounts.

Snapshot permissions

Preferences

You are sharing an unencrypted DB snapshot. When you share an unencrypted DB snapshot, you give the other account permission to make a copy of the DB snapshot and to restore a database from your DB snapshot.

DB snapshot

testoracletags-snap

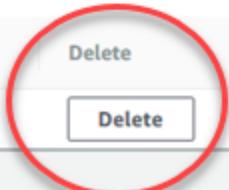
DB snapshot visibility

- Private
- Public

AWS account ID

Add

AWS account ID



Delete

Delete

Cancel

Save

6. Choose **Save** to save your changes.

AWS CLI

To share a DB snapshot, use the `aws rds modify-db-snapshot-attribute` command. Use the `--values-to-add` parameter to add a list of the IDs for the AWS accounts that are authorized to restore the manual snapshot.

The following example permits two AWS account identifiers, `123451234512` and `123456789012`, to restore the DB snapshot named `manual-snapshot1`, and removes the `all` attribute value to mark the snapshot as private.

```
aws rds modify-db-snapshot-attribute \
--db-snapshot-identifier manual-snapshot1 \
--attribute-name restore \
--values-to-add '[ "111122223333", "444455556666" ]'
```

To remove an AWS account identifier from the list, use the `--values-to-remove` parameter. The following example prevents AWS account ID `444455556666` from restoring the snapshot.

```
aws rds modify-db-snapshot-attribute \
--db-snapshot-identifier manual-snapshot1 \
--attribute-name restore \
--values-to-remove '[ "444455556666" ]'
```

API

You can also share a manual DB snapshot with other AWS accounts by using the Amazon RDS API. To do so, call the [ModifyDBSnapshotAttribute](#) action. Specify `restore` for `AttributeName`, and use the `ValuesToAdd` parameter to add a list of the IDs for the AWS accounts that are authorized to restore the manual snapshot.

To make a manual snapshot public and restorable by all AWS accounts, use the value `all`. However, take care not to add the `all` value for any manual snapshots that contain private information that you don't want to be available to all AWS accounts. Also, don't specify `all` for encrypted snapshots, because making such snapshots public isn't supported.

To remove sharing permission for an AWS account, use the [ModifyDBSnapshotAttribute](#) action with `AttributeName` set to `restore` and the `ValuesToRemove` parameter. To mark a manual snapshot as private, remove the value `all` from the values list for the `restore` attribute.

To list all of the AWS accounts permitted to restore a snapshot, use the [DescribeDBSnapshotAttributes](#) API action.

Restoring a DB Instance to a Specified Time

You can restore a DB instance to a specific point in time, creating a new DB instance . When you restore a DB instance to a point in time, the default DB security group is applied to the new DB instance. If you need custom DB security groups applied to your DB instance, you must apply them explicitly using the AWS Management Console, the AWS CLI `modify-db-instance` command, or the Amazon RDS API `ModifyDBInstance` action after the DB instance is available.

RDS uploads transaction logs for DB instances to Amazon S3 every 5 minutes. To determine the latest restorable time for a DB instance, use the AWS CLI `describe-db-instances` command and look at the value returned in the `LatestRestorableTime` field for the DB instance. In the AWS Management Console, this property is visible as the **Latest restore time** for the DB instance. You can restore to any point in time during your backup retention period.

Several of the database engines used by Amazon RDS have special considerations when restoring from a point in time. When you restore an Oracle DB instance to a point in time, you can specify a different Oracle DB engine, license model, and DBName (SID) to be used by the new DB instance. When you restore a SQL Server DB instance to a point in time, each database within that instance is restored to a point in time within 1 second of each other database within the instance. Transactions that span multiple databases within the instance may be restored inconsistently. Also, for a SQL Server DB instance, the `OFFLINE`, `EMERGENCY`, and `SINGLE_USER` modes are not currently supported. Setting any database into one of these modes will cause the latest restorable time to stop moving ahead for the whole instance.

Some actions, such as changing the recovery model of a SQL Server database, can break the sequence of logs that are used for point-in-time recovery. In some cases, Amazon RDS can detect this issue and the latest restorable time is prevented from moving forward; in other cases, such as when a SQL Server database uses the `BULK_LOGGED` recovery model, the break in log sequence is not detected. It may not be possible to restore a SQL Server DB instance to a point in time if there is a break in the log sequence. For these reasons, Amazon RDS does not support changing the recovery model of SQL Server databases.

You can restore a DB instance to a point in time using the AWS Management Console, the AWS CLI, or the RDS API.

AWS Management Console

To restore a DB instance to a specified time

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to restore.
4. For **Actions**, choose **Restore to point in time**.

The **Launch DB Instance** window appears.

5. Choose **Latest restorable time** to restore to the latest possible time, or choose **Custom** to choose a time.

If you chose **Custom**, enter the date and time that you want to restore the instance to.
6. For **DB instance identifier**, enter the name of the restored DB instance, and then complete the other options.
7. Choose **Launch DB Instance**.

CLI

To restore a DB instance to a specified time, use the AWS CLI command [restore-db-instance-to-point-in-time](#) to create a new DB instance.

Example

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-to-point-in-time \
--source-db-instance-identifier mysourcedbinstance \
--target-db-instance-identifier mytargetdbinstance \
--restore-time 2017-10-14T23:45:00.000Z
```

For Windows:

```
aws rds restore-db-instance-to-point-in-time ^
--source-db-instance-identifier mysourcedbinstance ^
--target-db-instance-identifier mytargetdbinstance ^
--restore-time 2017-10-14T23:45:00.000Z
```

API

To restore a DB instance to a specified time, call the Amazon RDS API

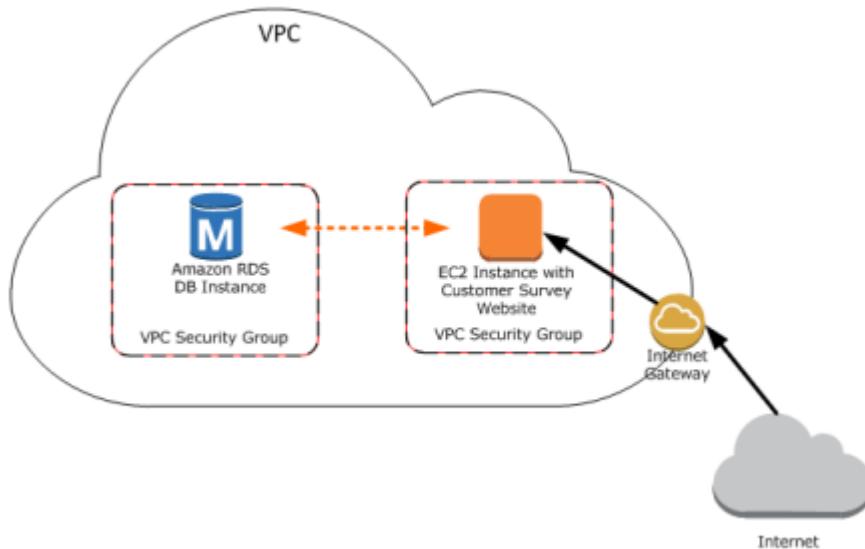
[RestoreDBInstanceToPointInTime](#) action with the following parameters:

- `SourceDBInstanceIdentifier`
- `TargetDBInstanceIdentifier`
- `RestoreTime`

Tutorial: Restore a DB Instance from a DB Snapshot

A common scenario when working with Amazon RDS is to have a DB instance that you work with occasionally but that you don't need full time. For example, you might have a quarterly customer survey that uses an Amazon Elastic Compute Cloud (Amazon EC2) instance to host a customer survey website and a DB instance that is used to store the survey results. One way to save money on such a scenario is to take a DB snapshot of the DB instance after the survey is completed, delete the DB instance, and then restore the DB instance when you need to conduct the survey again.

In the following illustration, you can see a possible scenario where an EC2 instance hosting a customer survey website is in the same Amazon Virtual Private Cloud (Amazon VPC) as a DB instance that retains the customer survey data. Note that each instance has its own security group; the EC2 instance security group allows access from the internet while the DB instance security group allows access only to and from the EC2 instance. When the survey is done, the EC2 instance can be stopped and the DB instance can be deleted after a final DB snapshot is created. When you need to conduct another survey, you can restart the EC2 instance and restore the DB instance from the DB snapshot.



For information about how to set up the needed VPC security groups for this scenario that allows the EC2 instance to connect with the DB instance, see [A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC \(p. 414\)](#).

You must create a DB snapshot before you can restore a DB instance from one. When you restore the DB instance, you provide the name of the DB snapshot to restore from, and then provide a name for the new DB instance that is created from the restore operation. You cannot restore from a DB snapshot to an existing DB instance; a new DB instance is created when you restore.

Prerequisites for Restoring a DB Instance from a DB Snapshot

Some settings on the restored DB instance are reset when the instance is restored, so you must retain the original resources to be able to restore the DB instance to its previous settings. For example, when you restore a DB instance from a DB snapshot, the default DB parameter and a default security group are

associated with the restored instance. That association means that the default security group does not allow access to the DB instance, and no custom parameter settings are available in the default parameter group. You need to retain the DB parameter group and security group associated with the DB instance that was used to create the DB snapshot.

The following are required before you can restore a DB instance from a DB snapshot:

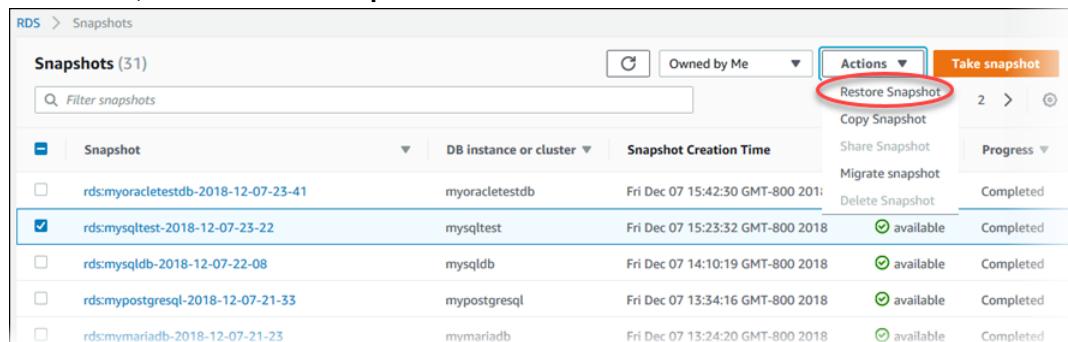
- You must have created a DB snapshot of a DB instance before you can restore a DB instance from that DB snapshot. For more information about creating a DB snapshot, see [Creating a DB Snapshot \(p. 216\)](#).
- You must retain the parameter group and security group associated with the DB instance you created the DB snapshot from.
- You need to determine the correct option group for the restored DB instance:
 - The option group associated with the DB snapshot that you restore from is associated with the restored DB instance once it is created. For example, if the DB snapshot you restore from uses Oracle Transparent Data Encryption (TDE), the restored DB instance uses the same option group, which had the TDE option.
 - You cannot use the option group associated with the original DB instance if you attempt to restore that instance into a different VPC or into a different platform. This restriction occurs because when an option group is assigned to a DB instance, it is also linked to the platform that the DB instance is on, either VPC or EC2-Classic (non-VPC). If a DB instance is in a VPC, the option group associated with the instance is linked to that VPC.
 - If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. Note that with persistent or permanent options, such as Oracle TDE, you must create a new option group that includes the persistent or permanent option when restoring a DB instance into a different VPC. For more information about working with option groups, see [Working with Option Groups \(p. 156\)](#).

Restoring a DB Instance from a DB Snapshot

You can use the procedure following to restore from a snapshot in the AWS Management Console.

To restore a DB instance from a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Snapshots**.
3. Choose the DB snapshot that you want to restore from.
4. For **Actions**, choose **Restore Snapshot**.



The **Restore DB Instance** page appears.

5. For **DB Instance Identifier** under **Settings**, enter the name that you want to use for the restored DB instance. If you are restoring from a DB instance that you deleted after you made the DB snapshot, you can use the name of that DB instance.
6. Choose **Restore DB Instance**.

Modifying a Restored DB Instance

As soon as the restore operation is complete, you should associate the custom security group used by the instance you restored from with any applicable custom DB parameter group that you might have. Only the default DB parameter and security groups are associated with the restored instance. If you want to restore the functionality of the DB instance to that of the DB instance that the snapshot was created from, you must modify the DB instance to use the security group and parameter group used by the previous DB instance.

You must apply any changes explicitly using the RDS console's **Modify** command, the `ModifyDBInstance` API, or the `aws rds modify-db-instance` command line tool, once the DB instance is available. We recommend that you retain parameter groups for any DB snapshots you have so that you can associate a restored instance with the correct parameter file.

You can modify other settings on the restored DB instance. For example, you can use a different storage type than the source DB snapshot. In this case the restoration process is slower because of the additional work required to migrate the data to the new storage type. In the case of restoring to or from Magnetic (Standard) storage, the migration process is the slowest, because Magnetic storage does not have the IOPS capability of Provisioned IOPS or General Purpose (SSD) storage.

The next steps assume that your DB instance is in a VPC. If your DB instance is not in a VPC, use the AWS Management Console to locate the DB security group you need for the DB instance.

To modify a restored DB instance to have the settings of the original DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the name of the DB instance created when you restored from the DB snapshot to display its details. Choose the **Connectivity** tab. The security group assigned to the DB instance might not allow access. If there are no inbound rules, no permissions exist that allow inbound access.

The screenshot shows the 'Connectivity' tab selected in the top navigation bar. Below it, the 'Security' section is highlighted with a red box. This section displays the following details:

Endpoint & port	Networking	Security
Endpoint restored-db-instance.cahj48jpj34l.us-west-2.rds.amazonaws.com	Availability zone us-west-2a	VPC security groups (active)
Port 3306	VPC	Public accessibility Yes
	Subnet group default	Certificate authority rds-ca-2015
	Subnets	Certificate authority d... Mar 5th, 2020

Security group (1)

Security group	Type
[Redacted]	No applicable security group roles

4. Choose **Modify**.
5. In the **Network & Security** section, choose the security group that you want to use for your DB instance. If you need to add rules to create a new security group to use with an EC2 instance, see [A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC \(p. 414\)](#) for more information.

You can also remove a security group by choosing the X associated with it.

Network & Security

Subnet group
Use this field to move the DB instance to a new subnet group in another VPC. [Learn more.](#)

default

Security group
List of DB security groups to associate with this DB instance.

Select security groups

default () () X

Certificate authority
Certificate authority for this DB instance

rds-ca-2015

Public accessibility [info](#)

Yes
EC2 instances and devices outside of the VPC hosting the DB instance will connect to the DB instances. You must also select one or more VPC security groups that specify which EC2 instances and devices can connect to the DB instance.

No
DB instance will not have a public IP address assigned. No EC2 instance or devices outside of the VPC will be able to connect.

6. Choose **Continue**, and then choose **Apply immediately**.
7. Choose **Modify DB Instance**.

After the instance status is available, choose the DB instance name to display its details. Choose the **Connectivity** tab, and confirm that the new security group has been applied, making the DB instance authorized for access.

Connectivity	Monitoring	Logs & events	Configuration	Maintenance & backups	Tags															
<h3>Connectivity</h3> <table border="1"> <tr> <td>Endpoint & port</td> <td>Networking</td> <td>Security</td> </tr> <tr> <td>Endpoint restored-db-instance.cahj48jpj34l.us-west-2.rds.amazonaws.com</td> <td>Availability zone us-west-2a</td> <td>VPC security groups rds-launch-wizard-9 (active)</td> </tr> <tr> <td>Port 3306</td> <td>VPC</td> <td>Public accessibility Yes</td> </tr> <tr> <td></td> <td>Subnet group default</td> <td>Certificate authority rds-ca-2015</td> </tr> <tr> <td></td> <td>Subnets</td> <td>Certificate authority date Mar 5th, 2020</td> </tr> </table>						Endpoint & port	Networking	Security	Endpoint restored-db-instance.cahj48jpj34l.us-west-2.rds.amazonaws.com	Availability zone us-west-2a	VPC security groups rds-launch-wizard-9 (active)	Port 3306	VPC	Public accessibility Yes		Subnet group default	Certificate authority rds-ca-2015		Subnets	Certificate authority date Mar 5th, 2020
Endpoint & port	Networking	Security																		
Endpoint restored-db-instance.cahj48jpj34l.us-west-2.rds.amazonaws.com	Availability zone us-west-2a	VPC security groups rds-launch-wizard-9 (active)																		
Port 3306	VPC	Public accessibility Yes																		
	Subnet group default	Certificate authority rds-ca-2015																		
	Subnets	Certificate authority date Mar 5th, 2020																		

Monitoring Amazon RDS

This section shows you how to monitor Amazon RDS.

Topics

- [Overview of Monitoring Amazon RDS \(p. 245\)](#)
- [Enhanced Monitoring \(p. 256\)](#)
- [Using Amazon RDS Performance Insights \(p. 266\)](#)
- [Using Amazon RDS Recommendations \(p. 282\)](#)
- [Using Amazon RDS Event Notification \(p. 287\)](#)
- [Viewing Amazon RDS Events \(p. 305\)](#)
- [Amazon RDS Database Log Files \(p. 307\)](#)
- [Logging Amazon RDS API Calls with AWS CloudTrail \(p. 338\)](#)

Overview of Monitoring Amazon RDS

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon RDS and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. Before you start monitoring Amazon RDS, we recommend that you create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

The next step is to establish a baseline for normal Amazon RDS performance in your environment, by measuring performance at various times and under different load conditions. As you monitor Amazon RDS, you should consider storing historical monitoring data. This stored data will give you a baseline to compare against with current performance data, identify normal performance patterns and performance anomalies, and devise methods to address issues.

For example, with Amazon RDS, you can monitor network throughput, I/O for read, write, and/or metadata operations, client connections, and burst credit balances for your DB instances. When performance falls outside your established baseline, you might need to change the instance class of your DB instance or the number of DB instances and Read Replicas that are available for clients in order to optimize your database availability for your workload.

In general, acceptable values for performance metrics depend on what your baseline looks like and what your application is doing. Investigate consistent or trending variances from your baseline. Advice about specific types of metrics follows:

- **High CPU or RAM consumption** – High values for CPU or RAM consumption might be appropriate, provided that they are in keeping with your goals for your application (like throughput or concurrency) and are expected.
- **Disk space consumption** – Investigate disk space consumption if space used is consistently at or above 85 percent of the total disk space. See if it is possible to delete data from the instance or archive data to a different system to free up space.
- **Network traffic** – For network traffic, talk with your system administrator to understand what expected throughput is for your domain network and Internet connection. Investigate network traffic if throughput is consistently lower than expected.
- **Database connections** – Consider constraining database connections if you see high numbers of user connections in conjunction with decreases in instance performance and response time. The best number of user connections for your DB instance will vary based on your instance class and the complexity of the operations being performed. You can determine the number of database connections by associating your DB instance with a parameter group where the `User_Connections` parameter is set to a value other than 0 (unlimited). You can either use an existing parameter group or create a new one. For more information, see [Working with DB Parameter Groups \(p. 169\)](#).
- **IOPS metrics** – The expected values for IOPS metrics depend on disk specification and server configuration, so use your baseline to know what is typical. Investigate if values are consistently different than your baseline. For best IOPS performance, make sure your typical working set will fit into memory to minimize read and write operations.

Monitoring Tools

AWS provides various tools that you can use to monitor Amazon RDS. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

Automated Monitoring Tools

You can use the following automated monitoring tools to watch Amazon RDS and report when something is wrong:

- **Amazon RDS Events** – Subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB snapshot, DB parameter group, or DB security group. For more information, see [Using Amazon RDS Event Notification \(p. 287\)](#).
- **Database log files** – View, download, or watch database log files using the Amazon RDS console or Amazon RDS API actions. You can also query some database log files that are loaded into database tables. For more information, see [Amazon RDS Database Log Files \(p. 307\)](#).
- **Amazon RDS Enhanced Monitoring** — Look at metrics in real time for the operating system. For more information, see [Enhanced Monitoring \(p. 256\)](#).

In addition, Amazon RDS integrates with Amazon CloudWatch for additional monitoring capabilities:

- **Amazon CloudWatch Metrics** – Amazon RDS automatically sends metrics to CloudWatch every minute for each active database. You are not charged additionally for Amazon RDS metrics in CloudWatch. For more information, see [the section called "Viewing DB Instance Metrics" \(p. 254\)](#).
- **Amazon CloudWatch Alarms** – You can watch a single Amazon RDS metric over a specific time period, and perform one or more actions based on the value of the metric relative to a threshold you set. For more information, see [Monitoring with Amazon CloudWatch \(p. 247\)](#)
- **Amazon CloudWatch Logs** – Most DB engines enable you to monitor, store, and access your database log files in CloudWatch Logs. For more information, see [Amazon CloudWatch Logs User Guide](#)

Manual Monitoring Tools

Another important part of monitoring Amazon RDS involves manually monitoring those items that the CloudWatch alarms don't cover. The Amazon RDS, CloudWatch, AWS Trusted Advisor and other AWS console dashboards provide an at-a-glance view of the state of your AWS environment. We recommend that you also check the log files on your DB instance.

- From the Amazon RDS console, you can monitor the following items for your resources:
 - The number of connections to a DB instance
 - The amount of read and write operations to a DB instance
 - The amount of storage that a DB instance is currently utilizing
 - The amount of memory and CPU being utilized for a DB instance
 - The amount of network traffic to and from a DB instance
- From the AWS Trusted Advisor dashboard, you can review the following cost optimization, security, fault tolerance, and performance improvement checks:
 - Amazon RDS Idle DB Instances
 - Amazon RDS Security Group Access Risk
 - Amazon RDS Backups
 - Amazon RDS Multi-AZ

For more information on these checks, see [Trusted Advisor Best Practices \(Checks\)](#).

- CloudWatch home page shows:
 - Current alarms and status
 - Graphs of alarms and resources
 - Service health status

In addition, you can use CloudWatch to do the following:

- Create [customized dashboards](#) to monitor the services you care about
- Graph metric data to troubleshoot issues and discover trends
- Search and browse all your AWS resource metrics
- Create and edit alarms to be notified of problems

Monitoring with Amazon CloudWatch

You can monitor DB instances using Amazon CloudWatch, which collects and processes raw data from Amazon RDS into readable, near real-time metrics. These statistics are recorded for a period of two weeks, so that you can access historical information and gain a better perspective on how your web application or service is performing. By default, Amazon RDS metric data is automatically sent to CloudWatch in 1-minute periods. For more information about CloudWatch, see [What Are Amazon CloudWatch, Amazon CloudWatch Events, and Amazon CloudWatch Logs?](#) in the *Amazon CloudWatch User Guide*.

Amazon RDS Metrics and Dimensions

When you use Amazon RDS resources, Amazon RDS sends metrics and dimensions to Amazon CloudWatch every minute. You can use the following procedures to view the metrics for Amazon RDS.

To view metrics using the Amazon CloudWatch console

Metrics are grouped first by the service namespace, and then by the various dimension combinations within each namespace.

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the AWS Region. From the navigation bar, choose the AWS Region where your AWS resources reside. For more information, see [Regions and Endpoints](#).
3. In the navigation pane, choose **Metrics**. Choose the **RDS** metric namespace.

The screenshot shows the AWS CloudWatch Metrics console. At the top, there are three tabs: "All metrics" (selected), "Graphed metrics", and "Graph options". Below the tabs, the navigation path is "All > RDS". A search bar contains the placeholder text "Search for any metric, dimension or resource id". The main content area displays "168 Metrics". Under "DbClusterIdentifier, EngineName", there are "3 Metrics". Under "Per-Database Metrics", there are "45 Metrics". Under "By Database Class", there are "45 Metrics".

4. Choose a metric dimension, for example **By Database Class**.
5. To sort the metrics, use the column heading. To graph a metric, select the check box next to the metric. To filter by resource, choose the resource ID, and then choose **Add to search**. To filter by metric, choose the metric name, and then choose **Add to search**.

The screenshot shows the AWS CloudWatch Metrics console with the "By Database Class" filter applied. The "Graphed metrics (1)" tab is selected. The navigation path is "All > RDS > By Database Class". The search bar placeholder is "Search for any metric, dimension or resource id". The main table lists metrics under "DatabaseClass (45)". The first row, "db.r4.large", has its checkbox checked and is highlighted. A context menu is open over this row, listing options: "Add to search", "Search for this only", "Remove from graph", "Graph this metric only", "Graph all search results", and "What is this?".

	DatabaseClass (45)	Metric Name
<input checked="" type="checkbox"/>	db.r4.large	BufferCacheHitRatio
<input type="checkbox"/>	db.t2.micro	Add to search
<input type="checkbox"/>	db.r4.large	Search for this only
<input type="checkbox"/>	db.r4.large	Remove from graph
<input type="checkbox"/>	db.t2.micro	Graph this metric only
<input type="checkbox"/>	db.t2.micro	Graph all search results
<input type="checkbox"/>	db.r4.large ▾	What is this?

To view metrics using the AWS CLI

- At a command prompt, use the following command:

```
aws cloudwatch list-metrics --namespace AWS/RDS
```

Amazon RDS Metrics

The AWS/RDS namespace includes the following metrics.

Metric	Description
BinLogDiskUsage	<p>The amount of disk space occupied by binary logs on the master. Applies to MySQL read replicas.</p> <p>Units: Bytes</p>
BurstBalance	<p>The percent of General Purpose SSD (gp2) burst-bucket I/O credits available.</p> <p>Units: Percent</p>
CPUUtilization	<p>The percentage of CPU utilization.</p> <p>Units: Percent</p>
CPUCreditUsage	<p>[T2 instances] The number of CPU credits spent by the instance for CPU utilization. One CPU credit equals one vCPU running at 100% utilization for one minute or an equivalent combination of vCPUs, utilization, and time (for example, one vCPU running at 50% utilization for two minutes or two vCPUs running at 25% utilization for two minutes).</p> <p>CPU credit metrics are available at a five-minute frequency only. If you specify a period greater than five minutes, use the <code>Sum</code> statistic instead of the <code>Average</code> statistic.</p> <p>Units: Credits (vCPU-minutes)</p>
CPUCreditBalance	<p>[T2 instances] The number of earned CPU credits that an instance has accrued since it was launched or started. For T2 Standard, the CPUCreditBalance also includes the number of launch credits that have been accrued.</p> <p>Credits are accrued in the credit balance after they are earned, and removed from the credit balance when they are spent. The credit balance has a maximum limit, determined by the instance size. Once the limit is reached, any new credits that are earned are discarded. For T2 Standard, launch credits do not count towards the limit.</p> <p>The credits in the CPUCreditBalance are available for the instance to spend to burst beyond its baseline CPU utilization.</p> <p>When an instance is running, credits in the CPUCreditBalance do not expire. When the instance stops, the CPUCreditBalance does not persist, and all accrued credits are lost.</p> <p>CPU credit metrics are available at a five-minute frequency only.</p>

Metric	Description
	Units: Credits (vCPU-minutes)
DatabaseConnections	The number of database connections in use. Units: Count
DiskQueueDepth	The number of outstanding IOs (read/write requests) waiting to access the disk. Units: Count
FailedSQLServerAgentJobs	The number of failed SQL Server Agent jobs during the last minute. Unit: Count/Minute
FreeableMemory	The amount of available random access memory. Units: Bytes
FreeStorageSpace	The amount of available storage space. Units: Bytes
MaximumUsedTransactionIDs	The maximum transaction ID that has been used. Applies to PostgreSQL. Units: Count
NetworkReceiveThroughput	The incoming (Receive) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication. Units: Bytes/second
NetworkTransmitThroughput	The outgoing (Transmit) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication. Units: Bytes/second
OldestReplicationSlotLag	The lagging size of the replica lagging the most in terms of WAL data received. Applies to PostgreSQL. Units: Megabytes
ReadIOPS	The average number of disk read I/O operations per second. Units: Count/Second
ReadLatency	The average amount of time taken per disk I/O operation. Units: Seconds
ReadThroughput	The average number of bytes read from disk per second. Units: Bytes/Second

Metric	Description
ReplicaLag	The amount of time a Read Replica DB instance lags behind the source DB instance. Applies to MySQL, MariaDB, and PostgreSQL Read Replicas. Units: Seconds
ReplicationSlotDiskUsage	The disk space used by replication slot files. Applies to PostgreSQL. Units: Megabytes
SwapUsage	The amount of swap space used on the DB instance. This metric is not available for SQL Server. Units: Bytes
TransactionLogsDiskUsage	The disk space used by transaction logs. Applies to PostgreSQL. Units: Megabytes
TransactionLogsGeneration	The size of transaction logs generated per second. Applies to PostgreSQL. Units: Megabytes/second
WriteIOPS	The average number of disk write I/O operations per second. Units: Count/Second
WriteLatency	The average amount of time taken per disk I/O operation. Units: Seconds
WriteThroughput	The average number of bytes written to disk per second. Units: Bytes/Second

Amazon RDS Dimensions

Amazon RDS metrics data can be filtered by using any of the dimensions in the following table:

Dimension	Description
DBInstanceIdentifier	This dimension filters the data you request for a specific database instance.
DBClusterIdentifier	This dimension filters the data you request for a specific Amazon Aurora DB cluster.
DBClusterIdentifier, Role	This dimension filters the data you request for a specific Aurora DB cluster, aggregating the metric by instance role (WRITER/READER). For example, you can aggregate metrics for all READER instances that belong to a cluster.
DatabaseClass	This dimension filters the data you request for all instances in a database class. For example, you can aggregate metrics for all instances that belong to the database class db.m1.small

Dimension	Description
EngineName	This dimension filters the data you request for the identified engine name only. For example, you can aggregate metrics for all instances that have the engine name mysql.
SourceRegion	This dimension filters the data you request for the specified region only. For example, you can aggregate metrics for all instances in the region us-east-1.

Creating CloudWatch Alarms to Monitor Amazon RDS

You can create a CloudWatch alarm that sends an Amazon SNS message when the alarm changes state. An alarm watches a single metric over a time period you specify, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon SNS topic or Auto Scaling policy.

Alarms invoke actions for sustained state changes only. CloudWatch alarms will not invoke actions simply because they are in a particular state, the state must have changed and been maintained for a specified number of periods. The following procedures show how to create alarms for Amazon RDS.

To set alarms using the CloudWatch console

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Choose **Alarms** and then choose **Create Alarm**. This launches the **Create Alarm Wizard**.
3. Choose **RDS Metrics** and scroll through the Amazon RDS metrics to locate the metric you want to place an alarm on. To display just the Amazon RDS metrics in this dialog box, search for the identifier of your resource. Choose the metric to create an alarm on and then choose **Next**.
4. Enter **Name**, **Description**, and **Whenever** values for the metric.
5. If you want CloudWatch to send you an email when the alarm state is reached, for **Whenever this alarm:**, choose **State is ALARM**. For **Send notification to:**, choose an existing SNS topic. If you choose **Create topic**, you can set the name and email addresses for a new email subscription list. This list is saved and appears in the field for future alarms.

Note

If you use **Create topic** to create a new Amazon SNS topic, the email addresses must be verified before they receive notifications. Emails are only sent when the alarm enters an alarm state. If this alarm state change happens before the email addresses are verified, they do not receive a notification.

6. At this point, the **Alarm Preview** area gives you a chance to preview the alarm you're about to create. Choose **Create Alarm**.

To set an alarm using the AWS CLI

- Call `put-metric-alarm`. For more information, see [AWS CLI Command Reference](#).

To set an alarm using the CloudWatch API

- Call `PutMetricAlarm`. For more information, see [Amazon CloudWatch API Reference](#)

Publishing Database Engine Logs to Amazon CloudWatch Logs

You can configure your Amazon RDS database engine to publish log data to a log group in Amazon CloudWatch Logs. With CloudWatch Logs, you can perform real-time analysis of the log data, and use CloudWatch to create alarms and view metrics. You can use CloudWatch Logs to store your log records in highly durable storage, which you can manage with the CloudWatch Logs Agent. For example, you can determine when to rotate log records from a host to the log service, so you can access the raw logs when you need to.

You can export logs for Amazon RDS MariaDB (all versions) and Amazon RDS MySQL (versions 5.6, 5.7, and 8.0).

Note

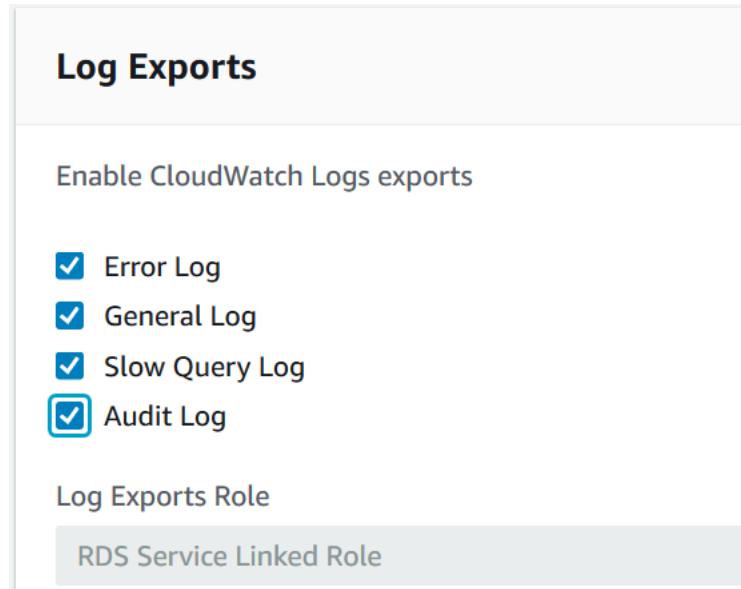
You must have a Service Linked Role before you enable log data publishing. For more information about Service Linked Roles, see the following: [Using Service-Linked Roles for Amazon RDS \(p. 409\)](#).

For specific requirements for these engines, see the following:

- the section called “[Publishing MariaDB Logs to Amazon CloudWatch Logs](#)” (p. 312)
- the section called “[Publishing MySQL Logs to CloudWatch Logs](#)” (p. 321)

Configuring CloudWatch Log Integration

To publish your database log files to CloudWatch Logs, choose which logs to publish. Make this choice in the **Advanced Settings** section when you create a new DB instance. You can also modify an existing DB instance to begin publishing.



After you have enabled publishing, Amazon RDS continuously streams all of the DB instance log records to a log group. For example, you have a log group `/aws/rds/instance/log_type` for each type of log that you publish. This log group is in the same AWS Region as the database instance that generates the log.

After you have published log records, you can use CloudWatch Logs to search and filter the records. For more information about searching and filtering logs, see [Searching and Filtering Log Data](#).

Viewing DB Instance Metrics

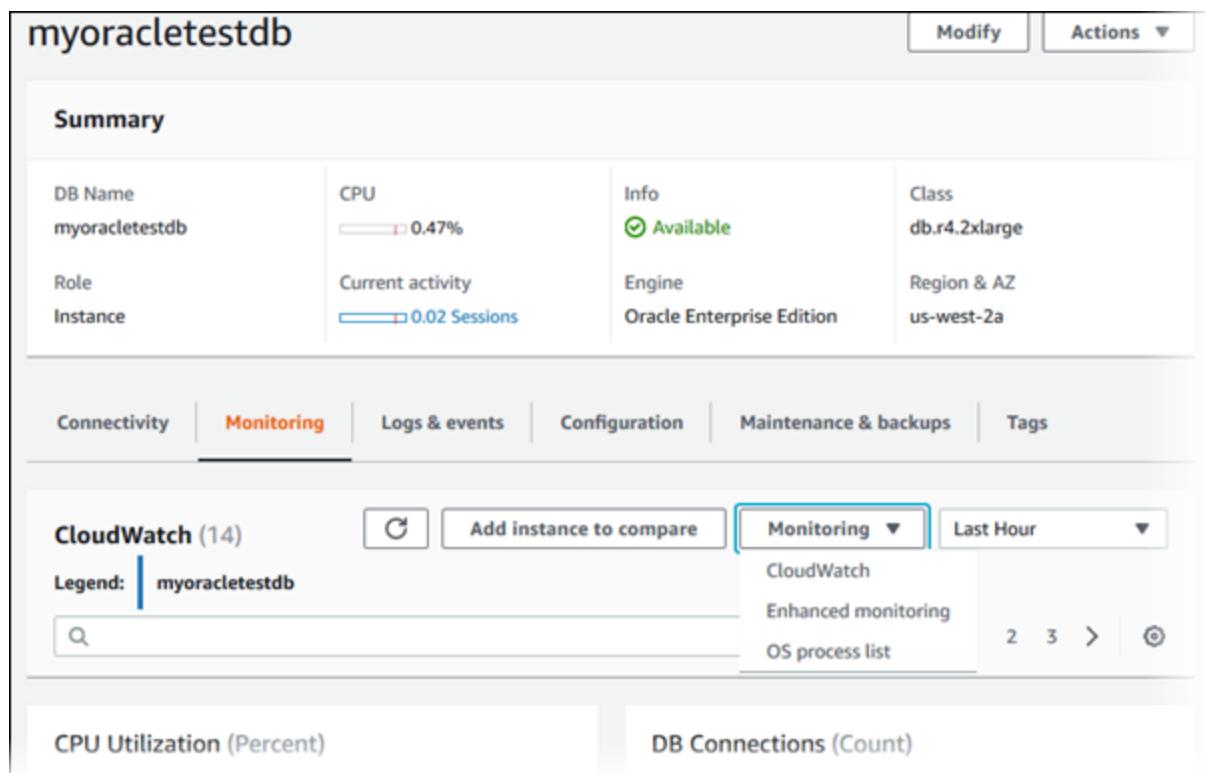
Amazon RDS provides metrics so that you can monitor the health of your DB instances. You can monitor both DB instance metrics and operating system (OS) metrics.

This section provides details on how you can view metrics for your DB instance using the RDS console and CloudWatch. For information on monitoring metrics for the operating system of your DB instance in real time using CloudWatch Logs, see [Enhanced Monitoring \(p. 256\)](#).

Viewing Metrics by Using the Console

To view DB and OS metrics for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the name of the DB instance you need information about to show its details.
4. Choose the **Monitoring** tab.
5. For **Monitoring**, choose the option for how you want to view your metrics from these:
 - **CloudWatch** – Shows a summary of DB instance metrics available from Amazon CloudWatch. Each metric includes a graph showing the metric monitored over a specific time span.
 - **Enhanced monitoring** – Shows a summary of OS metrics available for a DB instance with Enhanced Monitoring enabled. Each metric includes a graph showing the metric monitored over a specific time span.
 - **OS Process list** – Shows details for each process running in the selected instance.



Tip

You can choose the time range of the metrics represented by the graphs with the time range list.

You can choose any graph to bring up a more detailed view. You can also apply metric-specific filters to the data.

Viewing DB Instance Metrics with the CLI or API

Amazon RDS integrates with CloudWatch metrics to provide a variety of DB instance metrics. You can view CloudWatch metrics using the RDS console, AWS CLI, or API.

For a complete list of Amazon RDS metrics, go to [Amazon RDS Dimensions and Metrics](#) in the *Amazon CloudWatch User Guide*.

Viewing DB Metrics by Using the CloudWatch CLI

Note

The following CLI example requires the CloudWatch command line tools. For more information on CloudWatch and to download the developer tools, see the [Amazon CloudWatch product page](#). The `StartTime` and `EndTime` values supplied in this example are for illustrative purposes. You must substitute appropriate start and end time values for your DB instance.

To view usage and performance statistics for a DB instance

- Use the CloudWatch command `mon-get-stats` with the following parameters.

```
PROMPT>mon-get-stats FreeStorageSpace --dimensions="DBInstanceIdentifier=mydbinstance"
--statistics= Average
--namespace="AWS/RDS" --start-time 2009-10-16T00:00:00 --end-time 2009-10-16T00:02:00
```

Viewing DB Metrics by Using the CloudWatch API

The `StartTime` and `EndTime` values supplied in this example are for illustrative purposes. You must substitute appropriate start and end time values for your DB instance.

To view usage and performance statistics for a DB instance

- Call the CloudWatch API `GetMetricStatistics` with the following parameters:
 - `Statistics.member.1 = Average`
 - `Namespace = AWS/RDS`
 - `StartTime = 2009-10-16T00:00:00`
 - `EndTime = 2009-10-16T00:02:00`
 - `Period = 60`
 - `MeasureName = FreeStorageSpace`

Enhanced Monitoring

Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You can view the metrics for your DB instance using the console, or consume the Enhanced Monitoring JSON output from CloudWatch Logs in a monitoring system of your choice.

By default, Enhanced Monitoring metrics are stored in the CloudWatch Logs for 30 days. To modify the amount of time the metrics are stored in the CloudWatch Logs, change the retention for the `RDSOSMetrics` log group in the CloudWatch console. For more information, see [Change Log Data Retention in CloudWatch Logs](#) in the *Amazon CloudWatch Logs User Guide*.

The cost for using Enhanced Monitoring varies depends on several factors:

- You are only charged for Enhanced Monitoring that exceeds the free tier provided by Amazon CloudWatch Logs.

For more information about pricing, see [Amazon CloudWatch Pricing](#).

- A smaller monitoring interval results in more frequent reporting of OS metrics and increases your monitoring cost.
- Usage costs for Enhanced Monitoring are applied for each DB instance that Enhanced Monitoring is enabled for. Monitoring a large number of DB instances is more expensive than monitoring only a few.
- DB instances that support a more compute-intensive workload have more OS process activity to report and higher costs for Enhanced Monitoring.

Enhanced Monitoring Availability

Enhanced Monitoring is available for the following database engines:

- MariaDB
- Microsoft SQL Server
- MySQL version 5.5 or later
- Oracle
- PostgreSQL

Enhanced Monitoring is available for all DB instance classes except for `db.m1.small`.

Differences Between CloudWatch and Enhanced Monitoring Metrics

CloudWatch gathers metrics about CPU utilization from the hypervisor for a DB instance, and Enhanced Monitoring gathers its metrics from an agent on the instance. As a result, you might find differences between the measurements, because the hypervisor layer performs a small amount of work. The differences can be greater if your DB instances use smaller instance classes, because then there are likely more virtual machines (VMs) that are managed by the hypervisor layer on a single physical instance. Enhanced Monitoring metrics are useful when you want to see how different processes or threads on a DB instance use the CPU.

Setting Up for and Enabling Enhanced Monitoring

Before You Begin

Enhanced Monitoring requires permission to act on your behalf to send OS metric information to CloudWatch Logs. You grant Enhanced Monitoring the required permissions using an AWS Identity and Access Management (IAM) role.

The first time that you enable Enhanced Monitoring in the console, you can select the **Default** option for the **Monitoring Role** property to have RDS create the required IAM role. RDS then automatically creates a role named `rds-monitoring-role` for you, and uses it for the specified DB instance or Read Replica.

You can also create the required role before you enable Enhanced Monitoring, and then specify your new role's name when you enable Enhanced Monitoring. You must create this required role if you enable Enhanced Monitoring using the AWS CLI or the RDS API.

To create the appropriate IAM role to permit Amazon RDS to communicate with the Amazon CloudWatch Logs service on your behalf, take the following steps.

The user that enables Enhanced Monitoring must be granted the `PassRole` permission. For more information, see Example 2 in [Granting a User Permissions to Pass a Role to an AWS Service](#) in the *IAM User Guide*.

To create an IAM role for Amazon RDS Enhanced Monitoring

1. Open the [IAM Console](#) at <https://console.aws.amazon.com>.
2. In the navigation pane, choose **Roles**.
3. Choose **Create role**.
4. Choose the **AWS service** tab, and then choose **RDS** from the list of services.
5. Choose **RDS - Enhanced Monitoring**, and then choose **Next: Permissions**.
6. On the **Attached permissions policy** page, choose **AmazonRDSEnhancedMonitoringRole**, and then choose **Next: Review**.
7. For **Role Name**, type a name for your role, for example `emaccess`, and then choose **Create role**.

Enabling and Disabling Enhanced Monitoring

You can enable Enhanced Monitoring when you create a DB instance or Read Replica, or when you modify a DB instance. If you modify a DB instance to enable Enhanced Monitoring, you do not need to reboot your DB instance for the change to take effect.

You can enable Enhanced Monitoring in the RDS console when you do one of the following actions:

- **Create a DB Instance** – You can enable Enhanced Monitoring in the **Configure Advanced Settings** page.
- **Create a Read Replica** – You can enable Enhanced Monitoring in the **Configure Advanced Settings** page.
- **Modify a DB Instance** – You can enable Enhanced Monitoring in the **Modify DB Instance** page.

To enable Enhanced Monitoring by using the RDS console, scroll to the **Monitoring** section and do the following:

1. Choose **Enable enhanced monitoring** for your DB instance or Read Replica.

2. Set the **Monitoring Role** property to the IAM role that you created to permit Amazon RDS to communicate with Amazon CloudWatch Logs for you, or choose **Default** to have RDS create a role for you named `rds-monitoring-role`.
3. Set the **Granularity** property to the interval, in seconds, between points when metrics are collected for your DB instance or Read Replica. The **Granularity** property can be set to one of the following values: 1, 5, 10, 15, 30, or 60.

To disable Enhanced Monitoring, choose **Disable enhanced monitoring**.

The screenshot shows the 'Monitoring' section of the AWS RDS console. Under 'Enhanced monitoring', the 'Enable enhanced monitoring' option is selected (indicated by a blue dot). A tooltip explains that enhanced monitoring metrics are useful for seeing how different processes or threads use the CPU. The 'Disable enhanced monitoring' option is also present. Below this, there are two dropdown menus: 'Monitoring Role' set to 'rds-monitoring-role' and 'Granularity' set to '60 sec...'. Both dropdowns have a downward arrow indicating they are dropdown menus.

Enabling Enhanced Monitoring does not require your DB instance to restart.

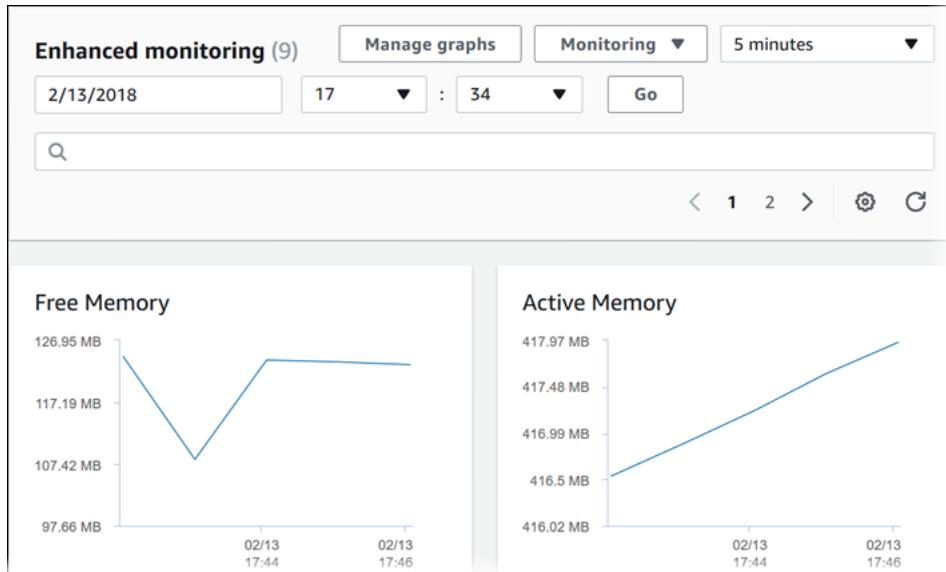
Note

The fastest that the RDS console refreshes is every 5 seconds. If you set the granularity to 1 second in the RDS console, you still see updated metrics only every 5 seconds. You can retrieve 1 second metric updates by using CloudWatch Logs.

Viewing Enhanced Monitoring

You can view OS metrics reported by Enhanced Monitoring in the RDS console by choosing the **Enhanced monitoring** view from the **Monitoring** drop-down.

The Enhanced Monitoring is shown following.



If you want to see details for the processes running on your DB instance, choose **OS process list** for **Monitoring**.

Process List view is shown following.

Process List						
<input type="text"/> Filter process list						
NAME	VIRT	RES	CPU%	MEM%	VMLIMIT	
postgres [3181] ^t	283.55 MB	17.11 MB	0.02	1.72		
postgres: rdsadmin rdsadmin localhost(40156) idle [2953] ^t	384.7 MB	9.51 MB	0.02	0.95		

The Enhanced Monitoring metrics shown in the Process List view are organized as follows:

- **RDS child processes** – Shows a summary of the RDS processes that support the DB instance, for example `aurora` for Amazon Aurora DB clusters and `mysqld` for MySQL DB instances. Process threads appear nested beneath the parent process. Process threads show CPU utilization only as other metrics are the same for all threads for the process. The console displays a maximum of 100 processes and threads. The results are a combination of the top CPU consuming and memory consuming processes and threads. If there are more than 50 processes and more than 50 threads, the console displays the top 50 consumers in each category. This display helps you identify which processes are having the greatest impact on performance.
- **RDS processes** – Shows a summary of the resources used by the RDS management agent, diagnostics monitoring processes, and other AWS processes that are required to support RDS DB instances.
- **OS processes** – Shows a summary of the kernel and system processes, which generally have minimal impact on performance.

The items listed for each process are:

- **VIRT** – Displays the virtual size of the process.
- **RES** – Displays the actual physical memory being used by the process.
- **CPU%** – Displays the percentage of the CPU bandwidth consumed by the process.
- **MEM%** – Displays the percentage of the total memory consumed by the process.

The monitoring data that is shown in the RDS console is retrieved from Amazon CloudWatch Logs. You can also retrieve the metrics for a DB instance as a log stream from CloudWatch Logs. For more information, see [Viewing Enhanced Monitoring by Using CloudWatch Logs \(p. 260\)](#).

Enhanced Monitoring metrics are not returned during the following:

- A failover of the DB instance.
- Changing the instance class of the DB instance (scale compute).

Enhanced Monitoring metrics are returned during a reboot of a DB instance because only the database engine is rebooted. Metrics for the operating system are still reported.

Viewing Enhanced Monitoring by Using CloudWatch Logs

After you have enabled Enhanced Monitoring for your DB instance, you can view the metrics for your DB instance using CloudWatch Logs, with each log stream representing a single DB instance being monitored. The log stream identifier is the resource identifier (`DbiResourceId`) for the DB instance.

To view Enhanced Monitoring log data

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, choose the region that your DB instance is in. For more information, go to [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.
3. Choose **Logs** in the navigation pane.
4. Choose **RDSOSMetrics** from the list of log groups.
5. Choose the log stream that you want to view from the list of log streams.

Available OS Metrics

The following tables list the OS metrics available using Amazon CloudWatch Logs.

Metrics for MariaDB, MySQL, Oracle, and PostgreSQL DB instances

Group	Metrics	Description
General	engine	The database engine for the DB instance.
	instanceID	The DB instance identifier.
	instanceResourceId	A region-unique, immutable identifier for the DB instance, also used as the log stream identifier.
	numVCpus	The number of virtual CPUs for the DB instance.
	timestamp	The time at which the metrics were taken.

Group	Metrics	Description
	uptime	The amount of time that the DB instance has been active.
	version	The version of the OS metrics' stream JSON format.
cpuUtilization	guest	The percentage of CPU in use by guest programs.
	idle	The percentage of CPU that is idle.
	irq	The percentage of CPU in use by software interrupts.
	nice	The percentage of CPU in use by programs running at lowest priority.
	steal	The percentage of CPU in use by other virtual machines.
	system	The percentage of CPU in use by the kernel.
	total	The total percentage of the CPU in use. This value includes the nice value.
	user	The percentage of CPU in use by user programs.
	wait	The percentage of CPU unused while waiting for I/O access.
diskIO	avgQueueLen	The number of requests waiting in the I/O device's queue.
	avgReqSz	The average request size, in kilobytes.
	await	The number of milliseconds required to respond to requests, including queue time and service time.
	device	The identifier of the disk device in use.
	readIOsPS	The number of read operations per second.
	readKb	The total number of kilobytes read.
	readKbPS	The number of kilobytes read per second.
	rrqmPS	The number of merged read requests queued per second.
	tps	The number of I/O transactions per second.
	util	The percentage of CPU time during which requests were issued.
	writeIOsPS	The number of write operations per second.
	writeKb	The total number of kilobytes written.
	writeKbPS	The number of kilobytes written per second.
	wrqmPS	The number of merged write requests queued per second.
fileSys	maxFiles	The maximum number of files that can be created for the file system.
	mountPoint	The path to the file system.
	name	The name of the file system.

Group	Metrics	Description
	total	The total number of disk space available for the file system, in kilobytes.
	used	The amount of disk space used by files in the file system, in kilobytes.
	usedFilePercent	The percentage of available files in use.
	usedFiles	The number of files in the file system.
	usedPercent	The percentage of the file-system disk space in use.
loadAverageMinutes	fifteen	The number of processes requesting CPU time over the last 15 minutes.
	five	The number of processes requesting CPU time over the last 5 minutes.
	one	The number of processes requesting CPU time over the last minute.
memory	active	The amount of assigned memory, in kilobytes.
	buffers	The amount of memory used for buffering I/O requests prior to writing to the storage device, in kilobytes.
	cached	The amount of memory used for caching file system-based I/O.
	dirty	The amount of memory pages in RAM that have been modified but not written to their related data block in storage, in kilobytes.
	free	The amount of unassigned memory, in kilobytes.
	hugePagesFree	The number of free huge pages. Huge pages are a feature of the Linux kernel.
	hugePagesRsvd	The number of committed huge pages.
	hugePagesSize	The size for each huge pages unit, in kilobytes.
	hugePagesSurp	The number of available surplus huge pages over the total.
	hugePagesTotal	The total number of huge pages for the system.
	inactive	The amount of least-frequently used memory pages, in kilobytes.
	mapped	The total amount of file-system contents that is memory mapped inside a process address space, in kilobytes.
	pageTables	The amount of memory used by page tables, in kilobytes.
	slab	The amount of reusable kernel data structures, in kilobytes.
	total	The total amount of memory, in kilobytes.
	writeback	The amount of dirty pages in RAM that are still being written to the backing storage, in kilobytes.

Group	Metrics	Description
network	interface	The identifier for the network interface being used for the DB instance.
	rx	The number of bytes received per second.
	tx	The number of bytes uploaded per second.
processList	cpuUsedPc	The percentage of CPU used by the process.
	id	The identifier of the process.
	memoryUsedPc	The amount of memory used by the process, in kilobytes.
	name	The name of the process.
	parentID	The process identifier for the parent process of the process.
	rss	The amount of RAM allocated to the process, in kilobytes.
	tgid	The thread group identifier, which is a number representing the process ID to which a thread belongs. This identifier is used to group threads from the same process.
	VIRT	The amount of virtual memory allocated to the process, in kilobytes.
swap	swap	The amount of swap memory available, in kilobytes.
	swap_in	The amount of memory, in kilobytes, swapped in from disk.
	swap_out	The amount of memory, in kilobytes, swapped out to disk.
	free	The amount of swap memory free, in kilobytes.
	committed	The amount of swap memory, in kilobytes, used as cache memory.
tasks	blocked	The number of tasks that are blocked.
	running	The number of tasks that are running.
	sleeping	The number of tasks that are sleeping.
	stopped	The number of tasks that are stopped.
	total	The total number of tasks.
	zombie	The number of child tasks that are inactive with an active parent task.

Metrics for Microsoft SQL Server DB instances

Group	Metrics	Description
General	engine	The database engine for the DB instance.
	instanceID	The DB instance identifier.

Group	Metrics	Description
	instanceResourceIdentifier	A region-unique, immutable identifier for the DB instance, also used as the log stream identifier.
	numVCpus	The number of virtual CPUs for the DB instance.
	timestamp	The time at which the metrics were taken.
	uptime	The amount of time that the DB instance has been active.
	version	The version of the OS metrics' stream JSON format.
cpuUtilization	idle	The percentage of CPU that is idle.
	kern	The percentage of CPU in use by the kernel.
	user	The percentage of CPU in use by user programs.
disks	name	The identifier for the disk.
	totalKb	The total space of the disk, in kilobytes.
	usedKb	The amount of space used on the disk, in kilobytes.
	usedPc	The percentage of space used on the disk.
	availKb	The space available on the disk, in kilobytes.
	availPc	The percentage of space available on the disk.
	rdCountPS	The number of read operations per second
	rdBytesPS	The number of bytes read per second.
	wrCountPS	The number of write operations per second.
	wBytesPS	The amount of bytes written per second.
memory	commitTotKb	The amount of pagefile-backed virtual address space in use, that is, the current commit charge. This value is composed of main memory (RAM) and disk (pagefiles).
	commitLimitKb	The maximum possible value for the commitTotKb metric. This value is the sum of the current pagefile size plus the physical memory available for pageable contents—excluding RAM that is assigned to non-pageable areas.
	commitPeakKb	The largest value of the commitTotKb metric since the operating system was last started.
	kernTotKb	The sum of the memory in the paged and non-paged kernel pools, in kilobytes.
	kernPagedKb	The amount of memory in the paged kernel pool, in kilobytes.
	kernNonpagedKb	The amount of memory in the non-paged kernel pool, in kilobytes.
	pageSize	The size of a page, in bytes.

Group	Metrics	Description
	physTotKb	The amount of physical memory, in kilobytes.
	physAvailKb	The amount of available physical memory, in kilobytes.
	sqlServerTotKb	The amount of memory committed to Microsoft SQL Server, in kilobytes.
	sysCacheKb	The amount of system cache memory, in kilobytes.
network	interface	The identifier for the network interface being used for the DB instance.
	rdBytesPS	The number of bytes received per second.
	wrBytesPS	The number of bytes sent per second.
processList	cpuUsedPc	The percentage of CPU used by the process.
	memUsedPc	The percentage of total memory used by the process.
	name	The name of the process.
	pid	The identifier of the process. This value is not present for processes that are owned by Amazon RDS.
	ppid	The process identifier for the parent of this process. This value is only present for child processes.
	tid	The thread identifier. This value is only present for threads. The owning process can be identified by using the pid value.
	workingSetKb	The amount of memory in the private working set plus the amount of memory that is in use by the process and can be shared with other processes, in kilobytes.
	workingSetPrivKb	The amount of memory that is in use by a process, but can't be shared with other processes, in kilobytes.
	workingSetShareableKb	The amount of memory that is in use by a process and can be shared with other processes, in kilobytes.
	virtKb	The amount of virtual address space the process is using, in kilobytes. Use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages.
system	handles	The number of handles that the system is using.
	processes	The number of processes running on the system.
	threads	The number of threads running on the system.

Using Amazon RDS Performance Insights

Amazon RDS Performance Insights monitors your Amazon RDS DB instance load so that you can analyze and troubleshoot your database performance. Amazon RDS Performance Insights is currently available for use with the following DB engines:

- Amazon Aurora with MySQL compatibility version 1.17.3 and higher 1.x versions
- Amazon RDS MySQL version 5.7.22 and higher 5.7 versions
- Amazon RDS MySQL version 5.6.41 and higher 5.6 versions
- Amazon Aurora with PostgreSQL compatibility
- Amazon RDS PostgreSQL version 10
- Amazon RDS Oracle (all versions)

Amazon RDS Performance Insights is not supported for MySQL 5.5 or MySQL 8.0.

For information about using Amazon Aurora, see the [Amazon Aurora User Guide](#).

Note

Performance Insights is not supported on db.t2 DB instance classes.

Performance Insights expands on existing Amazon RDS monitoring features to illustrate your database's performance and help you analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, SQL statements, hosts, or users. Performance Insights is on by default in the console create wizard for the Amazon Aurora MySQL, Amazon RDS MySQL, Amazon Aurora PostgreSQL, and Amazon RDS PostgreSQL DB engines. If you have more than one database on the DB instance, performance data for all of the databases is aggregated for the DB instance.

The central metric for Performance Insights is `DB_Load`, which represents the average number of active sessions for the DB engine. The `DB_Load` metric is collected every second. An *active session* is a connection that has submitted work to the DB engine and is waiting for a response from it. For example, if you submit a SQL query to the DB engine, the database session is active while the DB engine is processing that query.

By combining `DB_Load` with wait event data, you can get a complete picture of the state for an active session. Wait events vary by DB engine:

- For information about all MySQL wait events, see [Wait Event Summary Tables](#) in the MySQL documentation.
- For information about all PostgreSQL wait events, see [PostgreSQL Wait Events](#) in the PostgreSQL documentation.
- For information about all Oracle wait events, see [Descriptions of Wait Events](#) in the Oracle documentation.

Note

For Oracle, background processes sometimes do work without an associated SQL statement. In these cases, Performance Insights reports the type of background process (for example, LGWR, ARCO, PMON, and so on) concatenated with a colon and the wait class associated with that background process. For example, when the archiver is performing I/O, the Performance Insights report for it is similar to `ARC1:System_I/O`. Occasionally, the background process type is missing as well, and Performance Insights only reports the wait class, for example `:System_I/O`.

Session information is collected, aggregated, and displayed in the dashboard as the **Average Active Sessions** chart. The **Average Active Sessions** chart displays the **Max CPU** value as a line, so you can see if

active sessions are exceeding it or not. The **Max CPU** value is determined by the number of **vCPU** (virtual CPU) cores for your DB instance.

If you find that the load in the **Average Active Sessions** chart is often above the **Max CPU** line and the primary wait state is CPU, the system CPU is overloaded. In these cases, you might want to throttle connections to the instance, tune any SQL queries with a high CPU load, or consider a larger instance class. High and consistent instances of any wait state indicate that there might be bottlenecks or resource contention issues that you should resolve, even if the load doesn't cross the **Max CPU** line.

You can find an overview of Performance Insights in the following video.

[Using Performance Insights to Analyze Performance of Amazon Aurora PostgreSQL](#)

Topics

- [Enabling Performance Insights \(p. 267\)](#)
- [Access Control for Performance Insights \(p. 271\)](#)
- [Using the Performance Insights Dashboard \(p. 272\)](#)
- [Additional User Interface Features \(p. 278\)](#)
- [Performance Insights API \(p. 279\)](#)
- [Performance Insights Metrics Published to Amazon CloudWatch \(p. 279\)](#)
- [Logging Performance Insights Operations by Using AWS CloudTrail \(p. 280\)](#)

Enabling Performance Insights

To use Performance Insights, you must enable it on your DB instance.

AWS Management Console

You can use the console to enable Performance Insights when you create a new DB instance. You can also modify a DB instance to enable Performance Insights.

Topics

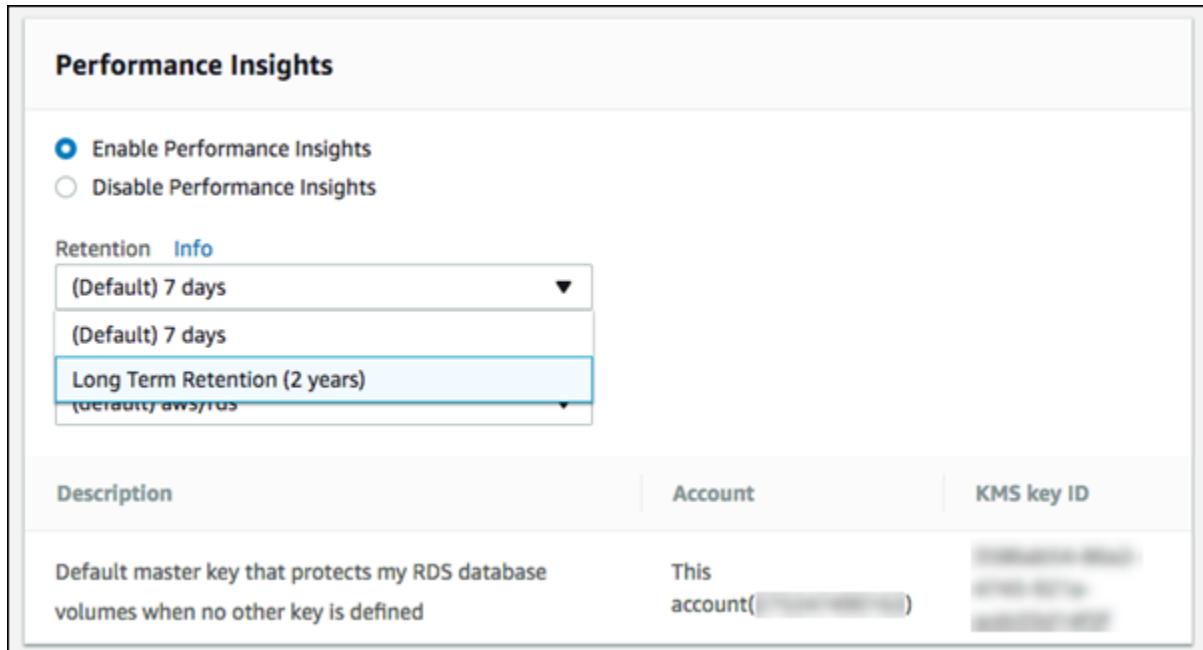
- [Enabling Performance Insights with the Console When Creating a DB Instance \(p. 267\)](#)
- [Enabling Performance Insights with the Console When Modifying a DB Instance \(p. 268\)](#)

Enabling Performance Insights with the Console When Creating a DB Instance

When you create a new DB instance, Performance Insights is enabled when you choose **Enable Performance Insights** in the **Performance Insights** section.

To create a DB instance, follow the instructions for your DB engine in [Creating an Amazon RDS DB Instance \(p. 113\)](#).

The following image shows the **Performance Insights** section.



You have the following options when you choose **Enable Performance Insights**:

- **Retention** – The amount of time to retain Performance Insights data. Choose either 7 days (the default) or 2 years.
- **Master key** – Specify your AWS Key Management Service (AWS KMS) key. Performance Insights encrypts all potentially sensitive data using your AWS KMS key. Data is encrypted in flight and at rest. For more information, see [Encrypting Amazon RDS Resources \(p. 389\)](#).

[Enabling Performance Insights with the Console When Modifying a DB Instance](#)

You can modify a DB instance to enable Performance Insights using the console.

To enable Performance Insights for a DB instance using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. Choose **Databases**.
3. Choose the DB instance that you want to modify, and choose **Modify**.
4. In the **Performance Insights** section, choose **Enable Performance Insights**.

You have the following options when you choose **Enable Performance Insights**:

- **Retention** – The amount of time to retain Performance Insights data. Choose either 7 days (the default) or 2 years.
 - **Master key** – Specify your AWS Key Management Service (AWS KMS) key. Performance Insights encrypts all potentially sensitive data using your AWS KMS key. Data is encrypted in flight and at rest. For more information, see [Encrypting Amazon RDS Resources \(p. 389\)](#).
5. Choose **Continue**.
 6. For **Scheduling of Modifications**, choose one of the following:
 - **Apply during the next scheduled maintenance window** – Wait to apply the **Performance Insights** modification until the next maintenance window.

- **Apply immediately** – Apply the **Performance Insights** modification as soon as possible.
7. Choose **Modify instance**.

CLI

When you create a new DB instance using the [create-db-instance](#) AWS CLI command, Performance Insights is enabled when you specify `--enable-performance-insights`.

You can also specify the `--enable-performance-insights` value using the following AWS CLI commands:

- [create-db-instance-read-replica](#)
- [modify-db-instance](#)
- [restore-db-instance-from-s3](#)

The following procedure describes how to enable Performance Insights for a DB instance using the AWS CLI.

To enable Performance Insights for a DB instance using the AWS CLI

- Call the [modify-db-instance](#) AWS CLI command and supply the following values:
 - `--db-instance-identifier` – The name of the DB instance.
 - `--enable-performance-insights`

The following example enables Performance Insights for `sample-db-instance`.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier sample-db-instance \
  --enable-performance-insights
```

For Windows:

```
aws rds modify-db-instance ^
  --db-instance-identifier sample-db-instance ^
  --enable-performance-insights
```

When you enable Performance Insights, you can optionally specify the amount of time, in days, to retain Performance Insights data with the `--performance-insights-retention-period` option. Valid values are 7 (the default) or 731 (2 years).

The following example enables Performance Insights for `sample-db-instance` and specifies that Performance Insights data is retained for two years.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier sample-db-instance \
```

```
--enable-performance-insights \
--performance-insights-retention-period 731
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier sample-db-instance ^
--enable-performance-insights ^
--performance-insights-retention-period 731
```

API

When you create a new DB instance using the [CreateDBInstance](#) action Amazon RDS API action, the Performance Schema is enabled when you set `EnablePerformanceInsights` to True.

You can also specify the `EnablePerformanceInsights` value using the following API actions:

- [ModifyDBInstance](#)
- [CreateDBInstanceReadReplica](#)
- [RestoreDBInstanceFromS3](#)

When you enable Performance Insights, you can optionally specify the amount of time, in days, to retain Performance Insights data with the `PerformanceInsightsRetentionPeriod` parameter. Valid values are 7 (the default) or 731 (2 years).

Enabling Performance Insights for Amazon RDS MySQL

For Amazon RDS MySQL, Performance Insights provides more detailed information when the Performance Schema feature of MySQL is enabled. The Performance Schema is enabled automatically when you create an Amazon RDS MySQL DB instance with Performance Insights enabled. When you create the DB instance with Performance Insights enabled, the following subset of Performance Schema parameters is set to the specified values automatically:

- `performance_schema=1`
- `performance-schema-consumer-events-waits-current=ON`
- `performance-schema-instrument='wait/%=ON'`
- `performance-schema-consumer-global instrumentation=ON`
- `performance-schema-consumer-thread-instrumentation=ON`

Performance Schema is enabled automatically only if your parameter group doesn't have an explicitly set value for the `performance_schema` parameter. You can examine the `performance_schema` parameter, and if the value of source is `user`, then you set a value. If you want the Performance Schema parameters to be set automatically, then unset the value for the `performance_schema` parameter. You can view the source of a parameter value by viewing the parameter in the AWS Management Console or by running the AWS CLI [describe-db-parameters](#) command.

When you change the value of the `performance_schema` parameter, a DB instance reboot is required. If you're creating a new DB instance with Performance Insights enabled, the `performance_schema` parameter is set to 1 (enabled) by default.

Without the Performance Schema enabled, Performance Insights displays database load broken down by the list state of the MySQL process. With Performance Schema enabled, Performance Insights displays database load broken down by detailed wait events.

For more information, see [Using the Performance Insights Dashboard \(p. 272\)](#).

Access Control for Performance Insights

To access Performance Insights, you must have the appropriate permissions from AWS Identity and Access Management (IAM). There are two options available for granting access:

1. Attach the `AmazonRDSFullAccess` managed policy to an IAM user or role.
2. Create a custom IAM policy and attach it to an IAM user or role.

AmazonRDSFullAccess Managed Policy

`AmazonRDSFullAccess` is an AWS-managed policy that grants access to all of the Amazon RDS API actions. The policy also grants access to related services that are used by the Amazon RDS console—for example, event notifications using Amazon SNS.

In addition, `AmazonRDSFullAccess` contains all the permissions needed for using Performance Insights. If you attach this policy to an IAM user or role, the recipient can use Performance Insights, in addition to all of the other features of the Amazon RDS console.

Using a Custom IAM Policy

For users who don't have full access with the `AmazonRDSFullAccess` policy, you can grant access to Performance Insights by creating or modifying a user-managed IAM policy. When you attach the policy to an IAM user or role, the recipient can use Performance Insights.

To create a custom policy

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. On the **Create Policy** page, choose the JSON tab.
5. Copy and paste the following.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "pi:*",  
            "Resource": "arn:aws:pi:*:metrics/rds/*"  
        }  
    ]  
}
```

6. Choose **Review policy**

Note

Currently, when you enter this policy, the **Visual editor** tab displays a warning that the `pi` resource is not recognized. You can ignore this warning.

7. Provide a name for the policy and optionally a description, and then choose **Create policy**.

You can now attach the policy to an IAM user or role. The following procedure assumes that you already have an IAM user available for this purpose.

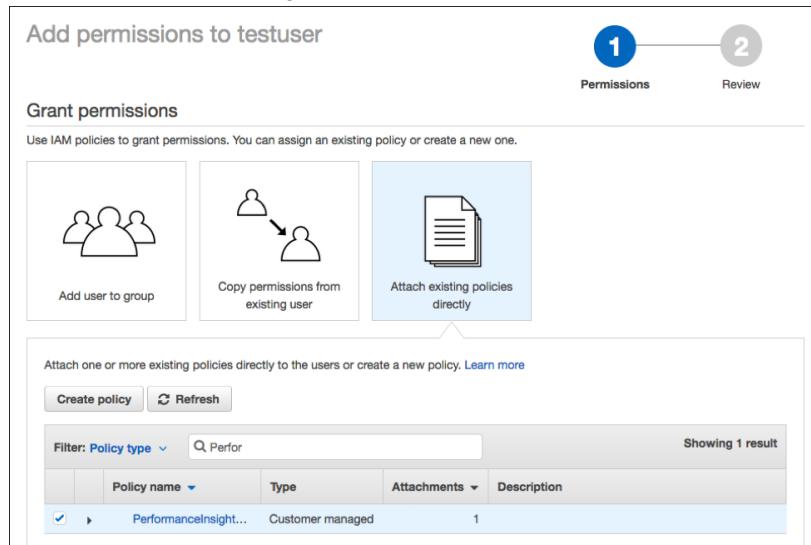
To attach the policy to an IAM user

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users**.
3. Choose an existing user from the list.

Important

To use Performance Insights, the user must have access to Amazon RDS in addition to the custom policy. For example, the `AmazonRDSReadOnlyAccess` predefined policy provides read-only access to Amazon RDS. For more information, see [AWS Managed \(Predefined\) Policies for Amazon RDS \(p. 349\)](#).

4. On the **Summary** page, choose **Add permissions**.
5. Choose **Attach existing policies directly**. For **Search**, type the first few characters of your policy name, as shown following.



6. Choose your policy, and then choose **Next: Review**.
7. Choose **Add permissions**.

Using the Performance Insights Dashboard

The Performance Insights dashboard contains database performance information to help you analyze and troubleshoot performance issues. On the main dashboard page, you can view information about the database load. You can also drill into details for a particular wait state, SQL query, host, or user.

Topics

- [Opening the Performance Insights Dashboard \(p. 272\)](#)
- [Performance Insights Dashboard Components \(p. 274\)](#)
- [Analyzing Database Load Using the Performance Insights Dashboard \(p. 277\)](#)

Opening the Performance Insights Dashboard

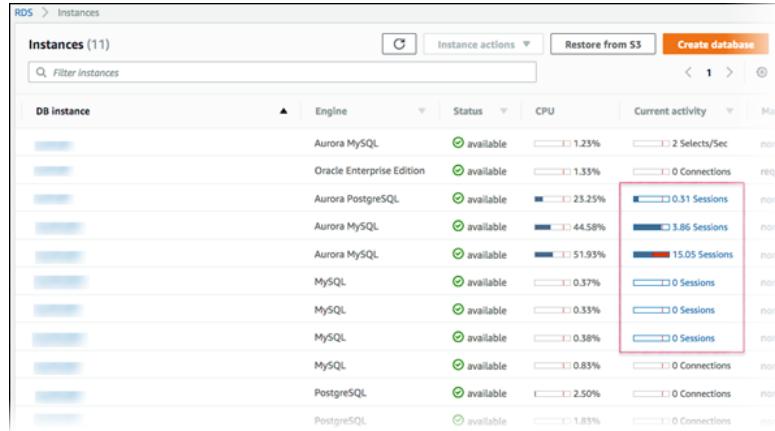
To see the Performance Insights dashboard, use the following procedure.

To view the Performance Insights dashboard in the AWS Management Console

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.

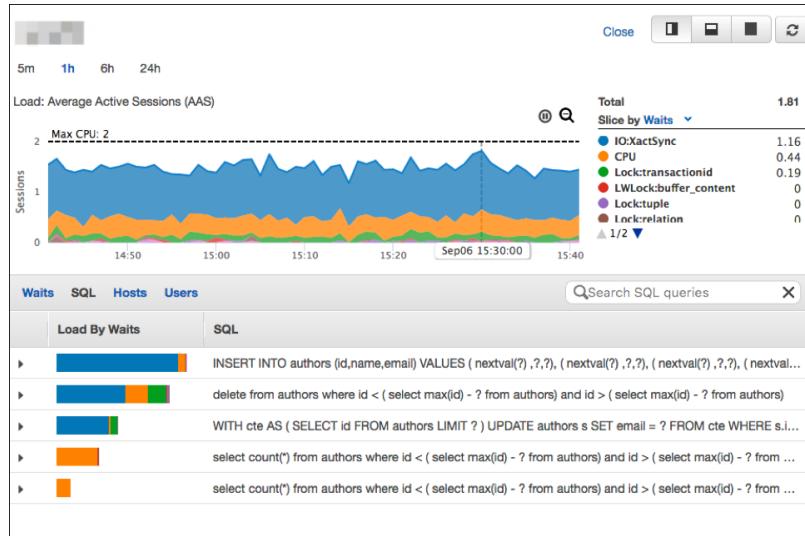
2. In the navigation pane, choose **Performance Insights**.
3. Choose a DB instance. The Performance Insights dashboard is displayed for that DB instance.

For DB instances with Performance Insights enabled, you can also reach the dashboard by choosing the **Sessions** item in the list of DB instances. Under **Current activity**, the **Sessions** item shows the database load in average active sessions over the last five minutes. The bar graphically shows the load. When the bar is empty, the DB instance is idle. As the load increases, the bar fills with blue. When the load passes the number of virtual CPUs (vCPUs) on the DB instance class, the bar turns red, indicating a potential bottleneck.



DB instance	Engine	Status	CPU	Current activity	Main
Aurora MySQL	available	1.23%	2 Selects/Sec	none	
Oracle Enterprise Edition	available	1.33%	0 Connections	none	
Aurora PostgreSQL	available	23.25%	0.31 Sessions	none	
Aurora MySQL	available	44.58%	3.86 Sessions	none	
Aurora MySQL	available	51.93%	15.05 Sessions	none	
MySQL	available	0.37%	0 Sessions	none	
MySQL	available	0.33%	0 Sessions	none	
MySQL	available	0.38%	0 Sessions	none	
MySQL	available	0.83%	0 Connections	none	
PostgreSQL	available	2.50%	0 Connections	none	
PostgreSQL	available	1.83%	0 Connections	none	

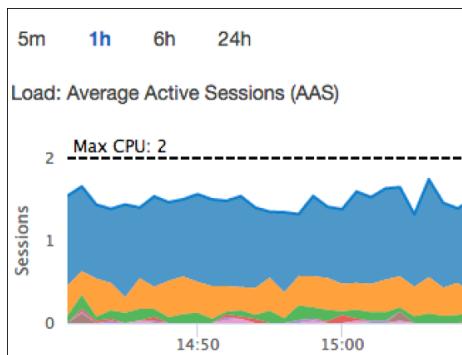
The following image shows the dashboard for a DB instance.



By default, the Performance Insights dashboard shows data for the last 60 minutes. You can modify it to display data for the last 5 minutes, 60 minutes, 5 hours, 24 hours, or 1 week. You can also show all of the data available.

The Performance Insight dashboard automatically refreshes with new data. The refresh rate depends on the amount of data displayed:

- 5 minutes refreshes every 5 seconds.
- 1 hour and 5 hours both refresh every minute.
- 24 hours refreshes every 5 minutes.
- 1 week refreshes every hour.



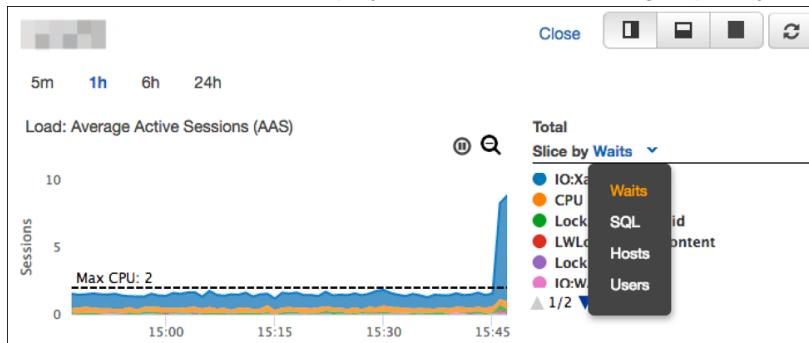
Performance Insights Dashboard Components

The dashboard is divided into two parts:

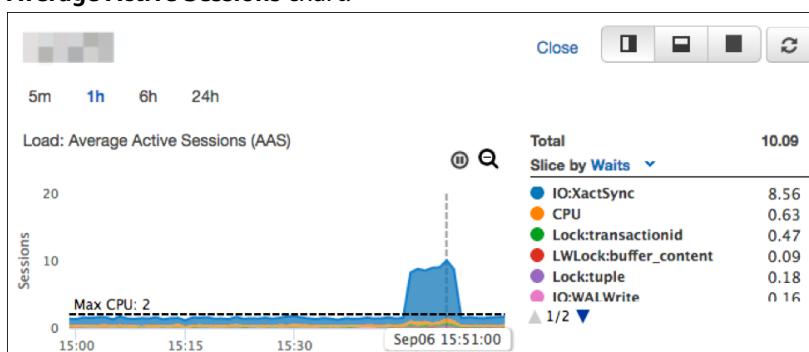
- Average Active Sessions chart** – Shows how the database load compares to DB instance capacity as represented by the **Max CPU** line.
- Top load items table** – Shows the top items contributing to database load.

Average Active Sessions Chart

The **Average Active Sessions** chart shows how the database load compares to DB instance capacity as represented by the **Max CPU** line. By default, load is shown as active sessions grouped by wait states. You can also choose instead to display load as active sessions grouped by SQL queries, hosts, or users.



To see details for any item for the selected time period in the legend, hover over that item on the **Average Active Sessions** chart.



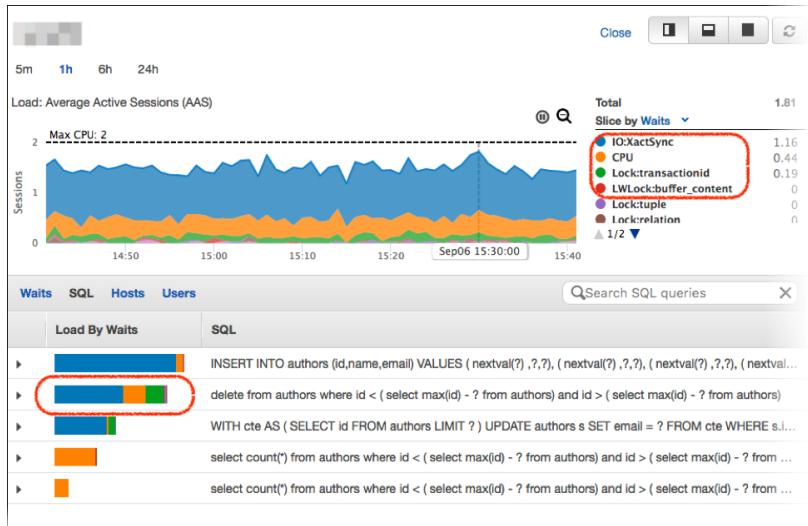
Top Load Items Table

The **Top Load Items** table shows the top items contributing to database load. By default, the top SQL queries that are contributing to the database load are shown. Queries are displayed as digests of multiple actual queries that are structurally similar but that possibly have different parameters. You can choose to display top wait states, hosts, or users instead.

	Load By Waits	SQL
▶		INSERT INTO authors (id,name,email) VALUES (nextval(?) ,? ,?), (nextval(?) ,? ,?), (nextval(?) ,? ,?), (nextval(?) ,? ,?)
▶		delete from authors where id < (select max(id) - ? from authors) and id > (select max(id) - ? from authors)
▶		WITH cte AS (SELECT id FROM authors LIMIT ?) UPDATE authors s SET email = ? FROM cte WHERE s...
▶		select count(*) from authors where id < (select max(id) - ? from authors) and id > (select max(id) - ? from ...
▶		select count(*) from authors where id < (select max(id) - ? from authors) and id > (select max(id) - ? from ...

The percentage of the database load associated with each top load item is illustrated in the **DB Load by Waits** column. This column reflects the load for that item by whatever grouping is currently selected in the **Average Active Sessions** chart. Take the case where the **Average Active Sessions** chart is grouping by hosts and you are looking at SQL queries in the top load items table. In this case, the **DB Load by Waits** bar reflects the load that query represents on the related host. Here it's colored-coded to map to the representation of that host in the **Average Active Sessions** chart.

For another example, suppose that the **Average Active Sessions** chart is grouping by wait states and you are looking at SQL queries in the top load items table. In this case, the **DB Load by Waits** bar is sized, segmented, and color-coded to show how much of a given wait state that query is contributing to. It also shows what wait states are affecting that query.



In the **Top Load Items** table, you can view the following types of identifiers (IDs) that are associated with SQL statements:

- **SQL ID** – An ID that the database uses to uniquely identify a SQL statement.

For Oracle and SQL Server DB instances, you can use a SQL ID to find a specific SQL statement.

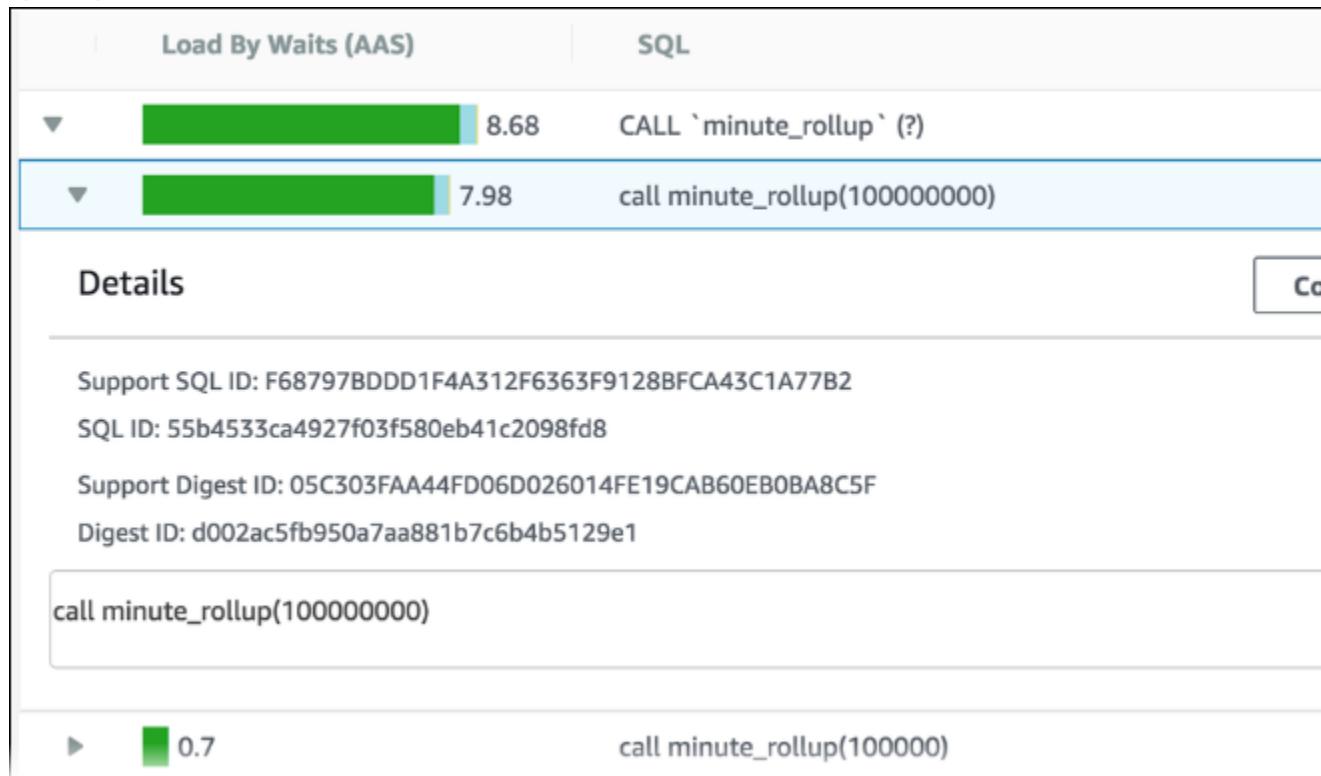
- **Support SQL ID** – A hash value of the SQL ID. This value is only for referencing a SQL ID when you are working with AWS Support. AWS Support doesn't have access to your actual SQL IDs and SQL text.
- **Digest ID** – An ID that the database uses to uniquely identify a SQL Digest. A SQL Digest can contain one or more SQL statements with literals removed and white space standardized. The literals are replaced with question marks (?).

For Amazon RDS MySQL and PostgreSQL DB instances, you can use a Digest ID to find a specific SQL Digest.

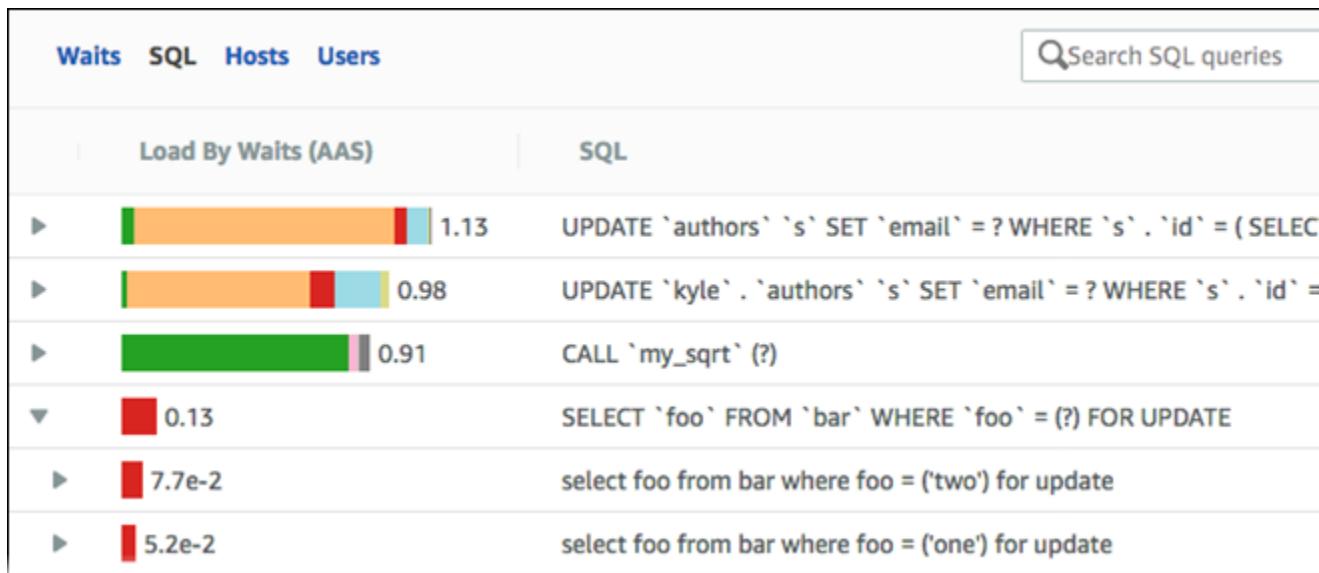
For Oracle and SQL Server DB instances, the Digest ID is the same as the SQL ID, and the top row in the **Top Load Items** table is the actual SQL statement, including the literals.

- **Support Digest ID** – A hash value of the Digest ID. This value is only for referencing a Digest ID when you are working with AWS Support. AWS Support doesn't have access to your actual Digest IDs and SQL text.

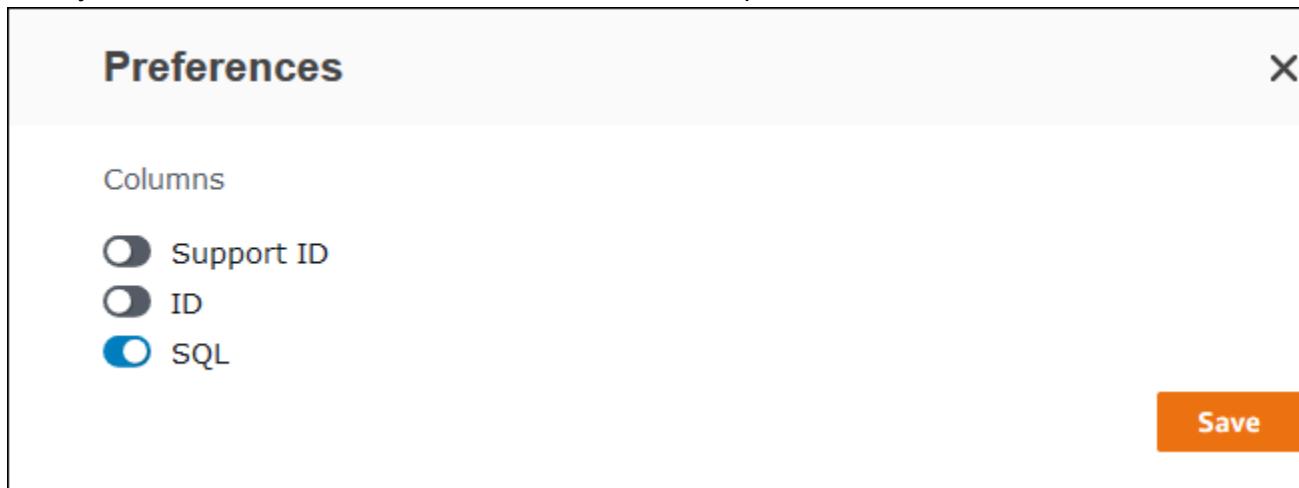
In the **Top Load Items** table, you can open a top statement to view its IDs. The following image shows an open top statement.



You can control the IDs that the **Top Load Items** table shows by choosing the **Preferences** icon.



When you choose the **Preferences** icon, the **Preferences** window opens.



Enable the IDs that you want to have visible in the **Top Load Items** table, and choose **Save**.

Analyzing Database Load Using the Performance Insights Dashboard

If the **Average Active Sessions** chart shows a bottleneck, you can find out where the load is coming from. To do so, look at the top load items table below the **Average Active Sessions** chart. Choose a particular item, like a SQL query or a user, to drill down into that item and see details about it.

DB load grouped by waits and top SQL queries is the default Performance Insights dashboard view, because this is the combination that typically provides the most insight into performance issues. DB load grouped by waits shows if there are any resource or concurrency bottlenecks in the database. In this case, the **SQL** tab of the top load items table shows which queries are driving that load.

Your typical workflow for diagnosing performance issues is as follows:

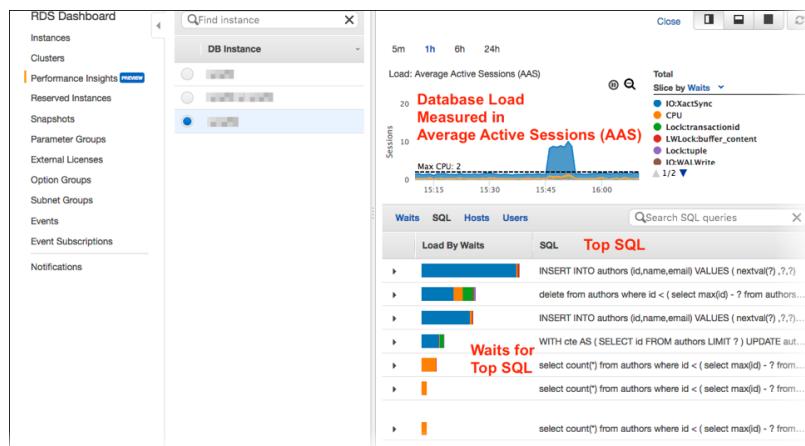
1. Review the **Average Active Sessions** chart and see if there are any incidents of database load exceeding the **Max CPU** line.

2. If there is, look at the **Average Active Sessions** chart and identify which wait state or states are primarily responsible.
3. Identify the digest queries causing the load by seeing which of the queries the **SQL** tab on the top load items table are contributing most to those wait states. You can identify these by the **DB Load by Wait** column.
4. Choose one of these digest queries in the **SQL** tab to expand it and see the child queries that it is composed of.

For example, in the dashboard following, **IO:XactSync** waits are a frequent issue. **CPU** wait is less, but it still contributes to load.

The first four roll-up queries in the **SQL** tab of the top load items table correlate strongly to the first state. Thus, those are the ones to drill into and examine the child queries of. You do so to determine how they are contributing to the performance issue.

The last three roll-up queries are the major contributors to CPU. These are the queries to investigate if CPU load is an issue.

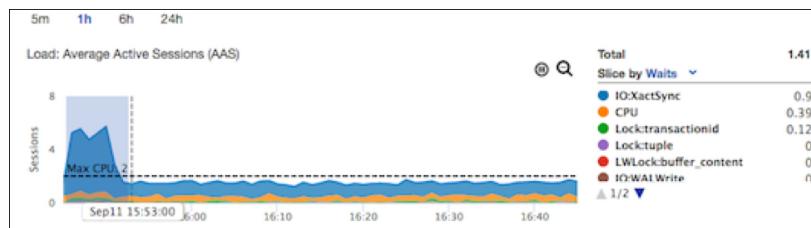


Additional User Interface Features

You can use other features of the Performance Insights user interface to help analyze performance data.

Click-and-Drag Zoom In

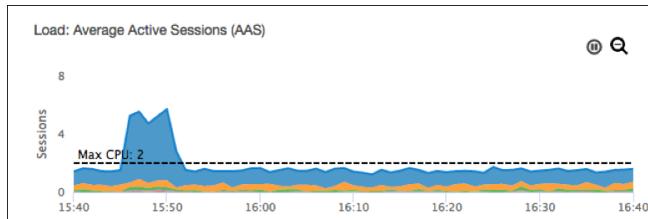
In the Performance Insights interface, you can choose a small portion of the load chart and zoom in on the detail.



To zoom in on a portion of the load chart, choose the start time and drag to the end of the time period you want. When you do this, the selected area is highlighted. When you release the mouse, the load chart zooms in on the selected region, and the **Top N** table is recalculated.

Pause and Zoom Out

In the upper-right corner of the load chart, you can find the **Pause** and **Zoom out** tools.



When you choose **Pause**, the load chart stops autorefreshing. When you choose **Pause** again, the chart resumes autorefreshing.

When you choose **Zoom out**, the load chart zooms out to the next largest time interval.

Performance Insights API

The Amazon RDS Performance Insights API provides visibility into the performance of your RDS instance, when Performance Insights is enabled for supported engine types. Amazon CloudWatch Logs provides the authoritative source for vended monitoring metrics for AWS services. Performance Insights offers a domain-specific view of database load measured as average active sessions and provided to API consumers as a two-dimensional time-series dataset. The time dimension of the data provides database load data for each time point in the queried time range. Each time point decomposes overall load in relation to the requested dimensions, such as `SQL`, `Wait-event`, `User`, or `Host`, measured at that time point.

For more information, see the [Amazon RDS Performance Insights API Reference](#).

Performance Insights Metrics Published to Amazon CloudWatch

Performance Insights automatically publishes metrics to Amazon CloudWatch. Each of these per second metrics represents the average over the last 60 seconds.

Metric	Description
DBLoad	The average number of active sessions for the DB engine.
DBLoadCPU	The number of active sessions where the wait event type is CPU.
DBLoadNonCPU	The number of active sessions where the wait event type is not CPU.

You can examine these metrics using the CloudWatch console, the AWS CLI, or the CloudWatch API.

For example, you can get the statistics for the `DBLoad` metric by running the [get-metric-statistics](#) command.

```
aws cloudwatch get-metric-statistics --region us-west-2 --namespace AWS/RDS --metric-name DBLoad --period 60 --statistics Sum --start-time 1532035185 --end-time 1532036185 --dimensions Name=DBInstanceIdentifier,Value=db-loadtest-0
```

This example generates output similar to the following.

```
{  
  "Datapoints": [  
    {  
      "Timestamp": "2018-07-19T21:30:00Z",  
      "Unit": "None",  
      "Sum": 1380.0  
    },  
    {  
      "Timestamp": "2018-07-19T21:34:00Z",  
      "Unit": "None",  
      "Sum": 1380.0  
    },  
    {  
      "Timestamp": "2018-07-19T21:35:00Z",  
      "Unit": "None",  
      "Sum": 1380.0  
    },  
    {  
      "Timestamp": "2018-07-19T21:31:00Z",  
      "Unit": "None",  
      "Sum": 1380.0  
    },  
    {  
      "Timestamp": "2018-07-19T21:32:00Z",  
      "Unit": "None",  
      "Sum": 1380.0  
    },  
    {  
      "Timestamp": "2018-07-19T21:29:00Z",  
      "Unit": "None",  
      "Sum": 8280.0  
    },  
    {  
      "Timestamp": "2018-07-19T21:33:00Z",  
      "Unit": "None",  
      "Sum": 1380.0  
    }  
  ],  
  "Label": "DBLoad"  
}
```

For more information about CloudWatch, see [What is Amazon CloudWatch?](#) in the *Amazon CloudWatch User Guide*.

Logging Performance Insights Operations by Using AWS CloudTrail

Performance Insights is integrated with AWS CloudTrail. CloudTrail captures low-level API requests made by or on behalf of Performance Insights in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures calls made from the Performance Insights in the RDS console or from the Performance Insights low-level API. Using the information collected by CloudTrail, you can determine what request was made to Performance Insights. You can also determine the source IP address it was made from, who made it, when it was made, and so on. CloudTrail logging is automatically enabled in your AWS account. To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

Any low-level API calls made to Performance Insights actions are tracked in log files. Performance Insights records are written together with other AWS service records in a log file. CloudTrail determines

when to create and write to a new file based on a time period and file size. The following API operations are supported:

- [DescribeDimensionKeys](#)
- [GetResourceMetrics](#)

Using Amazon RDS Recommendations

Amazon RDS provides automated recommendations for database resources. These recommendations provide best practice guidance by analyzing DB instance configuration, usage, and performance data.

You can find examples of these recommendations in the following table.

Type	Description	Recommendation	Additional Information
Engine version outdated	Your DB instance is not running the latest minor engine version.	We recommend that you upgrade to the latest version because it contains the latest security fixes and other improvements.	Upgrading a DB Instance Engine Version (p. 123)
Pending maintenance available	You have pending maintenance available on your DB instance.	We recommend that you perform the pending maintenance available on your DB instance. Updates to the operating system most often occur for security issues and should be done as soon as possible.	Maintaining a DB Instance (p. 117)
Automated backups disabled	Your DB instance has automated backups disabled.	We recommend that you enable automated backups on your DB instance. Automated backups enable point-in-time recovery of your DB instance. You receive backup storage up to the storage size of your DB instance at no additional charge.	Working With Backups (p. 208)
Magnetic volumes in use	Your DB instance is using magnetic storage.	Magnetic storage is not recommended for most DB instances. We recommend switching to General Purpose (SSD) storage or provisioned IOPS storage.	DB instance storage (p. 103)
EC2-Classic platform in use	Your DB instance is using the legacy EC2-Classic platform.	We recommend moving your DB instance to the EC2-VPC platform for better network access control. Amazon VPC provides a virtual network that is logically isolated from other virtual networks in the AWS Cloud.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412)
Enhanced Monitoring disabled	Your DB instance doesn't have Enhanced Monitoring enabled.	We recommend enabling Enhanced Monitoring. Enhanced Monitoring provides real-time operating system metrics for monitoring and troubleshooting.	Enhanced Monitoring (p. 256)
Encryption disabled	Your DB instance doesn't have encryption enabled.	We recommend enabling encryption. You can encrypt your existing Amazon RDS DB instances by restoring from an encrypted snapshot.	Encrypting Amazon RDS Resources (p. 389)
Previous generation	Your DB instance is running on a	Previous-generation DB instance classes have been replaced by DB	DB Instance Class (p. 82)

Type	Description	Recommendation	Additional Information
DB instance class in use	previous-generation DB instance class.	instance classes with better price, better performance, or both. We recommend running your DB instance on a later generation DB instance class.	

Amazon RDS generates recommendations periodically across all accounts and resources.

Topics

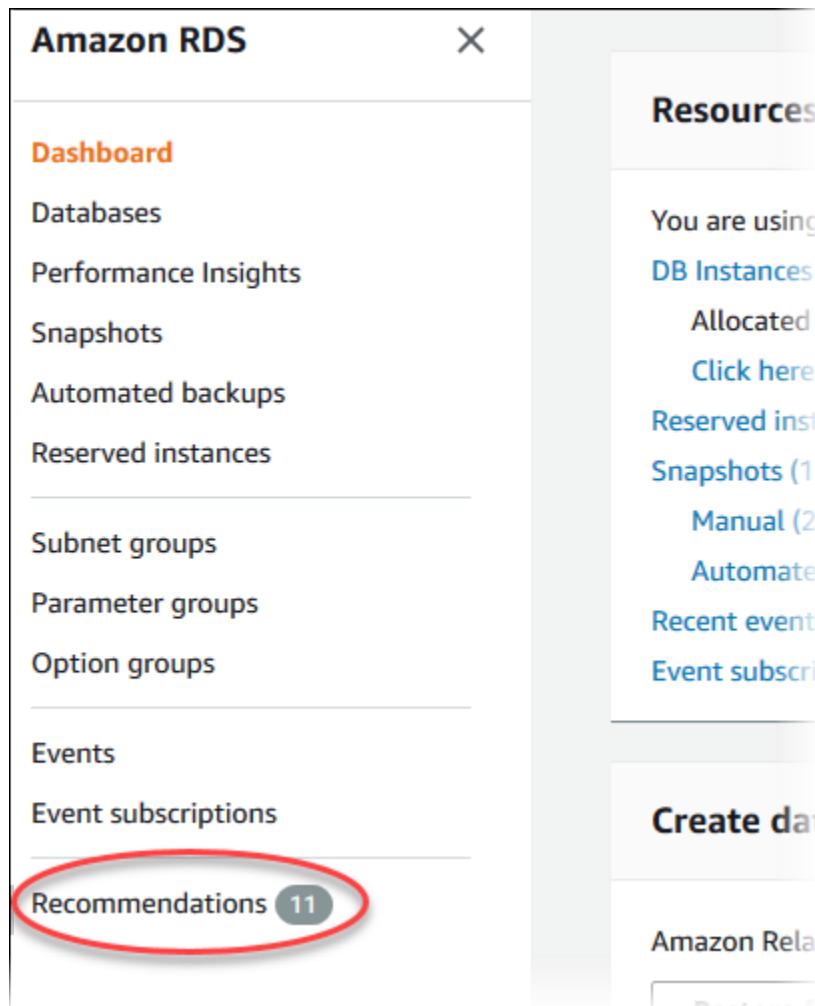
- [Responding to Amazon RDS Recommendations \(p. 283\)](#)

Responding to Amazon RDS Recommendations

You can find recommendations in the AWS Management Console. You can perform the recommended action immediately, schedule it for the next maintenance window, or dismiss it.

To respond to Amazon RDS recommendations

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Recommendations**.



The Recommendations page appears.

The Recommendations page displays four active recommendations:

- Enhanced monitoring disabled (2)**
DB instances that do not have Enhanced Monitoring enabled. [Info](#)
- Magnetic volume in use (1)**
DB instances using magnetic storage. [Info](#)
- Encryption disabled (1)**
DB instances that do not have encryption enabled. [Info](#)

3. On the **Recommendations** page, choose one of the following:
 - **Active** – Shows the current recommendations that you can apply, dismiss, or schedule.

- **Dismissed** – Shows the recommendations that have been dismissed. When you choose **Dismissed**, you can apply these dismissed recommendations.
- **Applied** – Shows the recommendations that are currently applied.
- **Scheduled** – Shows the recommendations that are scheduled but not yet applied. These recommendations will be applied in the next scheduled maintenance window.

From any list of recommendations, you can open a section to view the recommendations in that section.

Recommendations

Active (4) Dismissed (0) Applied (0) Scheduled (0)

▼ Enhanced monitoring disabled (2)
DB instances that do not have Enhanced Monitoring enabled. [Info](#)

DB instances

Filter recommendations Dismiss Apply now < 1 > ⌂

<input type="checkbox"/>	Resource ▾	Recommendation	Recommendation time ▲
<input type="checkbox"/>	mymariadb	Enhanced monitoring is not enabled on your DB instance. Enhanced monitoring provides real-time operating system metrics for monitoring and troubleshooting.	Mon Jul 16 10:02:41 GMT-700 2018
<input type="checkbox"/>	mysqladb	Enhanced monitoring is not enabled on your DB instance. Enhanced monitoring provides real-time operating system metrics for monitoring and troubleshooting.	Mon Jul 16 10:37:49 GMT-700 2018

► Magnetic volume in use (1)
DB instances using magnetic storage. [Info](#)

► Encryption disabled (1)
DB instances that do not have encryption enabled. [Info](#)

To configure preferences for displaying recommendations in each section, choose the **Preferences** icon.

The screenshot shows the 'Recommendations' page with the 'Active (4)' tab selected. There are three main sections: 'Enhanced monitoring disabled (2)', 'Magnetic volume in use (1)', and 'Encryption disabled (1)'. Each section lists DB instances and their status. The 'Enhanced monitoring disabled' section has two entries: 'mymariadb' and 'mysqldb'. The 'Apply now' button for the first entry is circled in red.

DB instance	Description	Recommendation time
mymariadb	Enhanced monitoring is not enabled on your DB instance. Enhanced monitoring provides real-time operating system metrics for monitoring and troubleshooting.	Mon Jul 16 10:02:41 GMT-700 2018
mysqldb	Enhanced monitoring is not enabled on your DB instance. Enhanced monitoring provides real-time operating system metrics for monitoring and troubleshooting.	Mon Jul 16 10:37:49 GMT-700 2018

From the **Preferences** window that appears, you can set display options. These options include the visible columns and the number of recommendations to display on the page.

4. Manage your active recommendations:

- Choose **Active** and open one or more sections to view the recommendations in them.
- Choose one or more recommendations and choose **Apply now** (to apply them immediately), **Apply in next maintenance window**, or **Dismiss**.

If the **Apply now** button appears for a recommendation but is unavailable (grayed out), the DB instance is not available. You can apply recommendations immediately only if the DB instance status is **available**. For example, you can't apply recommendations immediately to the DB instance if its status is **modifying**. In this case, wait for the DB instance to be available and apply the recommendation.

If the **Active** button doesn't appear for a recommendation, you can't apply the recommendation using the **Recommendations** page. You can modify the DB instance to apply the recommendation manually. For more information about modifying a DB instance, see [Modifying an Amazon RDS DB Instance \(p. 115\)](#).

Note

When you choose **Apply now**, a brief DB instance outage might result.

Using Amazon RDS Event Notification

Topics

- [Amazon RDS Event Categories and Event Messages \(p. 288\)](#)
- [Subscribing to Amazon RDS Event Notification \(p. 294\)](#)
- [Listing Your Amazon RDS Event Notification Subscriptions \(p. 297\)](#)
- [Modifying an Amazon RDS Event Notification Subscription \(p. 299\)](#)
- [Adding a Source Identifier to an Amazon RDS Event Notification Subscription \(p. 301\)](#)
- [Removing a Source Identifier from an Amazon RDS Event Notification Subscription \(p. 302\)](#)
- [Listing the Amazon RDS Event Notification Categories \(p. 303\)](#)
- [Deleting an Amazon RDS Event Notification Subscription \(p. 304\)](#)

Amazon RDS uses the Amazon Simple Notification Service (Amazon SNS) to provide notification when an Amazon RDS event occurs. These notifications can be in any notification form supported by Amazon SNS for an AWS Region, such as an email, a text message, or a call to an HTTP endpoint.

Amazon RDS groups these events into categories that you can subscribe to so that you can be notified when an event in that category occurs. You can subscribe to an event category for a DB instance, DB snapshot, DB parameter group, or DB security group. For example, if you subscribe to the Backup category for a given DB instance, you are notified whenever a backup-related event occurs that affects the DB instance. If you subscribe to a configuration change category for a DB security group, you are notified when the DB security group is changed. You also receive notification when an event notification subscription changes.

Event notifications are sent to the addresses that you provide when you create the subscription. You might want to create several different subscriptions, such as one subscription receiving all event notifications and another subscription that includes only critical events for your production DB instances. You can easily turn off notification without deleting a subscription by choosing **No** for **Enabled** in the Amazon RDS console or by setting the `Enabled` parameter to `false` using the AWS CLI or Amazon RDS API.

Note

Amazon RDS event notifications using SMS text messages are currently available for topic Amazon Resource Names (ARNs) and Amazon RDS resources in the US-East (Northern Virginia) Region. For more information on using text messages with SNS, see [Sending and Receiving SMS Notifications Using Amazon SNS](#).

Amazon RDS uses the ARN of an Amazon SNS topic to identify each subscription. The Amazon RDS console creates the ARN for you when you create the subscription. If you use the CLI or API, you create the ARN by using the Amazon SNS console or the Amazon SNS API when you create a subscription.

Billing for Amazon RDS event notification is through the Amazon Simple Notification Service (Amazon SNS). Amazon SNS fees apply when using event notification. For more information on Amazon SNS billing, see [Amazon Simple Notification Service Pricing](#).

The process for subscribing to Amazon RDS event notification is as follows:

1. Create an Amazon RDS event notification subscription by using the Amazon RDS console, AWS CLI, or API.
2. Amazon RDS sends an approval email or SMS message to the addresses you submitted with your subscription. To confirm your subscription, choose the link in the notification you were sent.
3. When you have confirmed the subscription, the status of your subscription is updated in the Amazon RDS console's **My Event Subscriptions** section.
4. You then begin to receive event notifications.

The following section lists all categories and events that you can be notified of. It also provides information about subscribing to and working with Amazon RDS event subscriptions.

Amazon RDS Event Categories and Event Messages

Amazon RDS generates a significant number of events in categories that you can subscribe to using the Amazon RDS Console, AWS CLI, or the API. Each category applies to a source type, which can be a DB instance, DB snapshot, DB security group, or DB parameter group.

The following table shows the event category and a list of events when a DB instance is the source type.

Category	Amazon RDS Event ID	Description
availability	RDS-EVENT-0006	The DB instance is restarting due to a previous controlled shutdown, or a recovery. The DB instance will be unavailable until the restart completes.
availability	RDS-EVENT-0004	The DB instance is undergoing a controlled shutdown.
availability	RDS-EVENT-0022	An error has occurred while restarting MySQL or MariaDB.
backup	RDS-EVENT-0001	A backup of the DB instance has started.
backup	RDS-EVENT-0002	A backup of the DB instance is complete.
configuration change	RDS-EVENT-0009	The DB instance has been added to a security group.
configuration change	RDS-EVENT-0024	The DB instance is being converted to a Multi-AZ DB instance.
configuration change	RDS-EVENT-0030	The DB instance is being converted to a Single-AZ DB instance.
configuration change	RDS-EVENT-0012	The DB instance class for this DB instance is being changed.
configuration change	RDS-EVENT-0018	The current storage settings for this DB instance are being changed.
configuration change	RDS-EVENT-0011	A parameter group for this DB instance has changed.
configuration change	RDS-EVENT-0092	A parameter group for this DB instance has finished updating.
configuration change	RDS-EVENT-0028	Automatic backups for this DB instance have been disabled.
configuration change	RDS-EVENT-0032	Automatic backups for this DB instance have been enabled.
configuration change	RDS-EVENT-0033	There are [count] users that match the master user name. Users not tied to a specific host have been reset.
configuration change	RDS-EVENT-0025	The DB instance has been converted to a Multi-AZ DB instance.

Category	Amazon RDS Event ID	Description
configuration change	RDS-EVENT-0029	The DB instance has been converted to a Single-AZ DB instance.
configuration change	RDS-EVENT-0014	The DB instance class for this DB instance has changed.
configuration change	RDS-EVENT-0017	The storage settings for this DB instance have changed.
configuration change	RDS-EVENT-0010	The DB instance has been removed from a security group.
configuration change	RDS-EVENT-0016	The master password for the DB instance has been reset.
configuration change	RDS-EVENT-0067	An attempt to reset the master password for the DB instance has failed.
configuration change	RDS-EVENT-0078	The Enhanced Monitoring configuration has been changed.
creation	RDS-EVENT-0005	A DB instance is being created.
deletion	RDS-EVENT-0003	The DB instance is being deleted.
failover	RDS-EVENT-0034	Amazon RDS is not attempting a requested failover because a failover recently occurred on the DB instance.
failover	RDS-EVENT-0013	A Multi-AZ failover that resulted in the promotion of a standby instance has started.
failover	RDS-EVENT-0015	A Multi-AZ failover that resulted in the promotion of a standby instance is complete. It may take several minutes for the DNS to transfer to the new primary DB instance.
failover	RDS-EVENT-0065	The instance has recovered from a partial failover.
failover	RDS-EVENT-0049	A Multi-AZ failover has completed.
failover	RDS-EVENT-0050	A Multi-AZ activation has started after a successful instance recovery.
failover	RDS-EVENT-0051	A Multi-AZ activation is complete. Your database should be accessible now.
failure	RDS-EVENT-0031	The DB instance has failed due to an incompatible configuration or an underlying storage issue. Begin a point-in-time-restore for the DB instance.
failure	RDS-EVENT-0036	The DB instance is in an incompatible network. Some of the specified subnet IDs are invalid or do not exist.

Category	Amazon RDS Event ID	Description
failure	RDS-EVENT-0035	The DB instance has invalid parameters. For example, MySQL could not start because a memory-related parameter is set too high for this instance class, so the customer action would be to modify the memory parameter and reboot the DB instance.
failure	RDS-EVENT-0058	Error while creating Statspack user account PERFSTAT. Please drop the account before adding the Statspack option.
failure	RDS-EVENT-0079	Enhanced Monitoring cannot be enabled without the enhanced monitoring IAM role. For information on creating the enhanced monitoring IAM role, see To create an IAM role for Amazon RDS Enhanced Monitoring (p. 257) .
failure	RDS-EVENT-0080	Enhanced Monitoring was disabled due to an error making the configuration change. It is likely that the enhanced monitoring IAM role is configured incorrectly. For information on creating the enhanced monitoring IAM role, see To create an IAM role for Amazon RDS Enhanced Monitoring (p. 257) .
failure	RDS-EVENT-0081	The IAM role that you use to access your Amazon S3 bucket for SQL Server native backup and restore is configured incorrectly. For more information, see Setting Up for Native Backup and Restore (p. 534) .
failure	RDS-EVENT-0082	Aurora was unable to copy backup data from an Amazon S3 bucket. It is likely that the permissions for Aurora to access the Amazon S3 bucket are configured incorrectly. For more information, see Migrating Data from an External MySQL Database to an Amazon Aurora MySQL DB Cluster .
low storage	RDS-EVENT-0089	The DB instance has consumed more than 90% of its allocated storage. You can monitor the storage space for a DB instance using the Free Storage Space metric. For more information, see Viewing DB Instance Metrics (p. 254) .
low storage	RDS-EVENT-0007	The allocated storage for the DB instance has been exhausted. To resolve this issue, you should allocate additional storage for the DB instance. For more information, see the RDS FAQ . You can monitor the storage space for a DB instance using the Free Storage Space metric. For more information, see Viewing DB Instance Metrics (p. 254) .
maintenance	RDS-EVENT-0026	Offline maintenance of the DB instance is taking place. The DB instance is currently unavailable.
maintenance	RDS-EVENT-0027	Offline maintenance of the DB instance is complete. The DB instance is now available.
notification	RDS-EVENT-0044	Operator-issued notification. For more information, see the event message.

Category	Amazon RDS Event ID	Description
notification	RDS-EVENT-0047	Patching of the DB instance has completed.
notification	RDS-EVENT-0048	Patching of the DB instance has been delayed.
notification	RDS-EVENT-0054	The MySQL storage engine you are using is not InnoDB, which is the recommended MySQL storage engine for Amazon RDS. For information about MySQL storage engines, see Supported Storage Engines for MySQL on Amazon RDS .
notification	RDS-EVENT-0055	<p>The number of tables you have for your DB instance exceeds the recommended best practices for Amazon RDS. Please reduce the number of tables on your DB instance.</p> <p>For information about recommended best practices, see Amazon RDS Basic Operational Guidelines (p. 70).</p>
notification	RDS-EVENT-0056	<p>The number of databases you have for your DB instance exceeds the recommended best practices for Amazon RDS. Please reduce the number of databases on your DB instance.</p> <p>For information about recommended best practices, see Amazon RDS Basic Operational Guidelines (p. 70).</p>
notification	RDS-EVENT-0064	The TDE key has been rotated. For information about recommended best practices, see Amazon RDS Basic Operational Guidelines (p. 70) .
notification	RDS-EVENT-0084	You attempted to convert a DB instance to Multi-AZ, but it contains in-memory file groups that are not supported for Multi-AZ. For more information, see Multi-AZ Deployments for Microsoft SQL Server (p. 551) .
notification	RDS-EVENT-0087	The DB instance has been stopped.
notification	RDS-EVENT-0088	The DB instance has been started.
notification	RDS-EVENT-0154	The DB instance is being started due to it exceeding the maximum allowed time being stopped.
notification	RDS-EVENT-0155	The DB instance has a DB engine minor version upgrade available.
read replica	RDS-EVENT-0045	<p>An error has occurred in the read replication process. For more information, see the event message.</p> <p>For information on troubleshooting Read Replica errors, see Troubleshooting a MySQL Read Replica Problem (p. 661).</p>

Category	Amazon RDS Event ID	Description
read replica	RDS-EVENT-0046	The Read Replica has resumed replication. This message appears when you first create a Read Replica, or as a monitoring message confirming that replication is functioning properly. If this message follows an RDS-EVENT-0045 notification, then replication has resumed following an error or after replication was stopped.
read replica	RDS-EVENT-0057	Replication on the Read Replica was terminated.
read replica	RDS-EVENT-0062	Replication on the Read Replica was manually stopped.
read replica	RDS-EVENT-0063	Replication on the Read Replica was reset.
recovery	RDS-EVENT-0020	Recovery of the DB instance has started. Recovery time will vary with the amount of data to be recovered.
recovery	RDS-EVENT-0021	Recovery of the DB instance is complete.
recovery	RDS-EVENT-0023	A manual backup has been requested but Amazon RDS is currently in the process of creating a DB snapshot. Submit the request again after Amazon RDS has completed the DB snapshot.
recovery	RDS-EVENT-0052	Recovery of the Multi-AZ instance has started. Recovery time will vary with the amount of data to be recovered.
recovery	RDS-EVENT-0053	Recovery of the Multi-AZ instance is complete.
recovery	RDS-EVENT-0066	The SQL Server DB instance is re-establishing its mirror. Performance will be degraded until the mirror is reestablished. A database was found with non-FULL recovery model. The recovery model was changed back to FULL and mirroring recovery was started. (<dbname>: <recovery model found>[,...])"
restoration	RDS-EVENT-0008	The DB instance has been restored from a DB snapshot.
restoration	RDS-EVENT-0019	The DB instance has been restored from a point-in-time backup.
security	RDS-EVENT-0068	The CloudHSM Classic partition password was decrypted by the system.

The following table shows the event category and a list of events when a DB parameter group is the source type.

Category	RDS Event ID	Description
configuration change	RDS-EVENT-0037	The parameter group was modified.

The following table shows the event category and a list of events when a DB security group is the source type.

Category	RDS Event ID	Description
configuration change	RDS-EVENT-0038	The security group has been modified.
failure	RDS-EVENT-0039	The Amazon EC2 security group owned by [user] does not exist; authorization for the security group has been revoked.

The following table shows the event category and a list of events when a DB snapshot is the source type.

Category	RDS Event ID	Description
creation	RDS-EVENT-0040	A manual DB snapshot is being created.
deletion	RDS-EVENT-0041	A DB snapshot has been deleted.
creation	RDS-EVENT-0042	A manual DB snapshot has been created.
restoration	RDS-EVENT-0043	A DB instance is being restored from a DB snapshot.
notification	RDS-EVENT-0059	Started the copy of the cross region DB snapshot [DB snapshot name] from source region [region name].
notification	RDS-EVENT-0060	Finished the copy of the cross region DB snapshot [DB snapshot name] from source region [region name] in [time] minutes.
notification	RDS-EVENT-0061	The copy of a cross region DB snapshot failed.
creation	RDS-EVENT-0090	An automated DB snapshot is being created.
creation	RDS-EVENT-0091	An automated DB snapshot has been created.

The following table shows the event category and a list of events when an Aurora DB cluster is the source type.

Category	RDS Event ID	Description
failover	RDS-EVENT-0069	A failover for the DB cluster has failed.
failover	RDS-EVENT-0070	A failover for the DB cluster has restarted.
failover	RDS-EVENT-0071	A failover for the DB cluster has finished.
failover	RDS-EVENT-0072	A failover for the DB cluster has begun within the same Availability Zone.
failover	RDS-EVENT-0073	A failover for the DB cluster has begun across Availability Zones.
failure	RDS-EVENT-0083	Aurora was unable to copy backup data from an Amazon S3 bucket. It is likely that the permissions for Aurora to access the Amazon S3 bucket are

Category	RDS Event ID	Description
		configured incorrectly. For more information, see Migrating Data from an External MySQL Database to an Amazon Aurora MySQL DB Cluster .
notification	RDS-EVENT-0076	Migration to an Aurora DB cluster failed.
notification	RDS-EVENT-0077	An attempt to convert a table from the source database to InnoDB failed during the migration to an Aurora DB cluster.
notification	RDS-EVENT-0149	A scaling point was not found.
notification	RDS-EVENT-0150	The DB cluster stopped.
notification	RDS-EVENT-0151	The DB cluster started.
notification	RDS-EVENT-0152	The DB cluster stop failed.
notification	RDS-EVENT-0153	The DB cluster is being started due to it exceeding the maximum allowed time being stopped.
notification	RDS-EVENT-0156	The DB cluster has a DB engine minor version upgrade available.

The following table shows the event category and a list of events when an Aurora DB cluster snapshot is the source type.

Category	RDS Event ID	Description
backup	RDS-EVENT-0074	Creation of a manual DB cluster snapshot has started.
backup	RDS-EVENT-0075	A manual DB cluster snapshot has been created.

Subscribing to Amazon RDS Event Notification

You can create an Amazon RDS event notification subscription so you can be notified when an event occurs for a given DB instance, DB snapshot, DB security group, or DB parameter group. The simplest way to create a subscription is with the RDS console. If you choose to create event notification subscriptions using the CLI or API, you must create an Amazon Simple Notification Service topic and subscribe to that topic with the Amazon SNS console or Amazon SNS API. You will also need to retain the Amazon Resource Name (ARN) of the topic because it is used when submitting CLI commands or API actions. For information on creating an SNS topic and subscribing to it, see [Getting Started with Amazon SNS](#).

You can specify the type of source you want to be notified of and the Amazon RDS source that triggers the event. These are defined by the **SourceType** (type of source) and the **SourceIdentifier** (the Amazon RDS source generating the event). If you specify both the **SourceType** and **SourceIdentifier**, such as **SourceType = db-instance** and **SourceIdentifier = myDBInstance1**, you receive all the DB instance events for the specified source. If you specify a **SourceType** but don't specify a **SourceIdentifier**, you receive notice of the events for that source type for all your Amazon RDS sources. If you don't specify either the **SourceType** or the **SourceIdentifier**, you are notified of events generated from all Amazon RDS sources belonging to your customer account.

Note

Event notifications might take up to five minutes to be delivered.

AWS Management Console

To subscribe to RDS event notification

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In navigation pane, choose **Event subscriptions**.
3. In the **Event subscriptions** pane, choose **Create event subscription**.
4. In the **Create event subscription** dialog box, do the following:
 - a. For **Name**, enter a name for the event notification subscription.
 - b. For **Send notifications to**, choose an existing Amazon SNS ARN for an Amazon SNS topic, or choose **create topic** to enter the name of a topic and a list of recipients.
 - c. For **Source type**, choose a source type.
 - d. Choose **Yes** to enable the subscription. If you want to create the subscription but to not have notifications sent yet, choose **No**.
 - e. Depending on the source type you selected, choose the event categories and sources that you want to receive event notifications for.
 - f. Choose **Create**.

The Amazon RDS console indicates that the subscription is being created.

Name	Status	Source Type	Enabled
Configchangerdpgres	active	Instances	Yes
Test	creating	Instances	Yes

CLI

To subscribe to RDS event notification, use the AWS CLI `create-event-subscription` command. Include the following required parameters:

- `--subscription-name`
- `--sns-topic-arn`

Example

For Linux, OS X, or Unix:

```
aws rds create-event-subscription \
--subscription-name myeventsubscription \
--sns-topic-arn arn:aws:sns:us-east-1:802#####:myawsuser-RDS \
--enabled
```

For Windows:

```
aws rds create-event-subscription ^
--subscription-name myeventsubscription ^
--sns-topic-arn arn:aws:sns:us-east-1:802#####:myawsuser-RDS ^
--enabled
```

API

To subscribe to Amazon RDS event notification, call the Amazon RDS API function [CreateEventSubscription](#). Include the following required parameters:

- `SubscriptionName`
- `SnsTopicArn`

Listing Your Amazon RDS Event Notification Subscriptions

You can list your current Amazon RDS event notification subscriptions.

AWS Management Console

To list your current Amazon RDS event notification subscriptions

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Event subscriptions**. The **Event subscriptions** pane shows all your event notification subscriptions.

Event subscriptions (2)		
	Name	Status
<input type="checkbox"/>	Configchangerdpgres	<input checked="" type="checkbox"/> active
<input type="checkbox"/>	Postgresnotification	<input checked="" type="checkbox"/> active

CLI

To list your current Amazon RDS event notification subscriptions, use the AWS CLI [describe-event-subscriptions](#) command.

Example

The following example describes all event subscriptions.

```
aws rds describe-event-subscriptions
```

The following example describes the `myfirsteventsubscription`.

```
aws rds describe-event-subscriptions --subscription-name myfirsteventsubscription
```

API

To list your current Amazon RDS event notification subscriptions, call the Amazon RDS API [DescribeEventSubscriptions](#) action.

Example

The following code example lists up to 100 event subscriptions.

```
https://rds.us-east-1.amazonaws.com/?Action=DescribeEventSubscriptions
```

```
&MaxRecords=100
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-east-1/rds/aws4_request
&X-Amz-Date=20140428T161907Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=4208679fe967783a1a149c826199080a066085d5a88227a80c6c0cadb3e8c0d4
```

The following example describes the `myfirsteventsSubscription`.

```
https://rds.us-east-1.amazonaws.com/
?Action=DescribeEventSubscriptions
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&SubscriptionName=myfirsteventsSubscription
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-east-1/rds/aws4_request
&X-Amz-Date=20140428T161907Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=4208679fe967783a1a149c826199080a066085d5a88227a80c6c0cadb3e8c0d4
```

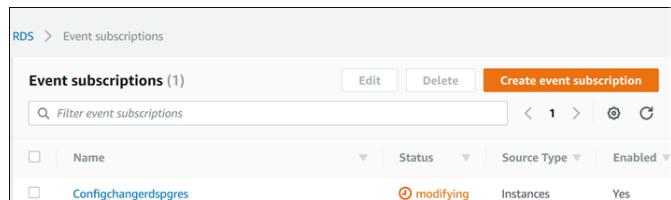
Modifying an Amazon RDS Event Notification Subscription

After you have created a subscription, you can change the subscription name, source identifier, categories, or topic ARN.

AWS Management Console

To modify an Amazon RDS event notification subscription

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Event subscriptions**.
3. In the **Event subscriptions** pane, choose the subscription that you want to modify and choose **Edit**.
4. Make your changes to the subscription in either the **Target** or **Source** section.
5. Choose **Edit**. The Amazon RDS console indicates that the subscription is being modified.



CLI

To modify an Amazon RDS event notification subscription, use the AWS CLI [modify-event-subscription](#) command. Include the following required parameter:

- `--subscription-name`

Example

The following code enables `myeventsSubscription`.

For Linux, OS X, or Unix:

```
aws rds modify-event-subscription \
--subscription-name myeventsSubscription \
--enabled
```

For Windows:

```
aws rds modify-event-subscription ^
--subscription-name myeventsSubscription ^
--enabled
```

API

To modify an Amazon RDS event, call the Amazon RDS API action [ModifyEventSubscription](#). Include the following required parameter:

- `SubscriptionName`

Adding a Source Identifier to an Amazon RDS Event Notification Subscription

You can add a source identifier (the Amazon RDS source generating the event) to an existing subscription.

AWS Management Console

You can easily add or remove source identifiers using the Amazon RDS console by selecting or deselecting them when modifying a subscription. For more information, see [Modifying an Amazon RDS Event Notification Subscription \(p. 299\)](#).

CLI

To add a source identifier to an Amazon RDS event notification subscription, use the AWS CLI [add-source-identifier-to-subscription](#) command. Include the following required parameters:

- `--subscription-name`
- `--source-identifier`

Example

The following example adds the source identifier `mysqldb` to the `myrdseventsSubscription` subscription.

For Linux, OS X, or Unix:

```
aws rds add-source-identifier-to-subscription \
  --subscription-name myrdseventsSubscription \
  --source-identifier mysqldb
```

For Windows:

```
aws rds add-source-identifier-to-subscription ^
  --subscription-name myrdseventsSubscription ^
  --source-identifier mysqldb
```

API

To add a source identifier to an Amazon RDS event notification subscription, call the Amazon RDS API [AddSourceIdentifierToSubscription](#). Include the following required parameters:

- `SubscriptionName`
- `SourceIdentifier`

Removing a Source Identifier from an Amazon RDS Event Notification Subscription

You can remove a source identifier (the Amazon RDS source generating the event) from a subscription if you no longer want to be notified of events for that source.

AWS Management Console

You can easily add or remove source identifiers using the Amazon RDS console by selecting or deselecting them when modifying a subscription. For more information, see [Modifying an Amazon RDS Event Notification Subscription \(p. 299\)](#).

CLI

To remove a source identifier from an Amazon RDS event notification subscription, use the AWS CLI `remove-source-identifier-from-subscription` command. Include the following required parameters:

- `--subscription-name`
- `--source-identifier`

Example

The following example removes the source identifier `mysqldb` from the `myrdseventsSubscription` subscription.

For Linux, OS X, or Unix:

```
aws rds remove-source-identifier-from-subscription \
--subscription-name myrdseventsSubscription \
--source-identifier mysqldb
```

For Windows:

```
aws rds remove-source-identifier-from-subscription ^
--subscription-name myrdseventsSubscription ^
--source-identifier mysqldb
```

API

To remove a source identifier from an Amazon RDS event notification subscription, use the Amazon RDS API `RemoveSourceIdentifierFromSubscription` command. Include the following required parameters:

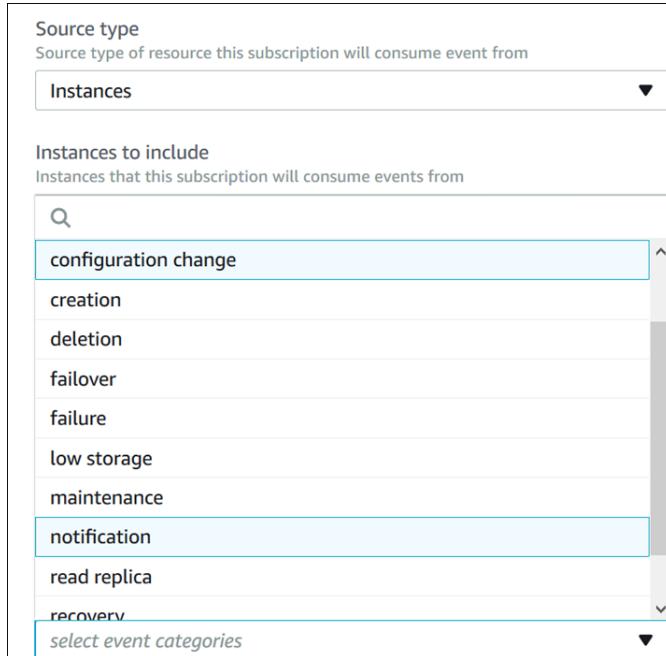
- `SubscriptionName`
- `SourceIdentifier`

Listing the Amazon RDS Event Notification Categories

All events for a resource type are grouped into categories. To view the list of categories available, use the following procedures.

AWS Management Console

When you create or modify an event notification subscription, the event categories are displayed in the Amazon RDS console. For more information, see [Modifying an Amazon RDS Event Notification Subscription \(p. 299\)](#).



CLI

To list the Amazon RDS event notification categories, use the AWS CLI `describe-event-categories` command. This command has no required parameters.

Example

```
aws rds describe-event-categories
```

API

To list the Amazon RDS event notification categories, use the Amazon RDS API `DescribeEventCategories` command. This command has no required parameters.

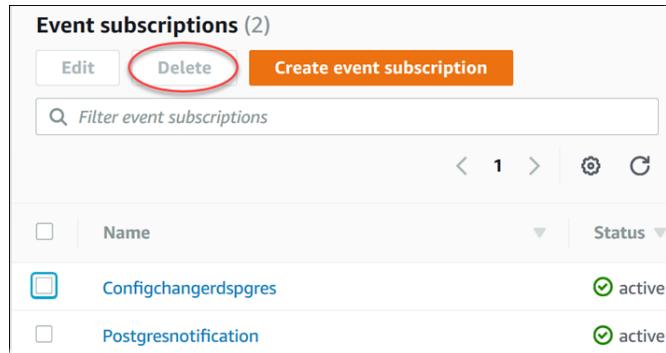
Deleting an Amazon RDS Event Notification Subscription

You can delete a subscription when you no longer need it. All subscribers to the topic will no longer receive event notifications specified by the subscription.

AWS Management Console

To delete an Amazon RDS event notification subscription

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **DB Event Subscriptions**.
3. In the **My DB Event Subscriptions** pane, choose the subscription that you want to delete.
4. Choose **Delete**.
5. The Amazon RDS console indicates that the subscription is being deleted.



The screenshot shows the 'Event subscriptions (2)' section of the AWS Management Console. At the top, there are three buttons: 'Edit', 'Delete' (which is circled in red), and 'Create event subscription'. Below these are search and filter options. The main area displays two event subscriptions in a table:

	Name	Status
<input type="checkbox"/>	Configchangedspgres	active
<input type="checkbox"/>	Postgresnotification	active

CLI

To delete an Amazon RDS event notification subscription, use the AWS CLI `delete-event-subscription` command. Include the following required parameter:

- `--subscription-name`

Example

The following example deletes the subscription `myrdssubscription`.

```
delete-event-subscription --subscription-name myrdssubscription
```

API

To delete an Amazon RDS event notification subscription, use the RDS API `DeleteEventSubscription` command. Include the following required parameter:

- `SubscriptionName`

Viewing Amazon RDS Events

Amazon RDS keeps a record of events that relate to your DB instances, DB snapshots, DB security groups, and DB parameter groups. This information includes the date and time of the event, the source name and source type of the event, and a message associated with the event.

You can retrieve events for your RDS resources through the AWS Management Console, which shows events from the past 24 hours. You can also retrieve events for your RDS resources by using the [describe-events](#) AWS CLI command, or the [DescribeEvents](#) RDS API action. If you use the AWS CLI or the RDS API to view events, you can retrieve events for up to the past 14 days.

AWS Management Console

To view all Amazon RDS instance events for the past 24 hours

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Events**. The available events appear in a list.
3. Use the **Filter** list to filter the events by type, and use the text box to the right of the **Filter** list to further filter your results. For example, the following screenshot shows a list of events filtered by the DB instance event type and containing the characters **1318**.

Events (7)	
<input type="text" value="1318"/>	X
< 1 >	⚙️ ⌂
Identifier	Type
feb1318	DB cluster snapshots
feb1318	DB cluster snapshots

CLI

To view all Amazon RDS instance events for the past 7 days

You can view all Amazon RDS instance events for the past 7 days by calling the [describe-events](#) AWS CLI command and setting the `--duration` parameter to 10080.

```
aws rds describe-events --duration 10080
```

API

To view all Amazon RDS instance events for the past 14 days

You can view all Amazon RDS instance events for the past 14 days by calling the [DescribeEvents](#) RDS API action and setting the `Duration` parameter to 20160.

```
https://rds.us-west-2.amazonaws.com/  
?Action=DescribeEvents
```

```
&Duration=20160
&MaxRecords=100
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140421/us-west-2/rds/aws4_request
&X-Amz-Date=20140421T194733Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=8e313cabcd9766c56a2886b5b298fd944e0b7cfa248953c82705fdd0374f27
```

Amazon RDS Database Log Files

You can view, download, and watch database logs using the Amazon RDS console, the AWS Command Line Interface (AWS CLI), or the Amazon RDS API. Viewing, downloading, or watching transaction logs is not supported.

For engine-specific information, see the following:

- [MariaDB Database Log Files \(p. 311\)](#)
- [Microsoft SQL Server Database Log Files \(p. 319\)](#)
- [MySQL Database Log Files \(p. 320\)](#)
- [Oracle Database Log Files \(p. 328\)](#)
- [PostgreSQL Database Log Files \(p. 334\)](#)

Viewing and Listing Database Log Files

You can view database log files for your DB engine by using the Amazon RDS console. You can list what log files are available for download or monitoring by using the AWS CLI or Amazon RDS API.

Note

If you can't view the list of log files for an existing Oracle DB instance, reboot the instance to view the list.

AWS Management Console

To view a database log file

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the name of the DB instance that has the log file that you want to view.
4. Choose the **Logs & events** tab.
5. Scroll down to the **Logs** section.
6. In the **Logs** section, choose the log that you want to view, and then choose **View**.

AWS CLI

To list the available database log files for a DB instance, use the AWS CLI `describe-db-log-files` command.

The following example returns a list of log files for a DB instance named `my-db-instance`.

Example

```
aws rds describe-db-log-files --db-instance-identifier my-db-instance
```

API

To list the available database log files for a DB instance, use the Amazon RDS API `DescribeDBLogFiles` action.

Downloading a Database Log File

You can use the Amazon RDS console, AWS CLI or API to download a database log file.

AWS Management Console

To download a database log file

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the name of the DB instance that has the log file that you want to view.
4. Choose the **Logs & events** tab.
5. Scroll down to the **Logs** section.
6. In the **Logs** section, choose the button next to the log that you want to download, and then choose **Download**.
7. Open the context (right-click) menu for the link provided, and then choose **Save Link As**. Enter the location where you want the log file to be saved, and then choose **Save**.



AWS CLI

To download a database log file, use the AWS CLI command `download-db-log-file-portion`. By default, this command downloads only the latest portion of a log file. However, you can download an entire file by specifying the parameter `--starting-token 0`.

The following example shows how to download the entire contents of a log file called `log/ERROR.4` and store it in a local file called `errorlog.txt`.

Example

For Linux, OS X, or Unix:

```
aws rds download-db-log-file-portion \
    --db-instance-identifier myexampledb \
    --starting-token 0 --output text \
    --log-file-name log/ERROR.4 > errorlog.txt
```

For Windows:

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier myexampledb ^
    --starting-token 0 --output text ^
    --log-file-name log/ERROR.4 > errorlog.txt
```

RDS API

To download a database log file, use the Amazon RDS API `DownloadDBLogFilePortion` action.

Watching a Database Log File

You can monitor the contents of a log file by using the Amazon RDS console.

AWS Management Console

To watch a database log file

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the name of the DB instance that has the log file that you want to view.
4. Choose the **Logs & events** tab.
5. In the **Logs** section, choose a log file, and then choose **Watch**.

Publishing Database Logs to Amazon CloudWatch Logs

In addition to viewing and downloading DB instance logs, you can publish logs to Amazon CloudWatch Logs. CloudWatch Logs lets you perform real-time analysis of the log data, store the data in highly durable storage, and manage the data with the CloudWatch Logs Agent. AWS retains log data published to CloudWatch Logs for an indefinite time period unless you specify a retention period. For more information, see [Change Log Data Retention in CloudWatch Logs](#).

For engine-specific information, see the following:

- the section called “[Publishing MariaDB Logs to Amazon CloudWatch Logs](#)” (p. 312)
- the section called “[Publishing MySQL Logs to CloudWatch Logs](#)” (p. 321)
- the section called “[Publishing Oracle Logs to Amazon CloudWatch Logs](#)” (p. 330)
- the section called “[Publishing PostgreSQL Logs to CloudWatch Logs](#)” (p. 335)

Reading Log File Contents Using REST

Amazon RDS provides a REST endpoint that allows access to DB instance log files. This is useful if you need to write an application to stream Amazon RDS log file contents.

The syntax is:

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

The following parameters are required:

- *DBInstanceIdentifier*—the name of the DB instance that contains the log file you want to download.
- *LogFileName*—the name of the log file to be downloaded.

The response contains the contents of the requested log file, as a stream.

The following example downloads the log file named *log/ERROR.6* for the DB instance named *sample-sql* in the *us-west-2* region.

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH///////////
wEaOAIXLhngC5zp9CyB1R6abwKrXHVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229afbf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

If you specify a nonexistent DB instance, the response consists of the following error:

- `DBInstanceNotFound`—*DBInstanceIdentifier* does not refer to an existing DB instance. (HTTP status code: 404)

MariaDB Database Log Files

You can monitor the MariaDB error log, slow query log, and the general log. The MariaDB error log is generated by default; you can generate the slow query and general logs by setting parameters in your DB parameter group. Amazon RDS rotates all of the MariaDB log files; the intervals for each type are given following.

You can monitor the MariaDB logs directly through the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs. You can also access MariaDB logs by directing the logs to a database table in the main database and querying that table. You can use the `mysqlbinlog` utility to download a binary log.

For more information about viewing, downloading, and watching file-based database logs, see [Amazon RDS Database Log Files \(p. 307\)](#).

Accessing MariaDB Error Logs

The MariaDB error log is written to the `<host-name>.err` file. You can view this file by using the Amazon RDS console or by retrieving the log using the Amazon RDS API, Amazon RDS CLI, or AWS SDKs. The `<host-name>.err` file is flushed every 5 minutes, and its contents are appended to `mysql-error-running.log`. The `mysql-error-running.log` file is then rotated every hour and the hourly files generated during the last 24 hours are retained. Each log file has the hour it was generated (in UTC) appended to its name. The log files also have a timestamp that helps you determine when the log entries were written.

MariaDB writes to the error log only on startup, shutdown, and when it encounters errors. A DB instance can go hours or days without new entries being written to the error log. If you see no recent entries, it's because the server did not encounter an error that resulted in a log entry.

Accessing the MariaDB Slow Query and General Logs

The MariaDB slow query log and the general log can be written to a file or a database table by setting parameters in your DB parameter group. For information about creating and modifying a DB parameter group, see [Working with DB Parameter Groups \(p. 169\)](#). You must set these parameters before you can view the slow query log or general log in the Amazon RDS console or by using the Amazon RDS API, AWS CLI, or AWS SDKs.

You can control MariaDB logging by using the parameters in this list:

- `slow_query_log`: To create the slow query log, set to 1. The default is 0.
- `general_log`: To create the general log, set to 1. The default is 0.
- `long_query_time`: To prevent fast-running queries from being logged in the slow query log, specify a value for the shortest query execution time to be logged, in seconds. The default is 10 seconds; the minimum is 0. If `log_output = FILE`, you can specify a floating point value that goes to microsecond resolution. If `log_output = TABLE`, you must specify an integer value with second resolution. Only queries whose execution time exceeds the `long_query_time` value are logged. For example, setting `long_query_time` to 0.1 prevents any query that runs for less than 100 milliseconds from being logged.
- `log_queries_not_using_indexes`: To log all queries that do not use an index to the slow query log, set this parameter to 1. The default is 0. Queries that do not use an index are logged even if their execution time is less than the value of the `long_query_time` parameter.
- `log_output option`: You can specify one of the following options for the `log_output` parameter:
 - **TABLE** (default)– Write general queries to the `mysql.general_log` table, and slow queries to the `mysql.slow_log` table.
 - **FILE**– Write both general and slow query logs to the file system. Log files are rotated hourly.

- **NONE**—Disable logging.

When logging is enabled, Amazon RDS rotates table logs or deletes log files at regular intervals. This measure is a precaution to reduce the possibility of a large log file either blocking database use or affecting performance. **FILE** and **TABLE** logging approach rotation and deletion as follows:

- When **FILE** logging is enabled, log files are examined every hour and log files older than 24 hours are deleted. In some cases, the remaining combined log file size after the deletion might exceed the threshold of 2 percent of a DB instance's allocated space. In these cases, the largest log files are deleted until the log file size no longer exceeds the threshold.
- When **TABLE** logging is enabled, in some cases log tables are rotated every 24 hours. This rotation occurs if the space used by the table logs is more than 20 percent of the allocated storage space or the size of all logs combined is greater than 10 GB. If the amount of space used for a DB instance is greater than 90 percent of the DB instance's allocated storage space, then the thresholds for log rotation are reduced. Log tables are then rotated if the space used by the table logs is more than 10 percent of the allocated storage space or the size of all logs combined is greater than 5 GB.

When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If the backup log table already exists, then it is deleted before the current log table is copied to the backup. You can query the backup log table if needed. The backup log table for the `mysql.general_log` table is named `mysql.general_log_backup`. The backup log table for the `mysql.slow_log` table is named `mysql.slow_log_backup`.

You can rotate the `mysql.general_log` table by calling the `mysql.rds_rotate_general_log` procedure. You can rotate the `mysql.slow_log` table by calling the `mysql.rds_rotate_slow_log` procedure.

Table logs are rotated during a database version upgrade.

Amazon RDS records both **TABLE** and **FILE** log rotation in an Amazon RDS event and sends you a notification.

To work with the logs from the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs, set the `log_output` parameter to **FILE**. Like the MariaDB error log, these log files are rotated hourly. The log files that were generated during the previous 24 hours are retained.

For more information about the slow query and general logs, go to the following topics in the MariaDB documentation:

- [Slow Query Log](#)
- [General Query Log](#)

Publishing MariaDB Logs to Amazon CloudWatch Logs

You can configure your Amazon RDS MariaDB DB instance to publish log data to a log group in Amazon CloudWatch Logs. With CloudWatch Logs, you can perform real-time analysis of the log data, and use CloudWatch to create alarms and view metrics. You can use CloudWatch Logs to store your log records in highly durable storage.

Amazon RDS publishes each MariaDB database log as a separate database stream in the log group. For example, if you configure the export function to include the slow query log, slow query data is stored in a slow query log stream in the `/aws/rds/instance/my_instance/slowquery` log group.

The error log is enabled by default. The following table summarizes the requirements for the other MariaDB logs.

Log	Requirement
Audit log	The DB instance must use a custom option group with the MARIADB_AUDIT_PLUGIN option.
General log	The DB instance must use a custom parameter group with the parameter setting general_log = 1 to enable the general log.
Slow query log	The DB instance must use a custom parameter group with the parameter setting slow_query_log = 1 to enable the slow query log.
Log output	The DB instance must use a custom parameter group with the parameter setting log_output = FILE to write logs to the file system and publish them to CloudWatch Logs.

AWS Management Console

To publish MariaDB logs to CloudWatch Logs from the console

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**.
4. In the **Log exports** section, choose the logs that you want to start publishing to CloudWatch Logs.
5. Choose **Continue**, and then choose **Modify DB Instance** on the summary page.

AWS CLI

You can publish a MariaDB logs with the AWS CLI. You can call the `modify-db-instance` command with the following parameters:

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

Note

A change to the `--cloudwatch-logs-export-configuration` option is always applied to the DB instance immediately. Therefore, the `--apply-immediately` and `--no-apply-immediately` options have no effect.

You can also publish MariaDB logs by calling the following AWS CLI commands:

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-from-s3`
- `restore-db-instance-to-point-in-time`

Run one of these AWS CLI commands with the following options:

- `--db-instance-identifier`

- `--enable-cloudwatch-logs-exports`
- `--db-instance-class`
- `--engine`

Other options might be required depending on the AWS CLI command you run.

Example

The following example modifies an existing MariaDB DB instance to publish log files to CloudWatch Logs. The `--cloudwatch-logs-export-configuration` value is a JSON object. The key for this object is `EnableLogTypes`, and its value is an array of strings with any combination of `audit`, `error`, `general`, and `slowquery`.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier mydbinstance \
  --cloudwatch-logs-export-configuration '{"EnableLogTypes": \
  ["audit", "error", "general", "slowquery"]}'
```

For Windows:

```
aws rds modify-db-instance ^
  --db-instance-identifier mydbinstance ^
  --cloudwatch-logs-export-configuration '{"EnableLogTypes": \
  ["audit", "error", "general", "slowquery"]}'
```

Example

The following command creates a MariaDB DB instance and publishes log files to CloudWatch Logs. The `--enable-cloudwatch-logs-exports` value is a JSON array of strings. The strings can be any combination of `audit`, `error`, `general`, and `slowquery`.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
  --db-instance-identifier mydbinstance \
  --enable-cloudwatch-logs-exports '[{"audit", "error", "general", "slowquery"}]' \
  --db-instance-class db.m4.large \
  --engine mariadb
```

For Windows:

```
aws rds create-db-instance ^
  --db-instance-identifier mydbinstance ^
  --enable-cloudwatch-logs-exports '[{"audit", "error", "general", "slowquery"}]' ^
  --db-instance-class db.m4.large ^
  --engine mariadb
```

RDS API

You can publish MariaDB logs with the RDS API. You can call the [ModifyDBInstance](#) action with the following parameters:

- `DBInstanceIdentifier`
- `CloudwatchLogsExportConfiguration`

Note

A change to the `CloudwatchLogsExportConfiguration` parameter is always applied to the DB instance immediately. Therefore, the `ApplyImmediately` parameter has no effect.

You can also publish MariaDB logs by calling the following RDS API actions:

- `CreateDBInstance`
- `RestoreDBInstanceFromDBSnapshot`
- `RestoreDBInstanceFromS3`
- `RestoreDBInstanceToPointInTime`

Run one of these RDS API actions with the following parameters:

- `DBInstanceIdentifier`
- `EnableCloudwatchLogsExports`
- `Engine`
- `DBInstanceClass`

Other parameters might be required depending on the AWS CLI command you run.

Log File Size

The MariaDB slow query log, error log, and the general log file sizes are constrained to no more than 2 percent of the allocated storage space for a DB instance. To maintain this threshold, logs are automatically rotated every hour and log files older than 24 hours are removed. If the combined log file size exceeds the threshold after removing old log files, then the largest log files are deleted until the log file size no longer exceeds the threshold.

Managing Table-Based MariaDB Logs

You can direct the general and slow query logs to tables on the DB instance by creating a DB parameter group and setting the `log_output` server parameter to `TABLE`. General queries are then logged to the `mysql.general_log` table, and slow queries are logged to the `mysql.slow_log` table. You can query the tables to access the log information. Enabling this logging increases the amount of data written to the database, which can degrade performance.

Both the general log and the slow query logs are disabled by default. In order to enable logging to tables, you must also set the `general_log` and `slow_query_log` server parameters to 1.

Log tables keep growing until the respective logging activities are turned off by resetting the appropriate parameter to 0. A large amount of data often accumulates over time, which can use up a considerable percentage of your allocated storage space. Amazon RDS does not allow you to truncate the log tables, but you can move their contents. Rotating a table saves its contents to a backup table and then creates a new empty log table. You can manually rotate the log tables with the following command line procedures, where the command prompt is indicated by `PROMPT>`:

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

To completely remove the old data and reclaim the disk space, call the appropriate procedure twice in succession.

Binary Logging Format

MariaDB on Amazon RDS supports the *row-based*, *statement-based*, and *mixed* binary logging formats. The default binary logging format is *mixed*. For details on the different MariaDB binary log formats, see [Binary Log Formats](#) in the MariaDB documentation.

If you plan to use replication, the binary logging format is important because it determines the record of data changes that is recorded in the source and sent to the replication targets. For information about the advantages and disadvantages of different binary logging formats for replication, see [Advantages and Disadvantages of Statement-Based and Row-Based Replication](#) in the MySQL documentation.

Important

Setting the binary logging format to row-based can result in very large binary log files. Large binary log files reduce the amount of storage available for a DB instance and can increase the amount of time to perform a restore operation of a DB instance.

Statement-based replication can cause inconsistencies between the source DB instance and a Read Replica. For more information, see [Unsafe Statements for Statement-based Replication](#) in the MariaDB documentation.

To set the MariaDB binary logging format

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Parameter groups**.
3. Choose the parameter group that is used by the DB instance that you want to modify.

You can't modify a default parameter group. If the DB instance is using a default parameter group, create a new parameter group and associate it with the DB instance.

For more information on DB parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

4. For **Parameter group actions**, choose **Edit**.
5. Set the `binlog_format` parameter to the binary logging format of your choice (**ROW**, **STATEMENT**, or **MIXED**).
6. Choose **Save changes** to save the updates to the DB parameter group.

Accessing MariaDB Binary Logs

You can use the `mysqlbinlog` utility to download binary logs in text format from MariaDB DB instances. The binary log is downloaded to your local computer. For more information about using the `mysqlbinlog` utility, go to [Using mysqlbinlog](#) in the MariaDB documentation.

To run the `mysqlbinlog` utility against an Amazon RDS instance, use the following options:

- Specify the `--read-from-remote-server` option.
- `--host`: Specify the DNS name from the endpoint of the instance.
- `--port`: Specify the port used by the instance.
- `--user`: Specify a MariaDB user that has been granted the replication slave permission.
- `--password`: Specify the password for the user, or omit a password value so the utility prompts you for a password.
- `--result-file`: Specify the local file that receives the output.
- Specify the names of one or more binary log files. To get a list of the available logs, use the SQL command `SHOW BINARY LOGS`.

For more information about mysqlbinlog options, go to [mysqlbinlog Options](#) in the MariaDB documentation.

The following is an example:

For Linux, OS X, or Unix:

```
mysqlbinlog \
--read-from-remote-server \
--host=mariadbinstance1.1234abcd.region.rds.amazonaws.com \
--port=3306 \
--user ReplUser \
--password <password> \
--result-file=/tmp/binlog.txt
```

For Windows:

```
mysqlbinlog ^
--read-from-remote-server ^
--host=mariadbinstance1.1234abcd.region.rds.amazonaws.com ^
--port=3306 ^
--user ReplUser ^
--password <password> ^
--result-file=/tmp/binlog.txt
```

Amazon RDS normally purges a binary log as soon as possible, but the binary log must still be available on the instance to be accessed by mysqlbinlog. To specify the number of hours for RDS to retain binary logs, use the `mysql.rds_set_configuration` stored procedure and specify a period with enough time for you to download the logs. After you set the retention period, monitor storage usage for the DB instance to ensure that the retained binary logs do not take up too much storage.

The following example sets the retention period to 1 day:

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

To display the current setting, use the `mysql.rds_show_configuration` stored procedure:

```
call mysql.rds_show_configuration;
```

Binary Log Annotation

In a MariaDB DB instance, you can use the `Annotate_rows` event to annotate a row event with a copy of the SQL query that caused the row event. This approach provides similar functionality to enabling the `binlog_rows_query_log_events` parameter on a DB instance on MySQL version 5.6 or later.

You can enable binary log annotations globally by creating a custom parameter group and setting the `binlog_annotation_row_events` parameter to 1. You can also enable annotations at the session level, by calling `SET SESSION binlog_annotation_row_events = 1`. Use the `replicate_annotation_row_events` to replicate binary log annotations to the slave instance if binary logging is enabled on it. No special privileges are required to use these settings.

The following is an example of a row-based transaction in MariaDB. The use of row-based logging is triggered by setting the transaction isolation level to read-committed.

```
CREATE DATABASE IF NOT EXISTS test;
USE test;
CREATE TABLE square(x INT PRIMARY KEY, y INT NOT NULL) ENGINE = InnoDB;
```

```
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN
INSERT INTO square(x, y) VALUES(5, 5 * 5);
COMMIT;
```

Without annotations, the binary log entries for the transaction look like the following:

```
BEGIN
/*!*/
# at 1163
# at 1209
#150922 7:55:57 server id 1855786460 end_log_pos 1209      Table_map: `test`.`square`
mapped to number 76
#150922 7:55:57 server id 1855786460 end_log_pos 1247      Write_rows: table id 76
flags: STMT_END_F
### INSERT INTO `test`.`square`
### SET
### @1=5
### @2=25
# at 1247
#150922 7:56:01 server id 1855786460 end_log_pos 1274      Xid = 62
COMMIT/*!*/;
```

The following statement enables session-level annotations for this same transaction, and disables them after committing the transaction:

```
CREATE DATABASE IF NOT EXISTS test;
USE test;
CREATE TABLE square(x INT PRIMARY KEY, y INT NOT NULL) ENGINE = InnoDB;
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET SESSION binlog_annotation_row_events = 1;
BEGIN;
INSERT INTO square(x, y) VALUES(5, 5 * 5);
COMMIT;
SET SESSION binlog_annotation_row_events = 0;
```

With annotations, the binary log entries for the transaction look like the following:

```
BEGIN
/*!*/
# at 423
# at 483
# at 529
#150922 8:04:24 server id 1855786460 end_log_pos 483 Annotate_rows:
#Q> INSERT INTO square(x, y) VALUES(5, 5 * 5)
#150922 8:04:24 server id 1855786460 end_log_pos 529 Table_map: `test`.`square` mapped
to number 76
#150922 8:04:24 server id 1855786460 end_log_pos 567 Write_rows: table id 76 flags:
STMT_END_F
### INSERT INTO `test`.`square`
### SET
### @1=5
### @2=25
# at 567
#150922 8:04:26 server id 1855786460 end_log_pos 594 Xid = 88
COMMIT/*!*/;
```

Microsoft SQL Server Database Log Files

You can access Microsoft SQL Server error logs, agent logs, trace files, and dump files by using the Amazon RDS console or APIs. For more information about viewing, downloading, and watching file-based database logs, see [Amazon RDS Database Log Files \(p. 307\)](#).

Retention Schedule

Log files are rotated each day and whenever your DB instance is restarted. The following is the retention schedule for Microsoft SQL Server logs on Amazon RDS.

Log Type	Retention Schedule
Error logs	A maximum of 30 error logs are retained. Amazon RDS may delete error logs older than 7 days.
Agent logs	A maximum of 10 agent logs are retained. Amazon RDS may delete agent logs older than 7 days.
Trace files	Trace files are retained according to the trace file retention period of your DB instance. The default trace file retention period is 7 days. To modify the trace file retention period for your DB instance, see Setting the Retention Period for Trace and Dump Files (p. 576) .
Dump files	Dump files are retained according to the dump file retention period of your DB instance. The default dump file retention period is 7 days. To modify the dump file retention period for your DB instance, see Setting the Retention Period for Trace and Dump Files (p. 576) .

Viewing the SQL Server Error Log by Using the rds_read_error_log Procedure

You can use the Amazon RDS stored procedure `rds_read_error_log` to view error logs and agent logs. For more information, see [Using the rds_read_error_log Procedure \(p. 575\)](#).

Related Topics

- [Using SQL Server Agent \(p. 574\)](#)
- [Working with Microsoft SQL Server Logs \(p. 575\)](#)
- [Working with Trace and Dump Files \(p. 576\)](#)

MySQL Database Log Files

You can monitor the MySQL error log, slow query log, and the general log. The MySQL error log is generated by default; you can generate the slow query and general logs by setting parameters in your DB parameter group. Amazon RDS rotates all of the MySQL log files; the intervals for each type are given following.

You can monitor the MySQL logs directly through the Amazon RDS console, Amazon RDS API, AWS CLI, or AWS SDKs. You can also access MySQL logs by directing the logs to a database table in the main database and querying that table. You can use the `mysqlbinlog` utility to download a binary log.

For more information about viewing, downloading, and watching file-based database logs, see [Amazon RDS Database Log Files \(p. 307\)](#).

Accessing MySQL Error Logs

The MySQL error log is written to the `mysql-error.log` file. You can view `mysql-error.log` by using the Amazon RDS console or by retrieving the log using the Amazon RDS API, Amazon RDS CLI, or AWS SDKs. `mysql-error.log` is flushed every 5 minutes, and its contents are appended to `mysql-error-running.log`. The `mysql-error-running.log` file is then rotated every hour and the hourly files generated during the last 24 hours are retained. Note that the retention period is different between Amazon RDS and Aurora.

Each log file has the hour it was generated (in UTC) appended to its name. The log files also have a timestamp that helps you determine when the log entries were written.

MySQL writes to the error log only on startup, shutdown, and when it encounters errors. A DB instance can go hours or days without new entries being written to the error log. If you see no recent entries, it's because the server did not encounter an error that would result in a log entry.

Accessing the MySQL Slow Query and General Logs

The MySQL slow query log and the general log can be written to a file or a database table by setting parameters in your DB parameter group. For information about creating and modifying a DB parameter group, see [Working with DB Parameter Groups \(p. 169\)](#). You must set these parameters before you can view the slow query log or general log in the Amazon RDS console or by using the Amazon RDS API, Amazon RDS CLI, or AWS SDKs.

You can control MySQL logging by using the parameters in this list:

- `slow_query_log`: To create the slow query log, set to 1. The default is 0.
- `general_log`: To create the general log, set to 1. The default is 0.
- `long_query_time`: To prevent fast-running queries from being logged in the slow query log, specify a value for the shortest query execution time to be logged, in seconds. The default is 10 seconds; the minimum is 0. If `log_output = FILE`, you can specify a floating point value that goes to microsecond resolution. If `log_output = TABLE`, you must specify an integer value with second resolution. Only queries whose execution time exceeds the `long_query_time` value are logged. For example, setting `long_query_time` to 0.1 prevents any query that runs for less than 100 milliseconds from being logged.
- `log_queries_not_using_indexes`: To log all queries that do not use an index to the slow query log, set to 1. The default is 0. Queries that do not use an index are logged even if their execution time is less than the value of the `long_query_time` parameter.
- `log_output option`: You can specify one of the following options for the `log_output` parameter.
 - **TABLE** (default)– Write general queries to the `mysql.general_log` table, and slow queries to the `mysql.slow_log` table.
 - **FILE**– Write both general and slow query logs to the file system. Log files are rotated hourly.

- **NONE**—Disable logging.

When logging is enabled, Amazon RDS rotates table logs or deletes log files at regular intervals. This measure is a precaution to reduce the possibility of a large log file either blocking database use or affecting performance. **FILE** and **TABLE** logging approach rotation and deletion as follows:

- When **FILE** logging is enabled, log files are examined every hour and log files older than 24 hours are deleted. In some cases, the remaining combined log file size after the deletion might exceed the threshold of 2 percent of a DB instance's allocated space. In these cases, the largest log files are deleted until the log file size no longer exceeds the threshold.
- When **TABLE** logging is enabled, in some cases log tables are rotated every 24 hours. This rotation occurs if the space used by the table logs is more than 20 percent of the allocated storage space or the size of all logs combined is greater than 10 GB. If the amount of space used for a DB instance is greater than 90 percent of the DB instance's allocated storage space, then the thresholds for log rotation are reduced. Log tables are then rotated if the space used by the table logs is more than 10 percent of the allocated storage space or the size of all logs combined is greater than 5 GB. You can subscribe to the `low_free_storage` event to be notified when log tables are rotated to free up space. For more information, see [Using Amazon RDS Event Notification \(p. 287\)](#).

When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If the backup log table already exists, then it is deleted before the current log table is copied to the backup. You can query the backup log table if needed. The backup log table for the `mysql.general_log` table is named `mysql.general_log_backup`. The backup log table for the `mysql.slow_log` table is named `mysql.slow_log_backup`.

You can rotate the `mysql.general_log` table by calling the `mysql.rds_rotate_general_log` procedure. You can rotate the `mysql.slow_log` table by calling the `mysql.rds_rotate_slow_log` procedure.

Table logs are rotated during a database version upgrade.

To work with the logs from the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs, set the `log_output` parameter to **FILE**. Like the MySQL error log, these log files are rotated hourly. The log files that were generated during the previous 24 hours are retained. Note that the retention period is different between Amazon RDS and Aurora.

For more information about the slow query and general logs, go to the following topics in the MySQL documentation:

- [The Slow Query Log](#)
- [The General Query Log](#)

Publishing MySQL Logs to CloudWatch Logs

You can configure your Amazon RDS MySQL DB instance to publish log data to a log group in Amazon CloudWatch Logs. With CloudWatch Logs, you can perform real-time analysis of the log data, and use CloudWatch to create alarms and view metrics. You can use CloudWatch Logs to store your log records in highly durable storage.

Amazon RDS publishes each MySQL database log as a separate database stream in the log group. For example, if you configure the export function to include the slow query log, slow query data is stored in a slow query log stream in the `/aws/rds/instance/my_instance/slowquery` log group.

The error log is enabled by default. The following table summarizes the requirements for the other MySQL logs.

Log	Requirement
Audit log	The DB instance must use a custom option group with the <code>MARIADB_AUDIT_PLUGIN</code> option.
General log	The DB instance must use a custom parameter group with the parameter setting <code>general_log = 1</code> to enable the general log.
Slow query log	The DB instance must use a custom parameter group with the parameter setting <code>slow_query_log = 1</code> to enable the slow query log.
Log output	The DB instance must use a custom parameter group with the parameter setting <code>log_output = FILE</code> to write logs to the file system and publish them to CloudWatch Logs.

Note

Publishing log files to CloudWatch Logs is only supported for MySQL versions 5.6, 5.7, and 8.0.

AWS Management Console

To publish MySQL logs to CloudWatch Logs using the console

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**.
4. In the **Log exports** section, choose the logs that you want to start publishing to CloudWatch Logs.
5. Choose **Continue**, and then choose **Modify DB Instance** on the summary page.

AWS CLI

You can publish MySQL logs with the AWS CLI. You can call the `modify-db-instance` command with the following parameters:

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

Note

A change to the `--cloudwatch-logs-export-configuration` option is always applied to the DB instance immediately. Therefore, the `--apply-immediately` and `--no-apply-immediately` options have no effect.

You can also publish MySQL logs by calling the following AWS CLI commands:

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-from-s3`
- `restore-db-instance-to-point-in-time`

Run one of these AWS CLI commands with the following options:

- `--db-instance-identifier`
- `--enable-cloudwatch-logs-exports`
- `--db-instance-class`
- `--engine`

Other options might be required depending on the AWS CLI command you run.

Example

The following example modifies an existing MySQL DB instance to publish log files to CloudWatch Logs. The `--cloudwatch-logs-export-configuration` value is a JSON object. The key for this object is `EnableLogTypes`, and its value is an array of strings with any combination of `audit`, `error`, `general`, and `slowquery`.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier mydbinstance \
  --cloudwatch-logs-export-configuration '{"EnableLogTypes": \
  ["audit", "error", "general", "slowquery"]}'
```

For Windows:

```
aws rds modify-db-instance ^
  --db-instance-identifier mydbinstance ^
  --cloudwatch-logs-export-configuration '{"EnableLogTypes": \
  ["audit", "error", "general", "slowquery"]}'
```

Example

The following example creates a MySQL DB instance and publishes log files to CloudWatch Logs. The `--enable-cloudwatch-logs-exports` value is a JSON array of strings. The strings can be any combination of `audit`, `error`, `general`, and `slowquery`.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
  --db-instance-identifier mydbinstance \
  --enable-cloudwatch-logs-exports '[{"audit", "error", "general", "slowquery"}]' \
  --db-instance-class db.m4.large \
  --engine MySQL
```

For Windows:

```
aws rds create-db-instance ^
  --db-instance-identifier mydbinstance ^
  --enable-cloudwatch-logs-exports '[{"audit", "error", "general", "slowquery"}]' ^
  --db-instance-class db.m4.large ^
  --engine MySQL
```

RDS API

You can publish MySQL logs with the RDS API. You can call the [ModifyDBInstance](#) action with the following parameters:

- `DBInstanceIdentifier`
- `CloudwatchLogsExportConfiguration`

Note

A change to the `CloudwatchLogsExportConfiguration` parameter is always applied to the DB instance immediately. Therefore, the `ApplyImmediately` parameter has no effect.

You can also publish MySQL logs by calling the following RDS API actions:

- [CreateDBInstance](#)
- [RestoreDBInstanceFromDBSnapshot](#)
- [RestoreDBInstanceFromS3](#)
- [RestoreDBInstanceToPointInTime](#)

Run one of these RDS API actions with the following parameters:

- `DBInstanceIdentifier`
- `EnableCloudwatchLogsExports`
- `Engine`
- `DBInstanceClass`

Other parameters might be required depending on the AWS CLI command you run.

Log File Size

The MySQL slow query log, error log, and the general log file sizes are constrained to no more than 2 percent of the allocated storage space for a DB instance. To maintain this threshold, logs are automatically rotated every hour and log files older than 24 hours are removed. If the combined log file size exceeds the threshold after removing old log files, then the largest log files are deleted until the log file size no longer exceeds the threshold.

For MySQL, there is a size limit on BLOBs written to the redo log. To account for this limit, ensure that the `innodb_log_file_size` parameter for your MySQL DB instance is 10 times larger than the largest BLOB data size found in your tables, plus the length of other variable length fields (`VARCHAR`, `VARBINARY`, `TEXT`) in the same tables. For information on how to set parameter values, see [Working with DB Parameter Groups \(p. 169\)](#). For information on the redo log BLOB size limit, go to [Changes in MySQL 5.6.20](#).

Managing Table-Based MySQL Logs

You can direct the general and slow query logs to tables on the DB instance by creating a DB parameter group and setting the `log_output` server parameter to `TABLE`. General queries are then logged to the `mysql.general_log` table, and slow queries are logged to the `mysql.slow_log` table. You can query the tables to access the log information. Enabling this logging increases the amount of data written to the database, which can degrade performance.

Both the general log and the slow query logs are disabled by default. In order to enable logging to tables, you must also set the `general_log` and `slow_query_log` server parameters to 1.

Log tables keep growing until the respective logging activities are turned off by resetting the appropriate parameter to 0. A large amount of data often accumulates over time, which can use up a considerable percentage of your allocated storage space. Amazon RDS does not allow you to truncate the log tables, but you can move their contents. Rotating a table saves its contents to a backup table and then creates a new empty log table. You can manually rotate the log tables with the following command line procedures, where the command prompt is indicated by **PROMPT>**:

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

To completely remove the old data and reclaim the disk space, call the appropriate procedure twice in succession.

Binary Logging Format

MySQL on Amazon RDS supports the *row-based*, *statement-based*, and *mixed* binary logging formats for MySQL version 5.6 and later. The default binary logging format is mixed. For DB instances running MySQL versions 5.1 and 5.5, only mixed binary logging is supported. For details on the different MySQL binary log formats, see [Binary Logging Formats](#) in the MySQL documentation.

If you plan to use replication, the binary logging format is important because it determines the record of data changes that is recorded in the source and sent to the replication targets. For information about the advantages and disadvantages of different binary logging formats for replication, see [Advantages and Disadvantages of Statement-Based and Row-Based Replication](#) in the MySQL documentation.

Important

Setting the binary logging format to row-based can result in very large binary log files. Large binary log files reduce the amount of storage available for a DB instance and can increase the amount of time to perform a restore operation of a DB instance.

Statement-based replication can cause inconsistencies between the source DB instance and a Read Replica. For more information, see [Determination of Safe and Unsafe Statements in Binary Logging](#) in the MySQL documentation.

To set the MySQL binary logging format

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Parameter groups**.
3. Choose the parameter group used by the DB instance you want to modify.

You can't modify a default parameter group. If the DB instance is using a default parameter group, create a new parameter group and associate it with the DB instance.

For more information on DB parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

4. From **Parameter group actions**, choose **Edit**.
5. Set the `binlog_format` parameter to the binary logging format of your choice (**ROW**, **STATEMENT**, or **MIXED**).
6. Choose **Save changes** to save the updates to the DB parameter group.

Important

Changing the `default.mysql5.6`, `default.mysql5.7`, or `default.mysql8.0` DB parameter group affects all MySQL version DB instances that use that parameter group. If you want to specify different binary logging formats for different MySQL 5.6, 5.7, or 8.0 DB instances in an AWS Region, you need to create your own DB parameter group. This parameter group identifies the different logging format and assigns that DB parameter group to the intended DB instances.

Accessing MySQL Binary Logs

You can use the `mysqlbinlog` utility to download or stream binary logs from Amazon RDS instances running MySQL 5.6 or later. The binary log is downloaded to your local computer, where you can perform actions such as replaying the log using the `mysql` utility. For more information about using the `mysqlbinlog` utility, go to [Using mysqlbinlog to Back Up Binary Log Files](#).

To run the `mysqlbinlog` utility against an Amazon RDS instance, use the following options:

- Specify the `--read-from-remote-server` option.
- `--host`: Specify the DNS name from the endpoint of the instance.
- `--port`: Specify the port used by the instance.
- `--user`: Specify a MySQL user that has been granted the replication slave permission.
- `--password`: Specify the password for the user, or omit a password value so that the utility prompts you for a password.
- To have the file downloaded in binary format, specify the `--raw` option.
- `--result-file`: Specify the local file to receive the raw output.
- Specify the names of one or more binary log files. To get a list of the available logs, use the SQL command `SHOW BINARY LOGS`.
- To stream the binary log files, specify the `--stop-never` option.

For more information about `mysqlbinlog` options, go to [mysqlbinlog - Utility for Processing Binary Log Files](#).

For example, see the following.

For Linux, OS X, or Unix:

```
mysqlbinlog \
--read-from-remote-server \
--host=MySQL56Instance1.cg034hpkmmt.region.rds.amazonaws.com \
--port=3306 \
--user ReplUser \
--password \
--raw \
--result-file=/tmp/ \
binlog.00098
```

For Windows:

```
mysqlbinlog ^
--read-from-remote-server ^
--host=MySQL56Instance1.cg034hpkmmt.region.rds.amazonaws.com ^
--port=3306 ^
--user ReplUser ^
--password ^
--raw ^
--result-file=/tmp/ ^
binlog.00098
```

Amazon RDS normally purges a binary log as soon as possible, but the binary log must still be available on the instance to be accessed by `mysqlbinlog`. To specify the number of hours for RDS to retain binary logs, use the `mysql.rds_set_configuration` stored procedure and specify a period with enough time for you to download the logs. After you set the retention period, monitor storage usage for the DB instance to ensure that the retained binary logs don't take up too much storage.

Note

The `mysql.rds_set_configuration` stored procedure is only available for MySQL version 5.6 or later.

The following example sets the retention period to 1 day.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

To display the current setting, use the `mysql.rds_show_configuration` stored procedure.

```
call mysql.rds_show_configuration;
```

Oracle Database Log Files

You can access Oracle alert logs, audit files, and trace files by using the Amazon RDS console or API. For more information about viewing, downloading, and watching file-based database logs, see [Amazon RDS Database Log Files \(p. 307\)](#).

The Oracle audit files provided are the standard Oracle auditing files. Amazon RDS supports the Oracle fine-grained auditing (FGA) feature. However, log access doesn't provide access to FGA events that are stored in the `SYS.FGA_LOG$` table and that are accessible through the `DBA_FGA_AUDIT_TRAIL` view.

The `DescribeDBLogFiles` API operation that lists the Oracle log files that are available for a DB instance ignores the `MaxRecords` parameter and returns up to 1,000 records.

Retention Schedule

The Oracle database engine might rotate logs files if they get very large. To retain audit or trace files, download them. Storing the files locally reduces your Amazon RDS storage costs and makes more space available for your data.

The following is the retention schedule for Oracle alert logs, audit files, and trace files on Amazon RDS.

Log Type	Retention Schedule
Alert logs	The text alert log is rotated daily with 30-day retention managed by Amazon RDS. The XML alert log is retained for at least seven days. You can access this log by using the <code>ALERTLOG</code> view.
Audit files	The default retention period for audit files is seven days. Amazon RDS might delete audit files older than seven days.
Trace files	The default retention period for trace files is seven days. Amazon RDS might delete trace files older than seven days.
Listener logs	The default retention period for the listener logs is seven days. Amazon RDS might delete listener logs older than seven days.

Note

Audit files and trace files share the same retention configuration.

Switching Online Log files

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.switch_logfile` to switch online log files. For more information, see [Switching Online Log Files \(p. 868\)](#).

Retrieving Archived Redo Logs

You can retain archived redo logs. For more information, see [Retaining Archived Redo Logs \(p. 871\)](#).

Working with Oracle Trace Files

Following, you can find descriptions of Amazon RDS procedures to create, refresh, access, and delete trace files.

[Listing Files](#)

You can use either of two procedures to allow access to any file in the `background_dump_dest` path. The first procedure refreshes a view containing a listing of all files currently in `background_dump_dest`.

```
exec rdsadmin.manage_tracefiles.refresh_tracefile_listing;
```

After the view is refreshed, use the following view to access the results.

```
rdsadmin.tracefile_listing
```

An alternative to the previous process is to use `FROM table` to stream nontable data in a table-like format to list database directory contents.

```
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP'));
```

The following query shows the text of a log file.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','alert_xxx.log'));
```

Generating Trace Files and Tracing a Session

Because there are no restrictions on `alter session`, many standard methods to generate trace files in Oracle remain available to an Amazon RDS DB instance. The following procedures are provided for trace files that require greater access.

Oracle Method	Amazon RDS Method
<code>oradebug hanganalyze 3</code>	<code>exec rdsadmin.manage_tracefiles.hanganalyze;</code>
<code>oradebug dump systemstate 266</code>	<code>exec rdsadmin.manage_tracefiles.dump_systemstate;</code>

You can use many standard methods to trace individual sessions connected to an Oracle DB instance in Amazon RDS. To enable tracing for a session, you can run subprograms in PL/SQL packages supplied by Oracle, such as the `DBMS_SESSION` and `DBMS_MONITOR` packages. For more information, see [Enabling Tracing for a Session](#) in the Oracle documentation.

Retrieving Trace Files

You can retrieve any trace file in `background_dump_dest` using a standard SQL query on an Amazon RDS-managed external table. To use this method, you must execute the procedure to set the location for this table to the specific trace file.

For example, you can use the `rdsadmin.tracefile_listing` view mentioned preceding to list all of the trace files on the system. You can then set the `tracefile_table` view to point to the intended trace file using the following procedure.

```
exec
rdsadmin.manage_tracefiles.set_tracefile_table_location('CUST01_ora_3260_SYSTEMSTATE.trc');
```

The following example creates an external table in the current schema with the location set to the file provided. You can retrieve the contents into a local file using a SQL query.

```
# eg: send the contents of the tracefile to a local file:
```

```
sqlplus user/password@TNS alias << EOF > /tmp/tracefile.txt
select * from tracefile_table;
EOF
```

Purging Trace Files

Trace files can accumulate and consume disk space. Amazon RDS purges trace files by default and log files that are older than seven days. You can view and set the trace file retention period using the `show_configuration` procedure. You should run the command `SET SERVEROUTPUT ON` so that you can view the configuration results.

The following example shows the current trace file retention period, and then sets a new trace file retention period.

```
# Show the current tracefile retention
SQL> exec rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:10080
DESCRIPTION:tracefile expiration specifies the duration in minutes before tracefiles in
bdbump are automatically deleted.

# Set the tracefile retention to 24 hours:
SQL> exec rdsadmin.rdsadmin_util.set_configuration('tracefile retention',1440);

#show the new tracefile retention
SQL> exec rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:1440
DESCRIPTION:tracefile expiration specifies the duration in minutes before tracefiles in
bdbump are automatically deleted.
```

In addition to the periodic purge process, you can manually remove files from the `background_dump_dest`. The following example shows how to purge all files older than five minutes.

```
exec rdsadmin.manage_tracefiles.purge_tracefiles(5);
```

You can also purge all files that match a specific pattern (if you do, don't include the file extension, such as .trc). The following example shows how to purge all files that start with `SCHPOC1_ora_5935`.

```
exec rdsadmin.manage_tracefiles.purge_tracefiles('SCHPOC1_ora_5935');
```

Publishing Oracle Logs to Amazon CloudWatch Logs

You can configure your Amazon RDS Oracle DB instance to publish log data to a log group in Amazon CloudWatch Logs. With CloudWatch Logs, you can analyze the log data, and use CloudWatch to create alarms and view metrics. You can use CloudWatch Logs to store your log records in highly durable storage.

Amazon RDS publishes each Oracle database log as a separate database stream in the log group. For example, if you configure the export function to include the audit log, audit data is stored in an audit log stream in the `/aws/rds/instance/my_instance/audit` log group.

Console

To publish Oracle DB logs to CloudWatch Logs from the AWS Management Console

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.

2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**.
4. In the **Log exports** section, choose the logs that you want to start publishing to CloudWatch Logs.
5. Choose **Continue**, and then choose **Modify DB Instance** on the summary page.

AWS CLI

To publish Oracle logs, you can use the [modify-db-instance](#) command with the following parameters:

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

Note

A change to the `--cloudwatch-logs-export-configuration` option is always applied to the DB instance immediately. Therefore, the `--apply-immediately` and `--no-apply-immediately` options have no effect.

You can also publish Oracle logs using the following commands:

- [create-db-instance](#)
- [restore-db-instance-from-db-snapshot](#)
- [restore-db-instance-from-s3](#)
- [restore-db-instance-to-point-in-time](#)

Example

The following example creates an Oracle DB instance with CloudWatch Logs publishing enabled. The `--enable-cloudwatch-logs-exports` value is a JSON array of strings. The strings can be any combination of `alert`, `audit`, `listener`, and `trace`.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
  --db-instance-identifier mydbinstance \
  --enable-cloudwatch-logs-exports '["trace","audit","alert","listener"]' \
  --db-instance-class db.m1.small \
  --engine oracle-se1
```

For Windows:

```
aws rds create-db-instance ^
  --db-instance-identifier mydbinstance ^
  --enable-cloudwatch-logs-exports '["trace","audit","alert","listener"]' ^
  --db-instance-class db.m1.small ^
  --engine oracle-se1
```

Example

The following example modifies an existing Oracle DB instance to publish log files to CloudWatch Logs. The `--cloudwatch-logs-export-configuration` value is a JSON object. The key for this

object is `EnableLogTypes`, and its value is an array of strings with any combination of `alert`, `audit`, `listener`, and `trace`.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--cloudwatch-logs-export-configuration '{"EnableLogTypes": \
["trace", "alert", "audit", "listener"]}'
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--cloudwatch-logs-export-configuration '{"EnableLogTypes": \
["trace", "alert", "audit", "listener"]}'
```

Example

The following example modifies an existing Oracle DB instance to disable publishing audit and listener log files to CloudWatch Logs. The `--cloudwatch-logs-export-configuration` value is a JSON object. The key for this object is `DisableLogTypes`, and its value is an array of strings with any combination of `alert`, `audit`, `listener`, and `trace`.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--cloudwatch-logs-export-configuration '{"DisableLogTypes": ["audit", "listener"]}'
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--cloudwatch-logs-export-configuration '{"DisableLogTypes": ["audit", "listener"]}'
```

RDS API

You can publish Oracle DB logs with the RDS API. You can call the [ModifyDBInstance](#) action with the following parameters:

- `DBInstanceIdentifier`
- `CloudwatchLogsExportConfiguration`

Note

A change to the `CloudwatchLogsExportConfiguration` parameter is always applied to the DB instance immediately. Therefore, the `ApplyImmediately` parameter has no effect.

You can also publish Oracle logs by calling the following RDS API operations:

- [CreateDBInstance](#)
- [RestoreDBInstanceFromDBSnapshot](#)

- [RestoreDBInstanceFromS3](#)
- [RestoreDBInstanceToPointInTime](#)

Run one of these RDS API actions with the following parameters:

- [DBInstanceIdentifier](#)
- [EnableCloudwatchLogsExports](#)
- [Engine](#)
- [DBInstanceClass](#)

Other parameters might be required depending on the RDS operation that you run.

Previous Methods for Accessing Alert Logs and Listener Logs

You can view the alert log using the Amazon RDS console. You can also use the following SQL statement to access the alert log.

```
select message_text from alertlog;
```

To access the listener log, use the following SQL statement.

```
select message_text from listenerlog;
```

Note

Oracle rotates the alert and listener logs when they exceed 10 MB, at which point they are unavailable from Amazon RDS views.

PostgreSQL Database Log Files

RDS PostgreSQL generates query and error logs. We write auto-vacuum information and rds_admin actions to the error log. PostgreSQL also logs connections, disconnections, and checkpoints to the error log. For more information, see the [Error Reporting and Logging](#) in the PostgreSQL documentation.

You can set the retention period for system logs using the `rds.log_retention_period` parameter in the DB parameter group associated with your DB instance. The unit for this parameter is minutes. For example, a setting of 1440 would retain logs for one day. The default value is 4320 (three days). The maximum value is 10080 (seven days). Note that your instance must have enough allocated storage to contain the retained log files.

You can enable query logging for your PostgreSQL DB instance by setting two parameters in the DB parameter group associated with your DB instance: `log_statement` and `log_min_duration_statement`. The `log_statement` parameter controls which SQL statements are logged. We recommend setting this parameter to `all` to log all statements when debugging issues in your DB instance. The default value is `none`. Alternatively, you can set this value to `ddl` to log all data definition language (DDL) statements (CREATE, ALTER, DROP, and so on) or to `mod` to log all DDL and data modification language (DML) statements (INSERT, UPDATE, DELETE, and so on).

The `log_min_duration_statement` parameter sets the limit in milliseconds of a statement to be logged. All SQL statements that run longer than the parameter setting are logged. This parameter is disabled and set to minus 1 (-1) by default. Enabling this parameter can help you find unoptimized queries.

If you are new to setting parameters in a DB parameter group and associating that parameter group with a DB instance, see [Working with DB Parameter Groups \(p. 169\)](#)

The following steps show how to set up query logging:

1. Set the `log_statement` parameter to `all`. The following example shows the information that is written to the `postgres.log` file:

```
2013-11-05 16:48:56 UTC::@[2952]:LOG: received SIGHUP, reloading configuration files
2013-11-05 16:48:56 UTC::@[2952]:LOG: parameter "log_min_duration_statement" changed to
"1"
```

Additional information is written to the `postgres.log` file when you execute a query. The following example shows the type of information written to the file after a query:

```
2013-11-05 16:41:07 UTC::@[2955]:LOG: checkpoint starting: time
2013-11-05 16:41:07 UTC::@[2955]:LOG: checkpoint complete: wrote 1 buffers (0.3%),
0 transaction log file(s) added, 0 removed, 1 recycled; write=0.000 s, sync=0.003 s,
total=0.012 s; sync files=1, longest=0.003 s, average=0.003 s
2013-11-05 16:45:14 UTC:[local]:master@postgres:[8839]:LOG: statement: SELECT d.datname
as "Name",
    pg_catalog.pg_get_userbyid(d.datdba) as "Owner",
    pg_catalog.pg_encoding_to_char(d.encoding) as "Encoding",
    d.datcollate as "Collate",
    d.datctype as "Ctype",
    pg_catalog.array_to_string(d.datacl, E'\n') AS "Access privileges"
FROM pg_catalog.pg_database d
ORDER BY 1;
2013-11-05 16:45:
```

2. Set the `log_min_duration_statement` parameter. The following example shows the information that is written to the `postgres.log` file when the parameter is set to 1:

```
2013-11-05 16:48:56 UTC::@[2952]:LOG: received SIGHUP, reloading configuration files
```

```
2013-11-05 16:48:56 UTC:[@:[2952]:LOG: parameter "log_min_duration_statement" changed to "1"
```

Additional information is written to the `postgres.log` file when you execute a query that exceeds the duration parameter setting. The following example shows the type of information written to the file after a query:

```
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG: statement: SELECT
c2.relname, i.indisprimary, i.indisunique, i.indisclustered, i.indisvalid,
pg_catalog.pg_get_indexdef(i.indexrelid, 0, true),
pg_catalog.pg_get_constraintdef(con.oid, true), contype, condeferrable, condeferred,
c2.reltablename
FROM pg_catalog.pg_class c, pg_catalog.pg_class c2, pg_catalog.pg_index i
LEFT JOIN pg_catalog.pg_constraint con ON (conrelid = i.indrelid AND conindid = i.indexrelid AND contype IN ('p','u','x'))
WHERE c.oid = '1255' AND c.oid = i.indrelid AND i.indexrelid = c2.oid
ORDER BY i.indisprimary DESC, i.indisunique DESC, c2.relname;
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG: duration: 3.367 ms
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG: statement: SELECT
c.oid::pg_catalog.regclass FROM pg_catalog.pg_class c, pg_catalog.pg_inherits i WHERE
c.oid=i.inhparent AND i.inhrelid = '1255' ORDER BY inhseqno;
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG: duration: 1.002 ms
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG: statement:
SELECT c.oid::pg_catalog.regclass FROM pg_catalog.pg_class c,
pg_catalog.pg_inherits i WHERE c.oid=i.inhrelid AND i.inhparent = '1255' ORDER BY
c.oid::pg_catalog.regclass::pg_catalog.text;
2013-11-05 16:51:18 UTC:[local]:master@postgres:[9193]:LOG: statement: select proname
from pg_proc;
2013-11-05 16:51:18 UTC:[local]:master@postgres:[9193]:LOG: duration: 3.469 ms
```

Publishing PostgreSQL Logs to CloudWatch Logs

You can configure your Amazon RDS for PostgreSQL DB instance to publish log data to a log group in Amazon CloudWatch Logs. With CloudWatch Logs, you can perform real-time analysis of the log data, and use CloudWatch to create alarms and view metrics. You can use CloudWatch Logs to store your log records in highly durable storage.

Note

Publishing log files to CloudWatch Logs is only supported for PostgreSQL versions 9.6.6 and above and 10.4 and above.

Following are the log types that can be published to CloudWatch Logs for Amazon RDS for PostgreSQL.

- Postgresql log
- Upgrade log

After you complete the configuration, Amazon RDS publishes the log events to log streams within a CloudWatch log group. For example, the PostgreSQL log data is stored within the log group `/aws/rds/instance/my_instance/postgresql`. To view your Amazon CloudWatch Logs, open <https://console.aws.amazon.com/cloudwatch/>.

AWS Management Console

To publish PostgreSQL logs to CloudWatch Logs using the console

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Select the DB instance that you want to modify, and then choose **Modify**.

4. In the **Log exports** section, choose the logs you want to start publishing to CloudWatch Logs.
5. Choose **Continue**, and then choose **Modify DB Instance** on the summary page.

AWS CLI

You can publish PostgreSQL logs with the AWS CLI. You can call the [modify-db-instance](#) command with the following parameters:

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

Note

A change to the `--cloudwatch-logs-export-configuration` option is always applied to the DB instance immediately. Therefore, the `--apply-immediately` and `--no-apply-immediately` options have no effect.

You can also publish PostgreSQL logs by calling the following AWS CLI commands:

- [create-db-instance](#)
- [restore-db-instance-from-db-snapshot](#)
- [restore-db-instance-to-point-in-time](#)

Run one of these AWS CLI commands with the following options:

- `--db-instance-identifier`
- `--enable-cloudwatch-logs-exports`
- `--db-instance-class`
- `--engine`

Other options might be required depending on the AWS CLI command you run.

Modify an instance to publish logs to CloudWatch Logs

Example

The following example modifies an existing PostgreSQL DB instance to publish log files to CloudWatch Logs. The `--cloudwatch-logs-export-configuration` value is a JSON object. The key for this object is `EnableLogTypes`, and its value is an array of strings with any combination of `postgresql` and `upgrade`.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--cloudwatch-logs-export-configuration '{"EnableLogTypes": ["postgresql", "upgrade"]}'
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--cloudwatch-logs-export-configuration '{"EnableLogTypes": ["postgresql", "upgrade"]}'
```

Example

Create an instance to publish logs to CloudWatch Logs

The following example creates a PostgreSQL DB instance and publishes log files to CloudWatch Logs. The `--enable-cloudwatch-logs-exports` value is a JSON array of strings. The strings can be any combination of `postgresql` and `upgrade`.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '[ "postgresql", "upgrade" ]' \
    --db-instance-class db.m4.large \
    --engine postgres
```

For Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports '[ "postgresql", "upgrade" ]' ^
    --db-instance-class db.m4.large ^
    --engine postgres
```

RDS API

You can publish PostgreSQL logs with the RDS API. You can call the [ModifyDBInstance](#) action with the following parameters:

- `DBInstanceIdentifier`
- `CloudwatchLogsExportConfiguration`

Note

A change to the `CloudwatchLogsExportConfiguration` parameter is always applied to the DB instance immediately. Therefore, the `ApplyImmediately` parameter has no effect.

You can also publish PostgreSQL logs by calling the following RDS API actions:

- [CreateDBInstance](#)
- [RestoreDBInstanceFromDBSnapshot](#)
- [RestoreDBInstanceToPointInTime](#)

Run one of these RDS API actions with the following parameters:

- `DBInstanceIdentifier`
- `EnableCloudwatchLogsExports`
- `Engine`
- `DBInstanceClass`

Other parameters might be required depending on the action you run.

Logging Amazon RDS API Calls with AWS CloudTrail

Amazon RDS is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon RDS. CloudTrail captures all API calls for Amazon RDS as events, including calls from the Amazon RDS console and from code calls to the Amazon RDS APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon RDS. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon RDS, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

Amazon RDS Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon RDS, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing Events with CloudTrail Event History](#).

For an ongoing record of events in your AWS account, including events for Amazon RDS, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all regions. The trail logs events from all regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#)

All Amazon RDS actions are logged by CloudTrail and are documented in the [Amazon RDS API Reference](#). For example, calls to the `CreateDBInstance`, `ModifyDBInstance`, and `CreateDBParameterGroup` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity Element](#).

Understanding Amazon RDS Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request

parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateDBInstance` action.

```
{  
    "eventVersion": "1.04",  
    "userIdentity": {  
        "type": "IAMUser",  
        "principalId": "AKIAIOSFODNN7EXAMPLE",  
        "arn": "arn:aws:iam::123456789012:user/johndoe",  
        "accountId": "123456789012",  
        "accessKeyId": "AKIAI44QH8DHBEEXAMPLE",  
        "userName": "johndoe"  
    },  
    "eventTime": "2018-07-30T22:14:06Z",  
    "eventSource": "rds.amazonaws.com",  
    "eventName": "CreateDBInstance",  
    "awsRegion": "us-east-1",  
    "sourceIPAddress": "72.21.198.65",  
    "userAgent": "aws-cli/1.15.42 Python/3.6.1 Darwin/17.7.0 botocore/1.10.42",  
    "requestParameters": {  
        "enableCloudwatchLogsExports": [  
            "audit",  
            "error",  
            "general",  
            "slowquery"  
        ],  
        "dBInstanceIdentifier": "test-instance",  
        "engine": "mysql",  
        "masterUsername": "myawsuser",  
        "allocatedStorage": 20,  
        "dBInstanceClass": "db.m1.small",  
        "masterUserPassword": "*****"  
    },  
    "responseElements": {  
        "dBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance",  
        "storageEncrypted": false,  
        "preferredBackupWindow": "10:27-10:57",  
        "preferredMaintenanceWindow": "sat:05:47-sat:06:17",  
        "backupRetentionPeriod": 1,  
        "allocatedStorage": 20,  
        "storageType": "standard",  
        "engineVersion": "5.6.39",  
        "dBInstancePort": 0,  
        "optionGroupMemberships": [  
            {  
                "status": "in-sync",  
                "optionGroupName": "default:mysql-5-6"  
            }  
        ],  
        "dBParameterGroups": [  
            {  
                "dBParameterGroupName": "default.mysql5.6",  
                "parameterApplyStatus": "in-sync"  
            }  
        ],  
        "monitoringInterval": 0,  
        "dBInstanceClass": "db.m1.small",  
        "readReplicaDBInstanceIdentifiers": [],  
        "dBSubnetGroup": {  
            "dBSubnetGroupName": "default",  
            "dBSubnetGroupDescription": "default",  
            "subnets": [  
            ]  
        }  
    }  
}
```

```
{  
    "subnetAvailabilityZone": {"name": "us-east-1b"},  
    "subnetIdentifier": "subnet-cbfff283",  
    "subnetStatus": "Active"  
},  
{  
    "subnetAvailabilityZone": {"name": "us-east-1e"},  
    "subnetIdentifier": "subnet-d7c825e8",  
    "subnetStatus": "Active"  
},  
{  
    "subnetAvailabilityZone": {"name": "us-east-1f"},  
    "subnetIdentifier": "subnet-6746046b",  
    "subnetStatus": "Active"  
},  
{  
    "subnetAvailabilityZone": {"name": "us-east-1c"},  
    "subnetIdentifier": "subnet-bac383e0",  
    "subnetStatus": "Active"  
},  
{  
    "subnetAvailabilityZone": {"name": "us-east-1d"},  
    "subnetIdentifier": "subnet-42599426",  
    "subnetStatus": "Active"  
},  
{  
    "subnetAvailabilityZone": {"name": "us-east-1a"},  
    "subnetIdentifier": "subnet-da327bf6",  
    "subnetStatus": "Active"  
}  
],  
"vpcId": "vpc-136a4c6a",  
"subnetGroupStatus": "Complete"  
},  
"masterUsername": "myawsuser",  
"multiAZ": false,  
"autoMinorVersionUpgrade": true,  
"engine": "mysql",  
"caCertificateIdentifier": "rds-ca-2015",  
"dbiResourceId": "db-ETDZIIXHEWY5N7GXVC4SH7H5IA",  
"dBSecurityGroups": [],  
"pendingModifiedValues": {  
    "masterUserPassword": "*****",  
    "pendingCloudwatchLogsExports": {  
        "logTypesToEnable": [  
            "audit",  
            "error",  
            "general",  
            "slowquery"  
        ]  
    }  
},  
"dBInstanceStatus": "creating",  
"publiclyAccessible": true,  
"domainMemberships": [],  
"copyTagsToSnapshot": false,  
"dBInstanceIdentifier": "test-instance",  
"licenseModel": "general-public-license",  
"iAMDatabaseAuthenticationEnabled": false,  
"performanceInsightsEnabled": false,  
"vpcSecurityGroups": [  
    {  
        "status": "active",  
        "vpcSecurityGroupId": "sg-f839b688"  
    }  
]
```

```
},
"requestID": "daf2e3f5-96a3-4df7-a026-863f96db793e",
"eventID": "797163d3-5726-441d-80a7-6eeb7464acd4",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}
```

Configuring Security in Amazon RDS

You can manage access to your Amazon RDS resources and your databases on a DB instance. The method you use to manage access depends on what type of task the user needs to perform with Amazon RDS:

- Run your DB instance in an Amazon Virtual Private Cloud (VPC) for the greatest possible network access control. For more information about creating a DB instance in a VPC, see [Using Amazon RDS with Amazon Virtual Private Cloud \(VPC\)](#).
- Use AWS Identity and Access Management (IAM) policies to assign permissions that determine who is allowed to manage RDS resources. For example, you can use IAM to determine who is allowed to create, describe, modify, and delete DB instances, tag resources, or modify security groups.
- Use security groups to control what IP addresses or Amazon EC2 instances can connect to your databases on a DB instance. When you first create a DB instance, its firewall prevents any database access except through rules specified by an associated security group.
- Use Secure Socket Layer (SSL) connections with DB instances running the MySQL, MariaDB, PostgreSQL, Oracle, or Microsoft SQL Server database engines. For more information on using SSL with a DB instance, see [Using SSL to Encrypt a Connection to a DB Instance \(p. 392\)](#).
- Use RDS encryption to secure your RDS instances and snapshots at rest. RDS encryption uses the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your RDS instance. For more information, see [Encrypting Amazon RDS Resources \(p. 389\)](#).
- Use network encryption and transparent data encryption with Oracle DB instances; for more information, see [Oracle Native Network Encryption \(p. 815\)](#) and [Oracle Transparent Data Encryption \(p. 836\)](#)
- Use the security features of your DB engine to control who can log in to the databases on a DB instance, just as you do if the database was on your local network.

Note

You only have to configure security for your use cases. You don't have to configure security access for processes that Amazon RDS manages, such as creating backups, replicating data between a master and a Read Replica, or other processes.

For more information on managing access to Amazon RDS resources and your databases on a DB instance, see the following topics.

Topics

- [Authentication and Access Control \(p. 342\)](#)
- [Encrypting Amazon RDS Resources \(p. 389\)](#)
- [Using SSL to Encrypt a Connection to a DB Instance \(p. 392\)](#)
- [Controlling Access with Security Groups \(p. 394\)](#)
- [Master User Account Privileges \(p. 407\)](#)
- [Using Service-Linked Roles for Amazon RDS \(p. 409\)](#)
- [Amazon Virtual Private Cloud \(VP Cs\) and Amazon RDS \(p. 412\)](#)

Authentication and Access Control

Access to Amazon RDS requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an Amazon RDS DB instance. The following sections provide details on how you can use [AWS Identity and Access Management \(IAM\)](#) and Amazon RDS to help secure your resources by controlling who can access them:

- [Authentication \(p. 343\)](#)
- [Access Control \(p. 343\)](#)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.
- **IAM user** – An [IAM user](#) is an identity within your AWS account that has specific custom permissions (for example, permissions to create a DB instance in Amazon RDS). You can use an IAM user name and password to sign in to secure AWS webpages like the [AWS Management Console](#), [AWS Discussion Forums](#), or the [AWS Support Center](#).

In addition to a user name and password, you can also generate [access keys](#) for each user. You can use these keys when you access AWS services programmatically, either through [one of the several SDKs](#) or by using the [AWS Command Line Interface \(CLI\)](#). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. Amazon RDS supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 Signing Process](#) in the [AWS General Reference](#).

- **IAM role** – An [IAM role](#) is an IAM identity that you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
 - **Federated user access** – Instead of creating an IAM user, you can use existing user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an [identity provider](#). For more information about federated users, see [Federated Users and Roles](#) in the *IAM User Guide*.
 - **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.
 - **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances](#) in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Amazon RDS resources. For example, you must have permissions to create an Amazon RDS DB instance, create a DB snapshot, add an event subscription, and so on.

The following sections describe how to manage permissions for Amazon RDS. We recommend that you read the overview first.

- [Overview of Managing Access Permissions to Your Amazon RDS Resources \(p. 344\)](#)
- [Using Identity-Based Policies \(IAM Policies\) for Amazon RDS \(p. 347\)](#)

Overview of Managing Access Permissions to Your Amazon RDS Resources

Every AWS resource is owned by an AWS account, and permissions to create or access the resources are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

Note

An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

- [Amazon RDS Resources and Operations \(p. 344\)](#)
- [Understanding Resource Ownership \(p. 345\)](#)
- [Managing Access to Resources \(p. 345\)](#)
- [Specifying Policy Elements: Actions, Effects, Resources, and Principals \(p. 347\)](#)
- [Specifying Conditions in a Policy \(p. 347\)](#)

Amazon RDS Resources and Operations

In Amazon RDS, the primary resource is a *DB instance*. Amazon RDS supports other resources that can be used with the primary resource such as *DB snapshots*, *parameter groups*, and *event subscriptions*. These are referred to as *subresources*.

These resources and subresources have unique Amazon Resource Names (ARNs) associated with them as shown in the following table.

Resource Type	ARN Format
DB cluster	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:cluster:<i>db-cluster-name</i></code>
DB cluster parameter group	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:cluster-pg:<i>cluster-parameter-group-name</i></code>
DB cluster snapshot	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:cluster-snapshot:<i>cluster-snapshot-name</i></code>
DB instance	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:db:<i>db-instance-name</i></code>
DB option group	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:og:<i>option-group-name</i></code>
DB parameter group	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:pg:<i>parameter-group-name</i></code>
DB snapshot	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:snapshot:<i>snapshot-name</i></code>

Resource Type	ARN Format
DB security group	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:secgrp:<i>security-group-name</i></code>
DB subnet group	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:subgrp:<i>subnet-group-name</i></code>
Event subscription	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:es:<i>subscription-name</i></code>
Read Replica	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:db:<i>db-instance-name</i></code>
Reserved DB instance	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:ri:<i>reserved-db-instance-name</i></code>

Amazon RDS provides a set of operations to work with the Amazon RDS resources. For a list of available operations, see [Actions](#).

Understanding Resource Ownership

A *resource owner* is the AWS account that created a resource. That is, the resource owner is the AWS account of the *principal entity* (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create an RDS resource, such as a DB instance, your AWS account is the owner of the RDS resource.
- If you create an IAM user in your AWS account and grant permissions to create RDS resources to that user, the user can create RDS resources. However, your AWS account, to which the user belongs, owns the RDS resources.
- If you create an IAM role in your AWS account with permissions to create RDS resources, anyone who can assume the role can create RDS resources. Your AWS account, to which the role belongs, owns the RDS resources.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

Note

This section discusses using IAM in the context of Amazon RDS. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What Is IAM?](#) in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM policies) and policies attached to a resource are referred to as *resource-based* policies. Amazon RDS supports only identity-based policies (IAM policies).

Topics

- [Identity-Based Policies \(IAM Policies\) \(p. 345\)](#)
- [Resource-Based Policies \(p. 346\)](#)

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – An account administrator can use a permissions policy that is associated with a particular user to grant permissions for that user to create an Amazon RDS resource, such as a DB instance.

- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:
 1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
 2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
 3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

For more information about using IAM to delegate permissions, see [Access Management](#) in the *IAM User Guide*.

The following is an example policy that allows the user with the ID 123456789012 to create DB instances for your AWS account. The policy requires that the name of the new DB instance begin with test. The new DB instance must also use the MySQL database engine and the db.t2.micro DB instance class. In addition, the new DB instance must use an option group and a DB parameter group that starts with default, and it must use the default subnet group.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowCreateDBInstanceOnly",  
            "Effect": "Allow",  
            "Action": [  
                "rds:CreateDBInstance"  
            ],  
            "Resource": [  
                "arn:aws:rds:*:123456789012:db:test*",  
                "arn:aws:rds:*:123456789012:og:default*",  
                "arn:aws:rds:*:123456789012:pg:default*",  
                "arn:aws:rds:*:123456789012:subgrp:default"  
            ],  
            "Condition": {  
                "StringEquals": {  
                    "rds:DatabaseEngine": "mysql",  
                    "rds:DatabaseClass": "db.t2.micro"  
                }  
            }  
        }  
    ]  
}
```

For more information about using identity-based policies with Amazon RDS, see [Using Identity-Based Policies \(IAM Policies\) for Amazon RDS \(p. 347\)](#). For more information about users, groups, roles, and permissions, see [Identities \(Users, Groups, and Roles\)](#) in the *IAM User Guide*.

Resource-Based Policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. Amazon RDS doesn't support resource-based policies.

Specifying Policy Elements: Actions, Effects, Resources, and Principals

For each Amazon RDS resource (see [Amazon RDS Resources and Operations \(p. 344\)](#)), the service defines a set of API operations (see [Actions](#)). To grant permissions for these API operations, Amazon RDS defines a set of actions that you can specify in a policy. Performing an API operation can require permissions for more than one action.

The following are the basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For more information, see [Amazon RDS Resources and Operations \(p. 344\)](#).
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, the `rds:DescribeDBInstances` permission allows the user permissions to perform the Amazon RDS `DescribeDBInstances` operation.
- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). Amazon RDS doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

For a table showing all of the Amazon RDS API actions and the resources that they apply to, see [Amazon RDS API Permissions: Actions, Resources, and Conditions Reference \(p. 351\)](#).

You can test IAM policies with the IAM policy simulator. It automatically provides a list of resources and parameters required for each AWS action, including Amazon RDS actions. The IAM policy simulator determines the permissions required for each of the actions that you specify. For information about the IAM policy simulator, see [Testing IAM Policies with the IAM Policy Simulator](#) in the *IAM User Guide*.

Specifying Conditions in a Policy

When you grant permissions, you can use the access policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see [Condition](#) in the *IAM User Guide*.

To express conditions, you use predefined condition keys. There are AWS-wide condition keys and RDS-specific keys that you can use as appropriate. For a complete list of AWS-wide keys, see [Available Keys for Conditions](#) in the *IAM User Guide*. For a complete list of RDS-specific keys, see [Using IAM Policy Conditions for Fine-Grained Access Control \(p. 367\)](#).

Using Identity-Based Policies (IAM Policies) for Amazon RDS

This topic provides examples of identity-based policies in which an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

Important

We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your Amazon RDS resources. For

more information, see [Overview of Managing Access Permissions to Your Amazon RDS Resources \(p. 344\)](#).

The sections in this topic cover the following:

- [Permissions Required to Use the Amazon RDS Console \(p. 349\)](#)
- [AWS Managed \(Predefined\) Policies for Amazon RDS \(p. 349\)](#)
- [Customer Managed Policy Examples \(p. 349\)](#)

The following is an example of an IAM policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowCreateDBInstanceOnly",  
            "Effect": "Allow",  
            "Action": [  
                "rds:CreateDBInstance"  
            ],  
            "Resource": [  
                "arn:aws:rds:*:123456789012:db:test*",  
                "arn:aws:rds:*:123456789012:og:default*",  
                "arn:aws:rds:*:123456789012:pg:default*",  
                "arn:aws:rds:*:123456789012:subgrp:default"  
            ],  
            "Condition": {  
                "StringEquals": {  
                    "rds:DatabaseEngine": "mysql",  
                    "rds:DatabaseClass": "db.t2.micro"  
                }  
            }  
        }  
    ]  
}
```

The policy includes a single statement that specifies the following permissions for the IAM user:

- The policy allows the IAM user to create a DB instance using the [CreateDBInstance](#) API action (this also applies to the [create-db-instance](#) AWS CLI command and the AWS Management Console).
- The **Resource** element specifies that the user can perform actions on or with resources. You specify resources using an Amazon Resources Name (ARN). This ARN includes the name of the service that the resource belongs to (`rds`), the AWS Region (* indicates any region in this example), the user account number (123456789012 is the user ID in this example), and the type of resource. For more information about creating ARNs, see [Working with Amazon Resource Names \(ARNs\) in Amazon RDS \(p. 181\)](#).

The **Resource** element in the example specifies the following policy constraints on resources for the user:

- The DB instance identifier for the new DB instance must begin with `test` (for example, `testCustomerData1`, `test-region2-data`).
- The option group for the new DB instance must begin with `default`.
- The DB parameter group for the new DB instance must begin with `default`.
- The subnet group for the new DB instance must be the `default` subnet group.
- The **Condition** element specifies that the DB engine must be MySQL and the DB instance class must be `db.t2.micro`. The **Condition** element specifies the conditions when a policy should take effect. You can add additional permissions or restrictions by using the **Condition** element. For more

information about specifying conditions, see [Using IAM Policy Conditions for Fine-Grained Access Control \(p. 367\)](#).

The policy doesn't specify the `Principal` element because in an identity-based policy you don't specify the principal who gets the permission. When you attach policy to a user, the user is the implicit principal. When you attach a permission policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the Amazon RDS API actions and the resources that they apply to, see [Amazon RDS API Permissions: Actions, Resources, and Conditions Reference \(p. 351\)](#).

Permissions Required to Use the Amazon RDS Console

For a user to work with the Amazon RDS console, that user must have a minimum set of permissions. These permissions allow the user to describe the Amazon RDS resources for their AWS account and to provide other related information, including Amazon EC2 security and network information.

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended for users with that IAM policy. To ensure that those users can still use the Amazon RDS console, also attach the `AmazonRDSReadOnlyAccess` managed policy to the user, as described in [AWS Managed \(Predefined\) Policies for Amazon RDS \(p. 349\)](#).

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the Amazon RDS API.

AWS Managed (Predefined) Policies for Amazon RDS

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see [AWS Managed Policies](#) in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon RDS:

- **AmazonRDSReadOnlyAccess** – Grants read-only access to all Amazon RDS resources for the root AWS account.
- **AmazonRDSFullAccess** – Grants full access to all Amazon RDS resources for the root AWS account.

You can also create custom IAM policies that allow users to access the required Amazon RDS API actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various Amazon RDS actions. These policies work when you are using RDS API actions, AWS SDKs, or the AWS CLI. When you are using the console, you need to grant additional permissions specific to the console, which is discussed in [Permissions Required to Use the Amazon RDS Console \(p. 349\)](#).

Note

All examples use the US West (Oregon) Region (`us-west-2`) and contain fictitious account IDs.

Examples

- [Example 1: Allow a User to Perform Any Describe Action on Any RDS Resource \(p. 350\)](#)
- [Example 2: Allow a User to Create a DB Instance That Uses the Specified DB Parameter and Security Groups \(p. 350\)](#)

- [Example 3: Prevent a User from Deleting a DB Instance \(p. 350\)](#)

Example 1: Allow a User to Perform Any Describe Action on Any RDS Resource

The following permissions policy grants permissions to a user to run all of the actions that begin with `Describe`. These actions show information about an RDS resource, such as a DB instance. The wildcard character (*) in the `Resource` element indicates that the actions are allowed for all Amazon RDS resources owned by the account.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowRDSDescribe",  
            "Effect": "Allow",  
            "Action": "rds:Describe*",  
            "Resource": "*"  
        }  
    ]  
}
```

Example 2: Allow a User to Create a DB Instance That Uses the Specified DB Parameter and Security Groups

The following permissions policy grants permissions to allow a user to only create a DB instance that must use the `mysql-production` DB parameter group and the `db-production` DB security group.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowMySQLProductionCreate",  
            "Effect": "Allow",  
            "Action": "rds>CreateDBInstance",  
            "Resource": [  
                "arn:aws:rds:us-west-2:123456789012:pg:mysql-production",  
                "arn:aws:rds:us-west-2:123456789012:secgrp:db-production"  
            ]  
        }  
    ]  
}
```

Example 3: Prevent a User from Deleting a DB Instance

The following permissions policy grants permissions to prevent a user from deleting a specific DB instance. For example, you might want to deny the ability to delete your production instances to any user that is not an administrator.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DenyDelete1",  
            "Effect": "Deny",  
            "Action": "rds>DeleteDBInstance",  
            "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-mysql-instance"  
        }  
    ]  
}
```

Amazon RDS API Permissions: Actions, Resources, and Conditions Reference

When you set up [access control \(p. 343\)](#) and write permissions policies that you can attach to an IAM identity (identity-based policies), you can use the following as a reference.

The following lists each Amazon RDS API operation. Included in the list are the corresponding actions for which you can grant permissions to perform the action, the AWS resource that you can grant the permissions for, and condition keys that you can include for fine-grained access control. You specify the actions in the policy's Action field, the resource value in the policy's Resource field, and conditions in the policy's Condition field. For more information about conditions, see [Using IAM Policy Conditions for Fine-Grained Access Control \(p. 367\)](#).

You can use AWS-wide condition keys in your Amazon RDS policies to express conditions. For a complete list of AWS-wide keys, see [Available Keys](#) in the *IAM User Guide*.

You can test IAM policies with the IAM policy simulator. It automatically provides a list of resources and parameters required for each AWS action, including Amazon RDS actions. The IAM policy simulator determines the permissions required for each of the actions that you specify. For information about the IAM policy simulator, see [Testing IAM Policies with the IAM Policy Simulator](#) in the *IAM User Guide*.

Note

To specify an action, use the `rds:` prefix followed by the API operation name (for example, `rds:CreateDBInstance`).

The following lists RDS API operations and their related actions, resources, and condition keys.

Topics

- [Amazon RDS Actions That Support Resource-Level Permissions \(p. 351\)](#)
- [Amazon RDS Actions That Don't Support Resource-Level Permissions \(p. 366\)](#)

Amazon RDS Actions That Support Resource-Level Permissions

Resource-level permissions refers to the ability to specify the resources on which users are allowed to perform actions. Amazon RDS has partial support for resource-level permissions. This means that for certain Amazon RDS actions, you can control when users are allowed to use those actions based on conditions that have to be fulfilled, or specific resources that users are allowed to use. For example, you can grant users permission to modify only specific DB instances.

The following lists RDS API operations and their related actions, resources, and condition keys.

RDS API Operations and Actions	Resources	Condition Keys
AddRoleToDBCluster <code>rds:AddRoleToDBCluster</code>	DB cluster <code>arn:aws:rds:region:account-id:cluster:db-cluster-name</code>	<code>rds:cluster-tag</code>
	IAM role <code>arn:aws:iam::account-id:role/role-name</code>	—
AddSourceIdentifierToSubscription	Events subscription	<code>rds:es-tag</code>

RDS API Operations and Actions	Resources	Condition Keys
	rds:AddSourceIdentifierToSubscriptionRegion:account-id:es:subscription-name	
AddTagsToResource	DB instance arn:aws:rds: region:account-id:db:db-instance-name	rds:db-tag rds:req-tag
	DB cluster arn:aws:rds: region:account-id:cluster:db-cluster-name	rds:cluster-tag rds:req-tag
	DB option group arn:aws:rds: region:account-id:og:option-group-name	rds:og-tag rds:req-tag
	DB parameter group arn:aws:rds: region:account-id:pg:parameter-group-name	rds:pg-tag rds:req-tag
	DB cluster parameter group arn:aws:rds: region:account-id:cluster-pg:cluster-parameter-group-name	rds:cluster-pg-tag rds:req-tag
	DB security group arn:aws:rds: region:account-id:secgrp:security-group-name	rds:secgrp-tag rds:req-tag
	DB subnet group arn:aws:rds: region:account-id:subgrp:subnet-group-name	rds:subgrp-tag rds:req-tag
	DB snapshot arn:aws:rds: region:account-id:snapshot:snapshot-name	rds:snapshot-tag rds:req-tag
	DB cluster snapshot arn:aws:rds: region:account-id:cluster-snapshot:cluster-snapshot-name	rds:cluster-snapshot-tag rds:req-tag
	Event subscription arn:aws:rds: region:account-id:es:subscription-name	rds:es-tag rds:req-tag

RDS API Operations and Actions	Resources	Condition Keys
	Reserved DB instance arn:aws:rds: <i>region:account-id:ri:reserved-db-instance-name</i>	rds:ri-tag rds:req-tag
ApplyPendingMaintenanceAction	DB instance rds:ApplyPendingMaintenanceAction: <i>region:account-id:db:db-instance-name</i>	rds:db-tag
AuthorizeDBSecurityGroup	DB security group rds:AuthorizeDBSecurityGroup: <i>region:account-id:secgrp:security-group-name</i>	rds:secgrp-tag
BacktrackDBCluster	DB cluster rds:BacktrackDBCluster: <i>region:account-id:cluster:db-cluster-name</i>	rds:cluster-tag
CopyDBClusterSnapshot	DB cluster snapshot rds:CopyDBClusterSnapshot: <i>region:account-id:cluster-snapshot:cluster-snapshot-name</i>	rds:cluster-snapshot-tag
CopyDBParameterGroup	DB parameter group rds:CopyDBParameterGroup: <i>region:account-id:pg:parameter-group-name</i>	rds:pg-tag
CopyDBSnapshot	DB snapshot rds:CopyDBSnapshot: <i>region:account-id:snapshot:snapshot-name</i>	rds:snapshot-tag
CopyOptionGroup	DB option group rds:CopyOptionGroup: <i>region:account-id:og:option-group-name</i>	rds:og-tag
CreateDBCluster	DB cluster rds:CreateDBCluster: <i>region:account-id:cluster:db-cluster-name</i>	rds:DatabaseEngine rds:DatabaseName rds:req-tag
	DB option group arn:aws:rds: <i>region:account-id:og:option-group-name</i>	rds:og-tag
	DB cluster parameter group arn:aws:rds: <i>region:account-id:cluster-pg:cluster-parameter-group-name</i>	rds:cluster-pg-tag

RDS API Operations and Actions	Resources	Condition Keys
	DB subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:subgrp-tag
CreateDBClusterEndpoint rds:CreateDBClusterEndpoint	DB cluster arn:aws:rds: <i>region:account-id:cluster:db-cluster-name</i>	rds:cluster-tag
	DB cluster endpoint arn:aws:rds: <i>region:account-id:cluster-endpoint:db-cluster-endpoint-identifier</i>	rds:endpointType
CreateDBClusterParameterGroup rds:CreateDBClusterParameterGroup	DB cluster parameter group arn:aws:rds: <i>region:account-id:cluster-pg:cluster-parameter-group-name</i>	rds:req-tag
CreateDBClusterSnapshot rds:CreateDBClusterSnapshot	DB cluster arn:aws:rds: <i>region:account-id:cluster:db-cluster-name</i>	rds:cluster-tag
	DB cluster snapshot arn:aws:rds: <i>region:account-id:cluster-snapshot:cluster-snapshot-name</i>	rds:req-tag
CreateDBInstance rds:CreateDBInstance	DB instance arn:aws:rds: <i>region:account-id:db:db-instance-name</i>	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:StorageSize rds:Vpc rds:req-tag
	DB option group arn:aws:rds: <i>region:account-id:og:option-group-name</i>	rds:og-tag
	DB parameter group arn:aws:rds: <i>region:account-id:pg:parameter-group-name</i>	rds:pg-tag

RDS API Operations and Actions	Resources	Condition Keys
	DB security group arn:aws:rds: <i>region:account-id:secgrp:security-group-name</i>	rds:secgrp-tag
	DB subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:subgrp-tag
	DB cluster arn:aws:rds: <i>region:account-id:cluster:db-cluster-name</i>	rds:cluster-tag
CreateDBInstanceReadReplica <code>rds:CreateDBInstanceReadReplica</code>	DB instance arn:aws:rds: <i>region:account-id:db:db-instance-name</i>	rds:DatabaseClass rds:Piops rds:req-tag
	DB option group arn:aws:rds: <i>region:account-id:og:option-group-name</i>	rds:og-tag
	DB subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:subgrp-tag
	DB parameter group arn:aws:rds: <i>region:account-id:pg:parameter-group-name</i>	rds:req-tag
CreateDBParameterGroup	DB parameter group	rds:req-tag
CreateDBSecurityGroup	DB security group arn:aws:rds: <i>region:account-id:secgrp:security-group-name</i>	rds:req-tag
CreateDBSnapshot <code>rds:CreateDBSnapshot</code>	DB instance arn:aws:rds: <i>region:account-id:db:db-instance-name</i>	rds:db-tag
	DB snapshot arn:aws:rds: <i>region:account-id:snapshot:snapshot-name</i>	rds:req-tag
CreateDBSubnetGroup	DB subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:req-tag

RDS API Operations and Actions	Resources	Condition Keys
CreateEventSubscription	Event subscription <code>rds:CreateEventSubscription</code> <code>arn:rds:region:account-id:es:subscription-name</code>	<code>rds:req-tag</code>
CreateOptionGroup	DB option group <code>rds:CreateOptionGroup</code> <code>arn:aws:rds:region:account-id:og:option-group-name</code>	<code>rds:req-tag</code>
DeleteDBCluster	DB cluster <code>rds:DeleteDBCluster</code> <code>arn:aws:rds:region:account-id:cluster:db-cluster-name</code>	<code>rds:cluster-tag</code>
	DB cluster snapshot <code>arn:aws:rds:region:account-id:cluster-snapshot:cluster-snapshot-name</code>	<code>rds:cluster-snapshot-tag</code>
DeleteDBClusterEndpoint	DB cluster endpoint <code>rds:DeleteDBClusterEndpoint</code> <code>arn:rds:region:account-id:cluster-endpoint:db-cluster-endpoint-identifier</code>	
DeleteDBClusterParameterGroup	DB cluster parameter group <code>rds:DeleteDBClusterParameterGroup</code> <code>arn:aws:rds:region:account-id:cluster-pg:cluster-parameter-group-name</code>	<code>rds:cluster-pg-tag</code>
DeleteDBClusterSnapshot	DB cluster snapshot <code>rds:DeleteDBClusterSnapshot</code> <code>arn:rds:region:account-id:cluster-snapshot:cluster-snapshot-name</code>	<code>rds:cluster-snapshot-tag</code>
DeleteDBInstance	DB instance <code>rds:DeleteDBInstance</code> <code>arn:aws:rds:region:account-id:db:db-instance-name</code>	<code>rds:db-tag</code>
DeleteDBParameterGroup	DB parameter group <code>rds:DeleteDBParameterGroup</code> <code>arn:rds:region:account-id:pg:parameter-group-name</code>	<code>rds:pg-tag</code>
DeleteDBSecurityGroup	DB security group <code>rds:DeleteDBSecurityGroup</code> <code>arn:rds:region:account-id:secgrp:security-group-name</code>	<code>rds:secgrp-tag</code>

RDS API Operations and Actions	Resources	Condition Keys
DeleteDBSnapshot	DB snapshot <code>rds:DeleteDBSnapshot arn:aws:rds:<i>region:account-id:snapshot:snapshot-name</i></code>	<code>rds:snapshot-tag</code>
DeleteDBSubnetGroup	DB subnet group <code>rds:DeleteDBSubnetGroup arn:aws:rds:<i>region:account-id:subgrp:subnet-group-name</i></code>	<code>rds:subgrp-tag</code>
DeleteEventSubscription	Event subscription <code>rds:DeleteEventSubscription arn:aws:rds:<i>region:account-id:es:subscription-name</i></code>	<code>rds:es-tag</code>
DeleteOptionGroup	DB option group <code>rds:DeleteOptionGroup arn:aws:rds:<i>region:account-id:og:option-group-name</i></code>	<code>rds:og-tag</code>
DescribeDBClusterEndpoints	DB cluster <code>rds:DescribeDBClusterEndpoints arn:aws:rds:<i>region:account-id:cluster:db-cluster-name</i></code>	<code>rds:cluster-tag</code>
	DB cluster endpoint <code>arn:aws:rds:<i>region:account-id:cluster-endpoint:db-cluster-endpoint-identifier</i></code>	
DescribeDBClusterParameterGroups	DB cluster parameter group <code>rds:DescribeDBClusterParameterGroups arn:aws:rds:<i>region:account-id:cluster-pg:cluster-parameter-group-name</i></code>	<code>rds:cluster-pg-tag</code>
DescribeDBClusterParameters	DB cluster parameter group <code>rds:DescribeDBClusterParameters arn:aws:rds:<i>region:account-id:cluster-pg:cluster-parameter-group-name</i></code>	<code>rds:cluster-pg-tag</code>
DescribeDBClusters	DB cluster <code>rds:DescribeDBClusters arn:aws:rds:<i>region:account-id:cluster:db-cluster-instance-name</i></code>	<code>rds:cluster-tag</code>
DescribeDBClusterSnapshots	DB cluster snapshot <code>rds:DescribeDBClusterSnapshots arn:aws:rds:<i>region:account-id:cluster-snapshot:cluster-snapshot-name</i></code>	<code>rds:cluster-snapshot-tag</code>

RDS API Operations and Actions	Resources	Condition Keys
DescribeDBEngineVersions	DB parameter group rds:DescribeDBEngineVersions arn:aws:rds:region:account-id:pg:parameter-group-name	rds:pg-tag
DescribeDBLogFiles	DB instance rds:DescribeDBLogFile arn:aws:rds:region:account-id:db:db-instance-name	rds:db-tag
DescribeDBParameterGroups	DB parameter group rds:DescribeDBParameterGroups arn:aws:rds:region:account-id:pg:parameter-group-name	rds:pg-tag
DescribeDBParameters	DB parameter group rds:DescribeDBParameters arn:aws:rds:region:account-id:pg:parameter-group-name	rds:pg-tag
DescribeDBSecurityGroups	DB security group rds:DescribeDBSecurityGroups arn:aws:rds:region:account-id:secgrp:security-group-name	rds:secgrp-tag
DescribeDBSnapshotAttributes	DB snapshot rds:DescribeDBSnapshotAttributes region:account-id:snapshot:snapshot-name	rds:snapshot-tag
DescribeDBSnapshots	DB instance rds:DescribeDBSnapshots arn:aws:rds:region:account-id:db:db-instance-name	rds:db-tag
	DB snapshot arn:aws:rds:region:account-id:snapshot:snapshot-name	rds:snapshot-tag
DescribeDBSubnetGroups	DB subnet group rds:DescribeDBSubnetGroups arn:aws:rds:region:account-id:subgrp:subnet-group-name	rds:subgrp-tag
DescribeEventSubscriptions	Event subscription rds:DescribeEventSubscriptions arn:aws:rds:region:account-id:es:subscription-name	rds:es-tag
DescribeOptionGroups	DB option group rds:DescribeOptionGroups arn:aws:rds:region:account-id:og:option-group-name	rds:og-tag

RDS API Operations and Actions	Resources	Condition Keys
DescribePendingMaintenanceActions	DB instance rds:DescribePendingMaintenanceActions <code>region:account-id:db:db-instance-name</code>	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:StorageSize rds:Vpc rds:db-tag
DescribeReservedDBInstancesOfferings	Reserved DB instance rds:DescribeReservedDBInstancesOfferings <code>region:account-id:ri:reserved-db-instance-name</code>	rds:DatabaseClass rds:MultiAz rds:ri-tag
DescribeReservedDBInstances	DB Offstances rds:DescribeReservedDBInstances <code>region:account-id:db:db-instance-name</code>	rds:DatabaseClass rds:MultiAz
DownloadDBLogFilePortion	DB instance rds:DownloadDBLogFilePortion <code>region:account-id:db:db-instance-name</code>	rds:db-tag
FailoverDBCluster	DB cluster rds:FailoverDBCluster <code>arn:aws:rds:region:account-id:cluster:db-cluster-instance-name</code>	rds:cluster-tag
ListTagsForResource	DB instance rds>ListTagsForResource <code>arn:aws:rds:region:account-id:db:db-instance-name</code>	rds:db-tag
	DB cluster arn:aws:rds:region:account-id:cluster:db-cluster-name	rds:cluster-tag
	DB option group arn:aws:rds:region:account-id:og:option-group-name	rds:og-tag
	DB parameter group arn:aws:rds:region:account-id:pg:parameter-group-name	rds:pg-tag

RDS API Operations and Actions	Resources	Condition Keys
	DB security group <code>arn:aws:rds:region:account-id:secgrp:security-group-name</code>	<code>rds:secgrp-tag</code>
	DB cluster parameter group <code>arn:aws:rds:region:account-id:cluster-pg:cluster-parameter-group-name</code>	<code>rds:cluster-pg-tag</code>
	DB subnet group <code>arn:aws:rds:region:account-id:subgrp:subnet-group-name</code>	<code>rds:subgrp-tag</code>
	DB snapshot <code>arn:aws:rds:region:account-id:snapshot:snapshot-name</code>	<code>rds:snapshot-tag</code>
	DB cluster snapshot <code>arn:aws:rds:region:account-id:cluster-snapshot:cluster-snapshot-name</code>	<code>rds:cluster-snapshot-tag</code>
	Event subscription <code>arn:aws:rds:region:account-id:es:subscription-name</code>	<code>rds:es-tag</code>
	Reserved DB instance <code>arn:aws:rds:region:account-id:ri:reserved-db-instance-name</code>	<code>rds:ri-tag</code>
ModifyDBCluster <code>rds:ModifyDBCluster</code>	DB cluster <code>arn:aws:rds:region:account-id:cluster:db-cluster-name</code>	<code>rds:cluster-tag</code>
	DB option group <code>arn:aws:rds:region:account-id:og:option-group-name</code>	<code>rds:og-tag</code>
	DB cluster parameter group <code>arn:aws:rds:region:account-id:cluster-pg:cluster-parameter-group-name</code>	<code>rds:cluster-pg-tag</code>

RDS API Operations and Actions	Resources	Condition Keys
ModifyDBClusterEndpoint	DB cluster endpoint <code>rds:ModifyDBClusterEndpoint</code> arn:aws:rds:region:account-id:cluster-endpoint:db-cluster-endpoint-identifier	<code>rds:endpointType</code>
ModifyDBClusterParameterGroup	DB cluster parameter group <code>rds:ModifyDBClusterParameterGroup</code> arn:aws:rds:region:account-id:cluster-pg:cluster-parameter-group-name	<code>rds:cluster-pg-tag</code>
ModifyDBClusterSnapshotAttribute	DB cluster snapshot <code>rds:ModifyDBClusterSnapshotAttribute</code> arn:aws:rds:region:account-id:cluster-snapshot:cluster-snapshot-name	<code>rds:cluster-snapshot-tag</code>
ModifyDBInstance	DB instance <code>rds:ModifyDBInstance</code> arn:aws:rds:region:account-id:db:db-instance-name	<code>rds:DatabaseClass</code> <code>rds:MultiAz</code> <code>rds:Piops</code> <code>rds:StorageSize</code> <code>rds:Vpc</code> <code>rds:db-tag</code>
	DB option group <code>arn:aws:rds:region:account-id:og:option-group-name</code>	<code>rds:og-tag</code>
	DB parameter group <code>arn:aws:rds:region:account-id:pg:parameter-group-name</code>	<code>rds:pg-tag</code>
	DB security group <code>arn:aws:rds:region:account-id:secgrp:security-group-name</code>	<code>rds:secgrp-tag</code>
ModifyDBParameterGroup	DB parameter group <code>rds:ModifyDBParameterGroup</code> arn:aws:rds:region:account-id:pg:parameter-group-name	<code>rds:pg-tag</code>
ModifyDBSnapshotAttribute	DB snapshot <code>rds:ModifyDBSnapshotAttribute</code> arn:aws:rds:region:account-id:snapshot:snapshot-name	<code>rds:snapshot-tag</code>

RDS API Operations and Actions	Resources	Condition Keys
ModifyDBSubnetGroup	DB subnet group rds:ModifyDBSubnetGroup <code>arn:aws:rds:region:account-id:subgrp:subnet-group-name</code>	rds:subgrp-tag
ModifyEventSubscription	Event subscription rds:ModifyEventSubscription <code>arn:aws:rds:region:account-id:es:subscription-name</code>	rds:es-tag
ModifyOptionGroup	DB option group rds:ModifyOptionGroup <code>arn:aws:rds:region:account-id:og:option-group-name</code>	rds:og-tag
PromoteReadReplica	DB instance rds:PromoteReadReplica <code>arn:aws:rds:region:account-id:db:db-instance-name</code>	rds:db-tag
PromoteReadReplicaDBCluster	DB cluster rds:PromoteReadReplicaDBCluster <code>arn:aws:rds:region:account-id:cluster:db-cluster-name</code>	
RebootDBInstance	DB instance rds:RebootDBInstance <code>arn:aws:rds:region:account-id:db:db-instance-name</code>	rds:db-tag
RemoveSourceIdentifierFromEventSubscription	Event subscription rds:RemoveSourceIdentifierFromEventSubscription <code>arn:aws:rds:region:account-id:es:subscription-name</code>	rds:es-tag
RemoveTagsFromResource	DB instance rds:RemoveTagsFromResource <code>arn:aws:rds:region:account-id:db:db-instance-name</code>	rds:db-tag rds:req-tag
	DB cluster arn:aws:rds:region:account-id:cluster:db-cluster-name	rds:cluster-tag rds:req-tag
	DB option group arn:aws:rds:region:account-id:og:option-group-name	rds:og-tag rds:req-tag
	DB parameter group arn:aws:rds:region:account-id:pg:parameter-group-name	rds:pg-tag rds:req-tag

RDS API Operations and Actions	Resources	Condition Keys
	DB cluster parameter group arn:aws:rds: <i>region:account-id:cluster-pg:cluster-parameter-group-name</i>	rds:cluster-pg-tag rds:req-tag
	DB security group arn:aws:rds: <i>region:account-id:secgrp:security-group-name</i>	rds:secgrp-tag rds:req-tag
	DB subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:subgrp-tag rds:req-tag
	DB snapshot arn:aws:rds: <i>region:account-id:snapshot:snapshot-name</i>	rds:snapshot-tag rds:req-tag
	DB cluster snapshot arn:aws:rds: <i>region:account-id:cluster-snapshot:cluster-snapshot-name</i>	rds:cluster-snapshot-tag rds:req-tag
	Event subscription arn:aws:rds: <i>region:account-id:es:subscription-name</i>	rds:es-tag rds:req-tag
	Reserved DB instance arn:aws:rds: <i>region:account-id:ri:reserved-db-instance-name</i>	rds:ri-tag rds:req-tag
ResetDBClusterParameter	DB cluster parameter group rds:ResetDBClusterParameter <i>ParameterGroup:region:account-id:cluster-pg:cluster-parameter-group-name</i>	rds:cluster-pg-tag
ResetDBParameterGroup	DB parameter group rds:ResetDBParameterGroup <i>Group:aws:rds:region:account-id:pg:parameter-group-name</i>	rds:pg-tag
RestoreDBClusterFromS3	DB cluster rds:RestoreDBClusterFromS3 <i>FromS3:aws:rds:region:account-id:cluster:db-cluster-instance-name</i>	rds:DatabaseEngine rds:DatabaseName rds:req-tag

RDS API Operations and Actions	Resources	Condition Keys
	DB cluster parameter group arn:aws:rds: <i>region:account-id:cluster-pg:cluster-parameter-group-name</i>	rds:cluster-pg-tag
	DB option group arn:aws:rds: <i>region:account-id:og:option-group-name</i>	rds:og-tag
	DB subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:subgrp-tag
RestoreDBClusterFromSnapshot	DB cluster rds:RestoreDBClusterFromSnapshot <i>region:account-id:cluster:db-cluster-instance-name</i>	rds:DatabaseEngine rds:DatabaseName rds:req-tag
	DB option group arn:aws:rds: <i>region:account-id:og:option-group-name</i>	rds:og-tag
	DB cluster snapshot arn:aws:rds: <i>region:account-id:cluster-snapshot:cluster-snapshot-name</i>	rds:cluster-snapshot-tag
	DB cluster rds:RestoreDBClusterToPointInTime <i>region:account-id:cluster:db-cluster-instance-name</i>	rds:req-tag
	DB option group arn:aws:rds: <i>region:account-id:og:option-group-name</i>	rds:og-tag
	DB subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:subgrp-tag

RDS API Operations and Actions	Resources	Condition Keys
RestoreDBInstanceFromDBSnapshot	DB instance rds:RestoreDBInstanceFromDBSnapshot <i>region:account-id:db:db-instance-name</i>	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:Vpc rds:req-tag
	DB option group arn:aws:rds: <i>region:account-id:og:option-group-name</i>	rds:og-tag
	DB snapshot arn:aws:rds: <i>region:account-id:snapshot:snapshot-name</i>	rds:snapshot-tag
	DB subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:subgrp-tag
RestoreDBInstanceToPointInTime	DB instance rds:RestoreDBInstanceToPointInTime <i>region:account-id:db:db-instance-name</i>	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:Vpc rds:req-tag
	DB option group arn:aws:rds: <i>region:account-id:og:option-group-name</i>	rds:og-tag
	DB snapshot arn:aws:rds: <i>region:account-id:snapshot:snapshot-name</i>	rds:snapshot-tag
	DB subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:subgrp-tag

RDS API Operations and Actions	Resources	Condition Keys
RevokeDBSecurityGroupIngress	DB security group rds:RevokeDBSecurityGroupIngress <i>region:account-id:secgrp:security-group-name</i>	rds:secgrp-tag
StartDBInstance	DB instance rds:StartDBInstance <i>arn:aws:rds:region:account-id:db:db-instance-name</i>	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:Vpc rds:db-tag
StopDBInstance	DB instance rds:StopDBInstance <i>arn:aws:rds:region:account-id:db:db-instance-name</i>	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:Vpc rds:db-tag

Amazon RDS Actions That Don't Support Resource-Level Permissions

You can use all Amazon RDS actions in an IAM policy to either grant or deny users permission to use that action. However, not all Amazon RDS actions support resource-level permissions, which enable you to specify the resources on which an action can be performed. The following Amazon RDS API actions currently don't support resource-level permissions. Therefore, to use these actions in an IAM policy, you must grant users permission to use all resources for the action by using a * wildcard for the `Resource` element in your statement.

- `rds:DescribeAccountAttributes`
- `rds:DescribeCertificates`
- `rds:DescribeDBClusterSnapshots`
- `rds:DescribeDBInstances`
- `rds:DescribeEngineDefaultClusterParameters`
- `rds:DescribeEngineDefaultParameters`
- `rds:DescribeEventCategories`
- `rds:DescribeEvents`
- `rds:DescribeOptionGroupOptions`

- `rds:DescribeOrderableDBInstanceOptions`
- `rds:DownloadCompleteDBLogFile`
- `rds:PurchaseReservedDBInstancesOffering`

Using IAM Policy Conditions for Fine-Grained Access Control

When you grant permissions in Amazon RDS, you can specify conditions that determine how a permissions policy takes effect.

Overview

In Amazon RDS, you have the option to specify conditions when granting permissions using an IAM policy (see [Access Control \(p. 343\)](#)). For example, you can:

- Allow users to create a DB instance only if they specify a particular database engine.
- Allow users to modify RDS resources that are tagged with a particular tag name and tag value.

There are two ways to specify conditions in an IAM policy for Amazon RDS:

- [Using Condition Keys \(p. 367\)](#)
- [Using Custom Tags \(p. 369\)](#)

Specifying Conditions: Using Condition Keys

AWS provides a set of predefined condition keys (AWS-wide condition keys) for all AWS services that support IAM for access control. For example, you can use the `aws:userid` condition key to require a specific AWS ID when requesting an action. For more information and a list of the AWS-wide condition keys, see [Available Keys for Conditions](#) in the *IAM User Guide*.

Note

Condition keys are case sensitive.

In addition Amazon RDS also provides its own condition keys that you can include in `Condition` elements in an IAM permissions policy. The following table shows the RDS condition keys that apply to RDS resources.

RDS Condition Key	Description	Value Type
<code>rds:DatabaseClass</code>	A type of DB instance class.	String
<code>rds:DatabaseEngine</code>	A database engine, such as MySQL.	String
<code>rds:DatabaseName</code>	The user-defined name of the database on the DB instance.	String
<code>rds:MultiAz</code>	A value that specifies whether the DB instance runs in multiple Availability Zones. To indicate that the DB instance is using Multi-AZ, specify <code>true</code> .	Boolean
<code>rds:Piops</code>	A value that contains the number of Provisioned IOPS (PIOPS) that the instance supports. To indicate a DB instance that does not have PIOPS enabled, specify 0.	Integer

RDS Condition Key	Description	Value Type
rds:StorageSize	The storage volume size (in GiB).	Integer
rds:Vpc	A value that specifies whether the DB instance runs in an Amazon Virtual Private Cloud (Amazon VPC). To indicate that the DB instance runs in an Amazon VPC, specify <code>true</code> .	Boolean
rds:req-tag	A value that limits the set of tag keys and values that can be used to tag a resource.	String

For example, the following `Condition` element uses a condition key and specifies the MySQL database engine. You could apply this to an IAM policy that allows permission to the `rds:CreateDBInstance` action to enable users to only create DB instances with the MySQL database engine. For an example of an IAM policy that uses this condition, see [Example Policies: Using Condition Keys \(p. 368\)](#).

```
"Condition": {"StringEquals": {"rds:DatabaseEngine": "mysql" } }
```

For a list of all of the RDS condition key identifiers and the RDS actions and resources that they apply to, see [Amazon RDS API Permissions: Actions, Resources, and Conditions Reference \(p. 351\)](#).

Example Policies: Using Condition Keys

Following are examples of how you can use condition keys in Amazon RDS IAM permissions policies.

Example 1: Grant Permission to Create a DB Instance that Uses a Specific DB Engine and Isn't MultiAZ

The following policy uses an RDS condition key and allows a user to create only DB instances that use the MySQL database engine and don't use MultiAZ. The `Condition` element indicates the requirement that the database engine is MySQL.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowMySQLCreate",
            "Effect": "Allow",
            "Action": "rds:CreateDBInstance",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseEngine": "mysql"
                },
                "Bool": {
                    "rds:MultiAz": false
                }
            }
        }
    ]
}
```

Example 2: Explicitly Deny Permission to Create DB Instances for Certain DB Instance Classes and Create DB Instances that Use Provisioned IOPS

The following policy explicitly denies permission to create DB instances that use the DB instance classes `r3.8xlarge` and `m4.10xlarge`, which are the largest and most expensive instances. This policy also prevents users from creating DB instances that use Provisioned IOPS, which incurs an additional cost.

Explicitly denying permission supersedes any other permissions granted. This ensures that identities do not accidentally get permission that you never want to grant.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyLargeCreate",
            "Effect": "Deny",
            "Action": "rds>CreateDBInstance",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseClass": [
                        "db.r3.8xlarge",
                        "db.m4.10xlarge"
                    ]
                }
            }
        },
        {
            "Sid": "DenyPIOPSCreate",
            "Effect": "Deny",
            "Action": "rds>CreateDBInstance",
            "Resource": "*",
            "Condition": {
                "NumericNotEquals": {
                    "rds:Piops": "0"
                }
            }
        }
    ]
}
```

Example 3: Limit the Set of Tag Keys and Values That Can Be Used to Tag a Resource

The following policy uses an RDS condition key and allows the addition of a tag with the key `stage` to be added to a resource with the values `test`, `qa`, and `production`.

```
{
    {
        "Version" : "2012-10-17",
        "Statement" : [
            {
                "Effect" : "Allow",
                "Action" : [ "rds:AddTagsToResource", "rds:RemoveTagsFromResource" ],
                "Resource" : "*",
                "Condition" : { "streq" : { "rds:req-tag/stage" : [ "test", "qa", "production" ] } }
            }
        ]
    }
}
```

Specifying Conditions: Using Custom Tags

RDS supports specifying conditions in an IAM policy using custom tags.

For example, if you add a tag named `environment` to your DB instances with values such as `beta`, `staging`, `production`, and so on, you can create a policy that restricts certain users to DB instances based on the `environment` tag value.

Note

Custom tag identifiers are case-sensitive.

The following table lists the RDS tag identifiers that you can use in a Condition element.

RDS Tag Identifier	Applies To
db-tag	DB instances, including Read Replicas
snapshot-tag	DB snapshots
ri-tag	Reserved DB instances
secgrp-tag	DB security groups
og-tag	DB option groups
pg-tag	DB parameter groups
subgrp-tag	DB subnet groups
es-tag	Event subscriptions
cluster-tag	DB clusters
cluster-pg-tag	DB cluster parameter groups
cluster-snapshot-tag	DB cluster snapshots

The syntax for a custom tag condition is as follows:

```
"Condition": {"StringEquals": {"rds:rds-tag-identifier/tag-name": ["value"]}} }
```

For example, the following Condition element applies to DB instances with a tag named environment and a tag value of production.

```
"Condition": {"StringEquals": {"rds:db-tag/environment": ["production"]}} }
```

For information about creating tags, see [Tagging Amazon RDS Resources \(p. 138\)](#).

Important

If you manage access to your RDS resources using tagging, we recommend that you secure access to the tags for your RDS resources. You can manage access to tags by creating policies for the AddTagsToResource and RemoveTagsFromResource actions. For example, the following policy denies users the ability to add or remove tags for all resources. You can then create policies to allow specific users to add or remove tags.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyTagUpdates",
            "Effect": "Deny",
            "Action": [
                "rds:AddTagsToResource",
                "rds:RemoveTagsFromResource"
            ],
            "Resource": "*"
        }
    ]
}
```

}

For a list of all of the condition key values, and the RDS actions and resources that they apply to, see [Amazon RDS API Permissions: Actions, Resources, and Conditions Reference \(p. 351\)](#).

Example Policies: Using Custom Tags

Following are examples of how you can use custom tags in Amazon RDS IAM permissions policies. For more information about adding tags to an Amazon RDS resource, see [Working with Amazon Resource Names \(ARNs\) in Amazon RDS \(p. 181\)](#).

Note

All examples use the us-west-2 region and contain fictitious account IDs.

Example 1: Grant Permission for Actions on a Resource with a Specific Tag with Two Different Values

The following policy allows permission to perform the `ModifyDBInstance` and `CreateDBSnapshot` APIs on instances with either the `stage` tag set to `development` or `test`.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowDevTestCreate",  
            "Effect": "Allow",  
            "Action": [  
                "rds:ModifyDBInstance",  
                "rds:CreateDBSnapshot"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "rds:db-tag/stage": [  
                        "development",  
                        "test"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

Example 2: Explicitly Deny Permission to Create a DB Instance that Uses Specified DB Parameter Groups

The following policy explicitly denies permission to create a DB instance that uses DB parameter groups with specific tag values. You might apply this policy if you require that a specific customer-created DB parameter group always be used when creating DB instances. Note that policies that use `Deny` are most often used to restrict access that was granted by a broader policy.

Explicitly denying permission supersedes any other permissions granted. This ensures that identities to not accidentally get permission that you never want to grant.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DenyProductionCreate",  
            "Effect": "Deny",  
            "Action": "rds:CreateDBInstance",  
            "Resource": "*"  
        }  
    ]  
}
```

```
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "rds:pg-tag/usage": "prod"
            }
        }
    ]
}
```

Example 3: Grant Permission for Actions on a DB Instance with an Instance Name that is Prefixed with a User Name

The following policy allows permission to call any API (except to `AddTagsToResource` or `RemoveTagsFromResource`) on a DB instance that has a DB instance name that is prefixed with the user's name and that has a tag called `stage` equal to `devo` or that has no tag called `stage`.

The `Resource` line in the policy identifies a resource by its Amazon Resource Name (ARN). For more information about using ARNs with Amazon RDS resources, see [Working with Amazon Resource Names \(ARNs\) in Amazon RDS \(p. 181\)](#).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowFullDevAccessNoTags",
            "Effect": "Allow",
            "NotAction": [
                "rds:AddTagsToResource",
                "rds:RemoveTagsFromResource"
            ],
            "Resource": "arn:aws:rds:*:123456789012:db:${aws:username}*",
            "Condition": {
                "StringEqualsIfExists": {
                    "rds:db-tag/stage": "devo"
                }
            }
        ]
    }
}
```

IAM Database Authentication for MySQL and PostgreSQL

You can authenticate to your DB instance using AWS Identity and Access Management (IAM) database authentication. IAM database authentication works with MySQL and PostgreSQL. With this authentication method, you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token.

An *authentication token* is a unique string of characters that Amazon RDS generates on request. Authentication tokens are generated using AWS Signature Version 4. Each token has a lifetime of 15 minutes. You don't need to store user credentials in the database, because authentication is managed externally using IAM. You can also still use standard database authentication.

IAM database authentication provides the following benefits:

- Network traffic to and from the database is encrypted using Secure Sockets Layer (SSL).
- You can use IAM to centrally manage access to your database resources, instead of managing access individually on each DB instance.

- For applications running on Amazon EC2, you can use profile credentials specific to your EC2 instance to access your database instead of a password, for greater security.

Topics

- [Availability for IAM Database Authentication \(p. 373\)](#)
- [MySQL Limitations for IAM Database Authentication \(p. 373\)](#)
- [PostgreSQL Limitations for IAM Database Authentication \(p. 373\)](#)
- [Enabling and Disabling IAM Database Authentication \(p. 374\)](#)
- [Creating and Using an IAM Policy for IAM Database Access \(p. 375\)](#)
- [Creating a Database Account Using IAM Authentication \(p. 378\)](#)
- [Connecting to Your DB Instance Using IAM Authentication \(p. 379\)](#)

Availability for IAM Database Authentication

IAM database authentication is available for the following database engines and instance classes:

- MySQL 5.6, minor version 5.6.34 or higher. All instance classes are supported, except for db.t2.micro, db.t2.small, and db.m1.small.
- MySQL 5.7, minor version 5.7.16 or higher. All instance classes are supported, except for db.t2.micro, db.t2.small, and db.m1.small.
- PostgreSQL versions 9.5.14, 9.6.9 or higher, and version 10.4 or higher.

Note

IAM database authentication is not supported for MySQL 5.5 or MySQL 8.0.

MySQL Limitations for IAM Database Authentication

When using IAM database authentication with MySQL, you are limited to a maximum of 20 new connections per second. If you are using a db.t2.micro instance class, the limit is 10 connections per second.

The database engines that work with Amazon RDS don't impose any limits on authentication attempts per second. However, when you use IAM database authentication, your application must generate an authentication token. Your application then uses that token to connect to the DB instance. If you exceed the limit of maximum new connections per second, then the extra overhead of IAM database authentication can cause connection throttling. The extra overhead can cause even existing connections to drop. For information about the maximum total connections for MySQL, see [Maximum MySQL connections \(p. 609\)](#)

We recommend the following when using the MySQL engine:

- Use IAM database authentication as a mechanism for temporary, personal access to databases.
- Use IAM database authentication only for workloads that can be easily retried.
- Don't use IAM database authentication if your application requires more than 20 new connections per second.

PostgreSQL Limitations for IAM Database Authentication

When using IAM database authentication with PostgreSQL, note the following limitations:

- The maximum number of connections for your database instance may be limited depending on the instance type and your workload.

- IAM database authentication is not supported with M5 instance types.

Enabling and Disabling IAM Database Authentication

By default, IAM database authentication is disabled on DB instances. You can enable IAM database authentication (or disable it again) using the AWS Management Console, AWS CLI, or the API.

IAM authentication for PostgreSQL DB instances require that the SSL value be 1. You cannot enable IAM authentication for a PostgreSQL DB instance if the SSL value is 0. You can't change the SSL value to 0 if IAM authentication is enabled for a PostgreSQL DB instance.

AWS Management Console

To create a new DB instance with IAM authentication by using the console, see either [Creating a DB Instance Running the MySQL Database Engine \(p. 597\)](#) or [Creating a DB Instance Running the PostgreSQL Database Engine \(p. 969\)](#).

Each creation workflow has a **Configure Advanced Settings** page, where you can enable IAM DB authentication. In that page's **Database Options** section, choose **Yes** for **Enable IAM DB Authentication**.

To enable or disable IAM authentication for an existing DB instance

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to modify.
4. Choose **Modify**.
5. In the **Database options** section, for **IAM DB authentication** choose **Enable IAM DB authentication** or **Disable**, and then choose **Continue**.
6. To apply the changes immediately, choose **Apply immediately**.
7. Choose **Modify DB instance**.

To restore a DB instance

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Snapshots**.
3. Choose the snapshot that you want to restore, and then choose **Restore Snapshot for Actions**.
4. In the **Settings** section, enter an identifier for the DB instance for **DB Instance Identifier**.
5. In the **Database options** section, for **IAM DB authentication**, choose **Enable IAM DB authentication** or **Disable**.
6. Choose **Restore DB Instance**.

AWS CLI

To create a new DB instance with IAM authentication by using the AWS CLI, use the `create-db-instance` command. Specify the `--enable-iam-database-authentication` option, as shown in the following example.

```
aws rds create-db-instance \
--db-instance-identifier mydbinstance \
--db-instance-class db.m3.medium \
--engine MySQL \
--allocated-storage 20 \
```

```
--master-username masterawsuser \
--master-user-password masteruserpassword \
--enable-iam-database-authentication
```

To update an existing DB cluster to have or not have IAM authentication, use the AWS CLI command [modify-db-instance](#). Specify either the --enable-iam-database-authentication or --no-enable-iam-database-authentication option, as appropriate.

By default, Amazon RDS performs the modification during the next maintenance window. If you want to override this and enable IAM DB authentication as soon as possible, use the --apply-immediately parameter.

The following example shows how to immediately enable IAM authentication for an existing DB instance.

```
aws rds modify-db-instance \
  --db-instance-identifier mydbinstance \
  --apply-immediately \
  --enable-iam-database-authentication
```

If you are restoring a DB instance, use one of the following AWS CLI commands:

- [restore-db-instance-to-point-in-time](#)
- [restore-db-instance-from-db-snapshot](#)

The IAM database authentication setting defaults to that of the source snapshot. To change this setting, set the --enable-iam-database-authentication or --no-enable-iam-database-authentication option, as appropriate.

RDS API

To create a new DB instance with IAM authentication by using the API, use the API operation [CreateDBInstance](#). Set the `EnableIAMDatabaseAuthentication` parameter to `true`.

To update an existing DB instance to have IAM authentication, use the API operation [ModifyDBInstance](#). Set the `EnableIAMDatabaseAuthentication` parameter to `true` to enable IAM authentication, or `false` to disable it.

If you are restoring a DB instance, use one of the following API actions:

- [RestoreDBInstanceToPointInTime](#)
- [RestoreDBInstanceFromDBSnapshot](#)

The IAM database authentication setting defaults to that of the source snapshot. To change this setting, set the `EnableIAMDatabaseAuthentication` parameter to `true` to enable IAM authentication, or `false` to disable it.

Creating and Using an IAM Policy for IAM Database Access

To allow an IAM user or role to connect to your DB instance, you must create an IAM policy. After that, you attach the policy to an IAM user or role.

Note

To learn more about IAM policies, see [Authentication and Access Control \(p. 342\)](#).

The following example policies allows an IAM user to connect to a DB instance using IAM database authentication.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "rds-db:connect"  
            ],  
            "Resource": [  
                "arn:aws:rds-db:us-east-2:1234567890:dbuser:db-ABCDEFGHIJKL01234/db_user"  
            ]  
        }  
    ]  
}
```

Note

Don't confuse the `rds-db:` prefix with other Amazon RDS action prefixes that begin with `rds:`. You use the `rds-db:` prefix and the `rds-db:connect` action only for IAM database authentication. They aren't valid in any other context.

Currently, the IAM console displays an error for policies with the `rds-db:connect` action. You can ignore this error.

The example policy includes a single statement with the following elements:

- **Effect** – Specify `Allow` to grant access to the DB instance. If you don't explicitly allow access, then access is denied by default.
- **Action** – Specify `rds-db:connect` to allow connection to the DB instance.
- **Resource** – Specify an Amazon Resource Name (ARN) that describes one database account in one DB instance. The ARN format is as follows.

```
arn:aws:rds-db:region:account-id:dbuser:dbi-resource-id/db-user-name
```

In this format, the following are so:

- **region** is the AWS Region for the Amazon RDS DB instance. In the example policy, the AWS Region is `us-east-2`.
- **account-id** is the AWS account number for the DB instance. In the example policy, the account number is `1234567890`.
- **dbi-resource-id** is the identifier for the DB instance. This identifier is unique to an AWS Region and never changes. In the example policy, the identifier is `db-ABCDEFGHIJKL01234`.

To find a DB instance resource ID in the AWS Management Console for Amazon RDS, choose the DB instance to see its details. Then choose the **Configuration** tab. The **Resource ID** is shown in the **Configuration Details** section.

Alternatively, you can use the AWS CLI command to list the identifiers and resource IDs for all of your DB instances in the current AWS Region, as shown following.

```
aws rds describe-db-instances \  
    --query "DBInstances[*].[DBInstanceIdentifier,DbiResourceId]"
```

- **db-user-name** is the name of the database account to associate with IAM authentication. In the example policy, the database account is `db_user`.

You can construct other ARNs to support various access patterns. The following policy allows access to two different database accounts in a DB instance .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-west-2:123456789012:dbuser:db-12ABC34DEFG5HIJ6KLMNOP78QR/jane_doe",
                "arn:aws:rds-db:us-west-2:123456789012:dbuser:db-12ABC34DEFG5HIJ6KLMNOP78QR/mary_roe"
            ]
        }
    ]
}
```

The following policy uses the "*" character to match all DB instances for a particular AWS account and AWS Region.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:1234567890:dbuser:*/db_user"
            ]
        }
    ]
}
```

The following policy matches all of the DB instances for a particular AWS account and AWS Region. However, the policy only grants access to DB instances that have a `jane_doe` database account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-west-2:123456789012:dbuser:*/jane_doe"
            ]
        }
    ]
}
```

The IAM user or role has access to only those databases that the database user does. For example, suppose that your DB instance has a database named `dev`, and another database named `test`. If the

database user `jane_doe` has access only to `dev`, any IAM users or roles that access that DB instance with the `jane_doe` user also have access only to `dev`. This access restriction is also true for other database objects, such as tables, views, and so on.

Attaching an IAM Policy to an IAM User or Role

After you create an IAM policy to allow database authentication, you need to attach the policy to an IAM user or role. For a tutorial on this topic, see [Create and Attach Your First Customer Managed Policy](#) in the *IAM User Guide*.

As you work through the tutorial, you can use one of the policy examples shown in this section as a starting point and tailor it to your needs. At the end of the tutorial, you have an IAM user with an attached policy that can make use of the `rds-db:connect` action.

Note

You can map multiple IAM users or roles to the same database user account. For example, suppose that your IAM policy specified the following resource ARN.

```
arn:aws:rds-db:us-west-2:123456789012:dbuser:db-12ABC34DEFG5HIJ6KLMNOP78QR/jane_doe
```

If you attach the policy to IAM users *Jane*, *Bob*, and *Diego*, then each of those users can connect to the specified DB instance using the `jane_doe` database account.

Creating a Database Account Using IAM Authentication

With IAM database authentication, you don't need to assign database passwords to the user accounts you create. If you remove an IAM user that is mapped to a database account, you should also remove the database account with the `DROP USER` statement.

Using IAM Authentication with PostgreSQL

To use IAM authentication with PostgreSQL, connect to the DB instance, create database users, and then grant them the `rds_iam` role as shown in the following example.

```
CREATE USER db_userx WITH LOGIN;
GRANT rds_iam TO db_userx;
```

Using IAM Authentication with MySQL

With MySQL, authentication is handled by `AWSAuthenticationPlugin`—an AWS-provided plugin that works seamlessly with IAM to authenticate your IAM users. Connect to the DB instance and issue the `CREATE USER` statement, as shown in the following example.

```
CREATE USER jane_doe IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
```

The `IDENTIFIED WITH` clause allows MySQL to use the `AWSAuthenticationPlugin` to authenticate the database account (`jane_doe`). The `AS 'RDS'` clause refers to the authentication method, and the specified database account must have the same name as the IAM user or role. In this example, both the database account and the IAM user or role must be named `jane_doe`.

Note

If you see the following message, it means that the AWS-provided plugin is not available for the current DB instance.

ERROR 1524 (HY000): Plugin 'AWSAuthenticationPlugin' is not loaded
To troubleshoot this error, verify that you are using a supported configuration and that
you have enabled IAM database authentication on your DB instance. For more information,
see [Availability for IAM Database Authentication \(p. 373\)](#) and [Enabling and Disabling IAM
Database Authentication \(p. 374\)](#).

After you create an account using `AWSAuthenticationPlugin`, you manage it in the same way as other database accounts. For example, you can modify account privileges with `GRANT` and `REVOKE` statements, or modify various account attributes with the `ALTER USER` statement.

Connecting to Your DB Instance Using IAM Authentication

With IAM database authentication, you use an authentication token when you connect to your DB instance. An *authentication token* is a string of characters that you use instead of a password. After you generate an authentication token, it's valid for 15 minutes before it expires. If you try to connect using an expired token, the connection request is denied.

Every authentication token must be accompanied by a valid signature, using AWS signature version 4. (For more information, see [Signature Version 4 Signing Process in the AWS General Reference](#).) The AWS CLI and the AWS SDK for Java can automatically sign each token you create.

You can use an authentication token when you connect to Amazon RDS from another AWS service, such as AWS Lambda. By using a token, you can avoid placing a password in your code. Alternatively, you can use the AWS SDK for Java to manually create and manually sign an authentication token.

After you have a signed IAM authentication token, you can connect to an Amazon RDS DB instance. Following, you can find out how to do this using either a command line tool or the AWS SDK for Java.

For more information, see [Use IAM authentication to connect with SQL Workbench/J to Amazon Aurora MySQL or Amazon RDS for MySQL](#).

Topics

- [Connecting to Your DB Instance from the Command Line: AWS CLI and mysql Client \(p. 379\)](#)
- [Connecting to Your DB Instance from the Command Line: AWS CLI and psql Client \(p. 381\)](#)
- [Connecting to Your DB Instance Using the AWS SDK for Java \(p. 382\)](#)

Connecting to Your DB Instance from the Command Line: AWS CLI and mysql Client

You can connect from the command line to an Amazon RDS DB instance with the AWS CLI and `mysql` command line tool as described following.

Topics

- [Generating an IAM Authentication Token \(p. 379\)](#)
- [Connecting to a DB Instance \(p. 380\)](#)

Generating an IAM Authentication Token

The following example shows how to get a signed authentication token using the AWS CLI.

```
aws rds generate-db-auth-token \
--hostname rdsmysql.cdgmuiqadpid.us-west-2.rds.amazonaws.com \
--port 3306 \
--region us-west-2 \
```

```
--username jane_doe
```

In the example, the parameters are as follows:

- --hostname – The host name of the DB instance that you want to access.
- --port – The port number used for connecting to your DB instance.
- --region – The AWS Region where the DB instance is running.
- --username – The database account that you want to access.

The first several characters of the token look like the following.

```
rdsmysql.cdgmuiadpid.us-west-2.rds.amazonaws.com:3306/?Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

Connecting to a DB Instance

The general format for connecting is shown following.

```
mysql --host=hostName --port=portNumber --ssl-ca=[full path]rds-combined-ca-bundle.pem --enable-cleartext-plugin --user=userName --password=authToken
```

The parameters are as follows:

- --host – The host name of the DB instance that you want to access.
- --port – The port number used for connecting to your DB instance.
- --ssl-ca – The SSL certificate file that contains the public key. For more information, see [Using SSL to Encrypt a Connection to a DB Instance \(p. 392\)](#).
- --enable-cleartext-plugin – A value that specifies that `AWSAuthenticationPlugin` must be used for this connection.
- --user – The database account that you want to access.
- --password – A signed IAM authentication token.

The authentication token consists of several hundred characters. It can be unwieldy on the command line. One way to work around this is to save the token to an environment variable, and then use that variable when you connect. The following example shows one way to perform this workaround.

```
RDSHOST="rdsmysql.cdgmuiadpid.us-west-2.rds.amazonaws.com"
TOKEN=$(aws rds generate-db-auth-token --hostname $RDSHOST --port 3306 --region us-west-2
--username jane_doe )"

mysql --host=$RDSHOST --port=3306 --ssl-ca=/sample_dir/rds-combined-ca-bundle.pem --enable-
cleartext-plugin --user=jane_doe --password=$TOKEN
```

When you connect using `AWSAuthenticationPlugin`, the connection is secured using SSL. To verify this, type the following at the `mysql>` command prompt.

```
show status like 'Ssl%';
```

The following lines in the output show more details.

Variable_name	Value
...	...
Ssl_cipher	AES256-SHA
...	...
Ssl_version	TLSv1.1
...	...

Connecting to Your DB Instance from the Command Line: AWS CLI and psql Client

You can connect from the command line to an Amazon RDS for PostgreSQL DB instance with the AWS CLI and psql command line tool as described following.

Topics

- [Generating an IAM Authentication Token \(p. 381\)](#)
- [Connecting to an Amazon RDS PostgreSQL Instance \(p. 381\)](#)

Generating an IAM Authentication Token

The authentication token consists of several hundred characters so it can be unwieldy on the command line. One way to work around this is to save the token to an environment variable, and then use that variable when you connect. The following example shows how to use the AWS CLI to get a signed authentication token using the `generate-db-auth-token` command, and store it in a `PGPASSWORD` environment variable.

```
export RDSSHOST="rdspostgres.cdgmuqiadpid.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSSHOST --port 5432 --
region us-west-2 --username jane_doe )"
```

In the example, the parameters to the `generate-db-auth-token` command are as follows:

- `--hostname` – The host name of the DB instance that you want to access.
- `--port` – The port number used for connecting to your DB instance.
- `--region` – The AWS Region where the DB instance is running.
- `--username` – The database account that you want to access.

The first several characters of the generated token look like the following.

```
rdspostgres.cdgmuqiadpid.us-west-2.rds.amazonaws.com:5432/?Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

Connecting to an Amazon RDS PostgreSQL Instance

The general format for using psql to connect is shown following.

```
psql "host=hostName port=portNumber sslmode=verify-full sslrootcert=certificateFile  
dbname=DBName user=userName"
```

The parameters are as follows:

- **host** – The host name of the DB instance that you want to access.
- **port** – The port number used for connecting to your DB instance.
- **sslmode** – The SSL mode to use. When you use `sslmode=verify-full`, the SSL connection verifies the DB instance endpoint against the endpoint in the SSL certificate.
- **sslrootcert** – The SSL certificate file that contains the public key. For more information, see [Using SSL with a PostgreSQL DB Instance](#).
- **dbname** – The database that you want to access.
- **user** – The database account that you want to access.

The following example shows using the command to connect. The example uses the environment variables that were set when the token was generated in the previous section.

```
psql "host=$RDSHOST port=5432 sslmode=verify-full sslrootcert=/sample_dir/rds-combined-ca-  
bundle.pem dbname=DBName user=jane_doe"
```

Connecting to Your DB Instance Using the AWS SDK for Java

You can connect from the command line to an Amazon RDS DB instance with the AWS SDK for Java as described following.

Topics

- [Generating an IAM Authentication Token \(p. 382\)](#)
- [Manually Constructing an IAM Authentication Token \(p. 383\)](#)
- [Connecting to a DB Instance \(p. 386\)](#)

Generating an IAM Authentication Token

If you are writing programs using the AWS SDK for Java, you can get a signed authentication token using the `RdsIamAuthTokenGenerator` class. Using this class requires that you provide AWS credentials. To do this, you create an instance of the `DefaultAWSCredentialsProviderChain` class. `DefaultAWSCredentialsProviderChain` uses the first AWS access key and secret key that it finds in the [default credential provider chain](#). For more information about AWS access keys, see [Managing Access Keys for IAM Users](#).

After you create an instance of `RdsIamAuthTokenGenerator`, you can call the `getAuthToken` method to obtain a signed token. Provide the AWS Region, host name, port number, and user name. The following code example illustrates how to do this.

```
package com.amazonaws.codesamples;  
  
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;  
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;  
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;  
  
public class GenerateRDSToken {  
  
    public static void main(String[] args) {
```

```

String region = "us-west-2";
String hostname = "rdsmysql.cdgmuqiadpid.us-west-2.rds.amazonaws.com";
String port = "3306";
String username = "jane_doe";

System.out.println(generateAuthToken(region, hostname, port, username));
}

static String generateAuthToken(String region, String hostName, String port, String
username) {

    RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
        .credentials(new DefaultAWSCredentialsProviderChain())
        .region(region)
        .build();

    String authToken = generator.getAuthToken(
        GetIamAuthTokenRequest.builder()
            .hostname(hostName)
            .port(Integer.parseInt(port))
            .userName(username)
            .build());
}

return authToken;
}

}

```

Manually Constructing an IAM Authentication Token

In Java, the easiest way to generate an authentication token is to use `RdsIamAuthTokenGenerator`. This class creates an authentication token for you, and then signs it using AWS signature version 4. For more information, see [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

However, you can also construct and sign an authentication token manually, as shown in the following code example.

```

package com.amazonaws.codesamples;

import com.amazonaws.SdkClientException;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.SigningAlgorithm;
import com.amazonaws.util.BinaryUtils;
import org.apache.commons.lang3.StringUtils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.SortedMap;
import java.util.TreeMap;

import static com.amazonaws.auth.internal.SignerConstants.AWS4_TERMINATOR;
import static com.amazonaws.util.StringUtils.UTF8;

public class CreateRDSAuthTokenManually {
    public static String httpMethod = "GET";
    public static String action = "connect";
    public static String canonicalURIParameter = "/";
    public static SortedMap<String, String> canonicalQueryParameters = new TreeMap();
    public static String payload = StringUtils.EMPTY;
    public static String signedHeader = "host";
}

```

```

public static String algorithm = "AWS4-HMAC-SHA256";
public static String serviceName = "rds-db";
public static String requestWithoutSignature;

public static void main(String[] args) throws Exception {

    String region = "us-west-2";
    String instanceName = "rdsmysql.cdgmuqiadpid.us-west-2.rds.amazonaws.com";
    String port = "3306";
    String username = "jane_doe";

    Date now = new Date();
    String date = new SimpleDateFormat("yyyyMMdd").format(now);
    String dateTimeStamp = new SimpleDateFormat("yyyyMMdd'T'HHmmssZ").format(now);
    DefaultAWSCredentialsProviderChain creds = new
DefaultAWSCredentialsProviderChain();
    String awsAccessKey = creds.getCredentials().getAWSAccessKeyId();
    String awsSecretKey = creds.getCredentials().getAWSSecretKey();
    String expiryMinutes = "900";

    System.out.println("Step 1: Create a canonical request:");
    String canonicalString = createCanonicalString(username, awsAccessKey, date,
dateTimeStamp, region, expiryMinutes, instanceName, port);
    System.out.println(canonicalString);
    System.out.println();

    System.out.println("Step 2: Create a string to sign:");
    String stringToSign = createStringToSign(dateTimeStamp, canonicalString,
awsAccessKey, date, region);
    System.out.println(stringToSign);
    System.out.println();

    System.out.println("Step 3: Calculate the signature:");
    String signature = BinaryUtils.toHex(calculateSignature(stringToSign,
newSigningKey(awsSecretKey, date, region, serviceName)));
    System.out.println(signature);
    System.out.println();

    System.out.println("Step 4: Add the signing info to the request");
    System.out.println	appendSignature(signature));
    System.out.println();

}

//Step 1: Create a canonical request date should be in format YYYYMMDD and dateTime
should be in format YYYYMMDDTHHMMSSZ
public static String createCanonicalString(String user, String accessKey, String date,
String dateTime, String region, String expiryPeriod, String hostName, String port) throws
Exception {
    canonicalQueryParameters.put("Action", action);
    canonicalQueryParameters.put("DBUser", user);
    canonicalQueryParameters.put("X-Amz-Algorithm", "AWS4-HMAC-SHA256");
    canonicalQueryParameters.put("X-Amz-Credential", accessKey + "%2F" + date + "%2F" +
region + "%2F" + serviceName + "%2Faws4_request");
    canonicalQueryParameters.put("X-Amz-Date", dateTime);
    canonicalQueryParameters.put("X-Amz-Expires", expiryPeriod);
    canonicalQueryParameters.put("X-Amz-SignedHeaders", signedHeader);
    String canonicalQueryString = "";
    while(!canonicalQueryParameters.isEmpty()) {
        String currentQueryParameter = canonicalQueryParameters.firstKey();
        String currentQueryParameterValue =
canonicalQueryParameters.remove(currentQueryParameter);
        canonicalQueryString = canonicalQueryString + currentQueryParameter + "=" +
currentQueryParameterValue;
        if (!currentQueryParameter.equals("X-Amz-SignedHeaders")) {
            canonicalQueryString += "&";
        }
    }
}

```

```

        }
    }
    String canonicalHeaders = "host:" + hostName + ":" + port + '\n';
    requestWithoutSignature = hostName + ":" + port + "/" + canonicalQueryString;

    String hashedPayload = BinaryUtils.toHex(hash(payload));
    return httpMethod + '\n' + canonicalURIParameter + '\n' + canonicalQueryString +
    '\n' + canonicalHeaders + '\n' + signedHeader + '\n' + hashedPayload;
}

//Step 2: Create a string to sign using sig v4
public static String createStringToSign(String date, String canonicalRequest,
String accessKey, String date, String region) throws Exception {
    String credentialScope = date + "/" + region + "/" + serviceName + "/aws4_request";
    return algorithm + '\n' + date + '\n' + canonicalRequest + '\n' +
BinaryUtils.toHex(hash(canonicalRequest));

}

//Step 3: Calculate signature
/**
 * Step 3 of the AWS Signature version 4 calculation. It involves deriving
 * the signing key and computing the signature. Refer to
 * http://docs.aws.amazon
 * .com/general/latest/gr/sigv4-calculate-signature.html
 */
public static byte[] calculateSignature(String stringToSign,
                                         byte[] signingKey) {
    return sign(stringToSign.getBytes(Charset.forName("UTF-8")), signingKey,
               SigningAlgorithm.HmacSHA256);
}

public static byte[] sign(byte[] data, byte[] key,
                        SigningAlgorithm algorithm) throws SdkClientException {
    try {
        Mac mac = algorithm.getMac();
        mac.init(new SecretKeySpec(key, algorithm.toString()));
        return mac.doFinal(data);
    } catch (Exception e) {
        throw new SdkClientException(
            "Unable to calculate a request signature: "
            + e.getMessage(), e);
    }
}

public static byte[] newSigningKey(String secretKey,
                                   String dateStamp, String regionName, String serviceName)
{
    byte[] kSecret = ("AWS4" + secretKey).getBytes(Charset.forName("UTF-8"));
    byte[] kDate = sign(dateStamp, kSecret, SigningAlgorithm.HmacSHA256);
    byte[] kRegion = sign(regionName, kDate, SigningAlgorithm.HmacSHA256);
    byte[] kService = sign(serviceName, kRegion,
                           SigningAlgorithm.HmacSHA256);
    return sign(AWS4_TERMINATOR, kService, SigningAlgorithm.HmacSHA256);
}

public static byte[] sign(String stringData, byte[] key,
                        SigningAlgorithm algorithm) throws SdkClientException {
    try {
        byte[] data = stringData.getBytes(UTF8);
        return sign(data, key, algorithm);
    } catch (Exception e) {
        throw new SdkClientException(
            "Unable to calculate a request signature: "
            + e.getMessage(), e);
    }
}

```

```

        }

    //Step 4: append the signature
    public static String appendSignature(String signature) {
        return requestWithoutSignature + "&X-Amz-Signature=" + signature;
    }

    public static byte[] hash(String s) throws Exception {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(UTF8));
            return md.digest();
        } catch (Exception e) {
            throw new SdkClientException(
                "Unable to compute hash while signing request: "
                + e.getMessage(), e);
        }
    }
}

```

Connecting to a DB Instance

The following code example shows how to generate an authentication token, and then use it to connect to an instance running MySQL.

To run this code example, you need the [AWS SDK for Java](#), found on the AWS site. In addition, you need the following:

- MySQL Connector/J. This code example was tested with `mysql-connector-java-5.1.33-bin.jar`.
- An intermediate certificate for Amazon RDS that is specific to an AWS Region. (For more information, see [Using SSL to Encrypt a Connection to a DB Instance \(p. 392\)](#).) At runtime, the class loader looks for the certificate in the same directory as this Java code example, so that the class loader can find it.
- Modify the values of the following variables as needed:
 - `RDS_INSTANCE_HOSTNAME` – The host name of the DB instance that you want to access.
 - `RDS_INSTANCE_PORT` – The port number used for connecting to your PostgreSQL DB instance.
 - `REGION_NAME` – The AWS Region where the DB instance is running.
 - `DB_USER` – The database account that you want to access.
 - `SSL_CERTIFICATE` – An SSL certificate for Amazon RDS that is specific to an AWS Region. To download a certificate for your AWS Region, see [Intermediate Certificates \(p. 393\)](#). Place the SSL certificate in the same directory as this Java program file, so that the class loader can find the certificate at runtime.

This code example obtains AWS credentials from the [default credential provider chain](#).

```

package com.amazonaws.samples;

import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.CertificateFactory;

```

```

import java.security.cert.X509Certificate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

import java.net.URL;

public class IAMDatabaseAuthenticationTester {
    //AWS Credentials of the IAM user with policy enabling IAM Database Authenticated
    access to the db by the db user.
    private static final DefaultAWSCredentialsProviderChain creds = new
DefaultAWSCredentialsProviderChain();
    private static final String AWS_ACCESS_KEY =
creds.getCredentials().getAWSAccessKeyId();
    private static final String AWS_SECRET_KEY = creds.getCredentials().getAWSSecretKey();

    //Configuration parameters for the generation of the IAM Database Authentication token
    private static final String RDS_INSTANCE_HOSTNAME = "rdsmysql.cdgmugiadpid.us-
west-2.rds.amazonaws.com";
    private static final int RDS_INSTANCE_PORT = 3306;
    private static final String REGION_NAME = "us-west-2";
    private static final String DB_USER = "jane_doe";
    private static final String JDBC_URL = "jdbc:mysql://" + RDS_INSTANCE_HOSTNAME + ":" +
RDS_INSTANCE_PORT;

    private static final String SSL_CERTIFICATE = "rds-ca-2015-us-west-2.pem";

    private static final String KEY_STORE_TYPE = "JKS";
    private static final String KEY_STORE_PROVIDER = "SUN";
    private static final String KEY_STORE_FILE_PREFIX = "sys-connect-via-ssl-test-cacerts";
    private static final String KEY_STORE_FILE_SUFFIX = ".jks";
    private static final String DEFAULT_KEY_STORE_PASSWORD = "changeit";

    public static void main(String[] args) throws Exception {
        //get the connection
        Connection connection = getDBConnectionUsingIam();

        //verify the connection is successful
        Statement stmt= connection.createStatement();
        ResultSet rs=stmt.executeQuery("SELECT 'Success!' FROM DUAL;");
        while (rs.next()) {
            String id = rs.getString(1);
            System.out.println(id); //Should print "Success!"
        }

        //close the connection
        stmt.close();
        connection.close();

        clearSslProperties();
    }

    /**
     * This method returns a connection to the db instance authenticated using IAM Database
     Authentication
     * @return
     * @throws Exception
     */
    private static Connection getDBConnectionUsingIam() throws Exception {
        setSslProperties();
        return DriverManager.getConnection(JDBC_URL, setMySqlConnectionProperties());
    }
}

```

```

/**
 * This method sets the mysql connection properties which includes the IAM Database
Authentication token
 * as the password. It also specifies that SSL verification is required.
 * @return
 */
private static Properties setMySqlConnectionProperties() {
    Properties mysqlConnectionProperties = new Properties();
    mysqlConnectionProperties.setProperty("verifyServerCertificate","true");
    mysqlConnectionProperties.setProperty("useSSL", "true");
    mysqlConnectionProperties.setProperty("user",DB_USER);
    mysqlConnectionProperties.setProperty("password",generateAuthToken());
    return mysqlConnectionProperties;
}

/**
 * This method generates the IAM Auth Token.
 * An example IAM Auth Token would look like follows:
 * btusiu123.cmz7kenwo2ye.rds.cn-north-1.amazonaws.com.cn:3306/?  

Action=connect&DBUser=iamtestuser&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-  

Date=20171003T010726Z&X-Amz-SignedHeaders=host&X-Amz-Expires=899&X-Amz-  

Credential=AKIAIPFXHGVDI5RNFO4AQ%2F20171003%2Fcn-north-1%2Frds-db%2Faws4_request&X-Amz-  

Signature=f9f45ef96c1f770cdad11a53e33ffa4c3730bc03fdee820cfdf1322eed15483b
 * @return
 */
private static String generateAuthToken() {
    BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY,
AWS_SECRET_KEY);

    RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
        .credentials(new
AWSStaticCredentialsProvider(awsCredentials)).region(REGION_NAME).build();
    return generator.getAuthToken(GetIamAuthTokenRequest.builder()

.hostname(RDS_INSTANCE_HOSTNAME).port(RDS_INSTANCE_PORT).userName(DB_USER).build());
}

/**
 * This method sets the SSL properties which specify the key store file, its type and
password:
 * @throws Exception
 */
private static void setSslProperties() throws Exception {
    System.setProperty("javax.net.ssl.trustStore", createKeyStoreFile());
    System.setProperty("javax.net.ssl.trustStoreType", KEY_STORE_TYPE);
    System.setProperty("javax.net.ssl.trustStorePassword", DEFAULT_KEY_STORE_PASSWORD);
}

/**
 * This method returns the path of the Key Store File needed for the SSL verification
during the IAM Database Authentication to
 * the db instance.
 * @return
 * @throws Exception
 */
private static String createKeyStoreFile() throws Exception {
    return createKeyStoreFile(createCertificate()).getPath();
}

/**
 * This method generates the SSL certificate
 * @return
 * @throws Exception
 */
private static X509Certificate createCertificate() throws Exception {

```

```
CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
URL url = new File(SSL_CERTIFICATE).toURI().toURL();
if (url == null) {
    throw new Exception();
}
try (InputStream certInputStream = url.openStream()) {
    return (X509Certificate) certFactory.generateCertificate(certInputStream);
}

/**
 * This method creates the Key Store File
 * @param rootX509Certificate - the SSL certificate to be stored in the KeyStore
 * @return
 * @throws Exception
 */
private static File createKeyStoreFile(X509Certificate rootX509Certificate) throws
Exception {
    File keyStoreFile = File.createTempFile(KEY_STORE_FILE_PREFIX,
KEY_STORE_FILE_SUFFIX);
    try (FileOutputStream fos = new FileOutputStream(keyStoreFile.getPath())) {
        KeyStore ks = KeyStore.getInstance(KEY_STORE_TYPE, KEY_STORE_PROVIDER);
        ks.load(null);
        ks.setCertificateEntry("rootCaCertificate", rootX509Certificate);
        ks.store(fos, DEFAULT_KEY_STORE_PASSWORD.toCharArray());
    }
    return keyStoreFile;
}

/**
 * This method clears the SSL properties.
 * @throws Exception
 */
private static void clearSslProperties() throws Exception {
    System.clearProperty("javax.net.ssl.trustStore");
    System.clearProperty("javax.net.ssl.trustStoreType");
    System.clearProperty("javax.net.ssl.trustStorePassword");
}
}
```

Encrypting Amazon RDS Resources

You can encrypt your Amazon RDS DB instances and snapshots at rest by enabling the encryption option for your Amazon RDS DB instances. Data that is encrypted at rest includes the underlying storage for a DB instances, its automated backups, Read Replicas, and snapshots.

Amazon RDS encrypted DB instances use the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon RDS DB instances. Once your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.

Topics

- [Overview of Encrypting Amazon RDS Resources \(p. 390\)](#)
- [Enabling Amazon RDS Encryption for a DB Instance \(p. 390\)](#)
- [Availability of Amazon RDS Encryption \(p. 391\)](#)
- [Managing Amazon RDS Encryption Keys \(p. 391\)](#)
- [Limitations of Amazon RDS Encrypted DB Instance \(p. 392\)](#)

Overview of Encrypting Amazon RDS Resources

Amazon RDS encrypted DB instances provide an additional layer of data protection by securing your data from unauthorized access to the underlying storage. You can use Amazon RDS encryption to increase data protection of your applications deployed in the cloud, and to fulfill compliance requirements for data-at-rest encryption.

Amazon RDS also supports encrypting an Oracle or SQL Server DB instance with Transparent Data Encryption (TDE). TDE can be used with encryption at rest, although using TDE and encryption at rest simultaneously might slightly affect the performance of your database. You must manage different keys for each encryption method. For more information on TDE, see [Oracle Transparent Data Encryption \(p. 836\)](#), [Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys \(p. 886\)](#), or [Microsoft SQL Server Transparent Data Encryption Support \(p. 561\)](#).

To manage the keys used for encrypting and decrypting your Amazon RDS resources, you use the [AWS Key Management Service \(AWS KMS\)](#). AWS KMS combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Using AWS KMS, you can create encryption keys and define the policies that control how these keys can be used. AWS KMS supports CloudTrail, so you can audit key usage to verify that keys are being used appropriately. Your AWS KMS keys can be used in combination with Amazon RDS and supported AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), and Amazon Redshift. For a list of services that support AWS KMS, go to [Supported Services](#) in the *AWS Key Management Service Developer Guide*.

For an Amazon RDS encrypted DB instance, all logs, backups, and snapshots are encrypted. A Read Replica of an Amazon RDS encrypted instance is also encrypted using the same key as the master instance when both are in the same region. If the master and Read Replica are in different regions, you encrypt using the encryption key for that region.

For encrypted and unencrypted Amazon RDS DB instances with cross-region Read Replicas, data sent between the source and the Read Replicas is encrypted.

Enabling Amazon RDS Encryption for a DB Instance

To enable encryption for a new DB instance, choose **Enable encryption** on the Amazon RDS console. For information on creating a DB instance, see one of the following topics:

- [Creating a DB Instance Running the MySQL Database Engine \(p. 597\)](#)
- [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#)
- [Creating a DB Instance Running the Microsoft SQL Server Database Engine \(p. 504\)](#)
- [Creating a DB Instance Running the PostgreSQL Database Engine \(p. 969\)](#)
- [Creating a DB Instance Running the MariaDB Database Engine \(p. 443\)](#)

If you use the `create-db-instance` AWS CLI command to create an encrypted RDS DB instance, set the `--storage-encrypted` parameter to true. If you use the `CreateDBInstance` API action, set the `StorageEncrypted` parameter to true.

When you create an encrypted DB instance, you can also supply the AWS KMS key identifier for your encryption key. If you don't specify an AWS KMS key identifier, then Amazon RDS uses your default encryption key for your new DB instance. AWS KMS creates your default encryption key for Amazon RDS for your AWS account. Your AWS account has a different default encryption key for each AWS Region.

Once you have created an encrypted DB instance, you cannot change the encryption key for that instance. Therefore, be sure to determine your encryption key requirements before you create your encrypted DB instance.

If you use the AWS CLI `create-db-instance` command to create an encrypted RDS DB instance, set the `--kms-key-id` parameter to the Amazon Resource Name (ARN) for the AWS KMS encryption key for the DB instance. If you use the Amazon RDS API `CreateDBInstance` action, set the `KmsKeyId` parameter to the ARN for your AWS KMS key for the DB instance.

You can use the ARN of a key from another account to encrypt an RDS DB instance. Or you might create a DB instance with the same AWS account that owns the AWS KMS encryption key used to encrypt that new DB instance. In this case, the AWS KMS key ID that you pass can be the AWS KMS key alias instead of the key's ARN.

Important

If Amazon RDS loses access to the encryption key for a DB instance—for example, when RDS access to a key is revoked—then the encrypted DB instance goes into a terminal state. In this case, you can only restore the DB instance from a backup. We strongly recommend that you always enable backups for encrypted DB instances to guard against the loss of encrypted data in your databases.

Availability of Amazon RDS Encryption

Amazon RDS encryption is currently available for all database engines and storage types. Amazon RDS encryption is not currently available in the China (Beijing) region.

Amazon RDS encryption is available for most DB instance classes. The following table lists DB instance classes that *do not support* Amazon RDS encryption:

Instance Type	Instance Class
General Purpose (M1)	db.m1.small
	db.m1.medium
	db.m1.large
	db.m1.xlarge
Memory Optimized (M2)	db.m2.xlarge
	db.m2.2xlarge
	db.m2.4xlarge
Burst Capable (T2)	db.t2.micro

Note

Encryption at rest is not available for DB instances running SQL Server Express Edition.

Managing Amazon RDS Encryption Keys

You can manage keys used for Amazon RDS encrypted DB instances using the [AWS Key Management Service \(AWS KMS\)](#) in the IAM console. If you want full control over a key, then you must create a customer-managed key.

You can't delete, revoke, or rotate default keys provisioned by AWS KMS. You can't share a snapshot that has been encrypted using the default AWS KMS encryption key of the AWS account that shared the snapshot.

You can view audit logs of every action taken with a customer-managed key by using [AWS CloudTrail](#).

Important

If you disable the key for an encrypted DB instance, you cannot read from or write to that DB instance. When Amazon RDS encounters a DB instance encrypted by a key that Amazon RDS doesn't have access to, Amazon RDS puts the DB instance into a terminal state. In this state, the DB instance is no longer available and the current state of the database can't be recovered. To restore the DB instance, you must re-enable access to the encryption key for Amazon RDS, and then restore the DB instance from a backup.

Limitations of Amazon RDS Encrypted DB Instance

The following limitations exist for Amazon RDS encrypted DB instance:

- You can only enable encryption for an Amazon RDS DB instance when you create it, not after the DB instance is created.

However, because you can encrypt a copy of an unencrypted DB snapshot, you can effectively add encryption to an unencrypted DB instance. That is, you can create a snapshot of your DB instance, and then create an encrypted copy of that snapshot. You can then restore a DB instance from the encrypted snapshot, and thus you have an encrypted copy of your original DB instance. For more information, see [Copying a Snapshot \(p. 221\)](#).

- DB instances that are encrypted can't be modified to disable encryption.
- You can't have an encrypted Read Replica of an unencrypted DB instance or an unencrypted Read Replica of an encrypted DB instance.
- Encrypted Read Replicas must be encrypted with the same key as the source DB instance.
- You can't restore an unencrypted backup or snapshot to an encrypted DB instance.
- To copy an encrypted snapshot from one region to another, you must specify the KMS key identifier of the destination region. This is because KMS encryption keys are specific to the region that they are created in.

The source snapshot remains encrypted throughout the copy process. AWS Key Management Service uses envelope encryption to protect data during the copy process. For more information about envelope encryption, see [Envelope Encryption](#).

Using SSL to Encrypt a Connection to a DB Instance

You can use SSL from your application to encrypt a connection to a DB instance running MySQL, MariaDB, SQL Server, Oracle, or PostgreSQL. Each DB engine has its own process for implementing SSL. To learn how to implement SSL for your DB instance, use the link following that corresponds to your DB engine:

- [Using SSL with a MariaDB DB Instance \(p. 439\)](#)
- [Using SSL with a Microsoft SQL Server DB Instance \(p. 555\)](#)
- [Using SSL with a MySQL DB Instance \(p. 592\)](#)
- [Using SSL with an Oracle DB Instance \(p. 723\)](#)
- [Using SSL with a PostgreSQL DB Instance \(p. 1068\)](#)

A root certificate that works for all regions can be downloaded at <https://s3.amazonaws.com/rds-downloads/rds-ca-2015-root.pem>. It is the trusted root entity and should work in most cases but might

fail if your application doesn't accept certificate chains. If your application doesn't accept certificate chains, download the AWS Region-specific certificate from the list of intermediate certificates found later in this section. You can download a root certificate for the AWS GovCloud regions at <https://s3-us-gov-west-1.amazonaws.com/rds-downloads/rds-GovCloud-Root-CA-2017.pem>.

Note

All certificates are only available for download using SSL connections.

A certificate bundle that contains both the intermediate and root certificates can be downloaded at <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>.

A certificate bundle that contains both the intermediate and root certificates for the AWS GovCloud regions can be downloaded at <https://s3-us-gov-west-1.amazonaws.com/rds-downloads/rds-combined-ca-us-gov-bundle.pem>.

If your application is on the Microsoft Windows platform and requires a PKCS7 file, you can download the PKCS7 certificate bundle that contains both the intermediate and root certificates at <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.p7b>.

Intermediate Certificates

You might need to use an intermediate certificate to connect to your region. For example, you must use an intermediate certificate to connect to the AWS GovCloud (US-West) region using SSL. If you need an intermediate certificate for a particular AWS Region, download the certificate from the following list:

[Asia Pacific \(Mumbai\)](#)

[Asia Pacific \(Tokyo\)](#)

[Asia Pacific \(Seoul\)](#)

[Asia Pacific \(Osaka-Local\)](#)

[Asia Pacific \(Singapore\)](#)

[Asia Pacific \(Sydney\)](#)

[Canada \(Central\)](#)

[China \(Beijing\)](#)

[China \(Ningxia\)](#)

[EU \(Frankfurt\)](#)

[EU \(Ireland\)](#)

[EU \(London\)](#)

[EU \(Paris\)](#)

[EU \(Stockholm\)](#)

[South America \(São Paulo\)](#)

[US East \(N. Virginia\)](#)

[US East \(Ohio\)](#)

[US West \(N. California\)](#)

[US West \(Oregon\)](#)

[AWS GovCloud \(US-East\) \(CA-2017\)](#)

[AWS GovCloud \(US-West\) \(CA-2017\)](#)

[AWS GovCloud \(US-West\) \(CA-2012\)](#)

Controlling Access with Security Groups

Security groups control the access that traffic has in and out of a DB instance. Three types of security groups are used with Amazon RDS: DB security groups, VPC security groups, and Amazon EC2 security groups. In simple terms, these work as follows:

- A DB security group controls access to EC2-Classic DB instances that are not in a VPC.
- A VPC security group controls access to DB instances and EC2 instances inside a VPC.
- An EC2 security group controls access to an EC2 instance.

By default, network access is turned off to a DB instance. You can specify rules in a security group that allows access from an IP address range, port, or EC2 security group. Once ingress rules are configured, the same rules apply to all DB instances that are associated with that security group. You can specify up to 20 rules in a security group.

DB Security Groups

DB security groups are used with DB instances that are not in a VPC and on the EC2-Classic platform. Each DB security group rule enables a specific source to access a DB instance that is associated with that DB security group. The source can be a range of addresses (for example, 203.0.113.0/24), or an EC2 security group. When you specify an EC2 security group as the source, you allow incoming traffic from all EC2 instances that use that EC2 security group. DB security group rules apply to inbound traffic only; outbound traffic is not currently permitted for DB instances.

You don't need to specify a destination port number when you create DB security group rules. The port number defined for the DB instance is used as the destination port number for all rules defined for the DB security group. DB security groups can be created using the Amazon RDS API actions or the Amazon RDS page of the AWS Management Console.

For more information about working with DB security groups, see [Working with DB Security Groups \(EC2-Classic Platform\) \(p. 399\)](#).

VPC Security Groups

Each VPC security group rule enables a specific source to access a DB instance in a VPC that is associated with that VPC security group. The source can be a range of addresses (for example, 203.0.113.0/24), or another VPC security group. By specifying a VPC security group as the source, you allow incoming traffic from all instances (typically application servers) that use the source VPC security group. VPC security groups can have rules that govern both inbound and outbound traffic, though the outbound traffic rules typically do not apply to DB instances. Outbound traffic rules only apply if the DB instance acts as a client. For example, outbound traffic rules apply to an Oracle DB instance with outbound database links. You must use the [Amazon EC2 API](#) or the **Security Group** option on the VPC Console to create VPC security groups.

When you create rules for your VPC security group that allow access to the instances in your VPC, you must specify a port for each range of addresses that the rule allows access for. For example, if you want to enable SSH access to instances in the VPC, then you create a rule allowing access to TCP port 22 for the specified range of addresses.

You can configure multiple VPC security groups that allow access to different ports for different instances in your VPC. For example, you can create a VPC security group that allows access to TCP port 80 for web servers in your VPC. You can then create another VPC security group that allows access to TCP port 3306 for RDS MySQL DB instances in your VPC.

For more information on VPC security groups, see [Security Groups](#) in the *Amazon Virtual Private Cloud User Guide*.

DB Security Groups vs. VPC Security Groups

The following table shows the key differences between DB security groups and VPC security groups.

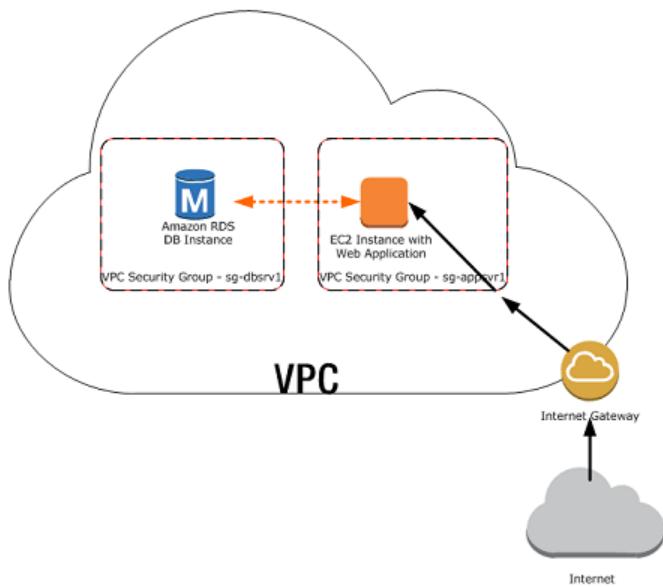
DB Security Group	VPC Security Group
Controls access to DB instances outside a VPC.	Controls access to DB instances in VPC.
Uses Amazon RDS API actions or the Amazon RDS page of the AWS Management Console to create and manage group and rules.	Uses Amazon EC2 API actions or the Amazon VPC page of the AWS Management Console to create and manage group and rules.
When you add a rule to a group, you don't need to specify port number or protocol.	When you add a rule to a group, specify the protocol as TCP. In addition, specify the same port number that you used to create the DB instances (or options) that you plan to add as members to the group.
Groups allow access from EC2 security groups in your AWS account or other accounts.	Groups allow access from other VPC security groups in your VPC only.

Security Group Scenario

A common use of an RDS instance in a VPC is to share data with an application server running in an Amazon EC2 instance in the same VPC, which is accessed by a client application outside the VPC. For this scenario, you use the RDS and VPC pages on the AWS Management Console or the RDS and EC2 API actions to create the necessary instances and security groups:

1. Create a VPC security group (for example, sg-appsrv1) and define inbound rules that use the IP addresses of the client application as the source. This security group allows your client application to connect to EC2 instances in a VPC that uses this security group.
2. Create an EC2 instance for the application and add the EC2 instance to the VPC security group (sg-appsrv1) that you created in the previous step. The EC2 instance in the VPC shares the VPC security group with the DB instance.
3. Create a second VPC security group (for example, sg-dbsrv1) and create a new rule by specifying the VPC security group that you created in step 1 (sg-appsrv1) as the source.
4. Create a new DB instance and add the DB instance to the VPC security group (sg-dbsrv1) that you created in the previous step. When you create the instance, use the same port number as the one specified for the VPC security group (sg-dbsrv1) rule that you created in step 3.

The following diagram shows this scenario.



For more information about using a VPC, see [Amazon Virtual Private Cloud \(VPCs\)](#) and [Amazon RDS](#) (p. 412).

Creating a VPC Security Group

You can create a VPC security group for a DB instance by using the VPC console. For information about creating a security group, see [Provide Access to Your DB Instance in Your VPC by Creating a Security Group \(p. 8\)](#) and [Security Groups in the Amazon Virtual Private Cloud User Guide](#).

Associating a Security Group with a DB Instance

You can associate a security group with a DB instance by using **Modify** on the RDS console, the `ModifyDBInstance` Amazon RDS API, or the `modify-db-instance` AWS CLI command.

For information about modifying a DB instance, see [Modifying an Amazon RDS DB Instance \(p. 115\)](#). For security group considerations when you restore a DB instance from a DB snapshot, see [Security Group Considerations \(p. 218\)](#).

Deleting DB VPC Security Groups

DB VPC security groups are an RDS mechanism to synchronize security information with a VPC security group. However, this synchronization is no longer required, because RDS has been updated to use VPC security group information directly.

Note

DB VPC security groups are deprecated, and they are different from DB security groups, VPC security groups, and EC2 security groups.

We strongly recommend that you delete any DB VPC security groups that you currently use. If you don't delete your DB VPC security groups, you might encounter unintended behaviors with your RDS DB instances, which can be as severe as losing access to a DB instance. The unintended behaviors are a result of an action such as an update to a DB instance, an option group, or similar. Such updates cause RDS to resynchronize the DB VPC security group with the VPC security group. This resynchronization can result in your security information being overwritten with incorrect and outdated security information. This result can have a severe impact on your access to your RDS DB instances.

How Can I Determine If I Have a DB VPC Security Group?

Because DB VPC security groups have been deprecated, they don't appear in the RDS console. However, you can call the [describe-db-security-groups](#) AWS CLI command or the [DescribeDBSecurityGroups](#) API action to determine if you have any DB VPC security groups.

In this case, you can call the `describe-db-security-groups` AWS CLI command with JSON specified as the output format. If you do, you can identify DB VPC security groups by the VPC identifier on the second line of the output for the security group as shown in the following example.

```
{  
    "DBSecurityGroups": [  
        {  
            "VpcId": "vpc-abcd1234",  
            "DBSecurityGroupDescription": "default:vpc-abcd1234",  
            "IPRanges": [  
                {  
                    "Status": "authorized",  
                    "CIDRIP": "xxx.xxx.xxx.xxx/n"  
                },  
                {  
                    "Status": "authorized",  
                    "CIDRIP": "xxx.xxx.xxx.xxx/n"  
                }  
            ],  
            "OwnerId": "123456789012",  
            "EC2SecurityGroups": [],  
            "DBSecurityGroupName": "default:vpc-abcd1234"  
        }  
    ]  
}
```

If you run the `DescribeDBSecurityGroups` API action, then you can identify DB VPC security groups using the `<VpcId>` response element as shown in the following example.

```
<DBSecurityGroup>  
    <EC2SecurityGroups/>  
    <DBSecurityGroupDescription>default:vpc-abcd1234</DBSecurityGroupDescription>  
    <IPRanges>  
        <IPRange>  
            <CIDRIP>xxx.xxx.xxx.xxx/n</CIDRIP>  
            <Status>authorized</Status>  
        </IPRange>  
        <IPRange>  
            <CIDRIP>xxx.xxx.xxx.xxx/n</CIDRIP>  
            <Status>authorized</Status>  
        </IPRange>  
    </IPRanges>  
    <VpcId>vpc-abcd1234</VpcId>  
    <OwnerId>123456789012</OwnerId>  
    <DBSecurityGroupName>default:vpc-abcd1234</DBSecurityGroupName>  
</DBSecurityGroup>
```

How Do I Delete a DB VPC Security Group?

Because DB VPC security groups don't appear in the RDS console, you must call the [delete-db-security-group](#) AWS CLI command or the [DeleteDBSecurityGroup](#) API action to delete a DB VPC security group.

After you delete a DB VPC security group, your DB instances in your VPC continue to be secured by the VPC security group for that VPC. The DB VPC security group that was deleted was merely a copy of the VPC security group information.

Review Your AWS CloudFormation Templates

Older versions of AWS CloudFormation templates can contain instructions to create a DB VPC security group. Because DB VPC security groups are not yet fully deprecated, they can still be created. Make sure that any AWS CloudFormation templates that you use to provision a DB instance with security settings don't also create a DB VPC security group. Don't use AWS CloudFormation templates that create an RDS `DBSecurityGroup` with an `EC2VpcId` as shown in the following example.

```
"DbSecurityByEC2SecurityGroup" : {
    "Type" : "AWS::RDS::DBSecurityGroup",
    "Properties" : {
        "GroupDescription" : "Ingress for Amazon EC2 security group",
        "EC2VpcId" : { "MyVPC" },
        "DBSecurityGroupIngress" : [ {
            "EC2SecurityGroupId" : "sg-b0ff1111",
            "EC2SecurityGroupOwnerId" : "111122223333"
        }, {
            "EC2SecurityGroupId" : "sg-ffd72222",
            "EC2SecurityGroupOwnerId" : "111122223333"
        } ]
    }
}
```

Instead, add security information for your RDS DB instances in a VPC using VPC security groups, as shown in the following example.

```
"DBInstance" : {
    "Type": "AWS::RDS::DBInstance",
    "Properties": {
        "DBName" : { "Ref" : "DBName" },
        "Engine" : "MySQL",
        "MultiAZ" : { "Ref": "MultiAZDatabase" },
        "MasterUsername" : { "Ref" : "<master_username>" },
        "DBInstanceClass" : { "Ref" : "DBCClass" },
        "AllocatedStorage" : { "Ref" : "DBAllocatedStorage" },
        "MasterUserPassword": { "Ref" : "<master_password>" },
        "VPCSecurityGroups" : [ { "Fn::GetAtt": [ "VPCSecurityGroup", "GroupId" ] } ]
    }
}
```

Working with DB Security Groups (EC2-Classic Platform)

By default, network access is turned off to a DB instance. You can specify rules in a *security group* that allows access from an IP address range, port, or EC2 security group. Once ingress rules are configured, the same rules apply to all DB instances that are associated with that security group. You can specify up to 20 rules in a security group.

Amazon RDS supports two different kinds of security groups. The one you use depends on which Amazon RDS platform you are on:

- **VPC security groups** – for the EC2-VPC platform.
- **DB security groups** – for the EC2-Classic platform.

You are most likely on the EC2-VPC platform (and must use VPC security groups) if any of the following are true:

- If you are a new Amazon RDS customer.
- If you have never created a DB instance before.
- If you are creating a DB instance in an AWS Region you have not used before.

Otherwise, if you are on the EC2-Classic platform, you use DB security groups to manage access to your Amazon RDS DB instances. For more information about the differences between DB security groups and VPC security groups, see [Controlling Access with Security Groups \(p. 394\)](#).

Note

To determine which platform you are on, see [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#).

If you are on the EC2-VPC platform, you must use VPC security groups instead of DB security groups. For more information about using a VPC, see [Amazon Virtual Private Cloud \(VPCs\) and Amazon RDS \(p. 412\)](#).

Topics

- [Creating a DB Security Group \(p. 399\)](#)
- [Listing Available DB Security Groups \(p. 401\)](#)
- [Viewing a DB Security Group \(p. 401\)](#)
- [Associating a DB Security Group with a DB Instance \(p. 402\)](#)
- [Authorizing Network Access to a DB Security Group from an IP Range \(p. 402\)](#)
- [Authorizing Network Access to a DB Instance from an Amazon EC2 Instance \(p. 404\)](#)
- [Revoking Network Access to a DB Instance from an IP Range \(p. 405\)](#)

Creating a DB Security Group

To create a DB security group, you need to provide a name and a description.

AWS Management Console

To create a DB security group

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. Choose **Security Groups** in the navigation pane on the left side of the window.

Note

If you are on the EC2-VPC platform, the **Security Groups** option does not appear in the navigation pane. In this case, you must use VPC security groups instead of DB security groups. For more information about using a VPC, see [Amazon Virtual Private Cloud \(VPCs\) and Amazon RDS \(p. 412\)](#).

3. Choose **Create DB Security Group**.
4. Type the name and description of the new DB security group in the **Name** and **Description** text boxes. The security group name can't contain spaces and can't start with a number.
5. Choose **Yes, Create**.

The DB security group is created.

A newly created DB security group doesn't provide access to a DB instance by default. You must specify a range of IP addresses or an Amazon EC2 security group that can have access to the DB instance. To specify IP addresses or an Amazon EC2 security group for a DB security group, see [Authorizing Network Access to a DB Security Group from an IP Range \(p. 402\)](#).

CLI

To create a DB security group, use the AWS CLI command `create-db-security-group`.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-security-group \
--db-security-group-name mydbsecuritygroup \
--db-security-group-description "My new security group"
```

For Windows:

```
aws rds create-db-security-group ^
--db-security-group-name mydbsecuritygroup ^
--db-security-group-description "My new security group"
```

A newly created DB security group doesn't provide access to a DB instance by default. You must specify a range of IP addresses or an Amazon EC2 security group that can have access to the DB instance. To specify IP addresses or an Amazon EC2 security group for a DB security group, see [Authorizing Network Access to a DB Security Group from an IP Range \(p. 402\)](#).

API

To create a DB security group, call the Amazon RDS function `CreateDBSecurityGroup` with the following parameters:

- `DBSecurityGroupName` = *mydbsecuritygroup*
- `Description` = *"My new security group"*

Example

```
https://rds.amazonaws.com/
?Action=CreateDBSecurityGroup
&DBSecurityGroupName=mydbsecuritygroup
&Description=My%20new%20db%20security%20group
&Version=2012-01-15
&SignatureVersion=2
```

```
&SignatureMethod=HmacSHA256
&Timestamp=2012-01-20T22%3A06%3A23.624Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

A newly created DB security group doesn't provide access to a DB instance by default. You must specify a range of IP addresses or an Amazon EC2 security group that can have access to the DB instance. To specify IP addresses or an Amazon EC2 security group for a DB security group, see [Authorizing Network Access to a DB Security Group from an IP Range \(p. 402\)](#).

Listing Available DB Security Groups

You can list which DB security groups have been created for your AWS account.

AWS Management Console

To list all available DB security groups for an AWS account

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. Choose **Security Groups** in the navigation pane on the left side of the window.

The available DB security groups appear in the **DB Security Groups** list.

CLI

To list all available DB security groups for an AWS account, Use the AWS CLI command `describe-db-security-groups` with no parameters.

Example

```
aws rds describe-db-security-groups
```

API

To list all available DB security groups for an AWS account, call `DescribeDBSecurityGroups` with no parameters.

Example

```
https://rds.amazonaws.com/
?Action=DescribeDBSecurityGroups
&MaxRecords=100
&Version=2009-10-16
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Viewing a DB Security Group

You can view detailed information about your DB security group to see what IP ranges have been authorized.

AWS Management Console

To view properties of a specific DB security group

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.

2. Choose **Security Groups** in the navigation pane on the left side of the window.
3. Select the details icon for the DB security group you want to view. The detailed information for the DB security group is displayed.

CLI

To view the properties of a specific DB security group use the AWS CLI `describe-db-security-groups`. Specify the DB security group you want to view.

Example

For Linux, OS X, or Unix:

```
aws rds describe-db-security-groups \
--db-security-group-name mydbsecuritygroup
```

For Windows:

```
aws rds describe-db-security-groups ^
--db-security-group-name mydbsecuritygroup
```

API

To view properties of a specific DB security group, call `DescribeDBSecurityGroups` with the following parameters:

- `DBSecurityGroupName=mydbsecuritygroup`

Example

```
https://rds.amazonaws.com/
?Action=DescribeDBSecurityGroups
&DBSecurityGroupName=mydbsecuritygroup
&Version=2009-10-16
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-16T22%3A23%3A07.107Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Associating a DB Security Group with a DB Instance

You can associate a DB security group with a DB instance using the RDS console's **Modify** option, the `ModifyDBInstance` Amazon RDS API, or the AWS CLI `modify-db-instance` command.

For information about modifying a DB instance, see [Modifying an Amazon RDS DB Instance \(p. 115\)](#).

Authorizing Network Access to a DB Security Group from an IP Range

By default, network access is turned off to a DB instance. If you want to access a DB instance that is not in a VPC, you must set access rules for a DB security group to allow access from specific EC2 security groups or CIDR IP ranges. You then must associate that DB instance with that DB security group. This

process is called *ingress*. Once ingress is configured for a DB security group, the same ingress rules apply to all DB instances associated with that DB security group.

Warning

Talk with your network administrator if you are intending to access a DB instance behind a firewall to determine the IP addresses you should use.

In following example, you configure a DB security group with an ingress rule for a CIDR IP range.

AWS Management Console

To configure a DB security group with an ingress rule for a CIDR IP range

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. Select **Security Groups** from the navigation pane on the left side of the console window.
3. Select the details icon for the DB security group you want to authorize.
4. In the details page for your security group, select **CIDR/IP** from the **Connection Type** drop-down list, type the CIDR range for the ingress rule you want to add to this DB security group into the **CIDR** text box, and choose **Authorize**.

Tip

The AWS Management Console displays a CIDR IP based on your connection below the CIDR text field. If you are not accessing the DB instance from behind a firewall, you can use this CIDR IP.

5. The status of the ingress rule is **authorizing** until the new ingress rule has been applied to all DB instances that are associated with the DB security group that you modified. After the ingress rule has been successfully applied, the status changes to **authorized**.

CLI

To configure a DB security group with an ingress rule for a CIDR IP range, use the AWS CLI command [authorize-db-security-group-ingress](#).

Example

For Linux, OS X, or Unix:

```
aws rds authorize-db-security-group-ingress \
--db-security-group-name mydbsecuritygroup \
--cidrip 192.168.1.10/27
```

For Windows:

```
aws rds authorize-db-security-group-ingress ^
--db-security-group-name mydbsecuritygroup ^
--cidrip 192.168.1.10/27
```

The command should produce output similar to the following.

```
SECURITYGROUP  mydbsecuritygroup  My new DBSecurityGroup
IP-RANGE      192.168.1.10/27  authorizing
```

API

To configure a DB security group with an ingress rule for a CIDR IP range, call the Amazon RDS API [AuthorizedDBSecurityGroupIngress](#) with the following parameters:

- DBSecurityGroupName = *mydbsecuritygroup*
- CIDRIP = *192.168.1.10/27*

Example

```
https://rds.amazonaws.com/
?Action=AuthorizeDBSecurityGroupIngress
&CIDRIP=192.168.1.10%2F27
&DBSecurityGroupName=mydbsecuritygroup
&Version=2009-10-16
&Action=AuthorizeDBSecurityGroupIngress
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-22T17%3A10%3A50.274Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Authorizing Network Access to a DB Instance from an Amazon EC2 Instance

If you want to access your DB instance from an Amazon EC2 instance, you must first determine if your EC2 instance and DB instance are in a VPC. If you are using a default VPC, you can assign the same EC2 or VPC security group that you used for your EC2 instance when you create or modify the DB instance that the EC2 instance accesses.

If your DB instance and EC2 instance are not in a VPC, you must configure the DB instance's security group with an ingress rule that allows traffic from the Amazon EC2 instance. You do this by adding the Amazon EC2 security group for the EC2 instance to the DB security group for the DB instance. In this example, you add an ingress rule to a DB security group for an Amazon EC2 security group.

Important

- Adding an ingress rule to a DB security group for an Amazon EC2 security group only grants access to your DB instances from Amazon EC2 instances associated with that Amazon EC2 security group.
- You can't authorize an Amazon EC2 security group that is in a different AWS Region than your DB instance. You can authorize an IP range, or specify an Amazon EC2 security group in the same AWS Region that refers to IP address in another AWS Region. If you specify an IP range, we recommend that you use the private IP address of your Amazon EC2 instance, which provides a more direct network route from your Amazon EC2 instance to your Amazon RDS DB instance, and doesn't incur network charges for data sent outside of the Amazon network.

AWS Management Console

To add an EC2 security group to a DB security group

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. From the navigation pane, choose **Security Groups**.
3. Select the details icon for the DB security group you want to grant access.
4. In the details page for your security group, choose **EC2 Security Group** for **Connection Type**, and then select the Amazon EC2 security group you want to use. Then choose **Authorize**.
5. The status of the ingress rule is **authorizing** until the new ingress rule has been applied to all DB instances that are associated with the DB security group that you modified. After the ingress rule has been successfully applied, the status changes to **authorized**.

CLI

To grant access to an Amazon EC2 security group, use the AWS CLI command `aws rds authorize-db-security-group-ingress`.

Example

For Linux, OS X, or Unix:

```
aws rds authorize-db-security-group-ingress \
    --db-security-group-name default \
    --ec2-security-group-name myec2group \
    --ec2-security-group-owner-id 987654321021
```

For Windows:

```
aws rds authorize-db-security-group-ingress ^
    --db-security-group-name default ^
    --ec2-security-group-name myec2group ^
    --ec2-security-group-owner-id 987654321021
```

The command should produce output similar to the following:

```
SECURITYGROUP  Name      Description
SECURITYGROUP  default   default
EC2-SECURITYGROUP  myec2group  987654321021  authorizing
```

API

To authorize network access to an Amazon EC2 security group, call that Amazon RDS API function, https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_AuthorizeDBSecurityGroupIngress.html with the following parameters:

- `EC2SecurityGroupName = myec2group`
- `EC2SecurityGroupOwnerId = 987654321021`

Example

```
https://rds.amazonaws.com/
    ?Action=AuthorizeDBSecurityGroupIngress
    &EC2SecurityGroupOwnerId=987654321021
    &EC2SecurityGroupName=myec2group
    &Version=2009-10-16
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2009-10-22T17%3A10%3A50.274Z
    &AWSAccessKeyId=<AWS Access Key ID>
    &Signature=<Signature>
```

Revoking Network Access to a DB Instance from an IP Range

You can easily revoke network access from a CIDR IP range to DB instances belonging to a DB security group by revoking the associated CIDR IP ingress rule.

In this example, you revoke an ingress rule for a CIDR IP range on a DB security group.

AWS Management Console

To revoke an ingress rule for a CIDR IP range on a DB Security Group

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. From the navigation pane, choose **Security Groups**.
3. Select the details icon for the DB security group that has the ingress rule you want to revoke.
4. In the details page for your security group, choose **Remove** next to the ingress rule you want to revoke.
5. The status of the ingress rule is **revoking** until the ingress rule has been removed from all DB instances that are associated with the DB security group that you modified. After the ingress rule has been successfully removed, the ingress rule is removed from the DB security group.

CLI

To revoke an ingress rule for a CIDR IP range on a DB security group, use the AWS CLI command `revoke-db-security-group-ingress`.

Example

For Linux, OS X, or Unix:

```
aws rds revoke-db-security-group-ingress \
--db-security-group-name mydbsecuritygroup \
--cidrip 192.168.1.1/27
```

For Windows:

```
aws rds revoke-db-security-group-ingress ^
--db-security-group-name mydbsecuritygroup ^
--cidrip 192.168.1.1/27
```

The command should produce output similar to the following.

```
SECURITYGROUP mydbsecuritygroup My new DBSecurityGroup
IP-RANGE 192.168.1.1/27 revoking
```

API

To revoke an ingress rule for a CIDR IP range on a DB security group, call the Amazon RDS API action https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_RevokeDBSecurityGroupIngress.html with the following parameters:

- `DBSecurityGroupName = mydbsecuritygroup`
- `CIDRIP = 192.168.1.10/27`

Example

```
https://rds.amazonaws.com/
```

```
?Action=RevokeDBSecurityGroupIngress
&DBSecurityGroupName=mydbsecuritygroup
&CIDRIP=192.168.1.10%2F27
&Version=2009-10-16
&SignatureVersion=2&SignatureMethod=HmacSHA256
&Timestamp=2009-10-22T22%3A32%3A12.515Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Master User Account Privileges

When you create a new DB instance, the default master user that you use gets certain privileges for that DB instance. The following table shows the privileges the master user gets for each of the database engines.

Important

We strongly recommend that you do not use the master user directly in your applications. Instead, adhere to the best practice of using a database user created with the minimal privileges required for your application.

Note

If you accidentally delete the permissions for the master user, you can restore them by modifying the DB instance and setting a new master user password. For more information about modifying a DB instance, see [Modifying an Amazon RDS DB Instance \(p. 115\)](#).

Database Engine	System Privilege	Role
MySQL and MariaDB	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER ON *.* WITH GRANT OPTION, REPLICATION SLAVE (only for Amazon RDS MySQL versions 5.6, 5.7 and 8.0, Amazon RDS MariaDB)	—
PostgreSQL	CREATE ROLE, CREATE DB, PASSWORD VALID UNTIL INFINITY, CREATE EXTENSION, ALTER EXTENSION, DROP EXTENSION, CREATE TABLESPACE, ALTER < OBJECT> OWNER, CHECKPOINT, PG_CANCEL_BACKEND(), PG_TERMINATE_BACKEND(), SELECT PG_STAT_REPLICATION, EXECUTE PG_STAT_STATEMENTS_RESET(), OWN POSTGRES_FDW_HANDLER(), OWN POSTGRES_FDW_VALIDATOR(), OWN POSTGRES_FDW, EXECUTE PG_BUFFERCACHE_PAGES(), SELECT PG_BUFFERCACHE	RDS_SUPERUSER
Oracle	ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, DROP ANY DIRECTORY, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, GRANT ANY OBJECT PRIVILEGE, RESTRICTED SESSION, EXEMPT REDACTION POLICY	AQ_ADMINISTRATOR_ROLE, AQ_USER_ROLE, CONNECT, CTXAPP, DBA, EXECUTE_CATALOG_ROLE, RECOVERY_CATALOG_OWNER, RESOURCE, SELECT_CATALOG_ROLE

Database Engine	System Privilege	Role
Microsoft SQL Server	ALTER ANY CONNECTION, ALTER ANY LINKED SERVER, ALTER ANY LOGIN, ALTER SERVER STATE, ALTER TRACE, CONNECT SQL, CREATE ANY DATABASE, VIEW ANY DATABASE, VIEW ANY DEFINITION, VIEW SERVER STATE, ALTER ANY SERVER ROLE, ALTER ANY USER	DB_OWNER (Database Level Role) PROCESSADMIN(Server Level Role) SETUPADMIN(Server Level Role) SQLAgentUserRole(Server Level Role)

Using Service-Linked Roles for Amazon RDS

Amazon RDS uses AWS Identity and Access Management (IAM) [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to Amazon RDS. Service-linked roles are predefined by Amazon RDS and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes using Amazon RDS easier because you don't have to manually add the necessary permissions. Amazon RDS defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon RDS can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete the roles only after first deleting their related resources. This protects your Amazon RDS resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see [AWS Services That Work with IAM](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-Linked Role Permissions for Amazon RDS

Amazon RDS uses the service-linked role named **AWSServiceRoleForRDS** – to allow Amazon RDS to call AWS services on behalf of your DB instances.

The AWSServiceRoleForRDS service-linked role trusts the following services to assume the role:

- `rds.amazonaws.com`

The role permissions policy allows Amazon RDS to complete the following actions on the specified resources:

- Actions on `ec2`:
 - `AssignPrivateIpAddresses`
 - `AuthorizeSecurityGroupIngress`
 - `CreateNetworkInterface`
 - `CreateSecurityGroup`
 - `DeleteNetworkInterface`
 - `DeleteSecurityGroup`
 - `DescribeAvailabilityZones`
 - `DescribeInternetGateways`
 - `DescribeSecurityGroups`
 - `DescribeSubnets`
 - `DescribeVpcAttribute`
 - `DescribeVpcs`
 - `ModifyNetworkInterfaceAttribute`
 - `RevokeSecurityGroupIngress`
 - `UnassignPrivateIpAddresses`
- Actions on `sns`:
 - `ListTopic`
 - `Publish`

- Actions on `cloudwatch:`
 - `PutMetricData`
 - `GetMetricData`
 - `CreateLogStream`
 - `PullLogEvents`
 - `DescribeLogStreams`
 - `CreateLogGroup`

Note

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. If you encounter the following error message:

**Unable to create the resource. Verify that you have permission to create service linked role.
Otherwise wait and try again later.**

Make sure you have the following permissions enabled:

```
{  
    "Action": "iam:CreateServiceLinkedRole",  
    "Effect": "Allow",  
    "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/  
AWSServiceRoleForRDS",  
    "Condition": {  
        "StringLike": {  
            "iam:AWSServiceName": "rds.amazonaws.com"  
        }  
    }  
}
```

For more information, see [Service-Linked Role Permissions](#) in the *IAM User Guide*.

Creating a Service-Linked Role for Amazon RDS

You don't need to manually create a service-linked role. When you create a DB instance, Amazon RDS creates the service-linked role for you.

Important

If you were using the Amazon RDS service before December 1, 2017, when it began supporting service-linked roles, then Amazon RDS created the `AWSServiceRoleForRDS` role in your account.

To learn more, see [A New Role Appeared in My IAM Account](#).

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create a DB instance, Amazon RDS creates the service-linked role for you again.

Editing a Service-Linked Role for Amazon RDS

Amazon RDS does not allow you to edit the `AWSServiceRoleForRDS` service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see [Editing a Service-Linked Role](#) in the *IAM User Guide*.

Deleting a Service-Linked Role for Amazon RDS

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or

maintained. However, you must delete all of your DB instances before you can delete the service-linked role.

Cleaning Up a Service-Linked Role

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no active sessions and remove any resources used by the role.

To check whether the service-linked role has an active session in the IAM console

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane of the IAM console, choose **Roles**. Then choose the name (not the check box) of the AWSServiceRoleForRDS role.
3. On the **Summary** page for the chosen role, choose the **Access Advisor** tab.
4. On the **Access Advisor** tab, review recent activity for the service-linked role.

Note

If you are unsure whether Amazon RDS is using the AWSServiceRoleForRDS role, you can try to delete the role. If the service is using the role, then the deletion fails and you can view the AWS Regions where the role is being used. If the role is being used, then you must wait for the session to end before you can delete the role. You cannot revoke the session for a service-linked role.

If you want to remove the AWSServiceRoleForRDS role, you must first delete *all* of your DB instances .

Deleting All of Your Instances

Use one of these procedures to delete each of your instances.

To delete an instance (console)

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**.
3. Choose the instance that you want to delete.
4. For **Actions**, choose **Delete**.
5. If you are prompted for **Create final Snapshot?**, choose **Yes** or **No**.
6. If you chose **Yes** in the previous step, for **Final snapshot name** enter the name of your final snapshot.
7. Choose **Delete**.

To delete an instance (CLI)

See [delete-db-instance](#) in the *AWS CLI Command Reference*.

To delete an instance (API)

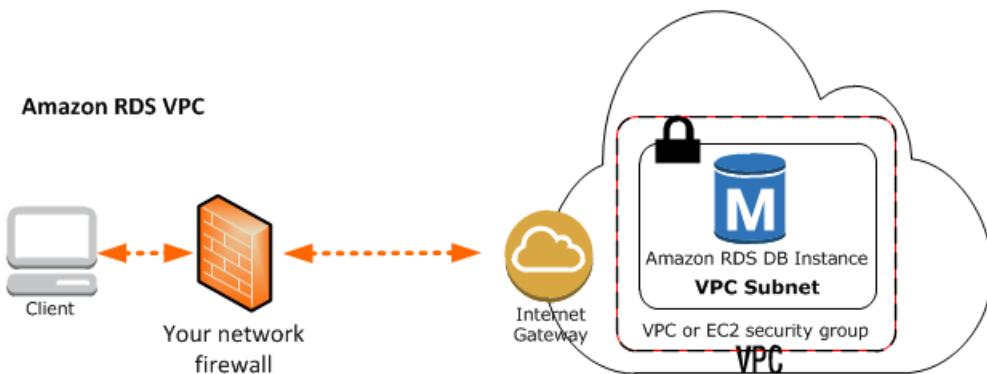
See [DeleteDBInstance](#) in the *Amazon RDS API Reference*.

You can use the IAM console, the IAM CLI, or the IAM API to delete the AWSServiceRoleForRDS service-linked role. For more information, see [Deleting a Service-Linked Role](#) in the *IAM User Guide*.

Amazon Virtual Private Cloud (VPCs) and Amazon RDS

There are two Amazon Elastic Compute Cloud (EC2) platforms that host Amazon RDS DB instances, *EC2-VPC* and *EC2-Classic*. Amazon Virtual Private Cloud (Amazon VPC) lets you launch AWS resources, such as Amazon RDS DB instances, into a virtual private cloud (VPC).

When you use an Amazon VPC, you have control over your virtual networking environment: you can select your own IP address range, create subnets, and configure routing and access control lists. The basic functionality of Amazon RDS is the same whether your DB instance is running in an Amazon VPC or not: Amazon RDS manages backups, software patching, automatic failure detection, and recovery. There is no additional cost to run your DB instance in Amazon VPC.



Accounts that support only the *EC2-VPC* platform have a default VPC. All new DB instances are created in the default VPC unless you specify otherwise. If you are a new Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in a region you have not used before, you are most likely on the *EC2-VPC* platform and have a default VPC.

Some legacy DB instances on the *EC2-Classic* platform are not in a VPC. The legacy *EC2-Classic* platform does not have a default VPC, but as is true for either platform, you can create your own VPC and specify that a DB instance be located in that VPC.

Topics

- [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#)
- [Scenarios for Accessing a DB Instance in a VPC \(p. 414\)](#)
- [Working with an Amazon RDS DB Instance in a VPC \(p. 420\)](#)
- [Updating the VPC for a DB Instance \(p. 425\)](#)
- [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#)

This documentation only discusses VPC functionality relevant to Amazon RDS DB instances. For more information about Amazon VPC, see [Amazon VPC Getting Started Guide](#) and [Amazon VPC User Guide](#).

Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform

Your AWS account and the region you select determines which of the two RDS platforms your DB instance is created on: *EC2-Classic* or *EC2-VPC*. The type of platform determines if you have a default

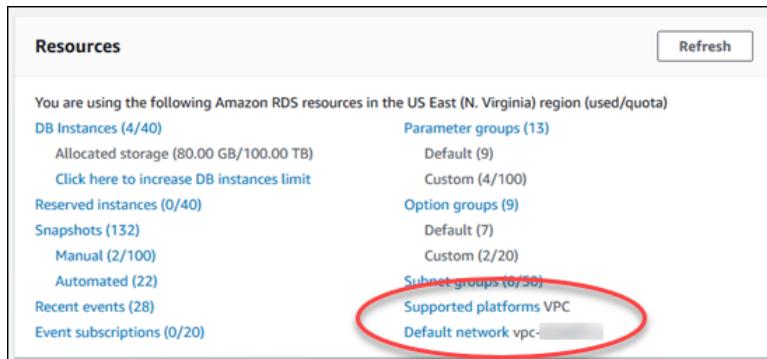
VPC, and which type of security group you use to provide access to your DB instance. The legacy *EC2-Classic* platform is the original platform used by Amazon RDS; if you are on this platform and want to use a VPC, you must create the VPC using the Amazon VPC console or Amazon VPC API. Accounts that only support the *EC2-VPC* platform have a default VPC where all DB instances are created, and you must use either an EC2 or VPC security group to provide access to the DB instance.

Note

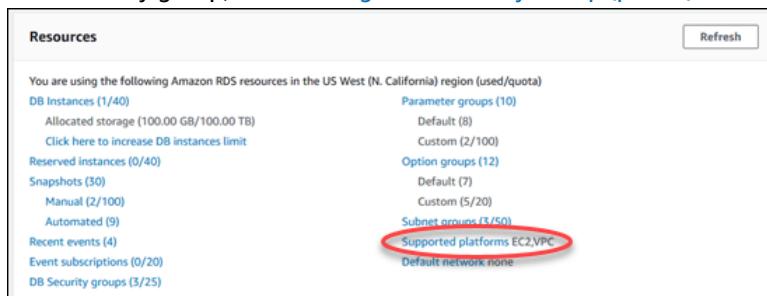
If you are a new Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in a region you have not used before, in almost all cases you are on the *EC2-VPC* platform and have a default VPC.

You can tell which platform your AWS account in a given region is using by looking at the dashboard on the RDS console or EC2 console. If you are a new Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in a region you have not used before, you might be redirected to the first-run console page and will not see the home page following.

If **Supported Platforms** indicates VPC, as shown following, your AWS account in the current region uses the *EC2-VPC* platform, and uses a default VPC. The name of the default VPC is shown below the supported platform. To provide access to a DB instance created on the *EC2-VPC* platform, you must create a VPC security group. For information about creating a VPC security group, see [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#).



If **Supported Platforms** indicates EC2 , VPC, as shown following, your AWS account in the current region uses the *EC2-Classic* platform, and you do not have a default VPC. To provide access to a DB instance created on the *EC2-Classic* platform, you must create a DB security group. For information about creating a DB security group, see [Creating a DB Security Group \(p. 399\)](#).



Note

- You can create a VPC on the *EC2-Classic* platform, but one is not created for you by default as it is on accounts that support the *EC2-VPC* platform.
- If you are interested in moving an existing DB instance into a VPC, you can use the AWS Management Console to do it easily. For more information. see [Moving a DB Instance Not in a VPC into a VPC \(p. 426\)](#).

Scenarios for Accessing a DB Instance in a VPC

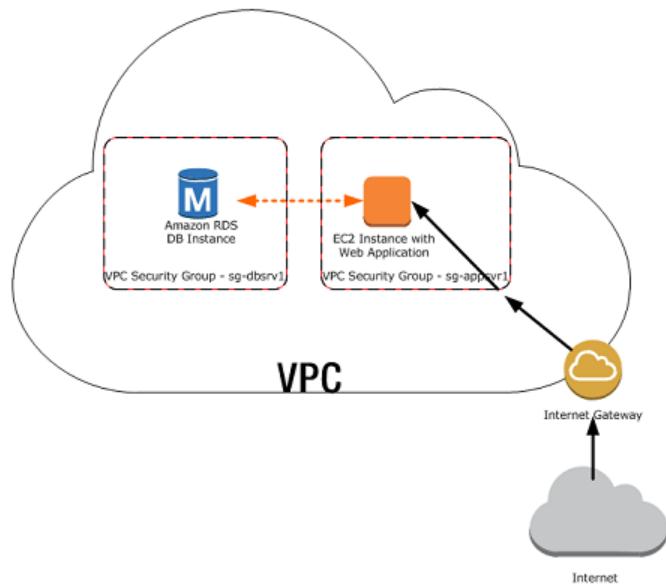
Amazon RDS supports the following scenarios for accessing a DB instance in a VPC:

DB Instance	Accessed By
In a VPC	An EC2 Instance in the Same VPC (p. 414)
	An EC2 Instance in a Different VPC (p. 415)
	An EC2 Instance Not in a VPC (p. 416)
	A Client Application Through the Internet (p. 417)
Not in a VPC	An EC2 Instance in a VPC (p. 417)
	An EC2 Instance Not in a VPC (p. 418)
	A Client Application Through the Internet (p. 419)

A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC

A common use of an RDS instance in a VPC is to share data with an application server that is running in an EC2 instance in the same VPC. This is the user scenario created if you use AWS Elastic Beanstalk to create an EC2 instance and a DB instance in the same VPC.

The following diagram shows this scenario.



The simplest way to manage access between EC2 instances and DB instances in the same VPC is to do the following:

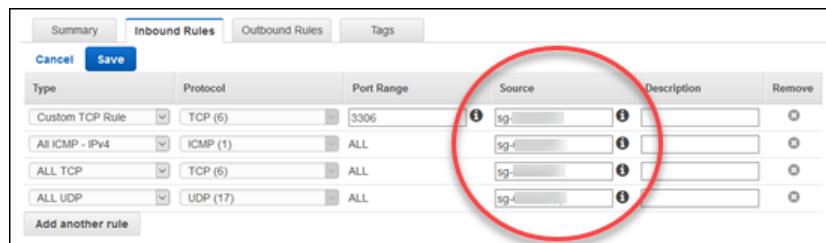
- Create a VPC security group that your DB instances will be in. This security group can be used to restrict access to the DB instances. For example, you can create a custom rule for this security group that allows TCP access using the port you assigned to the DB instance when you created it and an IP address you will use to access the DB instance for development or other purposes.

- Create a VPC security group that your EC2 instances (web servers and clients) will be in. This security group can, if needed, allow access to the EC2 instance from the Internet via the VPC's routing table. For example, you can set rules on this security group to allow TCP access to the EC2 instance over port 22.
- Create custom rules in the security group for your DB instances that allow connections from the security group you created for your EC2 instances. This would allow any member of the security group to access the DB instances.

For a tutorial that shows you how to create a VPC with both public and private subnets for this scenario, see [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#).

To create a rule in a VPC security group that allows connections from another security group, do the following:

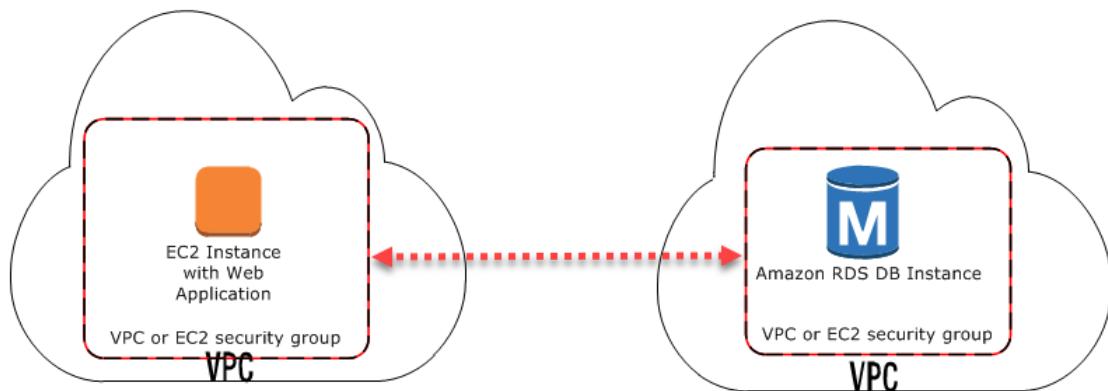
1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc>.
2. In the navigation pane, choose **Security Groups**.
3. Select or create a security group for which you want to allow access to members of another security group. In the scenario above, this would be the security group you will use for your DB instances. Choose the **Inbound Rules** tab, and then choose **Edit rule**.
4. On the **Edit inbound rules** page, choose **Add Rule**.
5. From **Type**, choose one of the **All ICMP** options. In the **Source** box, start typing the ID of the security group; this provides you with a list of security groups. Select the security group with members that you want to have access to the resources protected by this security group. In the scenario above, this would be the security group you will use for your EC2 instance.
6. Repeat the steps for the TCP protocol by creating a rule with **All TCP** as the **Type** and your security group in the **Source** box. If you intend to use the UDP protocol, create a rule with **All UDP** as the **Type** and your security group in the **Source** box.
7. Create a custom TCP rule that permits access via the port you used when you created your DB instance, such as port 3306 for MySQL. Enter your security group or an IP address you will use in the **Source** box.
8. Choose **Save** when you are done.



A DB Instance in a VPC Accessed by an EC2 Instance in a Different VPC

When your DB instance is in a different VPC from the EC2 instance you are using to access it, you can use VPC peering to access the DB instance.

The following diagram shows this scenario.

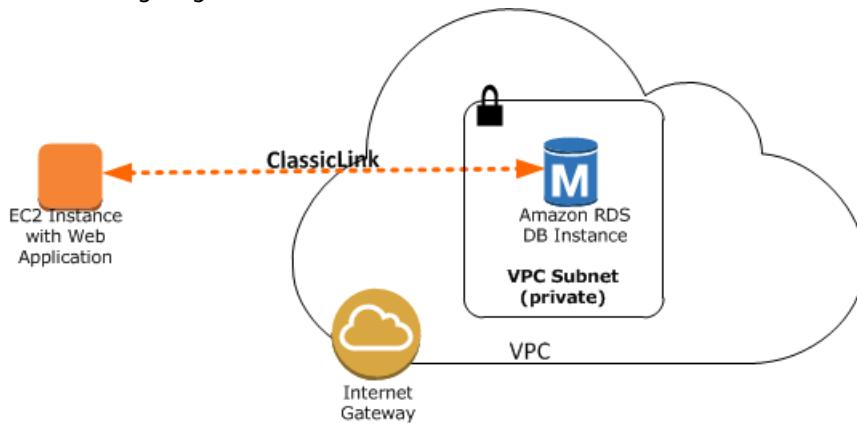


A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IP addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, with a VPC in another AWS account, or with a VPC in a different AWS Region. To learn more about VPC peering, see the [VPC documentation](#).

A DB Instance in a VPC Accessed by an EC2 Instance Not in a VPC

You can communicate between an Amazon RDS DB instance that is in a VPC and an EC2 instance that is not in an Amazon VPC by using *ClassicLink*. When you use Classic Link, an application on the EC2 instance can connect to the DB instance by using the RDS endpoint for the DB instance. ClassicLink is available at no charge.

The following diagram shows this scenario.



Using ClassicLink, you can connect an EC2 instance to a logically isolated database where you define the IP address range and control the access control lists (ACLs) to manage network traffic. You don't have to use public IP addresses or tunneling to communicate with the DB instance in the VPC. This arrangement provides you with higher throughput and lower latency connectivity for inter-instance communications.

To enable ClassicLink between a DB instance in a VPC and an EC2 instance not in a VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc>.

2. In the navigation pane, choose **Your VPCs**.
3. Choose the VPC used by the DB instance.
4. In **Actions**, choose **Enable ClassicLink**. In the confirmation dialog box, choose **Yes, Enable**.
5. On the EC2 console, select the EC2 instance you want to connect to the DB instance in the VPC.
6. In **Actions**, choose **ClassicLink**, and then choose **Link to VPC**.
7. On the **Link to VPC** page, choose the security group you want to use, and then choose **Link to VPC**.

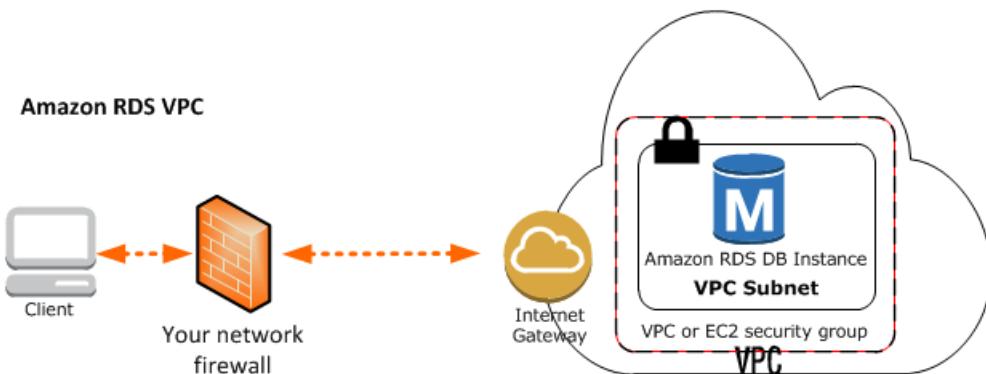
Note

The ClassicLink features are only visible in the consoles for accounts and regions that support EC2-Classic. For more information, see [ClassicLink](#) in the *Amazon EC2 User Guide for Linux Instances*.

A DB Instance in a VPC Accessed by a Client Application Through the Internet

To access a DB instance in a VPC from a client application through the internet, you configure a VPC with a single public subnet, and an Internet gateway to enable communication over the Internet.

The following diagram shows this scenario.



We recommend the following configuration:

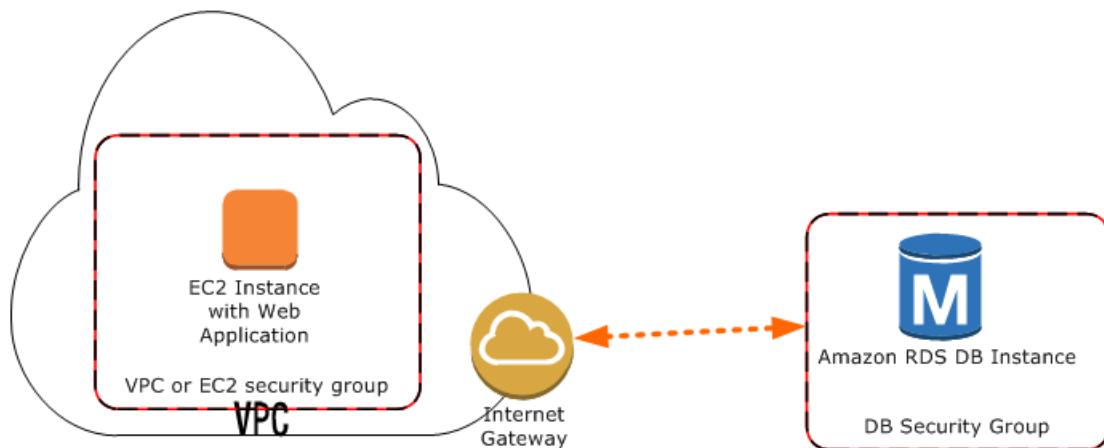
- A VPC of size /16 (for example CIDR: 10.0.0.0/16). This size provides 65,536 private IP addresses.
- A subnet of size /24 (for example CIDR: 10.0.0.0/24). This size provides 256 private IP addresses.
- An Amazon RDS DB instance that is associated with the VPC and the subnet. Amazon RDS assigns an IP address within the subnet to your DB instance.
- An Internet gateway which connects the VPC to the Internet and to other AWS products.
- A security group associated with the DB instance. The security group's inbound rules allow your client application to access to your DB instance.

For information about creating a DB instance in a VPC, see [Creating a DB Instance in a VPC \(p. 422\)](#).

A DB Instance Not in a VPC Accessed by an EC2 Instance in a VPC

In the case where you have an EC2 instance in a VPC and an RDS DB instance not in a VPC, you can connect them over the public Internet.

The following diagram shows this scenario.



Note

ClassicLink, as described in [A DB Instance in a VPC Accessed by an EC2 Instance Not in a VPC \(p. 416\)](#), is not available for this scenario.

To connect your DB instance and your EC2 instance over the public Internet, do the following:

- Ensure that the EC2 instance is in a public subnet in the VPC.
- Ensure that the RDS DB instance was marked as publicly accessible.
- A note about network ACLs here. A network ACL is like a firewall for your entire subnet. Therefore, all instances in that subnet are subject to network ACL rules. By default, network ACLs allow all traffic and you generally don't need to worry about them, unless you particularly want to add rules as an extra layer of security. A security group, on the other hand, is associated with individual instances, and you do need to worry about security group rules.
- Add the necessary ingress rules to the DB security group for the RDS DB instance.

An ingress rule specifies a network port and a CIDR/IP range. For example, you can add an ingress rule that allows port 3306 to connect to a MySQL RDS DB instance, and a CIDR/IP range of 203.0.113.25/32. For more information, see [Authorizing Network Access to a DB Security Group from an IP Range \(p. 402\)](#).

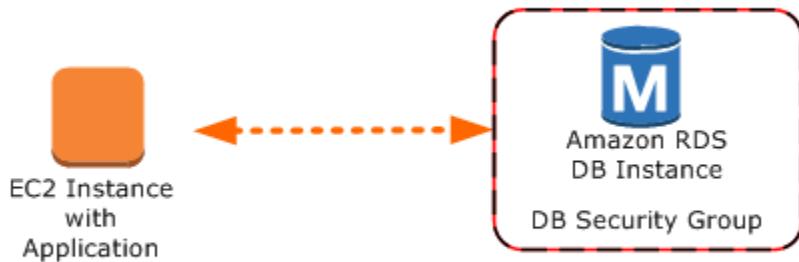
Note

If you are interested in moving an existing DB instance into a VPC, you can use the AWS Management Console to do it easily. For more information, see [Moving a DB Instance Not in a VPC into a VPC \(p. 426\)](#).

A DB Instance Not in a VPC Accessed by an EC2 Instance Not in a VPC

When neither your DB instance nor an application on an EC2 instance are in a VPC, you can access the DB instance by using its endpoint and port.

The following diagram shows this scenario.



You must create a security group for the DB instance that permits access from the port you specified when creating the DB instance. For example, you could use a connection string similar to this connection string used with *sqlplus* to access an Oracle DB instance:

```
PROMPT>sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<endpoint>)(PORT=<port number>))(CONNECT_DATA=(SID=<database name>)))'
```

For more information, see the following documentation.

Database Engine	Relevant Documentation
MariaDB	Connecting to a DB Instance Running the MariaDB Database Engine (p. 452)
Microsoft SQL Server	Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 514)
MySQL	Connecting to a DB Instance Running the MySQL Database Engine (p. 606)
Oracle	Connecting to a DB Instance Running the Oracle Database Engine (p. 751)
PostgreSQL	Connecting to a DB Instance Running the PostgreSQL Database Engine (p. 976)

Note

If you are interested in moving an existing DB instance into a VPC, you can use the AWS Management Console to do it easily. For more information, see [Moving a DB Instance Not in a VPC into a VPC \(p. 426\)](#).

A DB Instance Not in a VPC Accessed by a Client Application Through the Internet

New Amazon RDS customers can only create a DB instance in a VPC. However, you might need to connect to an existing Amazon RDS DB instance that is not in a VPC from a client application through the Internet.

The following diagram shows this scenario.



In this scenario, you must ensure that the DB security group for the RDS DB instance includes the necessary ingress rules for your client application to connect. An ingress rule specifies a network port and a CIDR/IP range. For example, you can add an ingress rule that allows port 3306 to connect to a MySQL RDS DB instance, and a CIDR/IP range of 203.0.113.25/32. For more information, see [Authorizing Network Access to a DB Security Group from an IP Range \(p. 402\)](#).

Warning

If you intend to access a DB instance behind a firewall, talk with your network administrator to determine the IP addresses you should use.

Note

If you are interested in moving an existing DB instance into a VPC, you can use the AWS Management Console to do it easily. For more information, see [Moving a DB Instance Not in a VPC into a VPC \(p. 426\)](#).

Working with an Amazon RDS DB Instance in a VPC

Unless you are working with a legacy DB instance, your DB instance is in a virtual private cloud (VPC). A virtual private cloud is a virtual network that is logically isolated from other virtual networks in the AWS Cloud. Amazon VPC lets you launch AWS resources, such as an Amazon RDS or Amazon EC2 instance, into a VPC. The VPC can either be a default VPC that comes with your account or one that you create. All VPCs are associated with your AWS account.

Your default VPC has three subnets you can use to isolate resources inside the VPC. The default VPC also has an Internet Gateway that can be used to provide access to resources inside the VPC from outside the VPC.

For a list of scenarios involving Amazon RDS DB instances in a VPC and outside of a VPC, see [Scenarios for Accessing a DB Instance in a VPC \(p. 414\)](#).

For a tutorial that shows you how to create a VPC that you can use with a common Amazon RDS scenario, see [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#).

To learn how to work with an Amazon RDS DB instances inside a VPC, see the following:

Topics

- [Working with a DB Instance in a VPC \(p. 420\)](#)
- [Working with DB Subnet Groups \(p. 421\)](#)
- [Hiding a DB Instance in a VPC from the Internet \(p. 422\)](#)
- [Creating a DB Instance in a VPC \(p. 422\)](#)

Working with a DB Instance in a VPC

Here are some tips on working with a DB instance in a VPC:

- Your VPC must have at least two subnets. These subnets must be in two different Availability Zones in the region where you want to deploy your DB instance. A subnet is a segment of a VPC's IP address range that you can specify and that lets you group instances based on your security and operational needs.
- If you want your DB instance in the VPC to be publicly accessible, you must enable the VPC attributes *DNS hostnames* and *DNS resolution*.
- Your VPC must have a DB subnet group that you create (for more information, see the next section). You create a DB subnet group by specifying the subnets you created. Amazon RDS uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet to assign to your DB instance.
- Your VPC must have a VPC security group that allows access to the DB instance.
- The CIDR blocks in each of your subnets must be large enough to accommodate spare IP addresses for Amazon RDS to use during maintenance activities, including failover and compute scaling.
- A VPC can have an *instance tenancy* attribute of either *default* or *dedicated*. All default VPCs have the instance tenancy attribute set to default, and a default VPC can support any DB instance class.

If you choose to have your DB instance in a dedicated VPC where the instance tenancy attribute is set to dedicated, the DB instance class of your DB instance must be one of the approved Amazon EC2 dedicated instance types. For example, the m3.medium EC2 dedicated instance corresponds to the db.m3.medium DB instance class. For information about instance tenancy in a VPC, go to [Using EC2 Dedicated Instances in the Amazon Virtual Private Cloud User Guide](#).

For more information about the instance types that can be in a dedicated instance, see [Amazon EC2 Dedicated Instances](#) on the EC2 pricing page.

- When an option group is assigned to a DB instance, it is linked to the supported platform the DB instance is on, either VPC or EC2-Classic (non-VPC). Furthermore, if a DB instance is in a VPC, the option group associated with the DB instance is linked to that VPC. This linkage means that you cannot use the option group assigned to a DB instance if you attempt to restore the DB instance into a different VPC or onto a different platform.
- If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the DB instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. Note that with persistent or permanent options, such as Oracle TDE, you must create a new option group that includes the persistent or permanent option when restoring a DB instance into a different VPC.

Working with DB Subnet Groups

Subnets are segments of a VPC's IP address range that you designate to group your resources based on security and operational needs. A DB subnet group is a collection of subnets (typically private) that you create in a VPC and that you then designate for your DB instances. A DB subnet group allows you to specify a particular VPC when creating DB instances using the CLI or API; if you use the console, you can just select the VPC and subnets you want to use.

Each DB subnet group should have subnets in at least two Availability Zones in a given region. When creating a DB instance in a VPC, you must select a DB subnet group. Amazon RDS uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet to associate with your DB instance. If the primary DB instance of a Multi-AZ deployment fails, Amazon RDS can promote the corresponding standby and subsequently create a new standby using an IP address of the subnet in one of the other Availability Zones.

When Amazon RDS creates a DB instance in a VPC, it assigns a network interface to your DB instance by using an IP address selected from your DB subnet group. However, we strongly recommend that you use the DNS name to connect to your DB instance because the underlying IP address can change during failover.

Note

For each DB instance that you run in a VPC, you should reserve at least one address in each subnet in the DB subnet group for use by Amazon RDS for recovery actions.

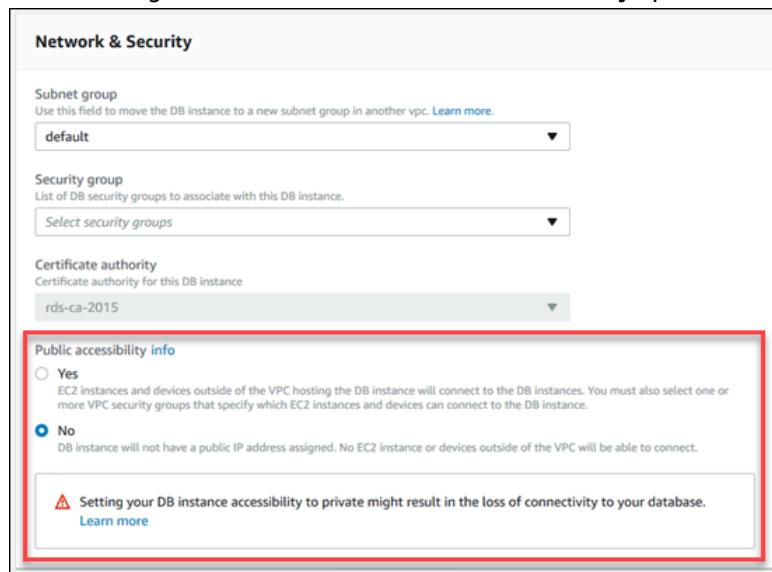
Hiding a DB Instance in a VPC from the Internet

One common Amazon RDS scenario is to have a VPC in which you have an EC2 instance with a public-facing web application and a DB instance with a database that is not publicly accessible. For example, you can create a VPC that has a public subnet and a private subnet. Amazon EC2 instances that function as web servers can be deployed in the public subnet, and the Amazon RDS DB instances are deployed in the private subnet. In such a deployment, only the web servers have access to the DB instances. For an illustration of this scenario, see [A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC \(p. 414\)](#).

When you launch a DB instance inside a VPC, you can designate whether the DB instance you create has a DNS that resolves to a public IP address by using the *Public accessibility* parameter. This parameter lets you designate whether there is public access to the DB instance. Note that access to the DB instance is ultimately controlled by the security group it uses, and that public access is not permitted if the security group assigned to the DB instance does not permit it.

You can modify a DB instance to turn on or off public accessibility by modifying the *Public accessibility* parameter. This parameter is modified just like any other DB instance parameter. For more information, see the modifying section for your DB engine.

The following illustration shows the **Public accessibility** option in the **Network & Security** section.



Creating a DB Instance in a VPC

The following procedures help you create a DB instance in a VPC. If your account has a default VPC, you can begin with step 3 because the VPC and DB subnet group have already been created for you. If your AWS account doesn't have a default VPC, or if you want to create an additional VPC, you can create a new VPC.

If you don't know if you have a default VPC, see [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#).

Note

If you want your DB instance in the VPC to be publicly accessible, you must update the DNS information for the VPC by enabling the VPC attributes *DNS hostnames* and *DNS resolution*. For

information about updating the DNS information for a VPC instance, see [Updating DNS Support for Your VPC](#).

Follow these steps to create a DB instance in a VPC:

- [Step 1: Create a VPC \(p. 423\)](#)
- [Step 2: Add Subnets to the VPC \(p. 423\)](#)
- [Step 3: Create a DB Subnet Group \(p. 423\)](#)
- [Step 4: Create a VPC Security Group \(p. 424\)](#)
- [Step 5: Create a DB Instance in the VPC \(p. 424\)](#)

Step 1: Create a VPC

If your AWS account does not have a default VPC or if you want to create an additional VPC, follow the instructions for creating a new VPC. See [Create a VPC with Private and Public Subnets \(p. 427\)](#) in the Amazon RDS documentation, or see [Step 1: Create a VPC](#) in the Amazon VPC documentation.

Step 2: Add Subnets to the VPC

Once you have created a VPC, you need to create subnets in at least two Availability Zones. You use these subnets when you create a DB subnet group. Note that if you have a default VPC, a subnet is automatically created for you in each Availability Zone in the region.

For instructions on how to create subnets in a VPC, see [Create a VPC with Private and Public Subnets \(p. 427\)](#) in the Amazon RDS documentation.

Step 3: Create a DB Subnet Group

A DB subnet group is a collection of subnets (typically private) that you create for a VPC and that you then designate for your DB instances. A DB subnet group allows you to specify a particular VPC when you create DB instances using the CLI or API. If you use the Amazon RDS console, you can just select the VPC and subnets you want to use. Each DB subnet group must have at least one subnet in at least two Availability Zones in the region.

Note

For a DB instance to be publicly accessible, the subnets in the DB subnet group must have an Internet gateway. For more information about Internet gateways for subnets, go to [Internet Gateways](#) in the Amazon VPC documentation.

When you create a DB instance in a VPC, you must select a DB subnet group. Amazon RDS then uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet. Amazon RDS creates and associates an Elastic Network Interface to your DB instance with that IP address. For Multi-AZ deployments, defining a subnet for two or more Availability Zones in a region allows Amazon RDS to create a new standby in another Availability Zone should the need arise. You need to do this even for Single-AZ deployments, just in case you want to convert them to Multi-AZ deployments at some point.

In this step, you create a DB subnet group and add the subnets you created for your VPC.

AWS Management Console

To create a DB subnet group

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Subnet groups**.
3. Choose **Create DB Subnet Group**.
4. For **Name**, type the name of your DB subnet group.

5. For **Description**, type a description for your DB subnet group.
6. For **VPC**, choose the VPC that you created.
7. In the **Add subnets** section, choose **Add all the subnets related to this VPC**.

The screenshot shows the 'Create DB subnet group' wizard. In the 'Subnet group details' section, the 'Name' field is set to 'mydbsubnetgroup' and the 'Description' field is set to 'My DB Subnet Group'. The 'VPC' dropdown is set to 'tutorial-vpc (vpc-971c12ee)'. In the 'Add subnets' section, there is a button labeled 'Add all the subnets related to this VPC'. Below it, there are dropdown menus for 'Availability zone' and 'Subnet', both currently set to 'select an availability zone' and 'select a subnet' respectively. A table titled 'Subnets in this subnet group (4)' lists four subnets with their details and a 'Remove' button next to each:

Availability zone	Subnet ID	CIDR block	Action
us-east-1a	subnet-d8c8e7f4	10.0.2.0/24	Remove
us-east-1f	subnet-718fdc7d	10.0.3.0/24	Remove
us-east-1a	subnet-cbc8e7e7	10.0.1.0/24	Remove
us-east-1a	subnet-0ccde220	10.0.0.0/24	Remove

At the bottom right of the wizard are 'Cancel' and 'Create' buttons.

8. Choose **Create**.

Your new DB subnet group appears in the DB subnet groups list on the RDS console. You can click the DB subnet group to see details, including all of the subnets associated with the group, in the details pane at the bottom of the window.

Step 4: Create a VPC Security Group

Before you create your DB instance, you must create a VPC security group to associate with your DB instance. For instructions on how to create a security group for your DB instance, see [Create a VPC Security Group for a Private Amazon RDS DB Instance \(p. 430\)](#) in the Amazon RDS documentation, or see [Security Groups for Your VPC](#) in the Amazon VPC documentation.

Step 5: Create a DB Instance in the VPC

In this step, you create a DB instance and use the VPC name, the DB subnet group, and the VPC security group you created in the previous steps.

Note

If you want your DB instance in the VPC to be publicly accessible, you must enable the VPC attributes *DNS hostnames* and *DNS resolution*. For information on updating the DNS information for a VPC instance, see [Updating DNS Support for Your VPC](#).

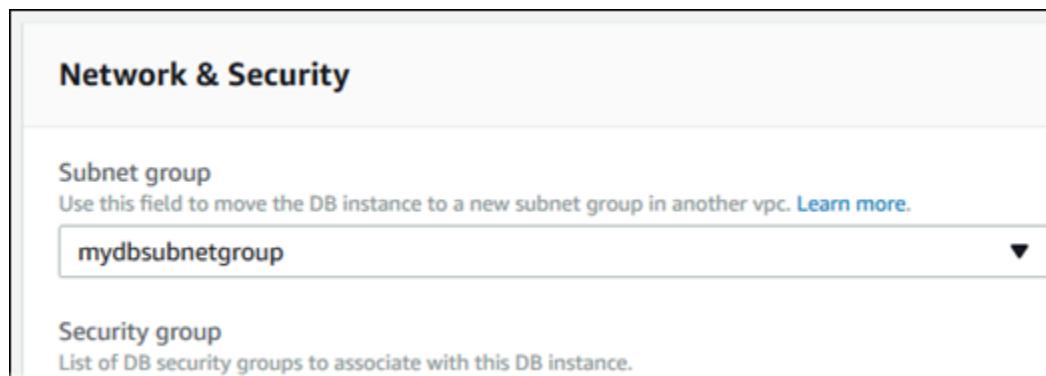
For details on how to create a DB instance for your DB engine, see the topic following that discusses your DB engine. For each engine, when prompted in the **Network & Security** section, enter the VPC name, the DB subnet group, and the VPC security group you created in the previous steps.

Database Engine	Relevant Documentation
MariaDB	Creating a DB Instance Running the MariaDB Database Engine (p. 443)
Microsoft SQL Server	Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 504)
MySQL	Creating a DB Instance Running the MySQL Database Engine (p. 597)
Oracle	Creating a DB Instance Running the Oracle Database Engine (p. 742)
PostgreSQL	Creating a DB Instance Running the PostgreSQL Database Engine (p. 969)

Updating the VPC for a DB Instance

You can use the AWS Management Console to easily move your DB instance to a different VPC.

For details on how to modify a DB instance for your DB engine, see the topic in the table following that discusses your DB engine. In the **Network & Security** section of the modify page, shown following, for **Subnet group**, enter the new subnet group. The new subnet group must be a subnet group in a new VPC.



Database Engine	Relevant Documentation
MariaDB	Modifying a DB Instance Running the MariaDB Database Engine (p. 456)
Microsoft SQL Server	Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 521)
MySQL	Modifying a DB Instance Running the MySQL Database Engine (p. 610)
Oracle	Modifying a DB Instance Running the Oracle Database Engine (p. 758)
PostgreSQL	Modifying a DB Instance Running the PostgreSQL Database Engine (p. 979)

Moving a DB Instance Not in a VPC into a VPC

Some legacy DB instances on the EC2-Classic platform are not in a VPC. If your DB instance is not in a VPC, you can use the AWS Management Console to easily move your DB instance into a VPC. Before you can move a DB instance not in a VPC, into a VPC, you must create the VPC.

Follow these steps to create a VPC for your DB instance.

- [Step 1: Create a VPC \(p. 423\)](#)
- [Step 2: Add Subnets to the VPC \(p. 423\)](#)
- [Step 3: Create a DB Subnet Group \(p. 423\)](#)
- [Step 4: Create a VPC Security Group \(p. 424\)](#)

After you create the VPC, follow these steps to move your DB instance into the VPC.

- [Updating the VPC for a DB Instance \(p. 425\)](#)

The following are some limitations to moving your DB instance into the VPC.

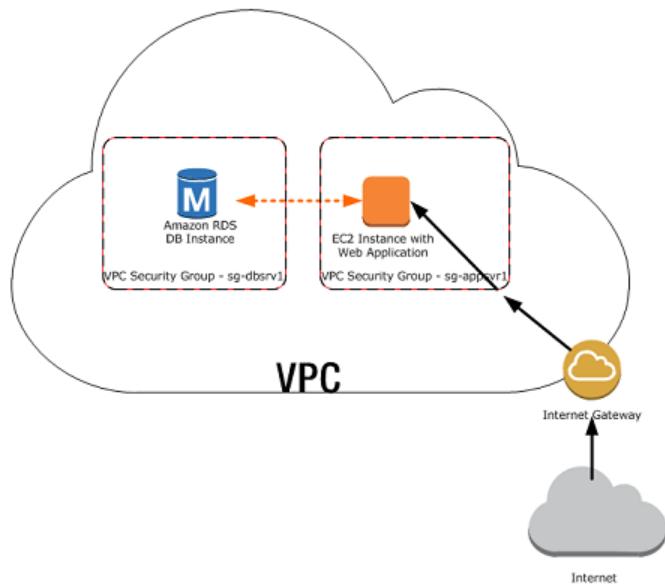
- Moving a Multi-AZ DB instance not in a VPC into a VPC is not currently supported.
- Moving a DB instance with Read Replicas not in a VPC into a VPC is not currently supported.

If you move your DB instance into a VPC, and you are using a custom option group with your DB instance, then you need to change the option group that is associated with your DB instance. Option groups are platform-specific, and moving to a VPC is a change in platform. To use a custom option group in this case, assign the default VPC option group to the DB instance, assign an option group that is used by other DB instances in the VPC you are moving to, or create a new option group and assign it to the DB instance. For more information, see [Working with Option Groups \(p. 156\)](#).

Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance

A common scenario includes an Amazon RDS DB instance in an Amazon VPC, that shares data with a web server that is running in the same VPC. In this tutorial you create the VPC for this scenario.

The following diagram shows this scenario. For information about other scenarios, see [Scenarios for Accessing a DB Instance in a VPC \(p. 414\)](#).



Because your Amazon RDS DB instance only needs to be available to your web server, and not to the public Internet, you create a VPC with both public and private subnets. The web server is hosted in the public subnet, so that it can reach the public Internet. The Amazon RDS DB instance is hosted in a private subnet. The web server is able to connect to the Amazon RDS DB instance because it is hosted within the same VPC, but the Amazon RDS DB instance is not available to the public Internet, providing greater security.

Create a VPC with Private and Public Subnets

Use the following procedure to create a VPC with both public and private subnets.

To create a VPC and subnets

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the top-right corner of the AWS Management Console, choose the region to create your VPC in. This example uses the US West (Oregon) region.
3. In the upper-left corner, choose **VPC Dashboard**. To begin creating a VPC, choose **Launch VPC Wizard**.
4. On the **Step 1: Select a VPC Configuration** page, choose **VPC with Public and Private Subnets**, and then choose **Select**.
5. On the **Step 2: VPC with Public and Private Subnets** page, set these values:
 - **IPv4 CIDR block:** 10.0.0.0/16
 - **IPv6 CIDR block:** No IPv6 CIDR Block

- **VPC name:** tutorial-vpc
- **Public subnet's IPv4 CIDR:** 10.0.0.0/24
- **Availability Zone:** us-west-2a
- **Public subnet name:** Tutorial public
- **Private subnet's IPv4 CIDR:** 10.0.1.0/24
- **Availability Zone:** us-west-2a
- **Private subnet name:** Tutorial Private 1
- **Instance type:** t2.small

Important

If you do not see the **Instance type** box in the console, click **Use a NAT instance instead**. This link is on the right.

Note

If the t2.small instance type is not listed, you can select a different instance type.

- **Key pair name:** No key pair
- **Service endpoints:** Skip this field.
- **Enable DNS hostnames:** Yes
- **Hardware tenancy:** Default

6. When you're finished, choose **Create VPC**.

Create Additional Subnets

You must have either two private subnets or two public subnets available to create an Amazon RDS DB subnet group for an RDS DB instance to use in a VPC. Because the RDS DB instance for this tutorial is private, add a second private subnet to the VPC.

To create an additional subnet

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. To add the second private subnet to your VPC, choose **VPC Dashboard**, choose **Subnets**, and then choose **Create subnet**.
3. On the **Create subnet** page, set these values:
 - **Name tag:** Tutorial private 2
 - **VPC:** Choose the VPC that you created in the previous step, for example: vpc-*identifier* (10.0.0.0/16) | tutorial-vpc
 - **Availability Zone:** us-west-2b

Note

Choose an Availability Zone that is different from the one that you chose for the first private subnet.

- **IPv4 CIDR block:** 10.0.2.0/24
4. When you're finished, choose **Create**. Next, choose **Close** on the confirmation page.
 5. To ensure that the second private subnet that you created uses the same route table as the first private subnet, choose **VPC Dashboard**, choose **Subnets**, and then choose the first private subnet that you created for the VPC, Tutorial private 1.
 6. Below the list of subnets, choose the **Route Table** tab, and note the value for **Route Table**—for example: rtb-98b613fd.
 7. In the list of subnets, deselect the first private subnet.
 8. In the list of subnets, choose the second private subnet Tutorial private 2, and choose the **Route Table** tab.

9. If the current route table is not the same as the route table for the first private subnet, choose **Edit route table association**. For **Route Table ID**, choose the route table that you noted earlier—for example: rtb-98b613fd. Next, to save your selection, choose **Save**.

Create a VPC Security Group for a Public Web Server

Next you create a security group for public access. To connect to public instances in your VPC, you add inbound rules to your VPC security group that allow traffic to connect from the internet.

To create a VPC security group

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. Choose **VPC Dashboard**, choose **Security Groups**, and then choose **Create security group**.
3. On the **Create security group** page, set these values:
 - **Security group name:** tutorial-securitygroup
 - **Description:** Tutorial Security Group
 - **VPC:** Choose the VPC that you created earlier, for example: vpc-*identifier* (10.0.0.0/16) | tutorial-vpc
4. To create the security group, choose **Create**. Next, choose **Close** on the confirmation page.

Note the security group ID because you will need it later in this tutorial.

To add inbound rules to the security group

1. Determine the IP address that you will use to connect to instances in your VPC. To determine your public IP address, you can use the service at <https://checkip.amazonaws.com>. An example of an IP address is 203.0.113.25/32.

If you are connecting through an Internet service provider (ISP) or from behind your firewall without a static IP address, you need to find out the range of IP addresses used by client computers.

Warning

If you use 0.0.0.0/0, you enable all IP addresses to access your public instances. This approach is acceptable for a short time in a test environment, but it's unsafe for production environments. In production, you'll authorize only a specific IP address or range of addresses to access your instances.

2. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
3. Choose **VPC Dashboard**, choose **Security Groups**, and then choose the tutorial-securitygroup security group that you created in the previous procedure.
4. Under the list of security groups, choose the **Inbound Rules** tab, and then choose **Edit rules**.
5. On the **Edit inbound rules** page, choose **Add Rule**.
6. Set the following values for your new inbound rule to allow Secure Shell (SSH) access to your EC2 instance. If you do this, you can connect to your EC2 instance to install the web server and other utilities, and to upload content for your web server.
 - **Type:** SSH
 - **Source:** The IP address or range from Step 1, for example: 203.0.113.25/32.
7. Choose **Add rule**.
8. Set the following values for your new inbound rule to allow HTTP access to your web server.
 - **Type:** HTTP
 - **Source:** 0.0.0.0/0.

9. To save your settings, choose **Save rules**. Next, choose **Close** on the confirmation page.

Create a VPC Security Group for a Private Amazon RDS DB Instance

To keep your Amazon RDS DB instance private, create a second security group for private access. To connect to private instances in your VPC, you add inbound rules to your VPC security group that allow traffic from your web server only.

To create a VPC security group

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. Choose **VPC Dashboard**, choose **Security Groups**, and then choose **Create security group**.
3. On the **Create security group** page, set these values:
 - **Security group name:** tutorial-db-securitygroup
 - **Description:** Tutorial DB Instance Security Group
 - **VPC:** Choose the VPC that you created earlier, for example: vpc-*identifier* (10.0.0.0/16) | tutorial-vpc
4. To create the security group, choose **Create**. Next, choose **Close** on the confirmation page.

To add inbound rules to the security group

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. Choose **VPC Dashboard**, choose **Security Groups**, and then choose the tutorial-db-securitygroup security group that you created in the previous procedure.
3. Under the list of security groups, choose the **Inbound Rules** tab, and then choose **Edit rules**.
4. On the **Edit inbound rules** page, choose **Add Rule**.
5. Set the following values for your new inbound rule to allow MySQL traffic on port 3306 from your EC2 instance. If you do this, you can connect from your web server to your DB instance to store and retrieve data from your web application to your database.
 - **Type:** MySQL/Aurora
 - **Source:** The identifier of the tutorial-securitygroup security group that you created previously in this tutorial, for example: sg-9edd5cfb.
6. To save your settings, choose **Save rules**. Next, choose **Close** on the confirmation page.

Create a DB Subnet Group

A DB subnet group is a collection of subnets that you create in a VPC and that you then designate for your DB instances. A DB subnet group allows you to specify a particular VPC when creating DB instances.

To create a DB subnet group

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Subnet groups**.
3. Choose **Create DB Subnet Group**.
4. On the **Create DB subnet group** page, set these values in **Subnet group details**:
 - **Name:** tutorial-db-subnet-group
 - **Description:** Tutorial DB Subnet Group

- **VPC:** tutorial-vpc (*vpc-*identifier**)
5. In the **Add subnets** section, choose **Add all the subnets related to this VPC**.
 6. Choose **Create**.

Your new DB subnet group appears in the DB subnet groups list on the RDS console. You can click the DB subnet group to see details, including all of the subnets associated with the group, in the details pane at the bottom of the window.

MariaDB on Amazon RDS

Amazon RDS supports DB instances running several versions of MariaDB. You can use the following major versions:

- MariaDB 10.3
- MariaDB 10.2
- MariaDB 10.1
- MariaDB 10.0

For more information about minor version support, see [MariaDB on Amazon RDS Versions \(p. 434\)](#).

You first use the Amazon RDS management tools or interfaces to create an Amazon RDS MariaDB DB instance. You can then use the Amazon RDS tools to perform management actions for the DB instance, such as reconfiguring or resizing the DB instance, authorizing connections to the DB instance, creating and restoring from backups or snapshots, creating Multi-AZ secondaries, creating Read Replicas, and monitoring the performance of the DB instance. You use standard MariaDB utilities and applications to store and access the data in the DB instance.

MariaDB is available in all of the AWS Regions. For more information about AWS Regions, see [Regions and Availability Zones \(p. 101\)](#).

You can use Amazon RDS for MariaDB databases to build HIPAA-compliant applications. You can store healthcare-related information, including protected health information (PHI), under an executed Business Associate Agreement (BAA) with AWS. For more information, see [HIPAA Compliance](#). AWS Services in Scope have been fully assessed by a third-party auditor and result in a certification, attestation of compliance, or Authority to Operate (ATO). For more information, see [AWS Services in Scope by Compliance Program](#).

Before creating your first DB instance, you should complete the steps in the setting up section of this guide. For more information, see [Setting Up for Amazon RDS \(p. 5\)](#).

Common Management Tasks for MariaDB on Amazon RDS

The following are the common management tasks you perform with an Amazon RDS DB instance running MariaDB, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Instance Classes, Storage, and PIOPS If you are creating a DB instance for production purposes, you should understand how instance classes, storage types, and Provisioned IOPS work in Amazon RDS.	DB Instance Class (p. 82) Amazon RDS Storage Types (p. 103)
Multi-AZ Deployments Provide high availability with synchronous standby replication in a different Availability Zone, automatic failover, fault tolerance for DB instances using Multi-AZ deployments, and Read Replicas.	High Availability (Multi-AZ) for Amazon RDS (p. 109)

Task Area	Relevant Documentation
Amazon Virtual Private Cloud (VPC) If your AWS account has a default VPC, then your DB instance is automatically created inside the default VPC. If your account does not have a default VPC, and you want the DB instance in a VPC, you must create the VPC and subnet groups before you create the DB instance.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412) Working with an Amazon RDS DB Instance in a VPC (p. 420)
Security Groups By default, DB instances are created with a firewall that prevents access to them. You therefore must create a security group with the correct IP addresses and network configuration to access the DB instance. The security group you create depends on what Amazon EC2 platform your DB instance is on, and whether you access your DB instance from an Amazon EC2 instance. In general, if your DB instance is on the <i>EC2-Classic</i> platform, you will need to create a DB security group; if your DB instance is on the EC2-VPC platform, you will need to create a VPC security group.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412) Controlling Access with Security Groups (p. 394)
Parameter Groups If your DB instance is going to require specific database parameters, you should create a parameter group before you create the DB instance.	Working with DB Parameter Groups (p. 169)
Importing and Exporting Data Establish procedures for importing or exporting data.	Importing Data into a MariaDB DB Instance (p. 476)
Replication You can offload read traffic from your primary MariaDB DB instance by creating Read Replicas.	Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances (p. 143)
Connecting to Your DB Instance Connect to your DB instance using a standard SQL client application.	Connecting to a DB Instance Running the MariaDB Database Engine (p. 452)
Backup and Restore When you create your DB instance, you can configure it to take automated backups. You can also back up and restore your databases manually by using full backup files (.bak files).	Working With Backups (p. 208)
Monitoring Monitor your RDS MariaDB DB instance by using Amazon CloudWatch RDS metrics, events, and Enhanced Monitoring. View log files for your RDS MariaDB DB instance.	Viewing DB Instance Metrics (p. 254) Viewing Amazon RDS Events (p. 305)
Log Files You can access the log files for your MariaDB DB instance.	Amazon RDS Database Log Files (p. 307) MariaDB Database Log Files (p. 311)

There are also advanced administrative tasks for working with DB instances running MariaDB. For more information, see the following documentation:

- [Parameters for MariaDB \(p. 480\)](#)
- [MariaDB on Amazon RDS SQL Reference \(p. 485\)](#)

MariaDB on Amazon RDS Versions

For MariaDB, version numbers are organized as version X.Y.Z. In Amazon RDS terminology, X.Y denotes the major version, and Z is the minor version number. For Amazon RDS implementations, a version change is considered major if the major version number changes, for example going from version 10.0 to 10.1. A version change is considered minor if only the minor version number changes, for example going from version 10.0.17 to 10.0.24.

Amazon RDS currently supports the following versions of MariaDB:

Major Version	Minor Version
MariaDB 10.3	<ul style="list-style-type: none">• 10.3.8 (supported in all AWS Regions)
MariaDB 10.2	<ul style="list-style-type: none">• 10.2.15 (supported in all AWS Regions)• 10.2.12 (supported in all AWS Regions)• 10.2.11 (supported in all AWS Regions)
MariaDB 10.1	<ul style="list-style-type: none">• 10.1.34 (supported in all AWS Regions)• 10.1.31 (supported in all AWS Regions)• 10.1.26 (supported in all AWS Regions)• 10.1.23 (supported in all AWS Regions)• 10.1.19 (supported in all AWS Regions)• 10.1.14 (supported in all AWS Regions except us-east-2)
MariaDB 10.0	<ul style="list-style-type: none">• 10.0.35 (supported in all AWS Regions)• 10.0.34 (supported in all AWS Regions)• 10.0.32 (supported in all AWS Regions)• 10.0.31 (supported in all AWS Regions)• 10.0.28 (supported in all AWS Regions)• 10.0.24 (supported in all AWS Regions)• 10.0.17 (supported in all AWS Regions except us-east-2, ca-central-1, eu-west-2)

For information about the Amazon RDS deprecation policy for MariaDB, see [Amazon RDS FAQs](#).

Version and Feature Support on Amazon RDS

MariaDB 10.3 Support on Amazon RDS

Amazon RDS supports the following versions of MariaDB 10.3:

- 10.3.8 (supported in all AWS Regions)

Amazon RDS supports the following new features for your DB instances running MariaDB version 10.3 or later:

- **Oracle compatibility** – PL/SQL compatibility parser, sequences, INTERSECT and EXCEPT to complement UNION, new TYPE OF and ROW TYPE OF declarations, and invisible columns
- **Temporal data processing** – System versioned tables for querying of past and present states of the database
- **Flexibility** – User-defined aggregates, storage-independent column compression, and proxy protocol support to relay the client IP address to the server
- **Manageability** – Instant ADD COLUMN operations and fast-fail data definition language (DDL) operations

For a list of all MariaDB 10.3 features and their documentation, see [Changes & Improvements in MariaDB 10.3](#) and [Release Notes - MariaDB 10.3 Series](#) on the MariaDB website.

For a list of unsupported features, see [Features Not Supported \(p. 436\)](#).

MariaDB 10.2 Support on Amazon RDS

Amazon RDS supports the following versions of MariaDB 10.2:

- 10.2.15 (supported in all AWS Regions)
- 10.2.12 (supported in all AWS Regions)
- 10.2.11 (supported in all AWS Regions)

Amazon RDS supports the following new features for your DB instances running MariaDB version 10.2 or later:

- ALTER USER
- Common Table Expressions
- Compressing Events to Reduce Size of the Binary Log
- CREATE USER — new options for limiting resource usage and TLS/SSL
- EXECUTE IMMEDIATE
- Flashback
- InnoDB — now the default storage engine instead of XtraDB
- InnoDB — set the buffer pool size dynamically
- JSON Functions
- Window Functions
- WITH

For a list of all MariaDB 10.2 features and their documentation, see [Changes & Improvements in MariaDB 10.2](#) and [Release Notes - MariaDB 10.2 Series](#) on the MariaDB website.

For a list of unsupported features, see [Features Not Supported \(p. 436\)](#).

MariaDB 10.1 Support on Amazon RDS

Amazon RDS supports the following versions of MariaDB 10.1:

- 10.1.34 (supported in all AWS Regions)

- 10.1.31 (supported in all AWS Regions)
- 10.1.26 (supported in all AWS Regions)
- 10.1.23 (supported in all AWS Regions)
- 10.1.19 (supported in all AWS Regions)
- 10.1.14 (supported in all AWS Regions except us-east-2)

Amazon RDS supports the following new features for your DB instances running MariaDB version 10.1 or later:

- Optimistic in-order parallel replication
- Page Compression
- XtraDB data scrubbing and defragmentation

For a list of all MariaDB 10.1 features and their documentation, see [Changes & Improvements in MariaDB 10.1](#) and [Release Notes - MariaDB 10.1 Series](#) on the MariaDB website.

For a list of unsupported features, see [Features Not Supported \(p. 436\)](#).

MariaDB 10.0 Support on Amazon RDS

Amazon RDS supports the following versions of MariaDB 10.0:

- 10.0.35 (supported in all AWS Regions)
- 10.0.34 (supported in all AWS Regions)
- 10.0.32 (supported in all AWS Regions)
- 10.0.31 (supported in all AWS Regions)
- 10.0.28 (supported in all AWS Regions)
- 10.0.24 (supported in all AWS Regions)
- 10.0.17 (supported in all AWS Regions except us-east-2, ca-central-1, eu-west-2)

For a list of all MariaDB 10.0 features and their documentation, see [Changes & Improvements in MariaDB 10.0](#) and [Release Notes - MariaDB 10.0 Series](#) on the MariaDB website.

For a list of unsupported features, see [Features Not Supported \(p. 436\)](#).

Features Not Supported

The following MariaDB features are not supported on Amazon RDS:

- Authentication plugin – GSSAPI
- Authentication plugin – Unix Socket
- AWS Key Management encryption plugin
- Delayed replication
- Encryption at rest for XtraDB and InnoDB
- HandlerSocket
- JSON table type
- MariaDB ColumnStore
- MariaDB Galera Cluster

- Multisource replication
- MyRocks storage engine
- Password validation plugin, `simple_password_check`, and `cracklib_password_check`
- Replication filters
- Spider storage engine
- Sphinx storage engine
- TokuDB storage engine
- Storage engine-specific object attributes, as described in [Engine-defined New Table/Field/Index Attributes](#) in the MariaDB documentation
- Table and tablespace encryption

To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application. Amazon RDS doesn't allow direct host access to a DB instance by using Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection.

Supported Storage Engines for MariaDB on Amazon RDS

While MariaDB supports multiple storage engines with varying capabilities, not all of them are optimized for recovery and data durability. InnoDB (for version 10.2 and higher) and XtraDB (for version 10.0 and 10.1) are the recommended and supported storage engines for MariaDB DB instances on Amazon RDS. Amazon RDS features such as Point-In-Time Restore and snapshot restore require a recoverable storage engine and are supported only for the recommended storage engine for the MariaDB version. Amazon RDS also supports Aria, although using Aria might have a negative impact on recovery in the event of an instance failure. However, if you need to use spatial indexes to handle geographic data on MariaDB 10.1 or 10.0, you should use Aria because spatial indexes are not supported by XtraDB. On MariaDB 10.2 and higher, the InnoDB storage engine supports spatial indexes.

Other storage engines are not currently supported by Amazon RDS for MariaDB.

MariaDB Security on Amazon RDS

Security for Amazon RDS MariaDB DB instances is managed at three levels:

- AWS Identity and Access Management controls who can perform Amazon RDS management actions on DB instances. When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations. For more information, see [Authentication and Access Control \(p. 342\)](#).
- When you create a DB instance, you use either a VPC security group or a DB security group to control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance. These connections can be made using Secure Socket Layer (SSL). In addition, firewall rules at your company can control whether devices running at your company can open connections to the DB instance.
- Once a connection has been opened to a MariaDB DB instance, authentication of the login and permissions are applied the same way as in a stand-alone instance of MariaDB. Commands such as `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE`, and `SET PASSWORD` work just as they do in stand-alone databases, as does directly modifying database schema tables.

When you create an Amazon RDS DB instance, the master user has the following default privileges:

- alter
- alter routine
- create
- create routine
- create temporary tables
- create user
- create view
- delete
- drop
- event
- execute
- grant option
- index
- insert
- lock tables
- process
- references
- reload

This privilege is limited on Amazon RDS MariaDB DB instances. It doesn't grant access to the FLUSH LOGS or FLUSH TABLES WITH READ LOCK operations.

- replication client
- replication slave
- select
- show databases
- show view
- trigger
- update

For more information about these privileges, see [User Account Management](#) in the MariaDB documentation.

Note

Although you can delete the master user on a DB instance, we don't recommend doing so. To recreate the master user, use the `ModifyDBInstance` API or the `modify-db-instance` AWS command line tool and specify a new master user password with the appropriate parameter.

If the master user does not exist in the instance, the master user is created with the specified password.

To provide management services for each DB instance, the `rdsadmin` user is created when the DB instance is created. Attempting to drop, rename, change the password for, or change privileges for the `rdsadmin` account results in an error.

To allow management of the DB instance, the standard `kill` and `kill_query` commands have been restricted. The Amazon RDS commands `mysql.rds_kill`, `mysql.rds_kill_query`, and `mysql.rds_kill_query_id` are provided for use in MariaDB and also MySQL so that you can terminate user sessions or queries on DB instances.

Using SSL with a MariaDB DB Instance

Amazon RDS supports Secure Sockets Layer (SSL) connections with DB instances running the MariaDB database engine.

Amazon RDS creates an SSL certificate and installs the certificate on the DB instance when Amazon RDS provisions the instance. These certificates are signed by a certificate authority. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks.

The public key is stored at <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>.

MariaDB uses yaSSL for secure connections in the following versions:

- MariaDB version 10.1.26 and earlier 10.1 versions
- MariaDB version 10.0.32 and earlier 10.0 versions

MariaDB uses OpenSSL for secure connections in the following versions:

- MariaDB 10.3 versions
- MariaDB 10.2 versions
- MariaDB version 10.1.31 and later 10.1 versions
- MariaDB version 10.0.34 and later 10.0 versions

Amazon RDS for MariaDB supports Transport Layer Security (TLS) versions 1.0, 1.1, and 1.2. The following table shows the TLS support for MySQL versions.

MariaDB Version	TLS 1.0	TLS 1.1	TLS 1.2
MariaDB 10.3	Supported	Supported	Supported
MariaDB 10.2	Supported	Supported	Supported
MariaDB 10.1	Supported	Supported for 10.1.31 and later 10.1 versions	Supported for 10.1.31 and later 10.1 versions
MariaDB 10.0	Supported	Supported for 10.0.34 and later 10.0 versions	Supported for 10.0.34 and later 10.0 versions

To encrypt connections using the default mysql client, launch the mysql client using the `--ssl-ca` parameter to reference the public key, as shown in the examples following.

The following example shows how to launch the client using the `--ssl-ca` parameter for MariaDB 10.2 and later.

```
mysql -h myinstance.c9akciq32.rds-us-east-1.amazonaws.com  
--ssl-ca=[full path]rds-combined-ca-bundle.pem --ssl-mode=REQUIRED
```

The following example shows how to launch the client using the `--ssl-ca` parameter for MariaDB 10.1 and earlier.

```
mysql -h myinstance.c9akciq32.rds-us-east-1.amazonaws.com  
--ssl-ca=[full path]rds-combined-ca-bundle.pem --ssl-verify-server-cert
```

You can require SSL connections for specific users accounts. For example, you can use one of the following statements, depending on your MariaDB version, to require SSL connections on the user account `encrypted_user`.

For MariaDB 10.2 and later, use the following statement.

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

For MariaDB 10.1 and earlier, use the following statement.

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL;
```

For more information on SSL connections with MariaDB, see [SSL Overview](#) in the MariaDB documentation.

Cache Warming

InnoDB (version 10.2 and later) and XtraDB (versions 10.0 and 10.1) cache warming can provide performance gains for your MariaDB DB instance by saving the current state of the buffer pool when the DB instance is shut down, and then reloading the buffer pool from the saved information when the DB instance starts up. This approach bypasses the need for the buffer pool to "warm up" from normal database use and instead preloads the buffer pool with the pages for known common queries. For more information on cache warming, see [Dumping and restoring the buffer pool](#) in the MariaDB documentation.

Cache warming is enabled by default on MariaDB 10.2 and higher DB instances. To enable it, set the `innodb_buffer_pool_dump_at_shutdown` and `innodb_buffer_pool_load_at_startup` parameters to 1 in the parameter group for your DB instance. Changing these parameter values in a parameter group affects all MariaDB DB instances that use that parameter group. To enable cache warming for specific MariaDB DB instances, you might need to create a new parameter group for those DB instances. For information on parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

Cache warming primarily provides a performance benefit for DB instances that use standard storage. If you use PIOPS storage, you don't commonly see a significant performance benefit.

Important

If your MariaDB DB instance doesn't shut down normally, such as during a failover, then the buffer pool state isn't saved to disk. In this case, MariaDB loads whatever buffer pool file is available when the DB instance is restarted. No harm is done, but the restored buffer pool might not reflect the most recent state of the buffer pool prior to the restart. To ensure that you have a recent state of the buffer pool available to warm the cache on startup, we recommend that you periodically dump the buffer pool "on demand." You can dump or load the buffer pool on demand.

You can create an event to dump the buffer pool automatically and at a regular interval. For example, the following statement creates an event named `periodic_buffer_pool_dump` that dumps the buffer pool every hour.

```
CREATE EVENT periodic_buffer_pool_dump
  ON SCHEDULE EVERY 1 HOUR
  DO CALL mysql.rds_innodb_buffer_pool_dump_now();
```

For more information, see [Events](#) in the MariaDB documentation.

Dumping and Loading the Buffer Pool on Demand

You can save and load the cache on demand using the following stored procedures:

- To dump the current state of the buffer pool to disk, call the [mysql.rds_innodb_buffer_pool_dump_now \(p. 710\)](#) stored procedure.
- To load the saved state of the buffer pool from disk, call the [mysql.rds_innodb_buffer_pool_load_now \(p. 710\)](#) stored procedure.
- To cancel a load operation in progress, call the [mysql.rds_innodb_buffer_pool_load_abort \(p. 710\)](#) stored procedure.

Database Parameters for MariaDB

By default, a MariaDB DB instance uses a DB parameter group that is specific to a MariaDB database. This parameter group contains some but not all of the parameters contained in the Amazon RDS DB parameter groups for the MySQL database engine. It also contains a number of new, MariaDB-specific parameters. For more information on the parameters available for the Amazon RDS MariaDB DB engine, see [Parameters for MariaDB \(p. 480\)](#).

Common DBA Tasks for MariaDB

Killing sessions or queries, skipping replication errors, working with InnoDB (version 10.2 and later) and XtraDB (versions 10.0 and 10.1) tablespaces to improve crash recovery times, and managing the global status history are common DBA tasks you might perform in a MariaDB DB instance. You can handle these tasks just as in an Amazon RDS MySQL DB instance, as described in [Common DBA Tasks for MySQL DB Instances \(p. 685\)](#). The crash recovery instructions there refer to the MySQL InnoDB engine, but they are applicable to a MariaDB instance running InnoDB or XtraDB as well.

Local Time Zone for MariaDB DB Instances

By default, the time zone for an RDS MariaDB DB instance is Universal Time Coordinated (UTC). You can set the time zone for your DB instance to the local time zone for your application instead.

To set the local time zone for a DB instance, set the `time_zone` parameter in the parameter group for your DB instance to one of the supported values listed later in this section. When you set the `time_zone` parameter for a parameter group, all DB instances and Read Replicas that are using that parameter group change to use the new local time zone. For information on setting parameters in a parameter group, see [Working with DB Parameter Groups \(p. 169\)](#).

After you set the local time zone, all new connections to the database reflect the change. If you have any open connections to your database when you change the local time zone, you won't see the local time zone update until after you close the connection and open a new connection.

You can set a different local time zone for a DB instance and one or more of its Read Replicas. To do this, use a different parameter group for the DB instance and the replica or replicas and set the `time_zone` parameter in each parameter group to a different local time zone.

If you are replicating across regions, then the replication master DB instance and the Read Replica use different parameter groups (parameter groups are unique to a region). To use the same local time zone for each instance, you must set the `time_zone` parameter in the instance's and Read Replica's parameter groups.

When you restore a DB instance from a DB snapshot, the local time zone is set to UTC. You can update the time zone to your local time zone after the restore is complete. If you restore a DB instance to a point in time, then the local time zone for the restored DB instance is the time zone setting from the parameter group of the restored DB instance.

You can set your local time zone to one of the following values.

Africa/Cairo	Asia/Bangkok	Australia/Darwin
Africa/Casablanca	Asia/Beirut	Australia/Hobart
Africa/Harare	Asia/Calcutta	Australia/Perth
Africa/Monrovia	Asia/Damascus	Australia/Sydney
Africa/Nairobi	Asia/Dhaka	Brazil/East
Africa/Tripoli	Asia/Irkutsk	Canada/Newfoundland
Africa/Windhoek	Asia/Jerusalem	Canada/Saskatchewan
America/Araguaina	Asia/Kabul	Europe/Amsterdam
America/Asuncion	Asia/Karachi	Europe/Athens
America/Bogota	Asia/Kathmandu	Europe/Dublin
America/Caracas	Asia/Krasnoyarsk	Europe/Helsinki
America/Chihuahua	Asia/Magadan	Europe/Istanbul
America/Cuiaba	Asia/Muscat	Europe/Kaliningrad
America/Denver	Asia/Novosibirsk	Europe/Moscow
America/Fortaleza	Asia/Riyadh	Europe/Paris
America/Guatemala	Asia/Seoul	Europe/Prague
America/Halifax	Asia/Shanghai	Europe/Sarajevo
America/Manaus	Asia/Singapore	Pacific/Auckland
America/Matamoros	Asia/Taipei	Pacific/Fiji
America/Monterrey	Asia/Tehran	Pacific/Guam
America/Montevideo	Asia/Tokyo	Pacific/Honolulu
America/Phoenix	Asia/Ulaanbaatar	Pacific/Samoa
America/Santiago	Asia/Vladivostok	US/Alaska
America/Tijuana	Asia/Yakutsk	US/Central
Asia/Amman	Asia/Yerevan	US/Eastern
Asia/Ashgabat	Atlantic/Azores	US/East-Indiana
Asia/Baghdad	Australia/Adelaide	US/Pacific
Asia/Baku	Australia/Brisbane	UTC

Creating a DB Instance Running the MariaDB Database Engine

The basic building block of Amazon RDS is the DB instance. The DB instance is where you create your MariaDB databases.

Important

You must complete the tasks in the [Setting Up for Amazon RDS \(p. 5\)](#) section before you can create or connect to a DB instance.

For an example that walks you through the process of creating and connecting to a sample DB instance, see [Creating a MariaDB DB Instance and Connecting to a Database on a MariaDB DB Instance \(p. 10\)](#).

AWS Management Console

To launch a MariaDB DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the AWS Management Console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
4. Choose **Create database** to open the **Select engine** page.

Select engine

Engine options

Amazon Aurora **Amazon Aurora** 

MySQL 

MariaDB 

PostgreSQL 

Oracle **ORACLE** 

Microsoft SQL Server 

MariaDB
MariaDB Community Edition is a MySQL-compatible database with strong support from the open source community, and extra features and performance optimizations.

- Supports database size up to 16 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.
- Supports global transaction ID (GTID) and thread pooling.
- Developed and supported by the MariaDB open source community.

Only enable options eligible for RDS Free Usage Tier [info](#)

[Cancel](#) **Next**

5. Choose **MariaDB**, and then choose **Next**.
6. The **Choose use case** page asks if you are planning to use the DB instance you are creating for production. If you are, choose **Production - MariaDB**. If you choose **Production - MariaDB**, the following are preselected in a later step:
 - **Multi-AZ failover option**
 - **Provisioned IOPS storage option**
 - **Enable deletion protection option**

We recommend these features for any production environment.

7. Choose **Next** to continue. The **Specify DB details** page appears.

On the **Specify DB details** page, specify your DB instance information. For information about each setting, see [Settings for MariaDB DB Instances \(p. 448\)](#).

Specify DB details

Instance specifications

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#) 

DB engine
MariaDB Community Edition

License model [Info](#)

DB engine version [Info](#)

DB instance class [Info](#)

Multi-AZ deployment [Info](#)

Create replica in different zone
Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

No

Storage type [Info](#)

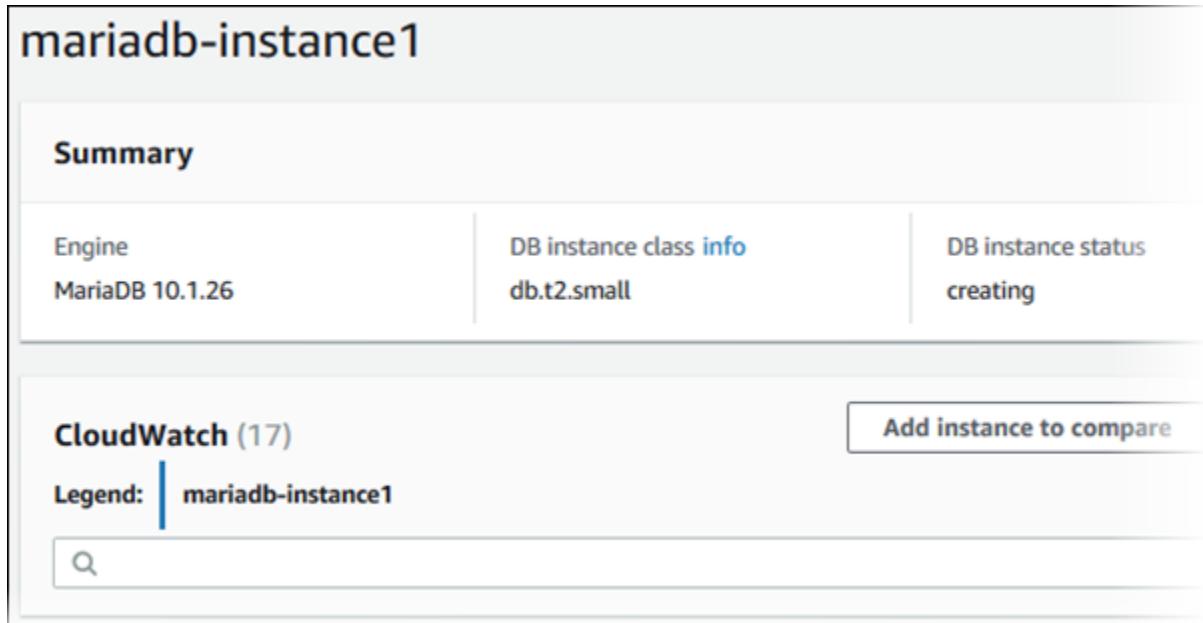
8. Choose **Next** to continue.

On the **Configure advanced settings** page, provide additional information that Amazon RDS needs to launch the DB instance. For information about each setting, see [Settings for MySQL DB Instances \(p. 601\)](#).

9. Choose **Create database**.

10. On the final page, choose **View DB instance details**.

On the RDS console, the details for the new DB instance appear. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and storage allocated, it could take several minutes for the new instance to be available.



CLI

To create a MariaDB DB instance by using the AWS CLI, call the `create-db-instance` command with the parameters below. For information about each setting, see [Settings for MariaDB DB Instances \(p. 448\)](#).

- `--db-instance-identifier`
- `--db-instance-class`
- `--db-security-groups`
- `--db-subnet-group`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--allocated-storage`
- `--backup-retention-period`

Note

If you require a specific minor version of MariaDB, include the `--engine-version` parameter.

Example

The following command creates a MariaDB instance named *mydbinstance*.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
--db-instance-identifier mydbinstance \
--db-instance-class db.m4.xlarge \
--engine mariadb \
--allocated-storage 20 \
--master-username masteruser \
--master-user-password masteruserpassword \
--backup-retention-period 3
```

For Windows:

```
aws rds create-db-instance ^
--db-instance-identifier mydbinstance ^
--db-instance-class db.m4.xlarge ^
--engine mariadb ^
--allocated-storage 20 ^
--master-username masteruser ^
--master-user-password masteruserpassword ^
--backup-retention-period 3
```

This command should produce output that begins with information that is similar to the following:

```
DBINSTANCE 20 True 3 rds-ca-2015 False arn:aws:rds:us-east-1:1234567890:db:mydbinstance
db.m4.xlarge mydbinstance creating 0 **** mariadb 10.1.26
```

API

To create a MariaDB DB instance by using the Amazon RDS API, call the [CreateDBInstance](#) action with the parameters below. For information about each setting, see [Settings for MariaDB DB Instances \(p. 448\)](#).

- AllocatedStorage
- BackupRetentionPeriod
- DBInstanceClass
- DBInstanceIdentifier
- DBSecurityGroups
- DBSubnetGroup
- Engine
- MasterUsername
- MasterUserPassword

Note

If you require a specific minor version of MariaDB, include the EngineVersion parameter.

Example

```
https://rds.us-west-2.amazonaws.com/
?Action=CreateDBInstance
&AllocatedStorage=20
&BackupRetentionPeriod=3
&DBInstanceClass=db.m4.xlarge
&DBInstanceIdentifier=mydbinstance
&DBName=mydatabase
&DBSecurityGroups.member.1=mysecuritygroup
&DBSubnetGroup=mydbsubnetgroup
&Engine=mariadb
&MasterUserPassword=masteruserpassword
&MasterUsername=masterawsuser
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140213/us-west-2/rds/aws4_request
&X-Amz-Date=20140213T162136Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=8052a76dfb18469393c5f0182cdab0ebc224a9c7c5c949155376c1c250fc7ec3
```

Settings for MariaDB DB Instances

The following table contains details about settings that you choose when you create a Maria DB instance.

Setting	Setting Description
Allocated storage	<p>The amount of storage to allocate for your DB instance (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance.</p> <p>For more information, see DB instance storage (p. 103).</p>
Auto minor version upgrade	<p>Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.</p>
Availability zone	<p>The availability zone for your DB instance. Use the default value of No Preference unless you want to specify an Availability Zone.</p> <p>For more information, see Regions and Availability Zones (p. 101).</p>
Backup retention period	<p>The number of days that you want automatic backups of your DB instance to be retained. For any non-trivial DB instance, you should set this value to 1 or greater.</p> <p>For more information, see Working With Backups (p. 208).</p>
Backup window	<p>The time period during which Amazon RDS automatically takes a backup of your DB instance. Unless you have a specific time that you want to have your database backup, use the default of No Preference.</p> <p>For more information, see Working With Backups (p. 208).</p>
Copy Tags To Snapshots	<p>Select this option to copy any DB instance tags to a DB snapshot when you create a snapshot.</p> <p>For more information, see Tagging Amazon RDS Resources (p. 138).</p>
Database name	<p>The name for the database on your DB instance. The name must contain 1 to 64 alpha-numeric characters. If you do not provide a name, Amazon RDS does not create a database on the DB instance you are creating.</p> <p>To create additional databases on your DB instance, connect to your DB instance and use the SQL command <code>CREATE DATABASE</code>. For more information, see Connecting to a DB Instance Running the MariaDB Database Engine (p. 452).</p>
Database port	<p>The port that you want to access the DB instance through. MariaDB installations default to port 3306. If you use a DB security group with your DB instance, this must be the</p>

Setting	Setting Description
	<p>same port value you provided when creating the DB security group.</p> <p>The firewalls at some companies block connections to the default MariaDB port. If your company firewall blocks the default port, choose another port for your DB instance.</p>
Deletion protection	<p>Enable deletion protection to prevent your DB instance from being deleted. If you create a production DB instance with the AWS Management Console, deletion protection is enabled by default. For more information, see Deleting a DB Instance (p. 135).</p>
DB engine version	The version of MariaDB that you want to use.
DB instance class	<p>The configuration for your DB instance.</p> <p>If possible, choose an instance class large enough that a typical query working set can be held in memory. When working sets are held in memory the system can avoid writing to disk, and this improves performance.</p> <p>For more information, see DB Instance Class (p. 82).</p>
DB instance identifier	The name for your DB instance. Your DB instance identifier can contain up to 63 alphanumeric characters, and must be unique for your account in the AWS Region you chose. You can add some intelligence to the name, such as including the AWS Region you chose, for example <code>mariadb-instance1</code> .
DB parameter group	<p>A parameter group for your DB instance. You can choose the default parameter group or you can create a custom parameter group.</p> <p>For more information, see Working with DB Parameter Groups (p. 169).</p>
Encryption	<p>Enable Encryption to enable encryption at rest for this DB instance.</p> <p>For more information, see Encrypting Amazon RDS Resources (p. 389).</p>
Enhanced monitoring	<p>Enable enhanced monitoring to gather metrics in real time for the operating system that your DB instance runs on.</p> <p>For more information, see Enhanced Monitoring (p. 256).</p>
License model	MariaDB has only one license model, general-public-license the general license agreement for MariaDB.
Log exports	Select the types of MariaDB database log files to generate. For more information, see MariaDB Database Log Files (p. 311) .

Setting	Setting Description
Maintenance window	The 30 minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose No Preference . For more information, see The Amazon RDS Maintenance Window (p. 120) .
Master username	The name that you use as the master user name to log on to your DB instance. For more information, and a list of the default privileges for the master user, see MariaDB Security on Amazon RDS (p. 437) .
Master password	The password for your master user account. The password must contain from 8 to 41 printable ASCII characters (excluding /, ", a space, and @).
Multi-AZ deployment	Create replica in different zone to create a standby mirror of your DB instance in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose No . For more information, see High Availability (Multi-AZ) for Amazon RDS (p. 109) .
Option group	An option group for your DB instance. You can choose the default option group or you can create a custom option group. For more information, see Working with Option Groups (p. 156) .
Public accessibility	Yes to give your DB instance a public IP address. This means that it is accessible outside the VPC (the DB instance also needs to be in a public subnet in the VPC). Choose No if you want the DB instance to only be accessible from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 422) .
Storage type	The storage type for your DB instance. For more information, see Amazon RDS Storage Types (p. 103) .
Subnet group	This setting depends on the platform you are on. If you are a new customer to AWS, choose default , which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC.

Setting	Setting Description
Virtual Private Cloud (VPC)	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, choose Not in VPC . For more information, see Amazon Virtual Private Cloud (VP Cs) and Amazon RDS (p. 412) .
VPC security groups	If you are a new customer to AWS, choose Create new VPC security group . Otherwise, choose Select existing VPC security groups , and select security groups you previously created. When you choose Create new VPC security group in the RDS console, a new security group is created with an inbound rule that allows access to the DB instance from the IP address detected in your browser. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 399) .

Related Topics

- [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#)
- [Connecting to a DB Instance Running the MariaDB Database Engine \(p. 452\)](#)
- [Modifying a DB Instance Running the MariaDB Database Engine \(p. 456\)](#)
- [Deleting a DB Instance \(p. 135\)](#)

Connecting to a DB Instance Running the MariaDB Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard MariaDB client application or utility to connect to the instance. In the connection string, you specify the DNS address from the DB instance endpoint as the host parameter, and specify the port number from the DB instance endpoint as the port parameter.

You can use the AWS Management Console, the AWS CLI [describe-db-instances](#) command, or the Amazon RDS API [DescribeDBInstances](#) action to list the details of an Amazon RDS DB instance, including its endpoint.

To find the endpoint for a MariaDB instance in the AWS Management Console

1. Open the RDS console and then choose **Databases** to display a list of your DB instances.
2. Choose the MariaDB DB instance name to display its details.
3. On the **Connectivity** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.

The screenshot shows the 'Summary' page for the 'mymariadb' database instance. At the top, there are sections for 'DB Name' (mymariadb) and 'Role' (Instance). Below these are tabs for 'Connectivity', 'Monitoring', 'Logs & events', 'Configuration', and 'Maintenance'. The 'Connectivity' tab is active. Under 'Endpoint & port', it shows 'Endpoint' as 'mymariadb' followed by a blurred IP address, and 'Port' as '3306'. The value 'mymariadb' is circled in red.

If an endpoint value is `mariadb-instance1.123456789012.us-east-1.rds.amazonaws.com:3306`, then you specify the following values in a MariaDB connection string:

- For host or host name, specify `mariadb-instance1.123456789012.us-east-1.rds.amazonaws.com`
- For port, specify 3306

You can connect to an Amazon RDS MariaDB DB instance by using tools like the `mysql` command line utility. For more information on using the `mysql` utility, go to [mysql Command-line Client](#) in the MariaDB documentation. One GUI-based application you can use to connect is HeidiSQL. For more information, see the [Download HeidiSQL](#) page.

Two common causes of connection failures to a new DB instance are the following:

- The DB instance was created using a security group that does not authorize connections from the device or Amazon EC2 instance where the MariaDB application or utility is running. If the DB instance was created in an Amazon VPC, it must have a VPC security group that authorizes the connections. If the DB instance was created outside of a VPC, it must have a DB security group that authorizes the connections.
- The DB instance was created using the default port of 3306, and your company has firewall rules blocking connections to that port from devices in your company network. To fix this failure, recreate the instance with a different port.

You can use SSL encryption on connections to an Amazon RDS MariaDB DB instance. For information, see [Using SSL with a MariaDB DB Instance \(p. 439\)](#).

Connecting from the mysql Utility

To connect to a DB instance using the mysql utility, type the following command at a command prompt on a client computer to connect to a database on a MariaDB DB instance. Substitute the DNS name (endpoint) for your DB instance for <endpoint>, the master user name you used for <mymasteruser>, and provide the master password you used when prompted for a password.

```
mysql -h <endpoint> -P 3306 -u <mymasteruser>
```

After you enter the password for the user, you see output similar to the following.

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 272
Server version: 5.5.5-10.0.17-MariaDB-log MariaDB Server

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql >
```

Connecting with SSL

Amazon RDS creates an SSL certificate for your DB instance when the instance is created. If you enable SSL certificate verification, then the SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. To connect to your DB instance using SSL, follow these steps:

To connect to a DB instance with SSL using the mysql utility

1. Download a root certificate that works for all regions from [here](#).
2. Enter the following command at a command prompt to connect to a DB instance with SSL using the mysql utility. For the -h parameter, substitute the DNS name for your DB instance. For the --ssl-ca parameter, substitute the SSL certificate file name as appropriate.

```
mysql -h mariadb-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-
ca-2015-root.pem
```

3. Include the --ssl-verify-server-cert parameter so that the SSL connection verifies the DB instance endpoint against the endpoint in the SSL certificate. For example:

```
mysql -h mariadb-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-ca-2015-root.pem --ssl-verify-server-cert
```

4. Enter the master user password when prompted.

You should see output similar to the following.

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 272
Server version: 5.5.5-10.0.17-MariaDB-log MariaDB Server

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql >
```

Maximum MariaDB Connections

The maximum number of connections allowed to an Amazon RDS MariaDB DB instance is based on the amount of memory available for the DB instance class of the DB instance. A DB instance class with more memory available results in a larger number of connections available. For more information on DB instance classes, see [DB Instance Class \(p. 82\)](#).

The connection limit for a DB instance is set by default to the maximum for the DB instance class for the DB instance. You can limit the number of concurrent connections to any value up to the maximum number of connections allowed using the `max_connections` parameter in the parameter group for the DB instance. For more information, see [Working with DB Parameter Groups \(p. 169\)](#).

You can retrieve the maximum number of connections allowed for an Amazon RDS MariaDB DB instance by executing the following query on your DB instance:

```
SELECT @@max_connections;
```

You can retrieve the number of active connections to an Amazon RDS MariaDB DB instance by executing the following query on your DB instance:

```
SHOW STATUS WHERE `variable_name` = 'Threads_connected';
```

Related Topics

- [Amazon RDS DB Instances \(p. 80\)](#)
- [Creating a DB Instance Running the MariaDB Database Engine \(p. 443\)](#)
- [Controlling Access with Security Groups \(p. 394\)](#)
- [Deleting a DB Instance \(p. 135\)](#)

Modifying a DB Instance Running the MariaDB Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS MariaDB DB instance, and describes the settings for MariaDB instances.

We recommend that you test any changes on a test instance before modifying a production instance, so that you fully understand the impact of each change. This is especially important when upgrading database versions.

After you modify your DB instance settings, you can apply the changes immediately, or apply them during the next maintenance window for the DB instance. Some modifications cause an interruption by restarting the DB instance.

AWS Management Console

To modify a MariaDB DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB instance** page appears.
4. Change any of the settings that you want. For information about each setting, see [Settings for MariaDB DB Instances \(p. 457\)](#).
5. When all the changes are as you want them, choose **Continue** and check the summary of modifications.
6. To apply the changes immediately, choose **Apply immediately**. Choosing this option can cause an outage in some cases. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify a MariaDB DB instance by using the AWS CLI, call the `modify-db-instance` command. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see [Settings for MariaDB DB Instances \(p. 457\)](#).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days). The code enables automatic minor version upgrades by using `--auto-minor-version-upgrade`. To disable automatic minor version upgrades, use `--no-auto-minor-version-upgrade`. The changes are applied during the next maintenance window by using `--no-apply-immediately`. Use `--apply-immediately` to apply the changes immediately. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
```

```
--db-instance-identifier mydbinstance \
--backup-retention-period 7 \
--auto-minor-version-upgrade \
--no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--backup-retention-period 7 ^
--auto-minor-version-upgrade ^
--no-apply-immediately
```

API

To modify a MariaDB instance by using the Amazon RDS API, call the [ModifyDBInstance](#) action. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see [Settings for MariaDB DB Instances \(p. 457\)](#).

Example

The following code modifies *mydbinstance* by setting the backup retention period to 1 week (7 days) and enabling automatic minor version upgrades. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
?Action=ModifyDBInstance
&ApplyImmediately=false
&AutoMinorVersionUpgrade=true
&BackupRetentionPeriod=7
&DBInstanceIdentifier=mydbinstance
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
&X-Amz-Date=20131016T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for MariaDB DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated storage	<p>The storage, in gigabytes, that you want to allocate for your DB instance.</p> <p>You can't modify allocated storage if the DB instance status is <i>storage-optimization</i> or if the allocated storage for the DB instance has been modified in the last six hours.</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	No downtime. Performance may be degraded during the change.

Setting	Setting Description	When the Change Occurs	Downtime Notes
	The maximum storage allowed depends on the storage type. For more information, see DB instance storage (p. 103) .		
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.	–	–
Backup retention period	<p>The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0.</p> <p>For more information, see Working With Backups (p. 208).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false and you change the setting from a non-zero value to another non-zero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.</p>	An outage occurs if you change from 0 to a non-zero value, or from a non-zero value to 0.
Backup window	<p>The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours.</p> <p>For more information, see Working With Backups (p. 208).</p>	The change is applied asynchronously, as soon as possible.	–
Certificate authority	The certificate that you want to use.	–	–
Copy tags to snapshots	<p>If you have any DB instance tags, this option copies them when you create a DB snapshot.</p> <p>For more information, see Tagging Amazon RDS Resources (p. 138).</p>	The change occurs immediately. This setting ignores the Apply immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Database port	<p>The port that you want to use to access the database.</p> <p>The port value must not match any of the port values specified for options in the option group for the DB instance.</p>	The change occurs immediately. This setting ignores the Apply immediately setting.	The DB instance is rebooted immediately.
DB engine version	<p>The version of the MariaDB database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications.</p> <p>For more information, see Upgrading the MariaDB DB Engine (p. 464).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change.
DB instance class	<p>The DB instance class that you want to use.</p> <p>For more information, see DB Instance Class (p. 82).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change.
DB instance identifier	<p>The DB instance identifier. This value is stored as a lowercase string.</p> <p>For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 126).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change. The DB instance is rebooted.
DB parameter group	<p>The parameter group that you want associated with the DB instance.</p> <p>For more information, see Working with DB Parameter Groups (p. 169).</p>	The parameter group change occurs immediately.	<p>An outage doesn't occur during this change. When you change the parameter group, changes to some parameters are applied to the DB instance immediately without a reboot. Changes to other parameters are applied only after the DB instance is rebooted.</p> <p>For more information, see Rebooting a DB Instance (p. 129).</p>

Setting	Setting Description	When the Change Occurs	Downtime Notes
Deletion protection	Enable deletion protection to prevent your DB instance from being deleted. For more information, see Deleting a DB Instance (p. 135) .	–	–
Enhanced monitoring	Enable enhanced monitoring to enable gathering metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 256) .	–	–
Log exports	Select the types of MariaDB database log files to publish to Amazon CloudWatch Logs. For more information, see MariaDB Database Log Files (p. 311) .	The change occurs immediately. This setting ignores the Apply immediately setting.	–
Maintenance window	The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours. If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied. For more information, see The Amazon RDS Maintenance Window (p. 120) .	The change occurs immediately. This setting ignores the Apply immediately setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.
Multi-AZ deployment	Yes to deploy your DB instance in multiple Availability Zones; otherwise, No . For more information, see Regions and Availability Zones (p. 101) .	If Apply immediately is set to true, the change occurs immediately. If Apply immediately is set to false, the change occurs during the next maintenance window.	–
New master password	The password for your master user. The password must contain from 8 to 41 alphanumeric characters.	The change is applied asynchronously, as soon as possible. This setting ignores the Apply immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Option group	<p>The option group that you want associated with the DB instance.</p> <p>For more information, see Working with Option Groups (p. 156).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	–
Public accessibility	<p>Yes to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. No to make the DB instance accessible only from inside the VPC.</p> <p>For more information, see Hiding a DB Instance in a VPC from the Internet (p. 422).</p>	The change occurs immediately. This setting ignores the Apply immediately setting.	–
Security group	<p>The security groups you want associated with the DB instance.</p> <p>For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 399).</p>	The change is applied asynchronously, as soon as possible. This setting ignores the Apply immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Storage type	<p>The storage type that you want to use.</p> <p>For more information, see Amazon RDS Storage Types (p. 103).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	<p>The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place.</p> <ul style="list-style-type: none"> • From General Purpose (SSD) to Magnetic. • From General Purpose (SSD) to Provisioned IOPS (SSD), if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ DB instance or for the source DB instance of a read replica. • From Magnetic to General Purpose (SSD). • From Magnetic to Provisioned IOPS (SSD). • From Provisioned IOPS (SSD) to Magnetic. • From Provisioned IOPS (SSD) to General Purpose (SSD), if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ

Setting	Setting Description	When the Change Occurs	Downtime Notes
			DB instance or for the source DB instance of a read replica.
Subnet Group	<p>The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC.</p> <p>For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 426).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change. The DB instance is rebooted.

Related Topics

- [Rebooting a DB Instance \(p. 129\)](#)
- [Connecting to a DB Instance Running the MariaDB Database Engine \(p. 452\)](#)
- [Upgrading the MariaDB DB Engine \(p. 464\)](#)
- [Deleting a DB Instance \(p. 135\)](#)

Upgrading the MariaDB DB Engine

When Amazon RDS supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades: major version upgrades and minor version upgrades. You must modify the DB instance manually to perform a major version upgrade. Minor version upgrades occur automatically if you enable auto minor version upgrades on your DB instance. In all other cases, you must modify the DB instance manually to perform a minor version upgrade.

For more information about MariaDB supported versions and version management, see [MariaDB on Amazon RDS Versions \(p. 434\)](#).

Topics

- [Overview of Upgrading \(p. 464\)](#)
- [Upgrading a MariaDB DB Instance \(p. 465\)](#)

Overview of Upgrading

Major version upgrades can contain database changes that are not backward-compatible with existing applications. As a result, Amazon RDS doesn't apply major version upgrades automatically; you must manually modify your DB instance. You should thoroughly test any upgrade before applying it to your production instances.

Unless you specify otherwise, your DB instance will automatically be upgraded to new MariaDB minor versions as they are supported by Amazon RDS. This patching occurs during your scheduled maintenance window. You can modify a DB instance to turn off automatic minor version upgrades.

Amazon RDS takes two DB snapshots during the upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken when the upgrade completes.

Note

Amazon RDS only takes DB snapshots if you have set the backup retention period for your DB instance to a number greater than 0. To change your backup retention period, see [Modifying a DB Instance Running the MariaDB Database Engine \(p. 456\)](#).

After the upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the first DB snapshot taken to create a new DB instance.

You control when to upgrade your DB instance to a new version supported by Amazon RDS. This level of control helps you maintain compatibility with specific database versions and test new versions with your application before deploying in production. When you are ready, you can perform version upgrades at the times that best fit your schedule.

If your DB instance is using read replication, you must upgrade all of the Read Replicas before upgrading the source instance.

If your DB instance is in a Multi-AZ deployment, both the primary and standby DB instances are upgraded. The primary and standby DB instances are upgraded at the same time and you will experience an outage until the upgrade is complete. The time for the outage varies based on your database engine, engine version, and the size of your DB instance.

If you are using a custom parameter group, and you perform a major version upgrade, you must specify either a default parameter group for the new DB engine version or create your own custom parameter group for the new DB engine version. Associating the new parameter group with the DB instance requires a customer-initiated database reboot after the upgrade completes. The instance's parameter group

status will show pending-reboot if the instance needs to be rebooted to apply the parameter group changes. An instance's parameter group status can be viewed in the AWS console or by using a "describe" call such as describe-db-instances.

Upgrading a MariaDB DB Instance

For information about manually or automatically upgrading a MariaDB DB instance, see [Upgrading a DB Instance Engine Version \(p. 123\)](#).

Migrating Data from a MySQL DB Snapshot to a MariaDB DB Instance

You can migrate an Amazon RDS MySQL DB snapshot to a new DB instance running MariaDB 10.1 using the AWS Management Console, AWS CLI, or Amazon RDS API. You must create the DB snapshot from an Amazon RDS DB instance running MySQL 5.6. To learn how to create an RDS MySQL DB snapshot, see [Creating a DB Snapshot \(p. 216\)](#).

After you migrate from MySQL to MariaDB, the MariaDB DB instance will be associated with the default DB parameter group and option group. After you restore the DB snapshot, you can associate a custom DB parameter group for the new DB instance. However, a MariaDB parameter group has a different set of configurable system variables. For information about the differences between MySQL and MariaDB system variables, see [System Variable Differences Between MariaDB 10.0 and MySQL 5.6](#). To learn about DB parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#). To learn about option groups, see [Working with Option Groups \(p. 156\)](#).

Incompatibilities Between MariaDB and MySQL

Incompatibilities between MySQL and MariaDB include the following:

- You can't migrate a DB snapshot created with MySQL 5.5 to MariaDB 10.1.
- You can't migrate a DB snapshot created with MySQL 5.6.40 or higher 5.6 version to MariaDB.
- You can't migrate a DB snapshot created with MySQL 5.7 to MariaDB.
- You can't migrate a DB snapshot created with MySQL 8.0 to MariaDB.
- You can't migrate an encrypted snapshot.
- If the source MySQL database uses a SHA256 password hash, you need to reset user passwords that are SHA256 hashed before you can connect to the MariaDB database. The following code shows how to reset a password that is SHA256 hashed:

```
SET old_passwords = 0;
UPDATE mysql.user SET plugin = 'mysql_native_password',
Password = PASSWORD('new_password')
WHERE (User, Host) = ('master_user_name', %);
FLUSH PRIVILEGES;
```

- If your RDS master user account uses the SHA-256 password hash, the password has to be reset using the RDS [modify-db-instance](#) AWS CLI command, [ModifyDBInstance](#) API operation, or the AWS Management Console. For information about modifying a MariaDB DB instance, see [Modifying a DB Instance Running the MariaDB Database Engine \(p. 456\)](#).
- MariaDB doesn't support the Memcached plugin; however, the data used by the Memcached plugin is stored as InnoDB tables. After you migrate a MySQL DB snapshot, you can access the data used by the Memcached plugin using SQL. For more information about the innodb_memcache database, see [InnoDB memcached Plugin Internals](#).

AWS Management Console

To migrate a MySQL DB snapshot to a MariaDB DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.

2. In the navigation pane, choose **Snapshots**, and then select the MySQL DB snapshot you want to migrate.
3. For **Actions**, choose **Migrate Snapshot**. The **Migrate Database** page appears.
4. For **Migrate to DB Engine**, choose **mariadb**.
5. On the **Migrate Database** page, provide additional information that RDS needs to launch the MariaDB DB instance.
 - **DB Engine Version:** Choose the version of the MariaDB database engine that you want to use. For more information, see [Upgrading the MariaDB DB Engine \(p. 464\)](#).
 - **DB Instance Class:** Choose a DB instance class that has the required storage and capacity for your database, for example db.r3.large. For any production application that requires fast and consistent I/O performance, we recommend Provisioned IOPS storage. For more information, see [Provisioned IOPS SSD Storage \(p. 105\)](#). MariaDB 10.1 does not support previous-generation DB instance classes. For more information, see [DB Instance Class \(p. 82\)](#).
 - **Multi-AZ Deployment:** Choose **Yes** to deploy your DB instance in multiple Availability Zones; otherwise, **No**. For more information, see [Regions and Availability Zones \(p. 101\)](#).
 - **DB Snapshot ID:** Type a name for the DB snapshot identifier.

The DB snapshot identifier has the following constraints:

- It must contain from 1 to 255 alphanumeric characters or hyphens.
- The character must be a letter.
- It cannot end with a hyphen or contain two consecutive hyphens.

If you are restoring from a shared manual DB snapshot, the DB snapshot identifier must be the Amazon Resource Name (ARN) of the shared DB snapshot.

- **DB Instance Identifier:** Type a name for the DB instance that is unique for your account in the AWS Region where the DB instance will reside. This identifier is used in the endpoint addresses for the instances in your DB instance.

The DB instance identifier has the following constraints:

- It must contain from 1 to 63 alphanumeric characters or hyphens.
- Its first character must be a letter.
- It cannot end with a hyphen or contain two consecutive hyphens.
- It must be unique for all DB instances for your AWS account, within an AWS Region.
- **Virtual Private Cloud (VPC):** If you have an existing VPC, then you can use that VPC with your MariaDB DB instance by selecting your VPC identifier, for example vpc-a464d1c1. For more information about VPC, see [Amazon Virtual Private Cloud \(VPCs\) and Amazon RDS \(p. 412\)](#).

Otherwise, you can choose to have Amazon RDS create a VPC for you by selecting Create a new VPC.

You cannot create MariaDB instances in the EC2 Classic Network.

- **Subnet group:** If you have an existing subnet group, then you can use that subnet group with your MariaDB DB instance by selecting your subnet group identifier, for example gs-subnet-group1.

Otherwise, you can choose to have Amazon RDS create a subnet group for you by selecting Create a new subnet group.

- **Public accessibility:** Choose **No** to specify that instances in your DB instance can only be accessed by resources inside your VPC. Choose **Yes** to specify that instances in your DB instance can be accessed by resources on the public network. The default is **Yes**.
- **Availability zone:** Choose the **Availability Zone** to host the primary instance for your MariaDB DB instance. To have Amazon RDS choose an **Availability Zone** for you, choose **No Preference**.

- **Database Port:** Type the default port to be used when connecting to instances in the DB instance. The default is 3306.

You might be behind a corporate firewall that doesn't allow access to default ports such as the MySQL default port 3306. In this case, provide a port value that your corporate firewall allows.

- **Option Group:** Choose the option group that you want associated with the DB instance. For more information, see [Working with Option Groups \(p. 156\)](#).
- **Encryption:** Choose **Enable Encryption** for your new MariaDB DB instance to be encrypted "at rest." If you choose **Enable Encryption**, you will be required to choose an AWS KMS encryption key as the **Master Key** value.
- **Auto minor version upgrade:** Choose **Enable auto minor version upgrade** to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window. The **Auto minor version upgrade** option only applies to upgrades to MySQL minor engine versions for your MariaDB DB instance. It doesn't apply to regular patches applied to maintain system stability.

Migrate Database

Migrate this database to a new DB Engine by selecting your desired options for the migrated instance.

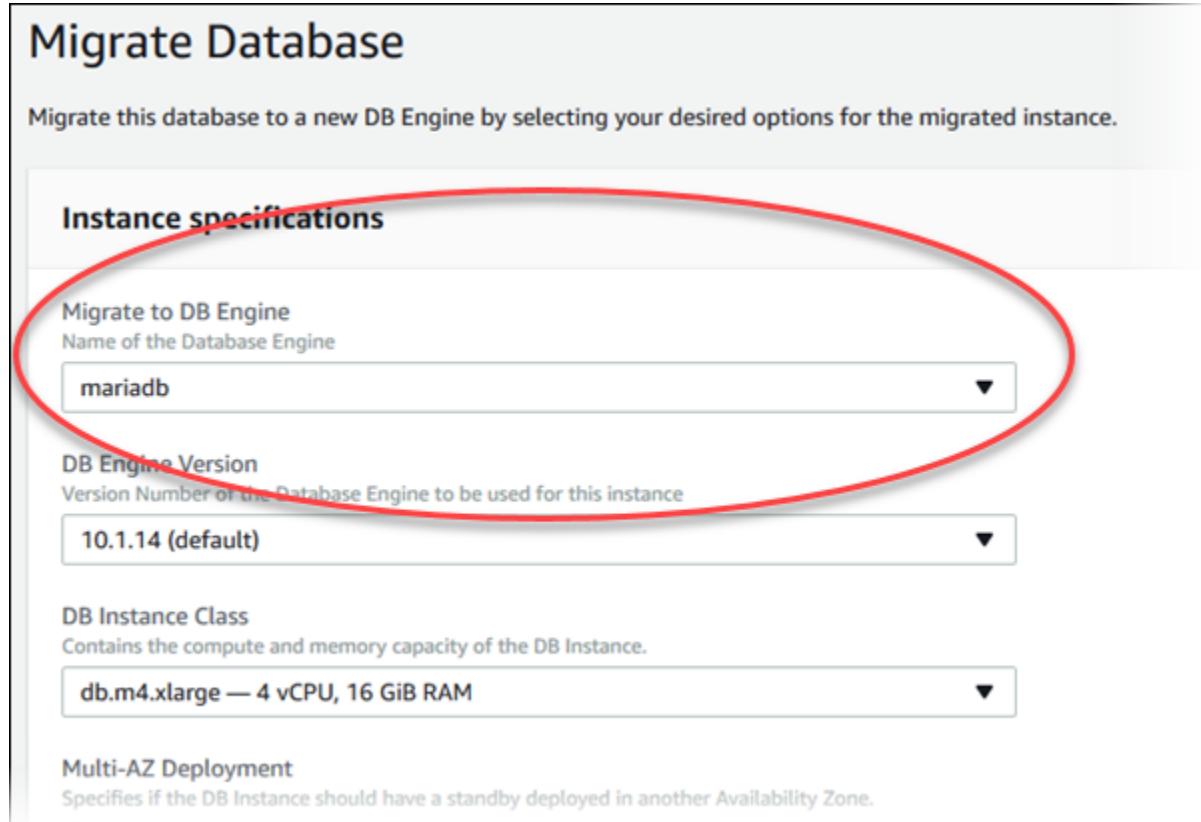
Instance specifications

Migrate to DB Engine
Name of the Database Engine
mariadb

DB Engine Version
Version Number of the Database Engine to be used for this instance
10.1.14 (default)

DB Instance Class
Contains the compute and memory capacity of the DB Instance.
db.m4.xlarge — 4 vCPU, 16 GiB RAM

Multi-AZ Deployment
Specifies if the DB Instance should have a standby deployed in another Availability Zone.



6. Choose **Migrate**.

CLI

To migrate data from a MySQL DB snapshot to a MariaDB DB instance, use the AWS CLI [restore-db-instance-from-db-snapshot](#) command with the following parameters:

- **--db-instance-identifier** – Name of the DB instance to create from the DB snapshot.

- **--db-snapshot-identifier** – The identifier for the DB snapshot to restore from.
- **--engine** – The database engine to use for the new instance.

Example

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-from-db-snapshot \
--db-instance-identifier newmariadbinstance \
--db-snapshot-identifier mysqlsnapshot \
--engine mariadb
```

For Windows:

```
aws rds restore-db-instance-from-db-snapshot \
--db-instance-identifier newmariadbinstance ^
--db-snapshot-identifier mysqlsnapshot ^
--engine mariadb
```

API

To migrate data from a MySQL DB snapshot to a MariaDB DB instance, call the Amazon RDS API action [RestoreDBInstanceFromDBSnapshot](#).

Example

```
https://rds.us-west-2.amazonaws.com/
?Action=RestoreDBInstanceFromDBSnapshot
&DBInstanceIdentifier= newmariadbinstance
&DBSnapshotIdentifier= mysqlsnapshot
&Engine= mariadb
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2013-09-09
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-west-2/rds/aws4_request
&X-Amz-Date=20140428T232655Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=78ac761e8c8f54a8c0727f4e67ad0a766fbb0024510b9aa34ea6d1f7df52fe92
```

Related Topics

- [Creating a DB Snapshot \(p. 216\)](#)
- [System Variable Differences Between MariaDB 10.0 and MySQL 5.6](#)
- [Working with DB Parameter Groups \(p. 169\)](#)
- [Working with Option Groups \(p. 156\)](#)

Working with MariaDB Replication

You usually use Read Replicas to configure replication between Amazon RDS DB instances. For general information about Read Replicas, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#). For specific information about working with Read Replicas on Amazon RDS MariaDB, see [Working with MariaDB Read Replicas \(p. 470\)](#).

You can also configure replication based on binary log coordinates for a MariaDB DB instance. For MariaDB instances, you can also configure replication based on global transaction IDs (GTIDs), which provides better crash safety. For more information, see [Configuring GTID-Based Replication into an Amazon RDS MariaDB DB instance \(p. 473\)](#).

The following are other replication options available with Amazon RDS MariaDB:

- You can set up replication between an Amazon RDS MariaDB DB instance and a MySQL or MariaDB instance that is external to Amazon RDS. For information about configuring replication with an external source, see [Replication with a MySQL or MariaDB Instance Running External to Amazon RDS \(p. 667\)](#).
- You can configure replication to import databases from a MySQL or MariaDB instance that is external to Amazon RDS, or to export databases to such instances. For more information, see [Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime \(p. 640\)](#) and [Exporting Data from a MySQL DB Instance by Using Replication \(p. 673\)](#).

For any of these replication options, you can use either row-based replication, statement-based, or mixed replication. Row-based replication only replicates the changed rows that result from a SQL statement. Statement-based replication replicates the entire SQL statement. Mixed replication uses statement-based replication when possible, but switches to row-based replication when SQL statements that are unsafe for statement-based replication are executed. In most cases, mixed replication is recommended. The binary log format of the DB instance determines whether replication is row-based, statement-based, or mixed. For information about setting the binary log format, see [Binary Logging Format \(p. 316\)](#).

Topics

- [Working with MariaDB Read Replicas \(p. 470\)](#)
- [Configuring GTID-Based Replication into an Amazon RDS MariaDB DB instance \(p. 473\)](#)

Working with MariaDB Read Replicas

This section contains specific information about working with Read Replicas on Amazon RDS MariaDB. For general information about Read Replicas and instructions for using them, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

Topics

- [Read Replica Configuration with MariaDB \(p. 471\)](#)
- [Read Replica Updates with MariaDB \(p. 471\)](#)
- [Multi-AZ Read Replica Deployments with MariaDB \(p. 471\)](#)
- [Monitoring MariaDB Read Replicas \(p. 471\)](#)
- [Starting and Stopping Replication with MariaDB Read Replicas \(p. 472\)](#)
- [Deleting Read Replicas with MariaDB \(p. 472\)](#)
- [Troubleshooting a MariaDB Read Replica Problem \(p. 472\)](#)

Read Replica Configuration with MariaDB

Before a MariaDB DB instance can serve as a replication source, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0. This requirement also applies to a Read Replica that is the source DB instance for another Read Replica.

You can create up to five Read Replicas from one DB instance. For replication to operate effectively, each Read Replica should have as the same amount of compute and storage resources as the source DB instance. If you scale the source DB instance, you should also scale the Read Replicas.

If a Read Replica is running any version of MariaDB, you can specify it as the source DB instance for another Read Replica. For example, you can create ReadReplica1 from MyDBInstance, and then create ReadReplica2 from ReadReplica1. Updates made to MyDBInstance are replicated to ReadReplica1 and then replicated from ReadReplica1 to ReadReplica2. You can't have more than four instances involved in a replication chain. For example, you can create ReadReplica1 from MySourceDBInstance, and then create ReadReplica2 from ReadReplica1, and then create ReadReplica3 from ReadReplica2, but you can't create a ReadReplica4 from ReadReplica3.

If you promote a MariaDB Read Replica that is in turn replicating to other Read Replicas, those Read Replicas remain active. Consider an example where MyDBInstance1 replicates to MyDBInstance2, and MyDBInstance2 replicates to MyDBInstance3. If you promote MyDBInstance2, replication from MyDBInstance1 to MyDBInstance2 no longer occurs, but MyDBInstance2 still replicates to MyDBInstance3.

To enable automatic backups on a Read Replica for Amazon RDS MariaDB, first create the Read Replica, then modify the Read Replica to enable automatic backups.

You can run multiple concurrent Read Replica create or delete actions that reference the same source DB instance, as long as you stay within the limit of five Read Replicas for the source instance.

Read Replica Updates with MariaDB

Read Replicas are designed to support read queries, but you might need occasional updates. For example, you might need to add an index to speed the specific types of queries accessing the replica. You can enable updates by setting the `read_only` parameter to **0** in the DB parameter group for the Read Replica.

Multi-AZ Read Replica Deployments with MariaDB

You can create a Read Replica from either single-AZ or Multi-AZ DB instance deployments. You use Multi-AZ deployments to improve the durability and availability of critical data, but you can't use the Multi-AZ secondary to serve read-only queries. Instead, you can create Read Replicas from high-traffic Multi-AZ DB instances to offload read-only queries. If the source instance of a Multi-AZ deployment fails over to the secondary, any associated Read Replicas automatically switch to use the secondary (now primary) as their replication source. For more information, see [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#).

You can create a Read Replica as a Multi-AZ DB instance. Amazon RDS creates a standby of your replica in another Availability Zone for failover support for the replica. Creating your Read Replica as a Multi-AZ DB instance is independent of whether the source database is a Multi-AZ DB instance.

Monitoring MariaDB Read Replicas

For MariaDB Read Replicas, you can monitor replication lag in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric. The `ReplicaLag` metric reports the value of the `Seconds_Behind_Master` field of the `SHOW SLAVE STATUS` command.

Common causes for replication lag for MariaDB are the following:

- A network outage.

- Writing to tables with indexes on a Read Replica. If the `read_only` parameter is not set to 0 on the Read Replica, it can break replication.
- Using a nontransactional storage engine such as MyISAM. Replication is only supported for the InnoDB storage engine on MariaDB 10.2 and later and the XtraDB storage engine on MariaDB 10.1 and earlier.

When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance. If the `ReplicaLag` metric returns -1, then replication is currently not active. `ReplicaLag = -1` is equivalent to `Seconds_Behind_Master = NULL`.

Starting and Stopping Replication with MariaDB Read Replicas

You can stop and restart the replication process on an Amazon RDS DB instance by calling the system stored procedures [mysql.rds_stop_replication \(p. 706\)](#) and [mysql.rds_start_replication \(p. 704\)](#). You can do this when replicating between two Amazon RDS instances for long-running operations such as creating large indexes. You also need to stop and start replication when importing or exporting databases. For more information, see [Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime \(p. 640\)](#) and [Exporting Data from a MySQL DB Instance by Using Replication \(p. 673\)](#).

If replication is stopped for more than 30 consecutive days, either manually or due to a replication error, Amazon RDS terminates replication between the master DB instance and all Read Replicas. It does so to prevent increased storage requirements on the master DB instance and long failover times. The Read Replica DB instance is still available. However, replication can't be resumed because the binary logs required by the Read Replica are deleted from the master DB instance after replication is terminated. You can create a new Read Replica for the master DB instance to reestablish replication.

Deleting Read Replicas with MariaDB

You must explicitly delete Read Replicas, using the same mechanisms for deleting a DB instance. If you delete the source DB instance without deleting the replicas, each replica is promoted to a standalone DB instance.

Troubleshooting a MariaDB Read Replica Problem

The replication technologies for MariaDB are asynchronous. Because they are asynchronous, occasional `BinLogDiskUsage` increases on the source DB instance and `ReplicaLag` on the Read Replica are to be expected. For example, a high volume of write operations to the source DB instance can occur in parallel. In contrast, write operations to the Read Replica are serialized using a single I/O thread, which can lead to a lag between the source instance and Read Replica. For more information about read-only replicas in the MariaDB documentation, go to [Replication Overview](#).

You can do several things to reduce the lag between updates to a source DB instance and the subsequent updates to the Read Replica, such as the following:

- Sizing a Read Replica to have a storage size and DB instance class comparable to the source DB instance.
- Ensuring that parameter settings in the DB parameter groups used by the source DB instance and the Read Replica are compatible. For more information and an example, see the discussion of the `max_allowed_packet` parameter later in this section.

Amazon RDS monitors the replication status of your Read Replicas and updates the `Replication State` field of the Read Replica instance to `Error` if replication stops for any reason. An example might be if DML queries run on your Read Replica conflict with the updates made on the source DB instance.

You can review the details of the associated error thrown by the MariaDB engine by viewing the `Replication Error` field. Events that indicate the status of the Read Replica are also generated,

including [RDS-EVENT-0045 \(p. 291\)](#), [RDS-EVENT-0046 \(p. 292\)](#), and [RDS-EVENT-0047 \(p. 291\)](#). For more information about events and subscribing to events, see [Using Amazon RDS Event Notification \(p. 287\)](#). If a MariaDB error message is returned, review the error in the [MariaDB error message documentation](#).

One common issue that can cause replication errors is when the value for the `max_allowed_packet` parameter for a Read Replica is less than the `max_allowed_packet` parameter for the source DB instance. The `max_allowed_packet` parameter is a custom parameter that you can set in a DB parameter group that is used to specify the maximum size of DML code that can be executed on the database. In some cases, the `max_allowed_packet` parameter value in the DB parameter group associated with a source DB instance is smaller than the `max_allowed_packet` parameter value in the DB parameter group associated with the source's Read Replica. In these cases, the replication process can throw an error (Packet bigger than '`max_allowed_packet`' bytes) and stop replication. You can fix the error by having the source and Read Replica use DB parameter groups with the same `max_allowed_packet` parameter values.

Other common situations that can cause replication errors include the following:

- Writing to tables on a Read Replica. If you are creating indexes on a Read Replica, you need to have the `read_only` parameter set to `0` to create the indexes. If you are writing to tables on the Read Replica, it might break replication.
- Using a non-transactional storage engine such as MyISAM. Read Replicas require a transactional storage engine. Replication is only supported for the InnoDB storage engine on MariaDB 10.2 and later and the XtraDB storage engine on MariaDB 10.1 and earlier.
- Using unsafe nondeterministic queries such as `SYSDATE()`. For more information, see [Determination of Safe and Unsafe Statements in Binary Logging](#).

If you decide that you can safely skip an error, you can follow the steps described in the section [Skipping the Current Replication Error \(p. 685\)](#). Otherwise, you can delete the Read Replica and create an instance using the same DB instance identifier so that the endpoint remains the same as that of your old Read Replica. If a replication error is fixed, the `Replication State` changes to *replicating*.

For MariaDB DB instances, in some cases Read Replicas can't be switched to the secondary if some binlog events aren't flushed during the failure. In these cases, you must manually delete and recreate the Read Replicas. You can reduce the chance of this happening by setting the following dynamic variable values: `sync_binlog=1`, `innodb_flush_log_at_trx_commit=1`, and `innodb_support_xa=1`. These settings might reduce performance, so test their impact before implementing the changes in a production environment.

Configuring GTID-Based Replication into an Amazon RDS MariaDB DB instance

You can set up GTID-based replication from an external MariaDB instance of version 10.0.24 or greater into an Amazon RDS MariaDB DB instance. Be sure to follow these guidelines when you set up an external replication master and a replica on Amazon RDS:

- Monitor failover events for the Amazon RDS MariaDB DB instance that is your replica. If a failover occurs, then the DB instance that is your replica might be recreated on a new host with a different network address. For information on how to monitor failover events, see [Using Amazon RDS Event Notification \(p. 287\)](#).
- Maintain the binlogs on your master instance until you have verified that they have been applied to the replica. This maintenance ensures that you can restore your master instance in the event of a failure.
- Turn on automated backups on your MariaDB DB instance on Amazon RDS. Turning on automated backups ensures that you can restore your replica to a particular point in time if you need to re-

synchronize your master and replica. For information on backups and Point-In-Time Restore, see [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#).

Note

The permissions required to start replication on an Amazon RDS MariaDB DB instance are restricted and not available to your Amazon RDS master user. Because of this, you must use the Amazon RDS [mysql.rds_set_external_master_gtid \(p. 485\)](#) and [mysql.rds_start_replication \(p. 704\)](#) commands to set up replication between your live database and your Amazon RDS MariaDB database.

To start replication between an external master instance and a MariaDB DB instance on Amazon RDS, use the following procedure.

To Start Replication

1. Make the source MariaDB instance read-only:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Get the current GTID of the external MariaDB instance. You can do this by using mysql or the query editor of your choice to run `SELECT @@gtid_current_pos;`.

The GTID is formatted as <domain-id>-<server-id>-<sequence-id>. A typical GTID looks something like **0-1234510749-1728**. For more information about GTIDs and their component parts, see [Global Transaction ID](#) in the MariaDB documentation.

3. Copy the database from the external MariaDB instance to the Amazon RDS MariaDB DB instance using mysqldump. For very large databases, you might want to use the procedure in [Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime \(p. 640\)](#).

Note

Make sure there is not a space between the `-p` option and the entered password.

For Linux, OS X, or Unix:

```
mysqldump \
    --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u <local_user> \
    -p<local_password> | mysql \
        --host=hostname \
        --port=3306 \
        -u <RDS_user_name> \
        -p <RDS_password>
```

For Windows:

```
mysqldump ^
    --databases <database_name> ^
    --single-transaction ^
    --compress ^
    --order-by-primary \
    -u <local_user> \
    -p<local_password> | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u <RDS_user_name> ^
```

```
-p <RDS_password>
```

Use the `--host`, `--user (-u)`, `--port` and `-p` options in the `mysql` command to specify the host name, user name, port, and password to connect to your Amazon RDS MariaDB DB instance. The host name is the DNS name from the Amazon RDS MariaDB DB instance endpoint, for example `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.

4. Make the source MariaDB instance writeable again:

```
mysql> SET GLOBAL read_only = OFF;  
mysql> UNLOCK TABLES;
```

5. In the Amazon RDS Management Console, add the IP address of the server that hosts the external MariaDB database to the VPC security group for the Amazon RDS MariaDB DB instance. For more information on modifying a VPC security group, go to [Security Groups for Your VPC](#) in the *Amazon Virtual Private Cloud User Guide*.

You might also need to configure your local network to permit connections from the IP address of your Amazon RDS MariaDB DB instance, so that it can communicate with your external MariaDB instance. To find the IP address of the Amazon RDS MariaDB DB instance, use the `host` command:

```
host <RDS_MariaDB_DB_host_name>
```

The host name is the DNS name from the Amazon RDS MariaDB DB instance endpoint.

6. Using the client of your choice, connect to the external MariaDB instance and create a MariaDB user to be used for replication. This account is used solely for replication and must be restricted to your domain to improve security. The following is an example:

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';
```

7. For the external MariaDB instance, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. For example, to grant the `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the '`repl_user`' user for your domain, issue the following command:

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.*  
TO 'repl_user'@'mydomain.com'  
IDENTIFIED BY '<password>';
```

8. Make the Amazon RDS MariaDB DB instance the replica. Connect to the Amazon RDS MariaDB DB instance as the master user and identify the external MariaDB database as the replication master by using the [mysql.rds_set_external_master_gtid \(p. 485\)](#) command. Use the GTID that you determined in Step 2. The following is an example:

```
CALL mysql.rds_set_external_master_gtid ('mymasterserver.mydomain.com', 3306,  
'repl_user', '<password>', '<GTID>', 0);
```

9. On the Amazon RDS MariaDB DB instance, issue the [mysql.rds_start_replication \(p. 704\)](#) command to start replication:

```
CALL mysql.rds_start_replication;
```

Importing Data into a MariaDB DB Instance

Following, you can find information about methods to import your MariaDB data to an Amazon RDS DB instance running MariaDB.

To do an initial data import into a MariaDB DB instance, you can use the procedures documented in [Restoring a Backup into an Amazon RDS MySQL DB Instance \(p. 631\)](#), as follows:

- To move data from an Amazon RDS MySQL DB instance, a MariaDB or MySQL instance in Amazon Elastic Compute Cloud (Amazon EC2) in the same VPC as your Amazon RDS MariaDB DB instance, or a small on-premises instance of MariaDB or MySQL, you can use the procedure documented in [Importing Data from a MySQL or MariaDB DB to an Amazon RDS MySQL or MariaDB DB Instance \(p. 638\)](#).
- To move data from a large or production on-premises instance of MariaDB or MySQL, you can use the procedure documented in [Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime \(p. 640\)](#).
- To move data from an instance of MariaDB or MySQL that is in EC2 in a different VPC than your Amazon RDS MariaDB DB instance, or to move data from any data source that can output delimited text files, you can use the procedure documented in [Importing Data From Any Source to a MySQL or MariaDB DB Instance \(p. 652\)](#).

You can also use AWS Database Migration Service (AWS DMS) to import data into an Amazon RDS DB instance. AWS DMS can migrate databases without downtime and, for many database engines, continue ongoing replication until you are ready to switch over to the target database. You can migrate to MariaDB from either the same database engine or a different database engine using AWS DMS. If you are migrating from a different database engine, you can use the AWS Schema Conversion Tool to migrate schema objects that are not migrated by AWS DMS. For more information about AWS DMS, see [What is AWS Database Migration Service](#).

You can configure replication into an Amazon RDS MariaDB DB instance using MariaDB global transaction identifiers (GTIDs) when the external instance is MariaDB version 10.0.24 or greater, or using binary log coordinates for MySQL instances or MariaDB instances on earlier versions than 10.0.24. Note that MariaDB GTIDs are implemented differently than MySQL GTIDs, which are not supported by Amazon RDS.

To configure replication into a MariaDB DB instance, you can use the following procedures:

- To configure replication into a MariaDB DB instance from an external MySQL instance or an external MariaDB instance running a version prior to 10.0.24, you can use the procedure documented in [Replication with a MySQL or MariaDB Instance Running External to Amazon RDS \(p. 667\)](#).
- To configure replication into a MariaDB DB instance from an external MariaDB instance running version 10.0.24 or greater, you can use the procedure documented in [Configuring GTID-Based Replication into an Amazon RDS MariaDB DB Instance \(p. 473\)](#).

Note

The mysql system database contains authentication and authorization information required to log into your DB instance and access your data. Dropping, altering, renaming, or truncating tables, data, or other contents of the mysql database in your DB instance can result in errors and might render the DB instance and your data inaccessible. If this occurs, the DB instance can be restored from a snapshot using the AWS CLI `restore-db-instance-from-db-snapshot` or recovered using `restore-db-instance-to-point-in-time` commands.

Options for MariaDB Database Engine

This appendix describes options, or additional features, that are available for Amazon RDS instances running the MariaDB DB engine. To enable these options, you add them to a custom option group, and then associate the option group with your DB instance. For more information about working with option groups, see [Working with Option Groups \(p. 156\)](#).

Amazon RDS supports the following options for MariaDB:

Option ID	Engine Versions
MARIADB_AUDIT_PLUGIN	MariaDB 10.0.24 and later

MariaDB Audit Plugin Support

Amazon RDS supports using the MariaDB Audit Plugin on MariaDB database instances. The MariaDB Audit Plugin records database activity such as users logging on to the database, queries run against the database, and more. The record of database activity is stored in a log file.

Audit Plugin Option Settings

Amazon RDS supports the following settings for the MariaDB Audit Plugin option.

Option Setting	Valid Values	Default Value	Description
SERVER_AUDIT	<code>/rdsdbdata/log/audit/</code>	<code>/rdsdbdata/log/audit/</code>	The location of the log file. The log file contains the record of the activity specified in SERVER_AUDIT_EVENTS. For more information, see Viewing and Listing Database Log Files (p. 307) and MariaDB Database Log Files (p. 311) .
SERVER_AUDIT_SIZE	<code>1000000000</code>	<code>1000000</code>	The size in bytes that when reached, causes the file to rotate. For more information, see Log File Size (p. 315) .
SERVER_AUDIT_ROTATIONS	<code>9</code>		The number of log rotations to save. For more information, see Log File Size (p. 315) and Downloading a Database Log File (p. 307) .
SERVER_AUDIT_EVENTS	<code>CONNECT, QUERY, TABLE</code>	<code>CONNECT, QUERY</code>	<p>The types of activity to record in the log. Installing the MariaDB Audit Plugin is itself logged.</p> <ul style="list-style-type: none">• CONNECT: Log successful and unsuccessful connections to the database, and disconnections from the database.• QUERY: Log the text of all queries run against the database.• TABLE: Log tables affected by queries when the queries are run against the database. <p>For MariaDB, CONNECT, QUERY, and TABLE are supported.</p>

Option Setting	Valid Values	Default Value	Description
			For MySQL, CONNECT and QUERY are supported.
SERVER_AUDIT	Multiple comma-separated values	None	Include only activity from the specified users. By default, activity is recorded for all users. If a user is specified in both SERVER_AUDIT_EXCL_USERS and SERVER_AUDIT_INCL_USERS, then activity is recorded for the user.
SERVER_AUDIT	Multiple comma-separated values	None	Exclude activity from the specified users. By default, activity is recorded for all users. If a user is specified in both SERVER_AUDIT_EXCL_USERS and SERVER_AUDIT_INCL_USERS, then activity is recorded for the user. <p>The rdsadmin user queries the database every second to check the health of the database. Depending on your other settings, this activity can possibly cause the size of your log file to grow very large, very quickly. If you don't need to record this activity, add the rdsadmin user to the SERVER_AUDIT_EXCL_USERS list.</p> <p>Note CONNECT activity is always recorded for all users, even if the user is specified for this option setting.</p>
SERVER_AUDIT_LOGGING	ON	ON	Logging is active. The only valid value is ON. Amazon RDS does not support deactivating logging. If you want to deactivate logging, remove the MariaDB Audit Plugin. For more information, see Removing the MariaDB Audit Plugin (p. 479) .
SERVER_AUDIT_QUERY_LENGTH	002147483647M1024		The limit on the length of the query string in a record.

Adding the MariaDB Audit Plugin

The general process for adding the MariaDB Audit Plugin to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the MariaDB Audit Plugin, you don't need to restart your DB instance. As soon as the option group is active, auditing begins immediately.

To add the MariaDB Audit Plugin

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group. Choose **mariadb** for **Engine**, and choose **10.0** or later for **Major engine version**. For more information, see [Creating an Option Group \(p. 157\)](#).

2. Add the **MARIADB_AUDIT_PLUGIN** option to the option group, and configure the option settings. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#). For more information about each setting, see [Audit Plugin Option Settings \(p. 477\)](#).
3. Apply the option group to a new or existing DB instance.
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the MariaDB Database Engine \(p. 443\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the MariaDB Database Engine \(p. 456\)](#).

Viewing and Downloading the MariaDB Audit Plugin Log

After you enable the MariaDB Audit Plugin, you access the results in the log files the same way you access any other text-based log files. The audit log files are located at `/rdsdbdata/log/audit/`. For information about viewing the log file in the console, see [Viewing and Listing Database Log Files \(p. 307\)](#). For information about downloading the log file, see [Downloading a Database Log File \(p. 307\)](#).

Modifying MariaDB Audit Plugin Settings

After you enable the MariaDB Audit Plugin, you can modify settings for the plugin. For more information about how to modify option settings, see [Modifying an Option Setting \(p. 164\)](#). For more information about each setting, see [Audit Plugin Option Settings \(p. 477\)](#).

Removing the MariaDB Audit Plugin

Amazon RDS doesn't support turning off logging in the MariaDB Audit Plugin. However, you can remove the plugin from a DB instance. When you remove the MariaDB Audit Plugin, the DB instance is restarted automatically to stop auditing.

To remove the MariaDB Audit Plugin from a DB instance, do one of the following:

- Remove the MariaDB Audit Plugin option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#)
- Modify the DB instance and specify a different option group that doesn't include the plugin. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see [Modifying a DB Instance Running the MariaDB Database Engine \(p. 456\)](#).

Parameters for MariaDB

By default, a MariaDB DB instance uses a DB parameter group that is specific to a MariaDB database. This parameter group contains some but not all of the parameters contained in the Amazon RDS DB parameter groups for the MySQL database engine. It also contains a number of new, MariaDB-specific parameters. The following MySQL parameters are not available in MariaDB-specific DB parameter groups:

- bind_address
- binlog_error_action
- binlog_gtid_simple_recovery
- binlog_max_flush_queue_time
- binlog_order_commits
- binlog_row_image
- binlog_rows_query_log_events
- binlogging_impossible_mode
- block_encryption_mode
- core_file
- default_tmp_storage_engine
- div_precision_increment
- end_markers_in_json
- enforce_gtid_consistency
- eq_range_index_dive_limit
- explicit_defaults_for_timestamp
- gtid_executed
- gtid-mode
- gtid_next
- gtid_owned
- gtid_purged
- log_bin_basename
- log_bin_index
- log_bin_use_v1_row_events
- log_slow_admin_statements
- log_slow_slave_statements
- log_throttle_queries_not_using_indexes
- master-info-repository
- optimizer_trace
- optimizer_trace_features
- optimizer_trace_limit
- optimizer_trace_max_mem_size
- optimizer_trace_offset
- relay_log_info_repository
- rpl_stop_slave_timeout
- slave_parallel_workers
- slave_pending_jobs_size_max
- slave_rows_search_algorithms

- storage_engine
- table_open_cache_instances
- timed_mutexes
- transaction_allow_batching
- validate_password
- validate_password_dictionary_file
- validate_password_length
- validate_password_mixed_case_count
- validate_password_number_count
- validate_password_policy
- validate_password_special_char_count

For more information on MySQL 5.6 parameters, go to the [MySQL 5.6 documentation](#).

The MariaDB-specific DB parameter groups also contain the following parameters that are applicable to MariaDB only. Acceptable ranges for the modifiable parameters are the same as specified in the MariaDB documentation except where noted. Amazon RDS MariaDB parameters are set to the default values of the storage engine you have selected.

- aria_block_size
- aria_checkpoint_interval
- aria_checkpoint_log_activity
- aria_force_start_after_recovery_failures
- aria_group_commit
- aria_group_commit_interval
- aria_log_dir_path
- aria_log_file_size
- aria_log_purge_type
- aria_max_sort_file_size
- aria_page_checksum
- aria_pagecache_age_threshold
- aria_pagecache_division_limit
- aria_recover

Amazon RDS MariaDB supports the values of NORMAL, OFF, and QUICK, but not FORCE or BACKUP.

- aria_repair_threads
- aria_sort_buffer_size
- aria_stats_method
- aria_sync_log_dir
- binlog_annotation_row_events
- binlog_commit_wait_count
- binlog_commit_wait_usec
- binlog_row_image (MariaDB version 10.1 and later)
- deadlock_search_depth_long
- deadlock_search_depth_short
- deadlock_timeout_long
- deadlock_timeout_short
- explicit_defaults_for_timestamp (MariaDB version 10.1 and later)

- extra_max_connections
- extra_port
- feedback
- feedback_send_retry_wait
- feedback_send_timeout
- feedback_url
- feedback_user_info
- gtid_domain_id
- gtid_strict_mode
- histogram_size
- histogram_type
- innodb_adaptive_hash_index_partitions
- innodb_background_scrub_data_check_interval (MariaDB version 10.1 and later)
- innodb_background_scrub_data_compressed (MariaDB version 10.1 and later)
- innodb_background_scrub_data_interval (MariaDB version 10.1 and later)
- innodb_background_scrub_data_uncompressed (MariaDB version 10.1 and later)
- innodb_buf_dump_status_frequency (MariaDB version 10.1 and later)
- innodb_buffer_pool_populate
- innodb_cleaner_lsn_age_factor
- innodb_compression_algorithm (MariaDB version 10.1 and later)
- innodb_corrupt_table_action
- innodb_defragment (MariaDB version 10.1 and later)
- innodb_defragment_fill_factor (MariaDB version 10.1 and later)
- innodb_defragment_fill_factor_n_recs (MariaDB version 10.1 and later)
- innodb_defragment_frequency (MariaDB version 10.1 and later)
- innodb_defragment_n_pages (MariaDB version 10.1 and later)
- innodb_defragment_stats_accuracy (MariaDB version 10.1 and later)
- innodb_empty_free_List_algorithm
- innodb_fake_changes
- innodb_fatal_semaphore_wait_threshold (MariaDB version 10.1 and later)
- innodb_foreground_preflush
- innodb_idle_flush_pct (MariaDB version 10.1 and later)
- innodb_immediate_scrub_data_uncompressed (MariaDB version 10.1 and later)
- innodb_instrument_semaphores (MariaDB version 10.1 and later)
- innodb_locking_fake_changes
- innodb_log_arch_dir
- innodb_log_arch_expire_sec
- innodb_log_archive
- innodb_log_block_size
- innodb_log_checksum_algorithm
- innodb_max_bitmap_file_size
- innodb_max_changed_pages
- innodb_prefix_index_cluster_optimization (MariaDB version 10.1 and later)
- innodb_sched_priority_cleaner
- innodb_scrub_log (MariaDB version 10.1 and later)
- innodb_scrub_log_speed (MariaDB version 10.1 and later)

- innodb_show_locks_held
- innodb_show_verbose_locks
- innodb_simulate_comp_failures
- innodb_stats_modified_counter
- innodb_stats_traditional
- innodb_use_atomic_writes
- innodb_use_fallocate
- innodb_use_global_flush_log_at_trx_commit
- innodb_use_stacktrace
- innodb_use_trim (MariaDB version 10.1 and later)
- join_buffer_space_limit
- join_cache_level
- key_cache_file_hash_size
- key_cache_segments
- max_digest_length (MariaDB version 10.1 and later)
- max_statement_time (MariaDB version 10.1 and later)
- mysql56_temporal_format (MariaDB version 10.1 and later)
- progress_report_time
- query_cache_strip_comments
- replicate_annotation_row_events
- replicate_do_db
- replicate_do_table
- replicate_events_marked_for_skip
- replicate_ignore_db
- replicate_ignore_table
- replicate_wild_ignore_table
- slave_domain_parallel_threads
- slave_parallel_max_queued
- slave_parallel_mode (MariaDB version 10.1 and later)
- slave_parallel_threads
- slave_run_triggers_for_rbr (MariaDB version 10.1 and later)
- sql_error_log_filename
- sql_error_log_rate
- sql_error_log_rotate
- sql_error_log_rotations
- sql_error_log_size_limit
- thread_handling
- thread_pool_idle_timeout
- thread_pool_max_threads
- thread_pool_min_threads
- thread_pool_oversubscribe
- thread_pool_size
- thread_pool_stall_limit
- transaction_write_set_extraction
- use_stat_tables
- userstat

For more information on MariaDB parameters, go to the [MariaDB documentation](#).

MariaDB on Amazon RDS SQL Reference

This appendix describes system stored procedures that are available for Amazon RDS instances running the MariaDB DB engine.

You can use all of the system stored procedures that are available for Amazon RDS MySQL DB instances for MariaDB DB instances also. These stored procedures are documented at [MySQL on Amazon RDS SQL Reference \(p. 692\)](#).

Additionally, the following system stored procedures are supported only for Amazon RDS DB instances running MariaDB:

- [mysql.rds_set_external_master_gtid \(p. 485\)](#)
- [mysql.rds_kill_query_id \(p. 487\)](#)

[mysql.rds_set_external_master_gtid](#)

Configures GTID-based replication from a MariaDB instance running external to Amazon RDS to an Amazon RDS MariaDB DB instance. This stored procedure is supported only where the external MariaDB instance is version 10.0.24 or greater. When setting up replication where one or both instances do not support MariaDB global transaction identifiers (GTIDs), use [mysql.rds_set_external_master \(p. 694\)](#).

Using GTIDs for replication provides crash-safety features not offered by binary log replication, so we recommend it in cases where the replicating instances support it.

Syntax

```
CALL mysql.rds_set_external_master_gtid(  
    host_name  
    , host_port  
    , replication_user_name  
    , replication_user_password  
    , gtid  
    , ssl_encryption  
);
```

Parameters

host_name

String. The host name or IP address of the MariaDB instance running external to Amazon RDS that will become the replication master.

host_port

Integer. The port used by the MariaDB instance running external to Amazon RDS to be configured as the replication master. If your network configuration includes SSH port replication that converts the port number, specify the port number that is exposed by SSH.

replication_user_name

String. The ID of a user with REPLICATION SLAVE permissions in the MariaDB DB instance to be configured as the Read Replica.

replication_user_password

String. The password of the user ID specified in *replication_user_name*.

gtid

String. The global transaction ID on the master that replication should start from.

You can use @@gtid_current_pos to get the current GTID if the replication master has been locked while you are configuring replication, so the binary log doesn't change between the points when you get the GTID and when replication starts.

Otherwise, if you are using mysqldump version 10.0.13 or greater to populate the slave instance prior to starting replication, you can get the GTID position in the output by using the --master-data or --dump-slave options. If you are not using mysqldump version 10.0.13 or greater, you can run the SHOW MASTER STATUS or use those same mysqldump options to get the binary log file name and position, then convert them to a GTID by running BINLOG_GTIID_POS on the external MariaDB instance:

```
SELECT BINLOG_GTIID_POS('<binary log file name>', <binary log file position>);
```

For more information about the MariaDB implementation of GTIDs, go to [Global Transaction ID](#) in the MariaDB documentation.

ssl_encryption

Integer. This option is not currently implemented. The default is 0.

Usage Notes

The mysql.rds_set_external_master_gtid procedure must be run by the master user. It must be run on the MariaDB DB instance that you are configuring as the replication slave of a MariaDB instance running external to Amazon RDS. Before running mysql.rds_set_external_master_gtid, you must have configured the instance of MariaDB running external to Amazon RDS as a replication master. For more information, see [Importing Data into a MariaDB DB Instance \(p. 476\)](#).

Warning

Do not use mysql.rds_set_external_master_gtid to manage replication between two Amazon RDS DB instances. Use it only when replicating with a MariaDB instance running external to RDS. For information about managing replication between Amazon RDS DB instances, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

After calling mysql.rds_set_external_master_gtid to configure an Amazon RDS DB instance as a Read Replica, you can call [mysql.rds_start_replication \(p. 704\)](#) on the replica to start the replication process. You can call [mysql.rds_reset_external_master \(p. 700\)](#) to remove the Read Replica configuration.

When mysql.rds_set_external_master_gtid is called, Amazon RDS records the time, user, and an action of "set master" in the mysql.rds_history and mysql.rds_replication_status tables.

Examples

When run on a MariaDB DB instance, the following example configures it as the replication slave of an instance of MariaDB running external to Amazon RDS.

```
call mysql.rds_set_external_master_gtid
('Sourcedb.some.com',3306,'ReplicationUser','SomePassword','0-123-456',0);
```

Related Topics

- [mysql.rds_reset_external_master \(p. 700\)](#)

- [mysql.rds_start_replication \(p. 704\)](#)
- [mysql.rds_stop_replication \(p. 706\)](#)

mysql.rds_kill_query_id

Terminates a query running against the MariaDB server.

Syntax

```
CALL mysql.rds_kill_query_id(queryID);
```

Parameters

queryID

Integer. The identity of the query to be terminated.

Usage Notes

To terminate a query running against the MariaDB server, use the `mysql.rds_kill_query_id` procedure and pass in the ID of that query. To obtain the query ID, query the MariaDB [Information Schema PROCESSLIST Table](#), as shown following:

```
SELECT USER, HOST, COMMAND, TIME, STATE, INFO, QUERY_ID FROM
    INFORMATION_SCHEMA.PROCESSLIST WHERE USER = '<user name>';
```

The connection to the MariaDB server is retained.

Related Topics

- [mysql.rds_kill \(p. 713\)](#)
- [mysql.rds_kill_query \(p. 713\)](#)

Examples

The following example terminates a query with a query ID of 230040:

```
call mysql.rds_kill_query_id(230040);
```

Microsoft SQL Server on Amazon RDS

Amazon RDS supports DB instances running several versions and editions of Microsoft SQL Server. The most recent supported version of each major version is shown following. For the full list of supported versions, editions, and RDS engine versions, see [Version and Feature Support on Amazon RDS \(p. 493\)](#).

- SQL Server 2017 RTM (CU) 14.00.3035.2.v1, released per [KB4293805](#) on 14 August 2018 .
- SQL Server 2016 SP2 (CU2 + Security Update) 13.00.5201.2.v1, released per [KB4458621](#) on 21 August 2018 .
- SQL Server 2014 SP2 CU10 12.00.5571.0, released per [KB4052725](#) on 16 January 2018.
- SQL Server 2012 SP4 GDR 11.00.7462.6, released per [KB4057116](#) on 12 January 2017.
- SQL Server 2008 R2 SP3 GDR 10.50.6560.0, released per [KB4057113](#) on 6 January 2018. Not available in US East (Ohio), Canada (Central), and EU (London)

For information about licensing for SQL Server, see [Licensing Microsoft SQL Server on Amazon RDS \(p. 503\)](#). For information about SQL Server builds, see this Microsoft support article about [the latest SQL Server builds](#).

With Amazon RDS, you can create DB instances and DB snapshots, point-in-time restores, and automated or manual backups. DB instances running SQL Server can be used inside a VPC. You can also use SSL to connect to a DB instance running SQL Server, and you can use TDE to encrypt data at rest. Amazon RDS currently supports Multi-AZ deployments for SQL Server using SQL Server Mirroring or Always On as a high-availability, failover solution.

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application such as Microsoft SQL Server Management Studio. Amazon RDS does not allow direct host access to a DB instance via Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection. When you create a DB instance, you are assigned to the *db_owner* role for all databases on that instance, and you have all database-level permissions except for those that are used for backups. Amazon RDS manages backups for you.

Before creating your first DB instance, you should complete the steps in the setting up section of this guide. For more information, see [Setting Up for Amazon RDS \(p. 5\)](#).

Common Management Tasks for Microsoft SQL Server on Amazon RDS

The following are the common management tasks you perform with an Amazon RDS SQL Server DB instance, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Instance Classes, Storage, and PIOPS If you are creating a DB instance for production purposes, you should understand how instance classes, storage types, and Provisioned IOPS work in Amazon RDS.	DB Instance Class Support for Microsoft SQL Server (p. 491) Amazon RDS Storage Types (p. 103)
Multi-AZ Deployments A production DB instance should use Multi-AZ deployments. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. Multi-AZ deployments for SQL Server are implemented using SQL Server's native Mirroring or Always On technology.	High Availability (Multi-AZ) for Amazon RDS (p. 109) Multi-AZ Deployments Using Microsoft SQL Server Mirroring or Always On (p. 498)
Amazon Virtual Private Cloud (VPC) If your AWS account has a default VPC, then your DB instance is automatically created inside the default VPC. If your account does not have a default VPC, and you want the DB instance in a VPC, you must create the VPC and subnet groups before you create the DB instance.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412) Working with an Amazon RDS DB Instance in a VPC (p. 420)
Security Groups By default, DB instances are created with a firewall that prevents access to them. You therefore must create a security group with the correct IP addresses and network configuration to access the DB instance. The security group you create depends on what Amazon EC2 platform your DB instance is on, and whether you will access your DB instance from an Amazon EC2 instance. In general, if your DB instance is on the <i>EC2-Classic</i> platform, you will need to create a DB security group; if your DB instance is on the <i>EC2-VPC</i> platform, you will need to create a VPC security group.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412) Controlling Access with Security Groups (p. 394)
Parameter Groups If your DB instance is going to require specific database parameters, you should create a parameter group before you create the DB instance.	Working with DB Parameter Groups (p. 169)
Option Groups If your DB instance is going to require specific database options, you should create an option group before you create the DB instance.	Options for the Microsoft SQL Server Database Engine (p. 559)
Connecting to Your DB Instance After creating a security group and associating it to a DB instance, you can connect to the DB instance using any standard SQL client application such as Microsoft SQL Server Management Studio.	Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 514)
Backup and Restore	Working With Backups (p. 208) Importing and Exporting SQL Server Databases (p. 533)

Task Area	Relevant Documentation
When you create your DB instance, you can configure it to take automated backups. You can also back up and restore your databases manually by using full backup files (.bak files).	
Monitoring You can monitor your SQL Server DB instance by using CloudWatch Amazon RDS metrics, events, and enhanced monitoring.	Viewing DB Instance Metrics (p. 254) Viewing Amazon RDS Events (p. 305)
Log Files You can access the log files for your SQL Server DB instance.	Amazon RDS Database Log Files (p. 307) Microsoft SQL Server Database Log Files (p. 319)

There are also advanced administrative tasks for working with SQL Server DB instances. For more information, see the following documentation:

- [Common DBA Tasks for Microsoft SQL Server \(p. 564\)](#).
- [Using Windows Authentication with a SQL Server DB Instance \(p. 578\)](#)
- [Accessing the tempdb Database \(p. 565\)](#)

Limits for Microsoft SQL Server DB Instances

The Amazon RDS implementation of Microsoft SQL Server on a DB instance have some limitations you should be aware of:

- You can create up to 30 databases on each of your DB instances running Microsoft SQL Server. The Microsoft system databases, such as `master` and `model`, don't count toward this limit.
- Some ports are reserved for Amazon RDS use and you can't use them when you create a DB instance.
- Amazon RDS for SQL Server does not support importing data into the `msdb` database.
- You can't rename databases on a DB instance in a SQL Server Multi-AZ deployment.
- The maximum storage size for SQL Server DB instances is the following:
 - General Purpose (SSD) storage: 16 TiB for all editions
 - Provisioned IOPS storage: 16 TiB for all editions
 - Magnetic storage: 1 TiB for all editions

If you have a scenario that requires a larger amount of storage, you can use sharding across multiple DB instances to get around the limit. This approach requires data-dependent routing logic in applications that connect to the sharded system. You can use an existing sharding framework, or you can write custom code to enable sharding. If you use an existing framework, the framework can't install any components on the same server as the DB instance.

- The minimum storage size for SQL Server DB instances is the following:
 - General Purpose (SSD) storage: 200 GiB for Enterprise and Standard editions, 20 GiB for Web and Express editions
 - Provisioned IOPS storage: 200 GiB for Enterprise and Standard editions, 100 GiB for Web and Express editions
 - Magnetic storage: 200 GiB for Enterprise and Standard editions, 20 GiB for Web and Express editions

- Amazon RDS doesn't support running SQL Server Analysis Services, SQL Server Integration Services, SQL Server Reporting Services, Data Quality Services, or Master Data Services on the same server as your Amazon RDS DB instance. To use these features, we recommend that you install SQL Server on an Amazon EC2 instance, or use an on-premise SQL Server instance, to act as the Reporting, Analysis, Integration, or Master Data Services server for your SQL Server DB instance on Amazon RDS. You can install SQL Server on an Amazon EC2 instance with Amazon EBS storage, pursuant to Microsoft licensing policies.
- Because of limitations in Microsoft SQL Server, restoring to a point in time before successful execution of a DROP DATABASE might not reflect the state of that database at that point in time. For example, the dropped database is typically restored to its state up to 5 minutes before the DROP DATABASE command was issued, which means that you can't restore the transactions made during those few minutes on your dropped database. To work around this, you can reissue the DROP DATABASE command after the restore operation is completed. Dropping a database removes the transaction logs for that database.

DB Instance Class Support for Microsoft SQL Server

The computation and memory capacity of a DB instance is determined by its DB instance class. The DB instance class you need depends on your processing power and memory requirements. For more information, see [DB Instance Class \(p. 82\)](#).

The following are the DB instance classes supported for Microsoft SQL Server.

SQL Server Edition	2017 and 2016 Support	2014, 2012, and 2008 R2 Support
Enterprise Edition	db.m4.xlarge–16xlarge db.r4.xlarge–16xlarge —	db.m4.xlarge–10xlarge db.r4.xlarge–8xlarge —
Standard Edition	db.m4.large–16xlarge, except db.m4.10xlarge db.r4.large–16xlarge —	db.m4.large–4xlarge db.r4.large–8xlarge —
Web Edition	db.m4.large–4xlarge db.r4.large–2xlarge db.t2.small–medium	db.m4.large–4xlarge db.r4.large–2xlarge db.t2.small–medium
Express Edition	— — db.t2.micro–medium	db.m1.small–small — db.t2.micro–medium

Microsoft SQL Server Security

The Microsoft SQL Server database engine uses role-based security. The master user name you use when you create a DB instance is a SQL Server Authentication login that is a member of the `processadmin`, `public`, and `setupadmin` fixed server roles.

Any user who creates a database is assigned to the `db_owner` role for that database and has all database-level permissions except for those that are used for backups. Amazon RDS manages backups for you.

The following server-level roles are not currently available in Amazon RDS:

- `bulkadmin`
- `dbcreator`
- `diskadmin`
- `securityadmin`
- `serveradmin`
- `sysadmin`

The following server-level permissions are not available on SQL Server DB instances:

- `ADMINISTER BULK OPERATIONS`
- `ALTER ANY CREDENTIAL`
- `ALTER ANY EVENT NOTIFICATION`
- `ALTER ANY EVENT SESSION`
- `ALTER ANY SERVER AUDIT`
- `ALTER RESOURCES`
- `ALTER SETTINGS` (You can use the DB Parameter Group APIs to modify parameters. For more information, see [Working with DB Parameter Groups \(p. 169\)](#).)
- `AUTHENTICATE SERVER`
- `CONTROL_SERVER`
- `CREATE DDL EVENT NOTIFICATION`
- `CREATE ENDPOINT`
- `CREATE TRACE EVENT NOTIFICATION`
- `EXTERNAL ACCESS ASSEMBLY`
- `SHUTDOWN` (You can use the RDS reboot option instead)
- `UNSAFE ASSEMBLY`
- `ALTER ANY AVAILABILITY GROUP` (SQL Server 2012 only)
- `CREATE ANY AVAILABILITY GROUP` (SQL Server 2012 only)

Compliance Program Support for Microsoft SQL Server DB Instances

AWS Services in Scope have been fully assessed by a third-party auditor and result in a certification, attestation of compliance, or Authority to Operate (ATO). For more information, see [AWS Services in Scope by Compliance Program](#).

HIPAA Support for Microsoft SQL Server DB Instances

You can use Amazon RDS for Microsoft SQL Server databases to build HIPAA-compliant applications. You can store healthcare-related information, including protected health information (PHI), under an executed Business Associate Agreement (BAA) with AWS. For more information, see [HIPAA Compliance](#).

Amazon RDS for SQL Server supports HIPAA for the following versions and editions:

- SQL Server 2017, 2016, 2014, and 2012: Enterprise, Standard, and Web Editions
- SQL Server 2008 R2: Enterprise Edition

To enable HIPAA support on your DB instance, set up the following three components.

Component	Details
Auditing	To set up auditing, set the parameter <code>rds.sqlserver_audit</code> to the value <code>fedramp_hipaa</code> . If your DB instance is not already using a custom DB parameter group, you must create a custom parameter group and attach it to your DB instance before you can modify the <code>rds.sqlserver_audit</code> parameter. For more information, see Working with DB Parameter Groups (p. 169) .
Transport Encryption	To set up transport encryption, force all connections to your DB instance to use Secure Sockets Layer (SSL). For more information, see Forcing Connections to Your DB Instance to Use SSL (p. 555) .
Encryption at Rest	To set up encryption at rest, you have two options: <ol style="list-style-type: none">1. If you are running Enterprise Edition, you can choose to use Transparent Data Encryption (TDE) to achieve encryption at rest. For more information, see Microsoft SQL Server Transparent Data Encryption Support (p. 561).2. You can set up encryption at rest by using AWS Key Management Service (AWS KMS) encryption keys. For more information, see Encrypting Amazon RDS Resources (p. 389).

SSL Support for Microsoft SQL Server DB Instances

You can use SSL to encrypt connections between your applications and your Amazon RDS DB instances running Microsoft SQL Server. You can also force all connections to your DB instance to use SSL. If you force connections to use SSL, it happens transparently to the client, and the client doesn't have to do any work to use SSL.

SSL is supported in all AWS Regions and for all supported SQL Server editions. For more information, see [Using SSL with a Microsoft SQL Server DB Instance \(p. 555\)](#).

Version and Feature Support on Amazon RDS

Microsoft SQL Server 2017 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2017:

- SQL Server 2017 RTM CU3 14.00.3015.40, released per [KB4052987](#) on 4 January 2018.
RDS API `EngineVersion` and CLI `engine-version: 14.00.3015.40.v1`
- Version 14.00.1000.169, RTM, for all editions, and all AWS Regions
RDS API `EngineVersion` and CLI `engine-version: 14.00.1000.169.v1`

SQL Server 2017 includes many new features, such as the following:

- Adaptive query processing
- Automatic plan correction
- GraphDB
- Resumable index rebuilds

For the full list of SQL Server 2017 features, see [What's New in SQL Server 2017](#) in the Microsoft documentation.

For a list of unsupported features, see [Features Not Supported and Features with Limited Support \(p. 498\)](#).

Microsoft SQL Server 2016 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2016:

- SQL Server 2016 SP1 CU7 13.00.4466.4, released per [KB4057119](#) on 4 January 2018.
RDS API `EngineVersion` and CLI `engine-version: 13.00.4466.4.v1`
- Version 13.00.4451.0, SP1 CU5, for all editions, and all AWS Regions
RDS API `EngineVersion` and CLI `engine-version: 13.00.4451.0.v1`
- Version 13.00.4422.0, SP1 CU2, for all editions, and all AWS Regions
RDS API `EngineVersion` and CLI `engine-version: 13.00.4422.0.v1`
- Version 13.00.2164.0, RTM CU2, for all editions, and all AWS Regions
RDS API `EngineVersion` and CLI `engine-version: 13.00.2164.0.v1`

Microsoft SQL Server 2014 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2014:

- SQL Server 2014 SP2 CU10 12.00.5571.0, released per [KB4052725](#) on 16 January 2018.
RDS API `EngineVersion` and CLI `engine-version: 12.00.5571.0.v1`
- Version 12.00.5546.0, SP2 CU5, for all editions and all AWS Regions
RDS API `EngineVersion` and CLI `engine-version: 12.00.5546.0.v1`
- Version 12.00.5000.0, SP2, for all editions and all AWS Regions
RDS API `EngineVersion` and CLI `engine-version: 12.00.5000.0.v1`
- Version 12.00.4422.0, SP1 CU2, for all editions except Enterprise Edition, and all AWS Regions except Canada (Central), and EU (London)

RDS API EngineVersion and CLI engine-version: 12.00.4422.0.v1

In addition to supported features of SQL Server 2012, Amazon RDS supports the new query optimizer available in SQL Server 2014, and also the delayed durability feature.

For a list of unsupported features, see [Features Not Supported and Features with Limited Support \(p. 498\)](#).

SQL Server 2014 supports all the parameters from SQL Server 2012 and uses the same default values. SQL Server 2014 includes one new parameter, backup checksum default. For more information, see [How to enable the CHECKSUM option if backup utilities do not expose the option](#) in the Microsoft documentation.

Microsoft SQL Server 2012 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2012:

- SQL Server 2012 SP4 GDR 11.00.7462.6, released per [KB4057116](#) on 12 January 2017.

RDS API EngineVersion and CLI engine-version: 11.00.7462.6.v1
- Version 11.00.6594.0, SP3 CU8, for all editions and all AWS Regions

RDS API EngineVersion and CLI engine-version: 11.00.6594.0.v1
- Version 11.00.6020.0, SP3, for all editions and all AWS Regions

RDS API EngineVersion and CLI engine-version: 11.00.6020.0.v1
- Version 11.00.5058.0, SP2, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)

RDS API EngineVersion and CLI engine-version: 11.00.5058.0.v1
- Version 11.00.2100.60, RTM, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)

RDS API EngineVersion and CLI engine-version: 11.00.2100.60.v1

For more information about SQL Server 2012, see [Features Supported by the Editions of SQL Server 2012](#) in the Microsoft documentation.

In addition to supported features of SQL Server 2008 R2, Amazon RDS supports the following SQL Server 2012 features:

- Columnstore indexes (Enterprise Edition)
- Online Index Create, Rebuild and Drop for XML, varchar(max), nvarchar(max), and varbinary(max) data types (Enterprise Edition)
- Flexible Server Roles
- Service Broker is supported, Service Broker endpoints are not supported
- Partially Contained Databases
- Sequences
- Transparent Data Encryption (Enterprise Edition only)
- THROW statement
- New and enhanced spatial types

- UTF-16 Support
- ALTER ANY SERVER ROLE server-level permission

For a list of unsupported features, see [Features Not Supported and Features with Limited Support \(p. 498\)](#).

Some SQL Server parameters have changed in SQL Server 2012.

- The following parameters have been removed from SQL Server 2012: `awe_enabled`, `precompute_rank`, and `sql_mail_xps`. These parameters were not modifiable in SQL Server DB Instances and their removal should have no impact on your SQL Server use.
- A new `contained_database_authentication` parameter in SQL Server 2012 supports partially contained databases. When you enable this parameter and then create a partially contained database, an authorized user's user name and password is stored within the partially contained database instead of in the master database. For more information about partially contained databases, see [Contained Databases](#) in the Microsoft documentation.

Microsoft SQL Server 2008 R2 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2008 R2:

- SQL Server 2008 R2 SP3 GDR 10.50.6560.0, released per [KB4057113](#) on 6 January 2018. Not available in US East (Ohio), Canada (Central), and EU (London)

RDS API `EngineVersion` and CLI `engine-version: 10.50.6560.0.v1`

- Version 10.50.6529.0, SP3 QFE, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)

RDS API `EngineVersion` and CLI `engine-version: 10.50.6529.0.v1`

- Version 10.50.6000.34, SP3, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)

RDS API `EngineVersion` and CLI `engine-version: 10.50.6000.34.v1`

- Version 10.50.2789.0, SP1, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)

RDS API `EngineVersion` and CLI `engine-version: 10.50.2789.0.v1`

For more information about SQL Server 2008 R2, see [Features Supported by the Editions of SQL Server 2008 R2](#) in the Microsoft documentation.

Amazon RDS supports the following SQL Server 2008 R2 features:

- Core database engine features
- SQL Server development tools:
 - Visual Studio integration
 - IntelliSense
- SQL Server management tools:
 - SQL Server Management Studio (SMS)
 - `sqlcmd`

- SQL Server Profiler (client side traces; workaround available for server side)
- SQL Server Migration Assistant (SSMA)
- Database Engine Tuning Advisor
- SQL Server Agent
- Safe CLR
- Full-text search (except semantic search)
- SSL
- Transparent Data Encryption (Enterprise Edition only)
- Spatial and location features
- Service Broker is supported, Service Broker endpoints are not supported
- Change Tracking
- Database Mirroring or Always On
- The ability to use an Amazon RDS SQL DB instance as a data source for Reporting, Analysis, and Integration Services that are running on a separate server.

For a list of unsupported features, see [Features Not Supported and Features with Limited Support \(p. 498\)](#).

Microsoft SQL Server Engine Version Management

With Amazon RDS, you control when to upgrade your SQL Server DB instance to new versions supported by Amazon RDS. You can maintain compatibility with specific SQL Server versions, test new versions with your application before deploying in production, and perform version upgrades on your own terms and timelines.

Currently, you perform all SQL Server database upgrades manually. For more information about upgrading a SQL Server DB instance, see [Upgrading the Microsoft SQL Server DB Engine \(p. 529\)](#).

Change Data Capture Support for Microsoft SQL Server DB Instances

Amazon RDS supports change data capture (CDC) for your DB instances running Microsoft SQL Server. CDC captures changes that are made to the data in your tables, and stores metadata about each change that you can access later. For more information, see [Change Data Capture](#) in the Microsoft documentation.

Amazon RDS supports CDC for the following SQL Server editions and versions:

- Microsoft SQL Server Enterprise Edition (2016, 2014, 2012, 2008 R2)
- Microsoft SQL Server Standard Edition (2016 version 13.00.4422.0 SP1 CU2 and later)

To use CDC with your Amazon RDS DB instances, first enable or disable CDC at the database level by using RDS-provided stored procedures. After that, any user that has the `db_owner` role for that database can use the native Microsoft stored procedures to control CDC on that database. For more information, see [Using Change Data Capture \(p. 570\)](#).

You can use CDC and AWS Database Migration Service to enable ongoing replication from SQL Server DB instances.

Features Not Supported and Features with Limited Support

The following Microsoft SQL Server features are not supported on Amazon RDS:

- Stretch database
- Backing up to Microsoft Azure Blob Storage
- Buffer pool extension
- BULK INSERT and OPENROWSET(BULK...) features
- Data Quality Services
- Database Log Shipping
- Database Mail
- Distribution Transaction Coordinator (MSDTC)
- File tables
- FILESTREAM support
- Maintenance Plans
- Performance Data Collector
- Policy-Based Management
- PolyBase
- R
- Replication
- Resource Governor
- SQL Server Audit
- Server-level triggers
- Service Broker endpoints
- T-SQL endpoints (all operations using CREATE ENDPOINT are unavailable)
- WCF Data Services

The following Microsoft SQL Server features have limited support on Amazon RDS:

- Distributed Queries / Linked Servers. For more information, see: [Implementing Linked Servers with Amazon RDS for Microsoft SQL Server](#).

Multi-AZ Deployments Using Microsoft SQL Server Mirroring or Always On

Amazon RDS supports Multi-AZ deployments for DB instances running Microsoft SQL Server by using SQL Server Database Mirroring or Always On. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. In the event of planned database maintenance or unplanned service disruption, Amazon RDS automatically fails over to the up-to-date secondary replica

so database operations can resume quickly without manual intervention. The primary and secondary instances use the same endpoint, whose physical network address transitions to the passive secondary replica as part of the failover process. You don't have to reconfigure your application when a failover occurs.

Amazon RDS manages failover by actively monitoring your Multi-AZ deployment and initiating a failover when a problem with your primary occurs. Failover doesn't occur unless the standby and primary are fully in sync. Amazon RDS actively maintains your Multi-AZ deployment by automatically repairing unhealthy DB instances and re-establishing synchronous replication. You don't have to manage anything. Amazon RDS handles the primary, the witness, and the standby instance for you. When you set up SQL Server Multi-AZ, RDS configures passive secondary instances for all of the databases on the instance.

For more information, see [Multi-AZ Deployments for Microsoft SQL Server \(p. 551\)](#).

Using Transparent Data Encryption to Encrypt Data at Rest

Amazon RDS supports Microsoft SQL Server Transparent Data Encryption (TDE), which transparently encrypts stored data. Amazon RDS uses option groups to enable and configure these features. For more information about the TDE option, see [Microsoft SQL Server Transparent Data Encryption Support \(p. 561\)](#).

Local Time Zone for Microsoft SQL Server DB Instances

The time zone of an Amazon RDS DB instance running Microsoft SQL Server is set by default. The current default is Universal Coordinated Time (UTC). You can set the time zone of your DB instance to a local time zone instead, to match the time zone of your applications.

You set the time zone when you first create your DB instance. You can create your DB instance by using the [AWS Management Console](#), the Amazon RDS API [CreateDBInstance](#) action, or the AWS CLI [create-db-instance](#) command.

If your DB instance is part of a Multi-AZ deployment (using SQL Server Mirroring or Always On), then when you fail over, your time zone remains the local time zone that you set. For more information, see [Multi-AZ Deployments Using Microsoft SQL Server Mirroring or Always On \(p. 498\)](#).

When you request a point-in-time restore, you specify the time to restore to in UTC. During the restore process, the time is translated to the time zone of the DB instance. For more information, see [Restoring a DB Instance to a Specified Time \(p. 237\)](#).

The following are limitations to setting the local time zone on your DB instance:

- You can't modify the time zone of an existing SQL Server DB instance.
- You can't restore a snapshot from a DB instance in one time zone to a DB instance in a different time zone.
- We strongly recommend that you don't restore a backup file from one time zone to a different time zone. If you restore a backup file from one time zone to a different time zone, you must audit your queries and applications for the effects of the time zone change. For more information, see [Importing and Exporting SQL Server Databases \(p. 533\)](#).

Supported Time Zones

You can set your local time zone to one of the values listed in the following table.

Time Zone	Standard Time Offset	Description	Notes
Afghanistan Standard Time	(UTC+04:30)	Kabul	
Alaskan Standard Time	(UTC-09:00)	Alaska	
Arabian Standard Time	(UTC+04:00)	Abu Dhabi, Muscat	
Atlantic Standard Time	(UTC-04:00)	Atlantic Time (Canada)	
AUS Central Standard Time	(UTC+09:30)	Darwin	
AUS Eastern Standard Time	(UTC+10:00)	Canberra, Melbourne, Sydney	
Belarus Standard Time	(UTC+03:00)	Minsk	This time zone does not observe daylight savings time.
Canada Central Standard Time	(UTC-06:00)	Saskatchewan	
Cape Verde Standard Time	(UTC-01:00)	Cabo Verde Is.	
Cen. Australia Standard Time	(UTC+09:30)	Adelaide	
Central America Standard Time	(UTC-06:00)	Central America	
Central Asia Standard Time	(UTC+06:00)	Astana	
Central Brazilian Standard Time	(UTC-04:00)	Cuiaba	
Central Europe Standard Time	(UTC+01:00)	Belgrade, Bratislava, Budapest, Ljubljana, Prague	
Central European Standard Time	(UTC+01:00)	Sarajevo, Skopje, Warsaw, Zagreb	
Central Pacific Standard Time	(UTC+11:00)	Solomon Islands, New Caledonia	
Central Standard Time	(UTC-06:00)	Central Time (US and Canada)	
Central Standard Time (Mexico)	(UTC-06:00)	Guadalajara, Mexico City, Monterrey	
China Standard Time	(UTC+08:00)	Beijing, Chongqing, Hong Kong, Urumqi	
E. Africa Standard Time	(UTC+03:00)	Nairobi	This time zone does not observe daylight savings time.

Time Zone	Standard Time Offset	Description	Notes
E. Australia Standard Time	(UTC+10:00)	Brisbane	
E. Europe Standard Time	(UTC+02:00)	Chisinau	
E. South America Standard Time	(UTC−03:00)	Brasilia	
Eastern Standard Time	(UTC−05:00)	Eastern Time (US and Canada)	
Georgian Standard Time	(UTC+04:00)	Tbilisi	
GMT Standard Time	(UTC)	Dublin, Edinburgh, Lisbon, London	This time zone is not the same as Greenwich Mean Time. This time zone does observe daylight savings time.
Greenland Standard Time	(UTC−03:00)	Greenland	
Greenwich Standard Time	(UTC)	Monrovia, Reykjavik	This time zone does not observe daylight savings time.
GTB Standard Time	(UTC+02:00)	Athens, Bucharest	
Hawaiian Standard Time	(UTC−10:00)	Hawaii	
India Standard Time	(UTC+05:30)	Chennai, Kolkata, Mumbai, New Delhi	
Jordan Standard Time	(UTC+02:00)	Amman	
Korea Standard Time	(UTC+09:00)	Seoul	
Middle East Standard Time	(UTC+02:00)	Beirut	
Mountain Standard Time	(UTC−07:00)	Mountain Time (US and Canada)	
Mountain Standard Time (Mexico)	(UTC−07:00)	Chihuahua, La Paz, Mazatlan	
US Mountain Standard Time	(UTC−07:00)	Arizona	This time zone does not observe daylight savings time.
New Zealand Standard Time	(UTC+12:00)	Auckland, Wellington	
Newfoundland Standard Time	(UTC−03:30)	Newfoundland	
Pacific SA Standard Time	(UTC−03:00)	Santiago	
Pacific Standard Time	(UTC−08:00)	Pacific Time (US and Canada)	
Pacific Standard Time (Mexico)	(UTC−08:00)	Baja California	

Time Zone	Standard Time Offset	Description	Notes
Russian Standard Time	(UTC+03:00)	Moscow, St. Petersburg, Volgograd	This time zone does not observe daylight savings time.
SA Pacific Standard Time	(UTC-05:00)	Bogota, Lima, Quito, Rio Branco	This time zone does not observe daylight savings time.
SE Asia Standard Time	(UTC+07:00)	Bangkok, Hanoi, Jakarta	
Singapore Standard Time	(UTC+08:00)	Kuala Lumpur, Singapore	
Tokyo Standard Time	(UTC+09:00)	Osaka, Sapporo, Tokyo	
US Eastern Standard Time	(UTC-05:00)	Indiana (East)	
UTC	UTC	Coordinated Universal Time	This time zone does not observe daylight savings time.
UTC-02	(UTC-02:00)	Coordinated Universal Time-02	
UTC-08	(UTC-08:00)	Coordinated Universal Time-08	
UTC-09	(UTC-09:00)	Coordinated Universal Time-09	
UTC-11	(UTC-11:00)	Coordinated Universal Time-11	
UTC+12	(UTC+12:00)	Coordinated Universal Time+12	
W. Australia Standard Time	(UTC+08:00)	Perth	
W. Central Africa Standard Time	(UTC+01:00)	West Central Africa	
W. Europe Standard Time	(UTC+01:00)	Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna	

Licensing Microsoft SQL Server on Amazon RDS

When you set up an Amazon RDS DB instance for Microsoft SQL Server, the software license is included.

This means that you don't need to purchase SQL Server licenses separately. AWS holds the license for the SQL Server database software. Amazon RDS pricing includes the software license, underlying hardware resources, and Amazon RDS management capabilities.

Amazon RDS supports the following Microsoft SQL Server editions:

- Enterprise
- Standard
- Web
- Express

Note

Licensing for SQL Server Web Edition supports only public and internet-accessible webpages, websites, web applications, and web services. This level of support is required for compliance with Microsoft's usage rights. For more information, see [AWS Service Terms](#).

Amazon RDS supports Multi-AZ deployments for DB instances running Microsoft SQL Server by using SQL Server Database Mirroring or Always On. There are no additional licensing requirements for Multi-AZ deployments. For more information, see [Multi-AZ Deployments for Microsoft SQL Server \(p. 551\)](#).

Restoring License-Terminated DB Instances

Amazon RDS takes snapshots of license-terminated DB instances. If your instance is terminated for licensing issues, you can restore it from the snapshot to a new DB instance. New DB instances have a license included.

For more information, see [Restoring License-Terminated DB Instances \(p. 573\)](#).

Development and Test

Because of licensing requirements, we can't offer SQL Server Developer edition on Amazon RDS. You can use Express edition for many development, testing, and other nonproduction needs. However, if you need the full feature capabilities of an enterprise-level installation of SQL Server, you must use a dedicated host environment. You can download and install SQL Server Developer edition (and other MSDN products) on Amazon EC2. Dedicated infrastructure is not required for Developer edition. By using your own host, you also gain access to other programmability features that are not accessible on Amazon RDS. For more information on the difference between SQL Server editions, see [Editions and supported features of SQL Server 2017](#) in the Microsoft documentation.

Related Topics

- [Microsoft SQL Server on Amazon RDS \(p. 488\)](#)
- [Creating a DB Instance Running the Microsoft SQL Server Database Engine \(p. 504\)](#)

Creating a DB Instance Running the Microsoft SQL Server Database Engine

The basic building block of Amazon RDS is the DB instance. Your Amazon RDS DB instance is similar to your on-premises Microsoft SQL Server. After you create your SQL Server DB instance, you can add one or more custom databases to it.

Important

You must complete the tasks in the [Setting Up for Amazon RDS \(p. 5\)](#) section before you can create or connect to a DB instance.

For an example that walks you through the process of creating and connecting to a sample DB instance, see [Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance \(p. 18\)](#).

AWS Management Console

To launch a SQL Server DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the Amazon RDS console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
4. Choose **Create database** to open the **Select engine** page.
5. Choose the **Microsoft SQL Server** icon.

Select engine

Engine options

Amazon Aurora
Amazon Aurora

MySQL


MariaDB


PostgreSQL


Oracle
ORACLE

Microsoft SQL Server


Microsoft SQL Server

Edition

SQL Server Express Edition
Affordable database management system that supports database sizes up to 10 GiB.

SQL Server Web Edition
In accordance with Microsoft's licensing policies, it can only be used to support public and Internet-accessible webpages, websites, web applications, and web services.

SQL Server Standard Edition
Core data management and business intelligence capabilities for mission-critical applications and mixed workloads.

SQL Server Enterprise Edition
Comprehensive high-end capabilities for mission-critical applications with demanding database workloads and business intelligence requirements.

Aurora global database feature is now available.
This feature is now available in our new database creation flow.

[Try it now](#)

Only enable options eligible for RDS Free Usage Tier [Info](#)

[Cancel](#) **Next**

6. Choose the SQL Server DB engine edition that you want to use. The SQL Server editions that are available vary by AWS Region.
7. For some editions, the **Use Case** step asks if you are planning to use the DB instance you are creating for production. If you are, choose **Production**. If you choose **Production**, the following are all preselected in a later step:
 - Multi-AZ failover option

- **Provisioned IOPS** storage option
- **Enable deletion protection** option

We recommend these features for any production environment.

8. Choose **Next** to continue. The **Specify DB Details** page appears.

On the **Specify DB Details** page, specify your DB instance information. For information about each setting, see [Settings for Microsoft SQL Server DB Instances \(p. 510\)](#).

Specify DB details

Instance specifications

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#) 

DB engine

Microsoft SQL Server Express Edition

License model [Info](#)

license-included

DB engine version [Info](#)

SQL Server 2017 14.00.3035.2.v1



Free tier

The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GiB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#).

Only enable options eligible for RDS Free Usage Tier [Info](#)

DB instance class [Info](#)

db.t2.small — 1 vCPU, 2 GiB RAM

Time zone (optional)

No preference

Storage type [Info](#)

9. Choose **Next** to continue. The **Configure Advanced Settings** page appears.

On the **Configure Advanced Settings** page, provide additional information that Amazon RDS needs to launch the DB instance. For information about each setting, see [Settings for Microsoft SQL Server DB Instances \(p. 510\)](#).

Configure advanced settings

Network & Security

Virtual Private Cloud (VPC) [Info](#)

VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-2aed394c)



Only VPCs with a corresponding DB subnet group are listed.

Subnet group [Info](#)

DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

default



Public accessibility [Info](#)

Yes

EC2 instances and devices outside of the VPC hosting the DB instance will connect to the DB instances. You must also select one or more VPC security groups that specify which EC2 instances and devices can connect to the DB instance.

No

DB instance will not have a public IP address assigned. No EC2 instance or devices outside of the VPC will be able to connect.

Availability zone [Info](#)

No preference



VPC security groups

Security groups have rules authorizing connections from all the EC2 instances and devices that need to access the DB instance.

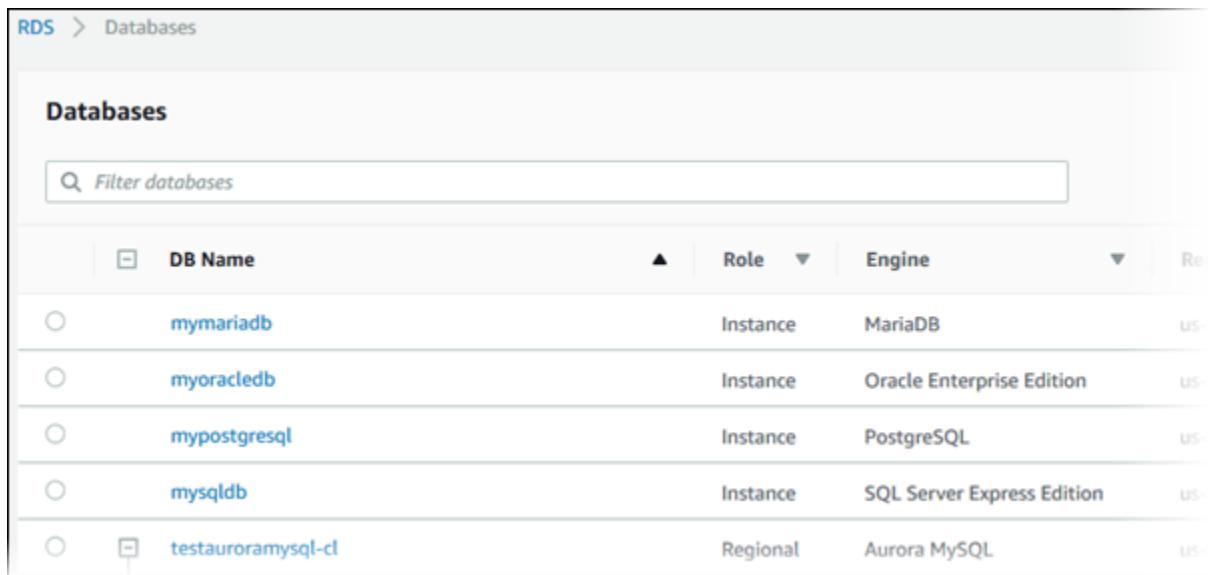
Create new VPC security group

Choose existing VPC security groups

10. Choose **Launch DB Instance**.

11. On the final page of the wizard, choose **Close**.

On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.



The screenshot shows the AWS RDS Databases console. At the top, there's a breadcrumb navigation: RDS > Databases. Below that is a section titled "Databases" with a search bar labeled "Filter databases". A table lists five databases:

DB Name	Role	Engine	Region
mymariadb	Instance	MariaDB	us-east-1
myoracledb	Instance	Oracle Enterprise Edition	us-east-1
mypostgresql	Instance	PostgreSQL	us-east-1
mysqldb	Instance	SQL Server Express Edition	us-east-1
testauroramysql-cl	Regional	Aurora MySQL	us-east-1

CLI

To create a Microsoft SQL Server DB instance by using the AWS CLI, call the [create-db-instance](#) command with the parameters below. For information about each setting, see [Settings for Microsoft SQL Server DB Instances \(p. 510\)](#).

- `--db-instance-identifier`
- `--db-instance-class`
- `--db-security-groups`
- `--db-subnet-group`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--allocated-storage`
- `--backup-retention-period`

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance
  --engine sqlserver-se \
  --db-instance-identifier mymssqlserver \
  --allocated-storage 250 \
  --db-instance-class db.m1.large \
  --db-security-groups mydbsecuritygroup \
  --db-subnet-group mydbsubnetgroup \
  --master-user-name masterawsuser \
  --master-user-password masteruserpassword \
  --backup-retention-period 3
```

For Windows:

```
aws rds create-db-instance ^
--engine sqlserver-se ^
--db-instance-identifier mydbinstance ^
--allocated-storage 250 ^
--db-instance-class db.m1.large ^
--db-security-groups mydbsecuritygroup ^
--db-subnet-group mydbsubnetgroup ^
--master-user-name masterawsuser ^
--master-user-password masteruserpassword ^
--backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE mydbinstance db.m1.large sqlserver-se 250 sa creating 3 **** n
10.50.2789
SECGROUP default active
PARAMGRP default.sqlserver-se-10.5 in-sync
```

API

To create a Microsoft SQL Server DB instance by using the Amazon RDS API, call the [CreateDBInstance](#) action with the parameters below. For information about each setting, see [Settings for Microsoft SQL Server DB Instances \(p. 510\)](#).

- `AllocatedStorage`
- `BackupRetentionPeriod`
- `DBInstanceState`
- `DBInstanceIdentifier`
- `DBSecurityGroups`
- `DBSubnetGroup`
- `Engine`
- `MasterUsername`
- `MasterUserPassword`

Example

```
https://rds.amazonaws.com/
?Action=CreateDBInstance
&AllocatedStorage=250
&BackupRetentionPeriod=3
&DBInstanceState=db.m1.large
&DBInstanceIdentifier=mydbinstance
&DBSecurityGroups.member.1=mysecuritygroup
&DBSubnetGroup=mydbsubnetgroup
&Engine=sqlserver-se
&MasterUserPassword=masteruserpassword
&MasterUsername=masterawsuser
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-1/rds/aws4_request
&X-Amz-Date=20140305T185838Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=b441901545441d3c7a48f63b5b1522c5b2b37c137500c93c45e209d4b3a064a3
```

Settings for Microsoft SQL Server DB Instances

The following table contains details about settings that you choose when you create a SQL Server DB instance.

Setting	Setting Description
Allocated Storage	The amount of storage to allocate for your DB instance (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information, see DB instance storage (p. 103) .
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.
Availability Zone	The availability zone for your DB instance. Use the default value of No Preference unless you want to specify an Availability Zone. For more information, see Regions and Availability Zones (p. 101) .
Backup Retention Period	The number of days that you want automatic backups of your DB instance to be retained. For any non-trivial DB instance, you should set this value to 1 or greater. For more information, see Working With Backups (p. 208) .
Backup Window	The time period during which Amazon RDS automatically takes a backup of your DB instance. Unless you have a specific time that you want to have your database backup, use the default of No Preference . For more information, see Working With Backups (p. 208) .
Copy Tags To Snapshots	Select this option to copy any DB instance tags to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .
Database Port	The port that you want to access the DB instance through. SQL Server installations default to port 1433. If you use a DB security group with your DB instance, this must be the same port value you provided when creating the DB security group.
DB Engine Version	The version of Microsoft SQL Server that you want to use.
DB Instance Class	The configuration for your DB instance. For example, a db.m1.small instance class equates to 1.7 GiB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.

Setting	Setting Description
	If possible, choose an instance class large enough that a typical query working set can be held in memory. When working sets are held in memory the system can avoid writing to disk, and this improves performance. For more information, see DB Instance Class (p. 82) and DB Instance Class Support for Microsoft SQL Server (p. 491) .
DB Instance Identifier	The name for your DB instance. Name your DB instances in the same way that you would name your on-premises servers. Your DB instance identifier can contain up to 63 alphanumeric characters, and must be unique for your account in the AWS Region you chose. You can add some intelligence to the name, such as including the AWS Region and DB engine you chose, for example sqlsv-instance1 .
DB Parameter Group	A parameter group for your DB instance. You can choose the default parameter group or you can create a custom parameter group. For more information, see Working with DB Parameter Groups (p. 169) .
Enable deletion protection	Enable deletion protection to prevent your DB instance from being deleted. If you create a production DB instance with the AWS Management Console, deletion protection is enabled by default. For more information, see Deleting a DB Instance (p. 135) .
Enable Encryption	Yes to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 389) .
Enable Enhanced Monitoring	Yes to gather metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 256) .
License Model	The license model that you want to use. Choose license-included to use the general license agreement for Microsoft SQL Server.
Maintenance Window	The 30 minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose No Preference . For more information, see The Amazon RDS Maintenance Window (p. 120) .
Master Username	The name that you use as the master user name to log on to your DB instance with all database privileges. The master user name is a SQL Server Authentication login that is a member of the <code>processadmin</code> , <code>public</code> , and <code>setupadmin</code> fixed server roles. For more information, see Microsoft SQL Server Security (p. 492) .

Setting	Setting Description
Master User Password	The password for your master user account. The password must contain from 8 to 128 printable ASCII characters (excluding /, ", a space, and @).
Multi-AZ Deployment	<p>Yes to create a passive secondary replica of your DB instance in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose No.</p> <p>For more information, see Multi-AZ Deployments for Microsoft SQL Server (p. 551).</p>
Option Group	<p>An option group for your DB instance. You can choose the default option group or you can create a custom option group.</p> <p>For more information, see Working with Option Groups (p. 156).</p>
Publicly Accessible	<p>Yes to give your DB instance a public IP address. This means that it is accessible outside the VPC (the DB instance also needs to be in a public subnet in the VPC). Choose No if you want the DB instance to only be accessible from inside the VPC.</p> <p>For more information, see Hiding a DB Instance in a VPC from the Internet (p. 422).</p>
Storage Type	<p>The storage type for your DB instance.</p> <p>For more information, see Amazon RDS Storage Types (p. 103).</p>
Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, choose default , which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC.
Time Zone	<p>The time zone for your DB instance. If you don't choose a time zone, your DB instance uses the default time zone.</p> <p>For more information, see Local Time Zone for Microsoft SQL Server DB Instances (p. 499).</p>
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, choose Not in VPC .
	<p>For more information, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 412).</p>

Setting	Setting Description
VPC Security Group	If you are a new customer to AWS, choose the default VPC. Otherwise, choose the VPC security group you previously created. When you choose Create new VPC security group in the RDS console, a new security group is created with an inbound rule that allows access to the DB instance from the IP address detected in your browser. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 399) .

Related Topics

- [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#)
- [Connecting to a DB Instance Running the Microsoft SQL Server Database Engine \(p. 514\)](#)
- [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#)
- [Deleting a DB Instance \(p. 135\)](#)

Connecting to a DB Instance Running the Microsoft SQL Server Database Engine

After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the DB instance. In this topic you connect to your DB instance by using either Microsoft SQL Server Management Studio (SSMS) or SQL Workbench/J.

For an example that walks you through the process of creating and connecting to a sample DB instance, see [Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance \(p. 18\)](#).

Connecting to Your DB Instance with Microsoft SQL Server Management Studio

In this procedure you connect to your sample DB instance by using Microsoft SQL Server Management Studio (SSMS). To download a stand-alone version of this utility, see [Download SQL Server Management Studio \(SSMS\)](#) in the Microsoft documentation.

To connect to a DB Instance using SSMS

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the upper-right corner of the Amazon RDS console, choose the AWS Region of your DB instance.
3. Find the DNS name and port number for your DB instance:
 - a. Open the RDS console and choose **Databases** to display a list of your DB instances.
 - b. Choose the name of your SQL Server DB instance to display its details.

mysqldb

Summary

DB Name mysqldb	CPU	Info Backing-up
Role Instance	Current activity 0 Connections	Engine SQL Server Express Edition

Connectivity Monitoring Logs & events Configuration Maintenance & backups Tags

Connectivity

Endpoint & port	Networking
Endpoint mysqldb rds.amazonaws.com	Availability zone us-west-2b
Port 1433	VPC vpc-2aed394c
	Subnet group

- c. On the **Connectivity** tab, copy the endpoint. Also, note the port used by the DB instance.
4. Start SQL Server Management Studio.

The **Connect to Server** dialog box appears.



5. Provide the information for your DB instance:

- a. For **Server type**, choose **Database Engine**.
- b. For **Server name**, enter the DNS name and port number of your DB Instance, separated by a comma.

Important

Change the colon between the DNS name and port number to a comma.

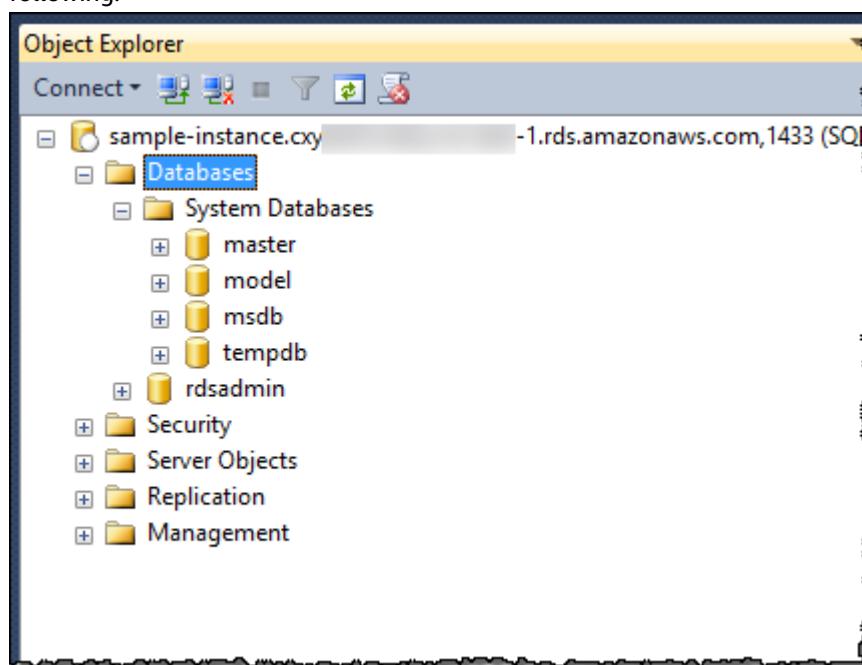
For example, your server name should look like the following.

```
sample-instance.cg034hpkmmjt.us-east-1.rds.amazonaws.com,1433
```

- c. For **Authentication**, choose **SQL Server Authentication**.
 - d. For **Login**, enter the master user name for your DB instance.
 - e. For **Password**, enter the password for your DB instance.
6. Choose **Connect**.

After a few moments, SSMS connects to your DB instance. If you can't connect to your DB instance, see [Security Group Considerations \(p. 519\)](#) and [Troubleshooting the Connection to Your SQL Server DB Instance \(p. 520\)](#).

7. Your SQL Server DB instance comes with SQL Server's standard built-in system databases (master, model, msdb, and tempdb). To explore the system databases, do the following:
- a. In SSMS, on the **View** menu, choose **Object Explorer**.
 - b. Expand your DB instance, expand **Databases**, and then expand **System Databases** as shown following.

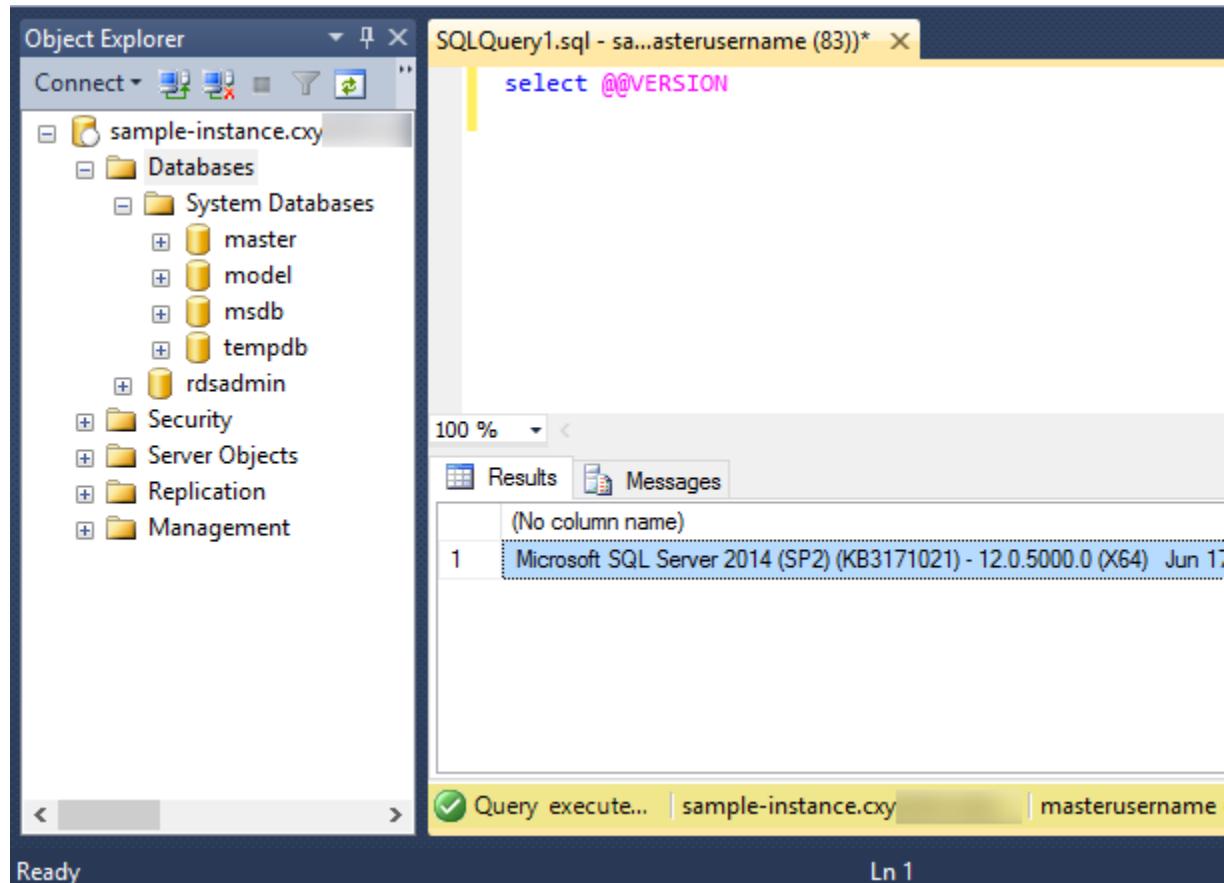


8. Your SQL Server DB instance also comes with a database named `rdsadmin`. Amazon RDS uses this database to store the objects that it uses to manage your database. The `rdsadmin` database also includes stored procedures that you can run to perform advanced tasks. For more information, see [Common DBA Tasks for Microsoft SQL Server \(p. 564\)](#).
9. You can now start creating your own databases and running queries against your DB instance and databases as usual. To run a test query against your DB instance, do the following:

- a. In SSMS, on the **File** menu point to **New** and then choose **Query with Current Connection**.
- b. Enter the following SQL query.

```
select @@VERSION
```

- c. Run the query. SSMS returns the SQL Server version of your Amazon RDS DB instance.



Connecting to Your DB Instance with SQL Workbench/J

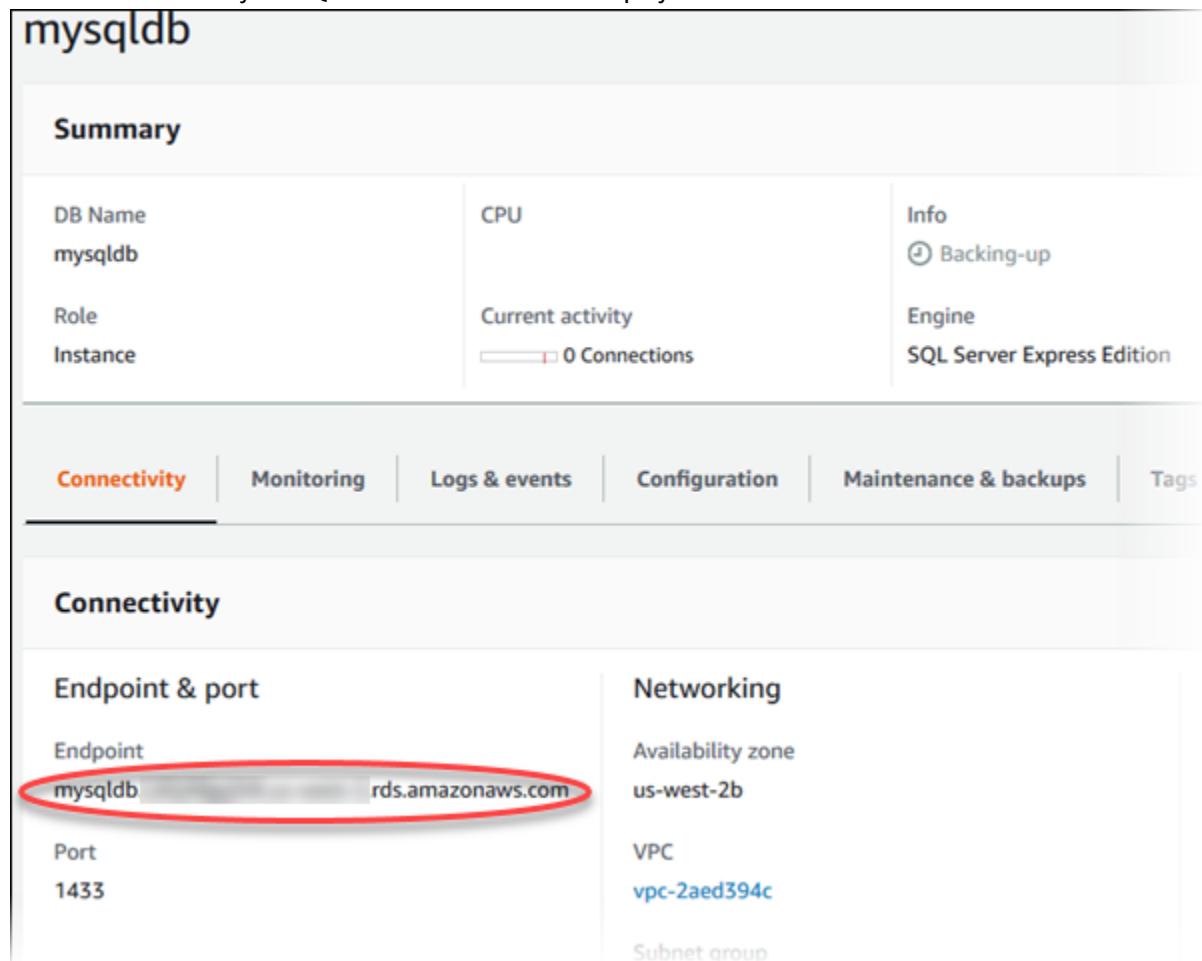
This example shows how to connect to a DB instance running the Microsoft SQL Server database engine by using the SQL Workbench/J database tool. To download SQL Workbench/J, see [SQL Workbench/J](#).

SQL Workbench/J uses JDBC to connect to your DB instance. You also need the JDBC driver for SQL Server. To download this driver, see [Microsoft JDBC Drivers 4.1 \(Preview\)](#) and [4.0 for SQL Server](#).

To connect to a DB instance using SQL Workbench

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the upper-right corner of the Amazon RDS console, choose the AWS Region of your DB instance.
3. Find the DNS name and port number for your DB instance:
 - a. Open the RDS console and then choose **Databases** to display a list of your DB instances.

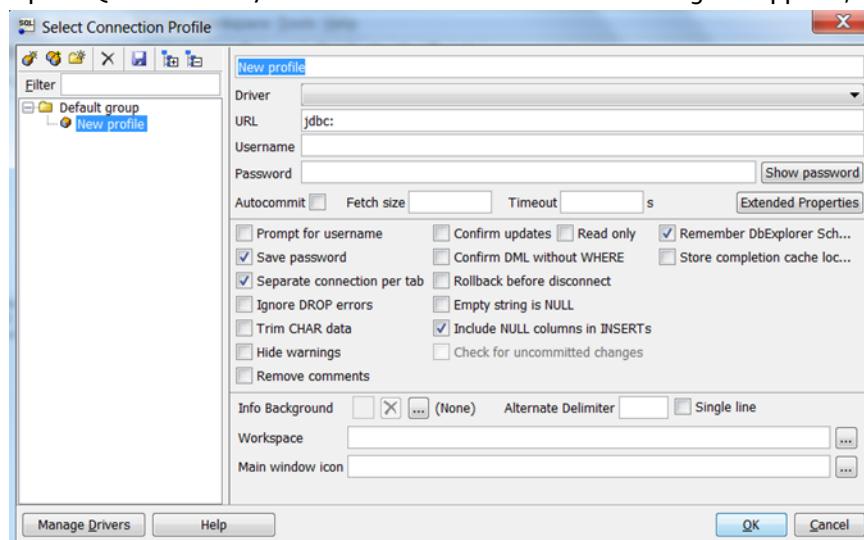
- b. Choose the name of your SQL Server DB instance to display its details.



The screenshot shows the 'mysqldb' DB instance summary and connectivity details. The 'mysqldb' instance is listed under the 'DB Name' column. Under the 'Role' column, it is listed as 'Instance'. Under the 'CPU' column, there is a progress bar indicating '0 Connections'. To the right, 'Info' shows 'Backing-up' and 'Engine' is listed as 'SQL Server Express Edition'. Below the summary, the 'Connectivity' tab is selected, showing the endpoint and port details. The 'Endpoint' field contains 'mysqlDb' and 'rds.amazonaws.com', which are both circled in red. The 'Port' field is listed as '1433'. On the right, the 'Networking' section shows the 'Availability zone' as 'us-west-2b' and 'VPC' as 'vpc-2aed394c'. A 'Subnet group' link is also present.

- c. On the **Connectivity** tab, copy the endpoint. Also, note the port used by the DB instance.

4. Open SQL Workbench/J. The **Select Connection Profile** dialog box appears, as shown following.



5. In the first box at the top of the dialog box, enter a name for the profile.

6. For **Driver**, choose **SQL JDBC 4.0**.
7. For **URL**, enter **jdbc:sqlserver://**, then enter the endpoint of your DB instance. For example, the URL value might be the following.

```
jdbc:sqlserver://sqlsvr-pdz.abcd12340.us-west-2.rds.amazonaws.com:1433
```

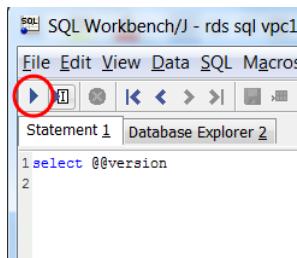
8. For **Username**, enter the master user name for the DB instance.
9. For **Password**, enter the password for the master user.
10. Choose the save icon in the dialog toolbar, as shown following.



11. Choose **OK**. After a few moments, SQL Workbench/J connects to your DB instance. If you can't connect to your DB instance, see [Security Group Considerations \(p. 519\)](#) and [Troubleshooting the Connection to Your SQL Server DB Instance \(p. 520\)](#).
12. In the query pane, enter the following SQL query.

```
select @@VERSION
```

13. Choose the execute icon in the toolbar, as shown following.



The query returns the version information for your DB instance, similar to the following.

```
Microsoft SQL Server 2012 - 11.0.2100.60 (X64)
```

Security Group Considerations

To connect to your DB instance, your DB instance must be associated with a security group that contains the IP addresses and network configuration that you use to access the DB instance. You may have associated your DB instance with an appropriate security group when you created your DB instance. If you assigned a default, non-configured security group when you created your DB instance, your DB instance firewall prevents connections.

If you need to create a new security group to enable access, the type of security group that you create will depend on what Amazon EC2 platform your DB instance is on. To determine your platform, see [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#). In general, if your DB instance is on the *EC2-Classic* platform, you create a DB security group; if your DB instance is on the *VPC* platform, you create a VPC security group. For instructions on creating a new security group, see [Controlling Access with Security Groups \(p. 394\)](#).

After you have created the new security group, you modify your DB instance to associate it with the security group. For more information, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).

You can enhance security by using SSL to encrypt connections to your DB instance. For more information, see [Using SSL with a Microsoft SQL Server DB Instance \(p. 555\)](#).

Troubleshooting the Connection to Your SQL Server DB Instance

The following are issues you might encounter when you attempt to connect to your SQL Server DB instance.

Issue	Troubleshooting Suggestions
Unable to connect to your DB instance.	For a newly-created DB instance, the DB instance has a status of creating until the DB instance is ready to use. When the state changes to available , you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.
Unable to connect to your DB instance.	If you can't send or receive communications over the port that you specified when you created the DB instance, you can't connect to the DB instance. Check with your network administrator to verify that the port you specified for your DB instance allows inbound and outbound communication.
Unable to connect to your DB instance.	<p>The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the security group for the DB instance might not match. The problem is most likely the egress or ingress rules on your firewall. For more information about security groups, see Controlling Access with Security Groups (p. 394).</p> <p>For a topic that walks you through the process of setting up rules for your security group, see Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 427).</p>
Could not open a connection to SQL Server – Microsoft SQL Server, Error: 53	<p>Make sure specified the server name correctly. For Server name, enter the DNS name and port number of your sample DB Instance, separated by a comma.</p> <p>Important Change the colon between the DNS name and port number to a comma.</p> <p>For example, your server name should look like the following:</p> <pre data-bbox="674 1501 1470 1537">sample-instance.cg034hpkmmjt.us-east-1.rds.amazonaws.com,1433</pre>
No connection could be made because the target machine actively refused it – Microsoft SQL Server, Error: 10061	You were able to reach the DB instance but the connection was refused. This is usually caused by specifying the user name or password incorrectly. Verify the user name and password and then retry.

Modifying a DB Instance Running the Microsoft SQL Server Database Engine

You can change the settings of a DB instance to accomplish tasks such as changing the instance class or renaming the instance. This topic guides you through modifying an Amazon RDS DB instance running Microsoft SQL Server, and describes the settings for SQL Server DB instances.

We recommend that you test any changes on a test instance before modifying a production instance, so that you fully understand the impact of each change. This is especially important when upgrading database versions.

After you modify your DB instance settings, you can apply the changes immediately, or apply them during the next maintenance window for the DB instance. Some modifications cause an interruption by restarting the DB instance.

AWS Management Console

To modify an SQL Server DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want. For information about each setting, see [Settings for Microsoft SQL Server DB Instances \(p. 522\)](#).
5. When all the changes are as you want them, choose **Continue**.
6. To apply the changes immediately, choose **Apply Immediately**. Choosing this option can cause an outage in some cases. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify a Microsoft SQL Server DB instance by using the AWS CLI, call the [modify-db-instance](#) command. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see [Settings for Microsoft SQL Server DB Instances \(p. 522\)](#).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days). The code enables automatic minor version upgrades by using `--auto-minor-version-upgrade`. To disable automatic minor version upgrades, use `--no-auto-minor-version-upgrade`. The changes are applied during the next maintenance window by using `--no-apply-immediately`. Use `--apply-immediately` to apply the changes immediately. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
```

```
--backup-retention-period 7 \
--auto-minor-version-upgrade \
--no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--backup-retention-period 7 ^
--auto-minor-version-upgrade ^
--no-apply-immediately
```

API

To modify a Microsoft SQL Server DB instance by using the Amazon RDS API, call the [ModifyDBInstance](#) action. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see [Settings for Microsoft SQL Server DB Instances \(p. 522\)](#).

Example

The following code modifies *mydbinstance* by setting the backup retention period to 1 week (7 days) and enabling automatic minor version upgrades. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
?Action=ModifyDBInstance
&ApplyImmediately=false
&AutoMinorVersionUpgrade=true
&BackupRetentionPeriod=7
&DBInstanceIdentifier=mydbinstance
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
&X-Amz-Date=20131016T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for Microsoft SQL Server DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated Storage	<p>The storage, in gibibytes, that you want to allocate for your DB instance. You can only increase the allocated storage, you can't reduce the allocated storage. The maximum storage allowed is 16 TiB.</p> <p>Warning Once Amazon RDS begins to modify your DB instance to increase the storage size</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	A short outage of a few minutes may occur. After that, the DB instance is online but in the storage-optimization state. Performance may be degraded during storage optimization. The

Setting	Setting Description	When the Change Occurs	Downtime Notes
	<p>or type, you can't submit another request to increase the storage size or type for 6 hours.</p> <p>You can't modify the storage of some older DB instances, and DB instances restored from older DB snapshots. The Allocated Storage option is disabled in the console if your DB instance isn't eligible. You can also check eligibility by using the AWS CLI command describe-valid-db-instance-modifications which returns the valid storage options for your DB instance.</p> <p>For more information, see DB instance storage (p. 103).</p>		storage optimization process is usually short, but can sometimes take up to and even beyond 24 hours.
Auto Minor Version Upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.	–	–
Backup Retention Period	<p>The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0.</p> <p>For more information, see Working With Backups (p. 208).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false and you change the setting from a nonzero value to another nonzero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.</p>	An outage occurs if you change from 0 to a nonzero value, or from a nonzero value to 0.
Backup Window	<p>The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours.</p> <p>For more information, see Working With Backups (p. 208).</p>	The change is applied asynchronously, as soon as possible.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Certificate Authority	The certificate that you want to use.	–	–
Copy Tags to Snapshots	If you have any DB instance tags, this option copies them when you create a DB snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .	The change occurs immediately. This setting ignores the Apply immediately setting.	–
Database Port	The port that you want to use to access the database. The port value must not match any of the port values specified for options in the option group for the DB instance.	The change occurs immediately. This setting ignores the Apply Immediately setting.	The DB instance is rebooted immediately.
DB Engine Version	The version of the SQL Server database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications. For more information, see Upgrading the Microsoft SQL Server DB Engine (p. 529) .	If Apply Immediately is set to true, the change occurs immediately. If Apply Immediately is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Class	The DB instance class that you want to use. For more information, see DB Instance Class (p. 82) and DB Instance Class Support for Microsoft SQL Server (p. 491) .	If Apply Immediately is set to true, the change occurs immediately. If Apply Immediately is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Identifier	The DB instance identifier. For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 126) .	If Apply Immediately is set to true, the change occurs immediately. If Apply Immediately is set to false, the change occurs during the next maintenance window.	An outage occurs during this change. The DB instance is rebooted.

Setting	Setting Description	When the Change Occurs	Downtime Notes
DB Parameter Group	<p>The parameter group that you want associated with the DB instance.</p> <p>For more information, see Working with DB Parameter Groups (p. 169).</p>	The parameter group change occurs immediately.	<p>An outage doesn't occur during this change. When you change the parameter group, changes to some parameters are applied to the DB instance immediately without a reboot. Changes to other parameters are applied only after the DB instance is rebooted.</p> <p>For more information, see Rebooting a DB Instance (p. 129).</p>
Domain	<p>The Active Directory Domain to move the instance to. Specify none to remove the instance from its current domain. The domain must exist prior to this operation.</p> <p>For more information, see Using Windows Authentication with a Microsoft SQL Server DB Instance (p. 578).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	<p>A brief outage occurs during this change. For single-AZ DB instances, the outage is approximately 5-10 minutes. For multi-AZ DB instances, the outage is approximately 1 minute.</p>
Domain IAM Role Name	<p>The name of the IAM role to use when accessing the Active Directory Service.</p> <p>For more information, see Using Windows Authentication with a Microsoft SQL Server DB Instance (p. 578).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	A brief outage occurs during this change.
Enable deletion protection	Enable deletion protection to prevent your DB instance from being deleted. For more information, see Deleting a DB Instance (p. 135) .	–	–
Enable Enhanced Monitoring	<p>Yes to enable gathering metrics in real time for the operating system that your DB instance runs on.</p> <p>For more information, see Enhanced Monitoring (p. 256).</p>	–	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
License Model	Choose license-included to use the general license agreement for Microsoft SQL Server.	If Apply Immediately is set to true, the change occurs immediately. If Apply Immediately is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
Maintenance Window	The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours. If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied. For more information, see The Amazon RDS Maintenance Window (p. 120) .	The change occurs immediately. This setting ignores the Apply Immediately setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.
Multi-AZ Deployment	Yes to have a passive secondary replica of your DB instance created in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. No for development and testing. If your DB instance is running Mirroring (not Always On) with SQL Server 2014, 2016, or 2017 Enterprise Edition, and has in-memory optimization enabled, disable in-memory optimization before you add Multi-AZ. Always On doesn't require this step. For more information, see Multi-AZ Deployments for Microsoft SQL Server (p. 551) .	If Apply Immediately is set to true, the change occurs immediately. If Apply Immediately is set to false, the change occurs during the next maintenance window.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
New Master Password	<p>The password for your master user. The password must contain from 8 to 128 printable ASCII characters (excluding /, ", a space, and @). By resetting the master password, you also reset permissions for the DB instance.</p> <p>For more information, see Resetting the DB Instance Owner Role Password (p. 1079).</p>	The change is applied asynchronously, as soon as possible. This setting ignores the Apply Immediately setting.	–
Option Group	<p>The option group that you want associated with the DB instance.</p> <p>For more information, see Working with Option Groups (p. 156).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	–
Publicly Accessible	<p>Yes to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. No to make the DB instance accessible only from inside the VPC.</p> <p>For more information, see Hiding a DB Instance in a VPC from the Internet (p. 422).</p>	The change occurs immediately. This setting ignores the Apply Immediately setting.	–
Security Group	<p>The security group you want associated with the DB instance.</p> <p>For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 399).</p>	The change is applied asynchronously, as soon as possible. This setting ignores the Apply Immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Storage Type	<p>The storage type that you want to use.</p> <p>You can't change from or to magnetic storage.</p> <p>For more information, see Amazon RDS Storage Types (p. 103).</p> <p>Warning Once Amazon RDS begins to modify your DB instance to change the storage size or type, you can't submit another request to change the storage size or type for 6 hours.</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	<p>The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place.</p> <ul style="list-style-type: none"> From General Purpose (SSD) to Provisioned IOPS (SSD). From Provisioned IOPS (SSD) to General Purpose (SSD).
Subnet Group	<p>The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC.</p> <p>For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 426).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change. The DB instance is rebooted.

Related Topics

- [Rebooting a DB Instance \(p. 129\)](#)
- [Connecting to a DB Instance Running the Microsoft SQL Server Database Engine \(p. 514\)](#)
- [Upgrading the Microsoft SQL Server DB Engine \(p. 529\)](#)
- [Deleting a DB Instance \(p. 135\)](#)

Upgrading the Microsoft SQL Server DB Engine

When Amazon RDS supports a new version of Microsoft SQL Server, you can upgrade your DB instances to the new version. Amazon RDS supports the following upgrades to a Microsoft SQL Server DB instance:

- Major Version Upgrades
- Minor Version Upgrades

In general, a major engine version upgrade can introduce changes that are not compatible with existing applications. In contrast, a minor version upgrade includes only changes that are backward-compatible with existing applications.

You must modify the DB instance manually to perform a major version upgrade. Minor version upgrades occur automatically if you enable auto minor version upgrades on your DB instance. In all other cases, you must modify the DB instance manually to perform a minor version upgrade.

For information about what SQL Server versions are available on Amazon RDS, see [Microsoft SQL Server on Amazon RDS \(p. 488\)](#).

Topics

- [Overview of Upgrading \(p. 529\)](#)
- [Major Version Upgrades \(p. 529\)](#)
- [Multi-AZ and In-Memory Optimization Considerations \(p. 530\)](#)
- [Option and Parameter Group Considerations \(p. 530\)](#)
- [Testing an Upgrade \(p. 531\)](#)
- [Upgrading a SQL Server DB Instance \(p. 532\)](#)

Overview of Upgrading

Amazon RDS takes two DB snapshots during the upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken after the upgrade completes.

Note

Amazon RDS only takes DB snapshots if you have set the backup retention period for your DB instance to a number greater than 0. To change your backup retention period, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).

After an upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the DB snapshot that was taken before the upgrade to create a new DB instance.

During a minor or major version upgrade of SQL Server, the **Free Storage Space** and **Disk Queue Depth** metrics will display -1. After the upgrade is complete, both metrics will return to normal.

Major Version Upgrades

Amazon RDS currently supports the following major version upgrades to a Microsoft SQL Server DB instance.

You can upgrade your existing DB instance to SQL Server 2017 from any version except SQL Server 2008. To upgrade from SQL Server 2008, first upgrade to one of the other versions first.

Current Version	Supported Upgrade Versions
SQL Server 2014	SQL Server 2017 SQL Server 2016
SQL Server 2012	SQL Server 2017 SQL Server 2016 SQL Server 2014
SQL Server 2008 R2	SQL Server 2016 SQL Server 2014 SQL Server 2012

Database Compatibility Level

You can use Microsoft SQL Server database compatibility levels to adjust some database behaviors to mimic previous versions of SQL Server. For more information, see [Compatibility Level](#) in the Microsoft documentation.

When you upgrade your DB instance, all existing databases remain at their original compatibility level. For example, if you upgrade from SQL Server 2012 to SQL Server 2014, all existing databases have a compatibility level of 110. Any new database created after the upgrade have compatibility level 120.

You can change the compatibility level of a database by using the ALTER DATABASE command. For example, to change a database named `customeracct` to be compatible with SQL Server 2014, issue the following command:

```
ALTER DATABASE customeracct SET COMPATIBILITY_LEVEL = 120
```

Multi-AZ and In-Memory Optimization Considerations

Amazon RDS supports Multi-AZ deployments for DB instances running Microsoft SQL Server by using SQL Server Database Mirroring or Always On. For more information, see [Multi-AZ Deployments for Microsoft SQL Server \(p. 551\)](#).

If your DB instance is in a Multi-AZ deployment, both the primary and standby instances are upgraded. Amazon RDS does rolling upgrades. You have an outage only for the duration of a failover.

SQL Server 2014/2016/2017 Enterprise Edition supports in-memory optimization.

Option and Parameter Group Considerations

Option Group Considerations

If your DB instance uses a custom option group, in some cases Amazon RDS can't automatically assign your DB instance a new option group. For example, when you upgrade to a new major version. In that

case, you must specify a new option group when you upgrade. We recommend that you create a new option group, and add the same options to it as your existing custom option group.

For more information, see [Creating an Option Group \(p. 157\)](#) or [Making a Copy of an Option Group \(p. 159\)](#).

Parameter Group Considerations

If your DB instance uses a custom parameter group, in some cases Amazon RDS can't automatically assign your DB instance a new parameter group. For example, when you upgrade to a new major version. In that case, you must specify a new parameter group when you upgrade. We recommend that you create a new parameter group, and configure the parameters as in your existing custom parameter group.

For more information, see [Creating a DB Parameter Group \(p. 170\)](#) or [Copying a DB Parameter Group \(p. 174\)](#).

Testing an Upgrade

Before you perform a major version upgrade on your DB instance, you should thoroughly test your database, and all applications that access the database, for compatibility with the new version. We recommend that you use the following procedure.

To test a major version upgrade

1. Review the upgrade documentation for the new version of the database engine to see if there are compatibility issues that might affect your database or applications:
 - [Upgrade to SQL Server 2016 or 2017](#)
 - [Upgrade to SQL Server 2014](#)
 - [Upgrade to SQL Server 2012](#)
2. If your DB instance uses a custom option group, create a new option group compatible with the new version you are upgrading to. For more information, see [Option Group Considerations \(p. 530\)](#).
3. If your DB instance uses a custom parameter group, create a new parameter group compatible with the new version you are upgrading to. For more information, see [Parameter Group Considerations \(p. 531\)](#).
4. Create a DB snapshot of the DB instance to be upgraded. For more information, see [Creating a DB Snapshot \(p. 216\)](#).
5. Restore the DB snapshot to create a new test DB instance. For more information, see [Restoring from a DB Snapshot \(p. 218\)](#).
6. Modify this new test DB instance to upgrade it to the new version, by using one of the following methods:
 - [Upgrading the Engine Version of a DB Instance Using the Console \(p. 123\)](#)
 - [Upgrading the Engine Version of a DB Instance Using the AWS CLI \(p. 124\)](#)
 - [Upgrading the Engine Version of a DB Instance Using the RDS API \(p. 124\)](#)
7. Evaluate the storage used by the upgraded instance to determine if the upgrade requires additional storage.
8. Run as many of your quality assurance tests against the upgraded DB instance as needed to ensure that your database and application work correctly with the new version. Implement any new tests needed to evaluate the impact of any compatibility issues you identified in step 1. Test all stored procedures and functions. Direct test versions of your applications to the upgraded DB instance.
9. If all tests pass, then perform the upgrade on your production DB instance. We recommend that you do not allow write operations to the DB instance until you confirm that everything is working correctly.

Upgrading a SQL Server DB Instance

For information about manually or automatically upgrading a SQL Server DB instance, see [Upgrading a DB Instance Engine Version \(p. 123\)](#).

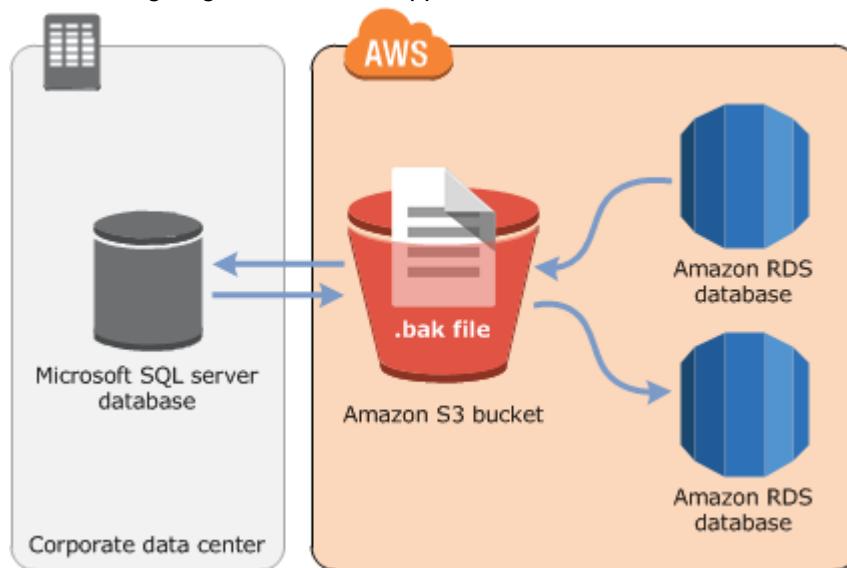
Importing and Exporting SQL Server Databases

Amazon RDS supports native backup and restore for Microsoft SQL Server databases using full backup files (.bak files). When you use RDS, you access files stored in Amazon S3 rather than using the local file system on the database server.

For example, you can create a full backup from your local server, store it on S3, and then restore it onto an existing Amazon RDS DB instance. You can also make backups from RDS, store them on S3, and then restore them wherever you wish.

Native backup and restore is available in all AWS Regions, and for both Single-AZ and Multi-AZ DB instances. Native backup and restore is available for all editions of Microsoft SQL Server supported on Amazon RDS.

The following diagram shows the supported scenarios.



Using native .bak files to back up and restore databases is usually the fastest way to backup and restore databases. There are many additional advantages to using native backup and restore. For example, you can do the following:

- Migrate databases to or from Amazon RDS.
- Move databases between Amazon RDS SQL Server DB instances.
- Migrate data, schemas, stored procedures, triggers and other database code inside a .bak file.
- Backup and restore single databases, instead of entire DB instances.
- Create copies of databases for development, testing, training, and demonstrations.
- Store and transfer backup files with Amazon S3, for an added layer of protection for disaster recovery.

Limitations and Recommendations

The following are some limitations to using native backup and restore:

- You can't back up to, or restore from, an Amazon S3 bucket in a different AWS Region than your Amazon RDS DB instance.

- We strongly recommend that you don't restore a backup file from one time zone to a different time zone. If you restore a backup file from one time zone to a different time zone, you must audit your queries and applications for the effects of the time zone change.
- Backing up databases larger than 1 TB in size is not supported.
- You can't restore databases larger than 4 TB in size. Restoring databases on SQL-Server Express is limited by the MSSQL edition to 10GB or less.
- You can't back up a database during the maintenance window, or any time Amazon RDS is in the process of taking a snapshot of the database.
- On Multi-AZ DB instances, you can only restore databases backed up in full recovery model.
- Calling the RDS procedures for backup/restore within a transaction is not supported.
- Backup files are encrypted with the specified KMS key using the "Encryption-Only" crypto mode. When you are restoring encrypted backup files, you should be aware they were encrypted with the "Encryption-Only" crypto mode.

We recommend that you use native backup and restore to migrate your database to Amazon RDS if your database can be offline while the backup file is created, copied, and restored. If your on-premises database can't be offline, we recommend that you use the AWS Database Migration Service to migrate your database to Amazon RDS. For more information, see [What Is AWS Database Migration Service?](#)

Native backup and restore is not intended to replace the data recovery capabilities of the cross-region snapshot copy feature. We recommend that you use snapshot copy to copy your database snapshot to another region for cross-region disaster recovery in Amazon RDS. For more information, see [Copying a Snapshot \(p. 221\)](#).

Setting Up for Native Backup and Restore

There are three components you'll need to set up for native backup and restore:

- An Amazon S3 bucket to store your backup files.

You need to have an S3 bucket to use for your backup files and then upload backups you want to migrate to RDS. If you already have an Amazon S3 bucket, you can use that. If you don't, you can [create a bucket](#). Alternatively, you can choose to have a new bucket created for you when you add the `SQLSERVER_BACKUP_RESTORE` option by using the AWS Management Console.

For information on using S3, see the Amazon Simple Storage Service Getting Started Guide for a simple introduction, or the Amazon Simple Storage Service Console User Guide for in-depth details.

- An AWS Identity and Access Management (IAM) role to access the bucket.

If you already have an IAM role, you can use that. If you don't have an IAM role, you can create a new one manually. Alternatively, you can choose to have a new IAM role created for you when you add the `SQLSERVER_BACKUP_RESTORE` option by using the AWS Management Console. If you want to create a new IAM role manually, or attach trust and permissions policies to an existing IAM role, take the approach discussed in the next section.

- The `SQLSERVER_BACKUP_RESTORE` option added to an option group on your DB instance.

To enable native backup and restore on your DB instance, you add the `SQLSERVER_BACKUP_RESTORE` option to an option group on your DB instance. For more information and instructions, see [Microsoft SQL Server Native Backup and Restore Support \(p. 559\)](#).

Manually Creating an IAM Role for Native Backup and Restore

If you want to manually create a new IAM role to use with native backup and restore, you create a role to delegate permissions from the Amazon RDS service to your Amazon S3 bucket. When you create an

IAM role, you attach trust and permissions policies. The trust policy allows rds to assume this role. The permission policy defines the actions this Role can do. For more information about creating the role, see [Creating a Role to Delegate Permissions to an AWS Service](#).

For the native backup and restore feature, use trust and permissions policies similar to the examples in this section. In following example, we use the service principle name `rds.amazonaws.com` as an alias for all service accounts. In the other examples, we specify an Amazon Resource Name (ARN) to identify another account, user, or role that we're granting access to in the trust policy.

Example Trust Policy for Native Backup and Restore

```
{  
    "Version": "2012-10-17",  
    "Statement":  
    [ {  
        "Effect": "Allow",  
        "Principal": {"Service": "rds.amazonaws.com"},  
        "Action": "sts:AssumeRole"  
    } ]  
}
```

The following example uses an ARN to specify a resource. For more information on using ARNs, see [Amazon Resource Names \(ARNs\)](#).

Example Permissions Policy for Native Backup and Restore Without Encryption Support

```
{  
    "Version": "2012-10-17",  
    "Statement":  
    [ {  
        "Effect": "Allow",  
        "Action":  
            [  
                "s3>ListBucket",  
                "s3:GetBucketLocation"  
            ],  
        "Resource": "arn:aws:s3:::bucket_name"  
    },  
    {  
        "Effect": "Allow",  
        "Action":  
            [  
                "s3.GetObject",  
                "s3:PutObject",  
                "s3>ListMultipartUploadParts",  
                "s3:AbortMultipartUpload"  
            ],  
        "Resource": "arn:aws:s3:::bucket_name/*"  
    } ]  
}
```

Example Permissions Policy for Native Backup and Restore with Encryption Support

If you want to encrypt your backup files, include an encryption key in your permissions policy. For more information about encryption keys, see [Getting Started](#) in the AWS Key Management Service (AWS KMS) documentation.

```
{  
    "Version": "2012-10-17",  
    "Statement":
```

```
[  
  {  
    "Effect": "Allow",  
    "Action":  
      [  
        "kms:DescribeKey",  
        "kms:GenerateDataKey",  
        "kms:Encrypt",  
        "kms:Decrypt"  
      ],  
    "Resource": "arn:aws:kms:region:account-id:key/key-id"  
  },  
  {  
    "Effect": "Allow",  
    "Action":  
      [  
        "s3>ListBucket",  
        "s3:GetBucketLocation"  
      ],  
    "Resource": "arn:aws:s3:::bucket_name"  
  },  
  {  
    "Effect": "Allow",  
    "Action":  
      [  
        "s3:GetObject",  
        "s3:PutObject",  
        "s3>ListMultipartUploadParts",  
        "s3:AbortMultipartUpload"  
      ],  
    "Resource": "arn:aws:s3:::bucket_name/*"  
  }  
]
```

Using Native Backup and Restore

After you have enabled and configured native backup and restore, you can start using it. First you connect to your Microsoft SQL Server database, and then call an Amazon RDS stored procedure to do the work. For instructions on connecting to your database, see [Connecting to a DB Instance Running the Microsoft SQL Server Database Engine \(p. 514\)](#).

Some of the stored procedures require that you provide an Amazon Resource Name (ARN) to your Amazon S3 bucket and file. The format for your ARN is `arn:aws:s3:::bucket_name/file_name`. Amazon S3 doesn't require an account number or region in ARNs. If you also provide an optional AWS KMS encryption key, the format for your ARN is `arn:aws:kms:region:account-id:key/key-id`. For more information, see [Amazon Resource Names \(ARNs\) and AWS Service Namespaces](#).

There are stored procedures for backing up your database, restoring your database, canceling tasks that are in progress, and tracking the status of the backup and restore tasks. For instructions on how to call each stored procedure, see the following subsections:

- [Backing Up a Database \(p. 536\)](#)
- [Restoring a Database \(p. 538\)](#)
- [Canceling a Task \(p. 538\)](#)
- [Tracking the Status of Tasks \(p. 538\)](#)

Backing Up a Database

To back up your database, you call the `rds_backup_database` stored procedure.

Note

You can't back up a database during the maintenance window, or when Amazon RDS is taking a snapshot.

The following parameters are required:

- **@source_db_name** – The name of the database to back up
- **@s3_arn_to_backup_to** – The bucket to use for the backup, plus the name of the file (Amazon S3 bucket + key ARN).

The file can have any extension, but `.bak` is traditional.

The following parameters are optional:

- **@kms_master_key_arn** – The key to encrypt the backup (KMS customer master key ARN).

For more information about encryption keys, see [Getting Started](#) in the AWS Key Management Service (AWS KMS) documentation.

- **@overwrite_S3_backup_file** – Defaults to 0
 - 0 – Don't overwrite the existing file. Return an error instead if the file already exists.
 - 1 – Overwrite an existing file that has the specified name, even if it isn't a backup file.
- **@type** – Defaults to `FULL`, not case sensitive
 - `differential` – Take a differential backup.
 - `full` – Take a full backup.

Example Differential Backup without Encryption

```
exec msdb.dbo.rds_backup_database
    @source_db_name='database_name',
    @s3_arn_to_backup_to='arn:aws:s3:::bucket_name/file_name_and_extension',
    @overwrite_S3_backup_file=1,
    @type='differential';
```

Example Full Backup with Encryption

```
exec msdb.dbo.rds_backup_database
    @source_db_name='database_name',
    @s3_arn_to_backup_to='arn:aws:s3:::bucket_name/file_name_and_extension',
    @kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id',
    @overwrite_S3_backup_file=1,
    @type='FULL';
```

The differential backup is based on the last full backup. For differential backups to work, you can't take a snapshot between the last full backup and the differential backup. If you want to take a differential backup, and a snapshot exists, make another full backup before proceeding with the differential.

You can look for the last full backup or snapshot using the following sample SQL:

```
select top 1
    database_name
    , backup_start_date
    , backup_finish_date
    from msdb.dbo.backupset
    where database_name='name-of-database'
    and type = 'D'
```

```
order by backup_start_date desc;
```

Restoring a Database

To restore your database, you call the `rds_restore_database` stored procedure.

The following parameters are required:

- `@restore_db_name` – The name of the database to restore.
- `@s3_arn_to_restore_from` – The Amazon S3 bucket that contains the backup file, and the name of the file.

The following parameters are optional:

- `@kms_master_key_arn` – If you encrypted the backup file, the key to use to decrypt the file.

Example Without Encryption

```
exec msdb.dbo.rds_restore_database
    @restore_db_name='database_name',
    @s3_arn_to_restore_from='arn:aws:s3:::bucket_name/file_name_and_extension';
```

Example With Encryption

```
exec msdb.dbo.rds_restore_database
    @restore_db_name='database_name',
    @s3_arn_to_restore_from='arn:aws:s3:::bucket_name/file_name_and_extension',
    @kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id';
```

Cancelling a Task

To cancel a backup or restore task, you call the `rds_cancel_task` stored procedure.

The following parameters are optional:

- `@db_name` – The name of the database to cancel the task for.
- `@task_id` – The ID of the task to cancel. You can get the task ID by calling `rds_task_status`.

Example

```
exec msdb.dbo.rds_cancel_task @task_id=1234;
```

Tracking the Status of Tasks

To track the status of your backup and restore tasks, you call the `rds_task_status` stored procedure. If you don't provide any parameters, the stored procedure returns the status of all tasks. The status for tasks is updated approximately every 2 minutes.

The following parameters are optional:

- `@db_name` – The name of the database to show the task status for.

- **@task_id** – The ID of the task to show the task status for.

Example

```
exec msdb.dbo.rds_task_status @db_name='database_name';
```

The `rds_task_status` stored procedure returns the following columns.

Column	Description
<code>task_id</code>	The ID of the task.
<code>task_type</code>	Either <code>BACKUP_DB</code> for a back up task, or <code>RESTORE_DB</code> for a restore task.
<code>database_name</code>	The name of the database that the task is associated with.
<code>% complete</code>	The progress of the task as a percentage.
<code>duration (mins)</code>	The amount of time spent on the task, in minutes.
<code>lifecycle</code>	<p>The status of the task. The possible statuses for a task are the following:</p> <ul style="list-style-type: none"> • <code>CREATED</code> – As soon as you call <code>rds_backup_database</code> or <code>rds_restore_database</code>, a task is created and the status is set to <code>CREATED</code>. • <code>IN_PROGRESS</code> – After a backup or restore task starts, the status is set to <code>IN_PROGRESS</code>. It can take up to 5 minutes for the status to change from <code>CREATED</code> to <code>IN_PROGRESS</code>. • <code>SUCCESS</code> – After a backup or restore task completes, the status is set to <code>SUCCESS</code>. • <code>ERROR</code> – If a backup or restore task fails, the status is set to <code>ERROR</code>. Read the <code>task_info</code> column for more information about the error. • <code>CANCEL_REQUESTED</code> – As soon as you call <code>rds_cancel_task</code>, the status of the task is set to <code>CANCEL_REQUESTED</code>. • <code>CANCELLED</code> – After a task is successfully canceled, the status of the task is set to <code>CANCELLED</code>.
<code>task_info</code>	<p>Additional information about the task.</p> <p>If an error occurs while backing up or restoring a database, this column contains information about the error. For a list of possible errors, and mitigation strategies, see Troubleshooting (p. 540).</p>
<code>last_updated</code>	The date and time that the task status was last updated. The status is updated after every 5% of progress.
<code>created_at</code>	The date and time that the task was created.
<code>overwrite_S3_backup_file</code>	The value of the <code>@overwrite_S3_backup_file</code> parameter specified when calling a backup task. For more information, see Backing Up a Database (p. 536) .

Compressing Backup Files

To save space in your Amazon S3 bucket, you can compress your backup files. For more information about compressing backup files, see [Backup Compression](#) in the Microsoft documentation.

Compressing your backup files is supported for the following database editions:

- Microsoft SQL Server Enterprise Edition
- Microsoft SQL Server Standard Edition

To turn on compression for your backup files, run the following code:

```
exec rdsadmin..rds_set_configuration 'S3 backup compression', 'true';
```

To turn off compression for your backup files, run the following code:

```
exec rdsadmin..rds_set_configuration 'S3 backup compression', 'false';
```

Troubleshooting

The following are issues you might encounter when you use native backup and restore.

Issue	Troubleshooting Suggestions
Access Denied	<p>The backup or restore process is unable to access the backup file. This is usually caused by issues like the following:</p> <ul style="list-style-type: none">• Referencing the incorrect bucket. Referencing the bucket using an incorrect format. Referencing a file name without using the ARN.• Incorrect permissions on the bucket file. For example, if it is created by a different account that is trying to access it now, add the correct permissions.• An IAM policy that is incorrect or incomplete. Your IAM role must include all the necessary elements, including for example, the correct version. These are highlighted in Importing and Exporting SQL Server Databases (p. 533).
BACKUP DATABASE WITH COMPRESSION is not supported on <edition_name> Edition	<p>Compressing your backup files is only supported for Microsoft SQL Server Enterprise Edition and Standard Edition.</p> <p>For more information, see Compressing Backup Files (p. 539).</p>
Key <ARN> does not exist	<p>You attempted to restore an encrypted backup, but didn't provide a valid encryption key. Check your encryption key and retry.</p> <p>For more information, see Restoring a Database (p. 538).</p>
Please reissue task with correct type and overwrite property	<p>If you attempt to back up your database and provide the name of a file that already exists, but set the overwrite property to false, the save operation fails. To fix this error, either provide the name of a file that doesn't already exist, or set the overwrite property to true.</p> <p>For more information, see Backing Up a Database (p. 536).</p> <p>It's also possible that you intended to restore your database, but called the <code>rds_backup_database</code> stored procedure accidentally. In that case, call the <code>rds_restore_database</code> stored procedure instead.</p> <p>For more information, see Restoring a Database (p. 538).</p>

Issue	Troubleshooting Suggestions
	If you intended to restore your database and called the <code>rds_restore_database</code> stored procedure, make sure that you provided the name of a valid backup file. For more information, see Using Native Backup and Restore (p. 536) .
Please specify a bucket that is in the same region as RDS instance	You can't back up to, or restore from, an Amazon S3 bucket in a different AWS Region than your Amazon RDS DB instance. You can use Amazon S3 replication to copy the backup file to the correct region. For more information, see Cross-Region Replication in the Amazon S3 documentation.
The specified bucket does not exist	Verify that you have provided the correct ARN for your bucket and file, in the correct format. For more information, see Using Native Backup and Restore (p. 536) .
User <ARN> is not authorized to perform <kms action> on resource <ARN>	You requested an encrypted operation, but didn't provide correct AWS KMS permissions. Verify that you have the correct permissions, or add them. For more information, see Setting Up for Native Backup and Restore (p. 534) .

Related Topics

- [Importing and Exporting SQL Server Data Using Other Methods \(p. 542\)](#)
- [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#)

Importing and Exporting SQL Server Data Using Other Methods

Following, you can find information about importing your Microsoft SQL Server data to Amazon RDS, and exporting your data from an Amazon RDS DB instance running SQL Server, by using snapshots.

If your scenario supports it, it is easier to move data in and out of Amazon RDS by using the native backup and restore functionality. For more information, see [Importing and Exporting SQL Server Databases \(p. 533\)](#).

Note

Amazon RDS for Microsoft SQL Server does not support importing data into the msdb database.

Importing Data into SQL Server on Amazon RDS by Using a Snapshot

To import data into a SQL Server DB instance by using a snapshot

1. Create a DB instance. For more information, see [Creating a DB Instance Running the Microsoft SQL Server Database Engine \(p. 504\)](#).
2. Stop applications from accessing the destination DB instance.

If you prevent access to your DB instance while you are importing data, data transfer is faster. Additionally, you won't need to worry about conflicts while data is being loaded if other applications cannot write to the DB instance at the same time. If something goes wrong and you have to roll back to a prior database snapshot, the only changes that you lose are the imported data, which you can import again after you resolve the issue.

For information about controlling access to your DB instance, see [Working with DB Security Groups \(EC2-Classic Platform\) \(p. 399\)](#).

3. Create a snapshot of the target database.

If the target database is already populated with data, we recommend that you take a snapshot of the database before you import the data. If something goes wrong with the data import or you want to discard the changes, you can restore the database to its previous state by using the snapshot. For information about database snapshots, see [Creating a DB Snapshot \(p. 216\)](#).

Note

When you take a database snapshot, I/O operations to the database are suspended for about 10 seconds while the backup is in progress.

4. Disable automated backups on the target database.

Disabling automated backups on the target DB instance will improve performance while you are importing your data because Amazon RDS doesn't log transactions when automatic backups are disabled. However, there are some things to consider. Because automated backups are required to perform a point-in-time recovery, you won't be able to restore the database to a specific point in time while you are importing data. Additionally, any automated backups that were created on the DB instance are erased. You can still use previous snapshots to recover the database, and any snapshots that you have taken will remain available. For information about automated backups, see [Working With Backups \(p. 208\)](#).

5. Disable foreign key constraints, if applicable.

If you need to disable foreign key constraints, you can do so with the following script.

```
--Disable foreign keys on all tables
```

```
DECLARE @table_name SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;

OPEN table_cursor;
FETCH NEXT FROM table_cursor INTO @table_name;

WHILE @@FETCH_STATUS = 0 BEGIN
    SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' NOCHECK CONSTRAINT ALL';
    EXEC (@cmd);
    FETCH NEXT FROM table_cursor INTO @table_name;
END

CLOSE table_cursor;
DEALLOCATE table_cursor;

GO
```

6. Drop indexes, if applicable.
7. Disable triggers, if applicable.

If you need to disable triggers, you can do so with the following script.

```
--Disable triggers on all tables
DECLARE @enable BIT = 0;
DECLARE @trigger SYSNAME;
DECLARE @table SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
    table_object.name table_name
FROM sysobjects trigger_object
JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
WHERE trigger_object.type = 'TR';

OPEN trigger_cursor;
FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

WHILE @@FETCH_STATUS = 0 BEGIN
    IF @enable = 1
        SET @cmd = 'ENABLE ';
    ELSE
        SET @cmd = 'DISABLE ';

    SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+'' ON
dbo.'+QUOTENAME(@table)+'' ;
    EXEC (@cmd);
    FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
END

CLOSE trigger_cursor;
DEALLOCATE trigger_cursor;

GO
```

8. Query the source SQL Server instance for any logins that you want to import to the destination DB instance.

SQL Server stores logins and passwords in the `master` database. Because Amazon RDS doesn't grant access to the `master` database, you cannot directly import logins and passwords into your destination DB instance. Instead, you must query the `master` database on the source SQL Server instance to generate a data definition language (DDL) file that includes all logins and passwords that you want to add to the destination DB instance, and also role memberships and permissions that you want to transfer.

For information about querying the master database, see [How to Transfer the Logins and the Passwords Between Instances of SQL Server 2005 and SQL Server 2008](#) in the Microsoft Knowledge Base.

The output of the script is another script that you can run on the destination DB instance. The script in the Knowledge Base article has the following code:

```
p.type IN
```

Every place p.type appears, use the following code instead:

```
p.type = 'S'
```

9. Import the data using the method in [Import the Data \(p. 545\)](#).
10. Grant applications access to the target DB instance.

When your data import is complete, you can grant access to the DB instance to those applications that you blocked during the import. For information about controlling access to your DB instance, see [Working with DB Security Groups \(EC2-Classic Platform\) \(p. 399\)](#).

11. Enable automated backups on the target DB instance.

For information about automated backups, see [Working With Backups \(p. 208\)](#).

12. Enable foreign key constraints.

If you disabled foreign key constraints earlier, you can now enable them with the following script.

```
--Enable foreign keys on all tables
DECLARE @table_name SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;

OPEN table_cursor;
FETCH NEXT FROM table_cursor INTO @table_name;

WHILE @@FETCH_STATUS = 0 BEGIN
    SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' CHECK CONSTRAINT ALL';
    EXEC (@cmd);
    FETCH NEXT FROM table_cursor INTO @table_name;
END

CLOSE table_cursor;
DEALLOCATE table_cursor;
```

13. Enable indexes, if applicable.
14. Enable triggers, if applicable.

If you disabled triggers earlier, you can now enable them with the following script.

```
--Enable triggers on all tables
DECLARE @enable BIT = 1;
DECLARE @trigger SYSNAME;
DECLARE @table SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
    table_object.name table_name
FROM sysobjects trigger_object
JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
WHERE trigger_object.type = 'TR';
```

```
OPEN trigger_cursor;
FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

WHILE @@FETCH_STATUS = 0 BEGIN
    IF @enable = 1
        SET @cmd = 'ENABLE ';
    ELSE
        SET @cmd = 'DISABLE ';

    SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+'' ON
dbo.'+QUOTENAME(@table)+'' ;
    EXEC (@cmd);
    FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
END

CLOSE trigger_cursor;
DEALLOCATE trigger_cursor;
```

Import the Data

Microsoft SQL Server Management Studio is a graphical SQL Server client that is included in all Microsoft SQL Server editions except the Express Edition. SQL Server Management Studio Express is available from Microsoft as a free download. To find this download, see [the Microsoft website](#).

Note

SQL Server Management Studio is available only as a Windows-based application.

SQL Server Management Studio includes the following tools, which are useful in importing data to a SQL Server DB instance:

- Generate and Publish Scripts Wizard
- Import and Export Wizard
- Bulk copy

Generate and Publish Scripts Wizard

The Generate and Publish Scripts Wizard creates a script that contains the schema of a database, the data itself, or both. If you generate a script for a database in your local SQL Server deployment, you can then run the script to transfer the information that it contains to an Amazon RDS DB instance.

Note

For databases of 1 GiB or larger, it is more efficient to script only the database schema and then use the Import and Export Wizard or the bulk copy feature of SQL Server to transfer the data.

For detailed information about the Generate and Publish Scripts Wizard, see the [Microsoft SQL Server documentation](#).

In the wizard, pay particular attention to the advanced options on the **Set Scripting Options** page to ensure that everything you want your script to include is selected. For example, by default, database triggers are not included in the script.

When the script is generated and saved, you can use SQL Server Management Studio to connect to your DB instance and then run the script.

Import and Export Wizard

The Import and Export Wizard creates a special Integration Services package, which you can use to copy data from your local SQL Server database to the destination DB instance. The wizard can filter which tables and even which tuples within a table are copied to the destination DB instance.

Note

The Import and Export Wizard works well for large datasets, but it might not be the fastest way to remotely export data from your local deployment. For an even faster way, consider the SQL Server bulk copy feature.

For detailed information about the Import and Export Wizard, see the [Microsoft SQL Server documentation](#).

In the wizard, on the **Choose a Destination** page, do the following:

- For **Server Name**, type the name of the endpoint for your DB instance.
- For the server authentication mode, choose **Use SQL Server Authentication**.
- For **User name** and **Password**, type the credentials for the master user that you created for the DB instance.

Bulk Copy

The SQL Server bulk copy feature is an efficient means of copying data from a source database to your DB instance. Bulk copy writes the data that you specify to a data file, such as an ASCII file. You can then run bulk copy again to write the contents of the file to the destination DB instance.

This section uses the **bcp** utility, which is included with all editions of SQL Server. For detailed information about bulk import and export operations, see [the Microsoft SQL Server documentation](#).

Note

Before you use bulk copy, you must first import your database schema to the destination DB instance. The Generate and Publish Scripts Wizard, described earlier in this topic, is an excellent tool for this purpose.

The following command connects to the local SQL Server instance to generate a tab-delimited file of a specified table in the C:\ root directory of your existing SQL Server deployment. The table is specified by its fully qualified name, and the text file has the same name as the table that is being copied.

```
bcp dbname.schema_name.table_name out C:\table_name.txt -n -S localhost -U username -P password -b 10000
```

The preceding code includes the following options:

- **-n** specifies that the bulk copy will use the native data types of the data to be copied.
- **-S** specifies the SQL Server instance that the *bcp* utility will connect to.
- **-U** specifies the user name of the account that will log in to the SQL Server instance.
- **-P** specifies the password for the user specified by **-U**.
- **-b** specifies the number of rows per batch of imported data.

Note

There might be other parameters that are important to your import situation. For example, you might need the **-E** parameter that pertains to identity values. For more information; see the full description of the command line syntax for the **bcp** utility in [the Microsoft SQL Server documentation](#).

For example, suppose a database named *store* that uses the default schema, *dbo*, contains a table named *customers*. The user account *admin*, with the password *insecure*, will copy 10,000 rows of the *customers* table to a file named *customers.txt*.

```
bcp store.dbo.customers out C:\customers.txt -n -S localhost -U admin -P insecure -b 10000
```

After you generate the data file, if you have created the database and schema on the target DB instance, you can upload the data to your DB instance by using a similar command. In this case, you will use the `in` argument to specify an input file instead of `out` to specify an output file. Instead of using `localhost` to specify the local SQL Server instance, you will specify the endpoint of your DB instance. If you use a port other than 1433, you will specify that, too. The user name and password are the master user and password for your DB instance. The syntax is as follows.

```
bcp dbname.schema_name.table_name in C:\table_name.txt -n -S endpoint,port -  
U master_user_name -P master_user_password -b 10000
```

To continue the previous example, suppose the master user name is `admin`, and the password is `insecure`. The endpoint for the DB instance is `rds.ckz2kqd4qsn1.us-east-1.rds.amazonaws.com`, and you use port 4080. The command is as follows.

```
bcp store.dbo.customers in C:\customers.txt -n -S rds.ckz2kqd4qsn1.us-east-1.rds.amazonaws.com,4080 -U admin -P insecure -b 10000
```

Exporting Data from SQL Server on Amazon RDS

You can choose one of the following options to export data from an Amazon RDS SQL DB instance :

- **Native database backup using a full backup file (.bak)** – Using .bak files to backup databases is heavily optimized, and is usually the fastest way to export data. For more information, see [Importing and Exporting SQL Server Databases \(p. 533\)](#).
- **SQL Server Import and Export Wizard** – For more information, see [SQL Server Import and Export Wizard \(p. 547\)](#).
- **SQL Server Generate and Publish Scripts Wizard and bcp utility** – For more information, see [SQL Server Generate and Publish Scripts Wizard and bcp Utility \(p. 548\)](#).

SQL Server Import and Export Wizard

You can use the SQL Server Import and Export Wizard to copy one or more tables, views, or queries from your Amazon RDS SQL DB instance to another data store. This choice is best if the target data store is not SQL Server. For more information, see [SQL Server Import and Export Wizard](#) in the SQL Server documentation.

The SQL Server Import and Export Wizard is available as part of Microsoft SQL Server Management Studio, a graphical SQL Server client that is included in all Microsoft SQL Server editions except the Express Edition. SQL Server Management Studio is available only as a Windows-based application. SQL Server Management Studio Express is available from Microsoft as a free download. To find this download, see [the Microsoft website](#).

To use the SQL Server Import and Export Wizard to export data

1. In SQL Server Management Studio, connect to your Amazon RDS SQL DB instance. For details on how to do this, see [Connecting to a DB Instance Running the Microsoft SQL Server Database Engine \(p. 514\)](#).
2. In **Object Explorer**, expand **Databases**, open the context (right-click) menu for the source database, choose **Tasks**, and then choose **Export Data**. The wizard appears.
3. On the **Choose a Data Source** page, do the following:
 - a. For **Data source**, choose **SQL Server Native Client 11.0**.
 - b. Verify that the **Server name** box shows the endpoint of your Amazon RDS SQL DB instance.

- c. Select **Use SQL Server Authentication**. For **User name** and **Password**, type the master user name and password of your Amazon RDS SQL DB.
 - d. Verify that the **Database** box shows the database from which you want to export data.
 - e. Choose **Next**.
4. On the **Choose a Destination** page, do the following:
 - a. For **Destination**, choose **SQL Server Native Client 11.0**.
- Note**
Other target data sources are available, include .NET Framework data providers, OLE DB providers, SQL Server Native Client providers, ADO.NET providers, Microsoft Office Excel, Microsoft Office Access, and the Flat File source. If you choose to target one of these data sources, skip the remainder of step 4 and see [Choose a Destination](#) in the SQL Server documentation for details on the connection information to provide.
- b. For **Server name**, type the server name of the target SQL Server DB instance.
 - c. Choose the appropriate authentication type. Type a user name and password if necessary.
 - d. For **Database**, choose the name of the target database, or choose **New** to create a new database to contain the exported data.

If you choose **New**, see [Create Database](#) in the SQL Server documentation for details on the database information to provide.
- e. Choose **Next**.
5. On the **Table Copy or Query** page, choose **Copy data from one or more tables or views** or **Write a query to specify the data to transfer**. Choose **Next**.
 6. If you chose **Write a query to specify the data to transfer**, you see the **Provide a Source Query** page. Type or paste in a SQL query, and then choose **Parse** to verify it. Once the query validates, choose **Next**.
 7. On the **Select Source Tables and Views** page, do the following:
 - a. Select the tables and views that you want to export, or verify that the query you provided is selected.
 - b. Choose **Edit Mappings** and specify database and column mapping information. For more information, see [Column Mappings](#) in the SQL Server documentation.
 - c. (Optional) To see a preview of data to be exported, select the table, view, or query, and then choose **Preview**.
 - d. Choose **Next**.
 8. On the **Run Package** page, verify that **Run immediately** is selected. Choose **Next**.
 9. On the **Complete the Wizard** page, verify that the data export details are as you expect. Choose **Finish**.
 10. On the **The execution was successful** page, choose **Close**.

SQL Server Generate and Publish Scripts Wizard and bcp Utility

You can use the SQL Server Generate and Publish Scripts Wizard to create scripts for an entire database or just selected objects. You can run these scripts on a target SQL Server DB instance to recreate the scripted objects. You can then use the bcp utility to bulk export the data for the selected objects to the target DB instance. This choice is best if you want to move a whole database (including objects other than tables) or large quantities of data between two SQL Server DB instances. For a full description of the bcp command line syntax, see [bcp Utility](#) in the Microsoft SQL Server documentation.

The SQL Server Generate and Publish Scripts Wizard is available as part of Microsoft SQL Server Management Studio, a graphical SQL Server client that is included in all Microsoft SQL Server editions

API Version 2014-10-31

except the Express Edition. SQL Server Management Studio is available only as a Windows-based application. SQL Server Management Studio Express is available from Microsoft as a [free download](#).

To use the SQL Server Generate and Publish Scripts Wizard and the bcp utility to export data

1. In SQL Server Management Studio, connect to your Amazon RDS SQL DB instance. For details on how to do this, see [Connecting to a DB Instance Running the Microsoft SQL Server Database Engine \(p. 514\)](#).
2. In **Object Explorer**, expand the **Databases** node and select the database you want to script.
3. Follow the instructions in [Generate and Publish Scripts Wizard](#) in the SQL Server documentation to create a script file.
4. In SQL Server Management Studio, connect to your target SQL Server DB instance.
5. With the target SQL Server DB instance selected in **Object Explorer**, choose **Open** on the **File** menu, choose **File**, and then open the script file.
6. If you have scripted the entire database, review the CREATE DATABASE statement in the script to make sure the database is being created in the location and with the parameters that you want. For more information, see [CREATE DATABASE](#) in the SQL Server documentation.
7. If you are creating database users in the script, check to see if server logins exist on the target DB instance for those users. If not, create logins for those users; the scripted commands to create the database users will fail otherwise. For more information, see [Create a Login](#) in the SQL Server documentation.
8. Choose **!Execute** on the SQL Editor menu to execute the script file and create the database objects. When the script finishes, verify that all database objects exist as expected.
9. Use the bcp utility to export data from the Amazon RDS SQL DB instance into files. Open a command prompt and type the following command.

```
bcp database_name.schema_name.table_name out data_file -n -S aws_rds_sql_endpoint -U
username -P password
```

The preceding code includes the following options:

- *table_name* is the name of one of the tables that you've recreated in the target database and now want to populate with data.
- *data_file* is the full path and name of the data file to be created.
- *-n* specifies that the bulk copy will use the native data types of the data to be copied.
- *-S* specifies the SQL Server DB instance to export from.
- *-U* specifies the user name to use when connecting to the SQL Server DB instance.
- *-P* specifies the password for the user specified by *-U*.

The following shows an example command.

```
bcp world.dbo.city out C:\Users\JohnDoe\city.dat -n -S sql-jdoe.1234abcd.us-
west-2.rds.amazonaws.com,1433 -U JohnDoe -P ClearTextPassword
```

Repeat this step until you have data files for all of the tables you want to export.

10. Prepare your target DB instance for bulk import of data by following the instructions at [Basic Guidelines for Bulk Importing Data](#) in the SQL Server documentation.
11. Decide on a bulk import method to use after considering performance and other concerns discussed in [About Bulk Import and Bulk Export Operations](#) in the SQL Server documentation.
12. Bulk import the data from the data files you created using the bcp utility, following the instructions at either [Import and Export Bulk Data by Using the bcp Utility](#) or [Import Bulk Data by Using BULK](#)

[INSERT or OPENROWSET\(BULK...\)](#) in the SQL Server documentation, depending on what you decided in step 11.

Related Topics

- [Importing and Exporting SQL Server Databases \(p. 533\)](#)

Multi-AZ Deployments for Microsoft SQL Server

Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. In the event of planned database maintenance or unplanned service disruption, Amazon RDS automatically fails over to the up-to-date secondary DB instance. This functionality lets database operations resume quickly without manual intervention. The primary and standby instances use the same endpoint, whose physical network address transitions to the secondary replica as part of the failover process. You don't have to reconfigure your application when a failover occurs.

Amazon RDS supports Multi-AZ deployments for Microsoft SQL Server by using either SQL Server Database Mirroring or Always On availability groups. Amazon RDS monitors and maintains the health of your Multi-AZ deployment. If problems occur, RDS automatically repairs unhealthy DB instances, reestablishes synchronization, and initiates failovers. Failover only occurs if the standby and primary are fully in sync. You don't have to manage anything.

When you set up SQL Server Multi-AZ, RDS automatically configures all databases on the instance to use Mirroring or Always On. Amazon RDS handles the primary, the witness, and the secondary DB instance for you.

Because configuration is automatic, RDS selects Mirroring or Always On based on the version of SQL Server that you deploy. Amazon RDS supports Multi-AZ with Mirroring or Always On for the following SQL Server versions and editions, with the noted exceptions:

- SQL Server 2017: Standard and Enterprise Editions (Always On not yet supported)
- SQL Server 2016: Standard and Enterprise Editions (Always On only in 13.0.5216.0 or later)
- SQL Server 2014: Standard and Enterprise Editions
- SQL Server 2012: Standard and Enterprise Editions
- SQL Server 2008 R2: Standard and Enterprise Editions

Amazon RDS supports Multi-AZ for SQL Server in all AWS Regions, with the following exceptions:

- US West (N. California): Neither Mirroring or Always On are supported here
- Asia Pacific (Sydney): Supported for [DB instances in VPCs](#)
- Asia Pacific (Tokyo): Supported for [DB instances in VPCs](#)
- South America (São Paulo): Supported on all [DB instance classes](#) except m1 or m2

Adding Multi-AZ to a Microsoft SQL Server DB Instance

When you create a new SQL Server DB instance using the AWS Management Console, you can add Multi-AZ with Mirroring or Always On. You do so by choosing **Yes (Mirroring / Always On)** from the **Multi-AZ Deployment** list on the **Specify DB Details** page. For more information, see [Creating a DB Instance Running the Microsoft SQL Server Database Engine \(p. 504\)](#).

When you modify an existing SQL Server DB instance using the AWS Management Console, you can add Multi-AZ with Mirroring or Always On by choosing **Yes (Mirroring / Always On)** from the **Multi-AZ Deployment** list on the **Modify DB Instance** page. For more information, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).

Microsoft SQL Server Multi-AZ Deployment Notes and Recommendations

The following are some restrictions when working with Multi-AZ deployments for Microsoft SQL Server DB instances:

- Cross-region Multi-AZ is not currently supported.
- You can't configure the secondary to accept database read activity.
- Multi-AZ with Always On supports in-memory optimization.
- You can't rename a database on a SQL Server DB instance that is in a SQL Server Multi-AZ deployment. If you need to rename a database on such an instance, first turn off Multi-AZ for the DB instance, then rename the database. Finally, turn Multi-AZ back on for the DB instance.
- You can only restore Multi-AZ DB instances that are backed up using the full recovery model.

The following are some notes about working with Multi-AZ deployments for Microsoft SQL Server DB instances:

- Amazon RDS exposes the Always On [availability group listener endpoint](#). The endpoint is visible in the console, and is returned by the `DescribeDBInstances` API as an entry in the endpoints field.
- Amazon RDS supports [availability group multisubnet failovers](#).
- To use SQL Server Multi-AZ with a SQL Server DB instance in a VPC, first create a DB subnet group that has subnets in at least two distinct Availability Zones. Then assign the DB subnet group to the primary replica of the SQL Server DB instance.
- When a DB instance is modified to be a Multi-AZ deployment, during the modification it has a status of **modifying**. Amazon RDS creates the standby, and makes a backup of the primary DB instance. After the process is complete, the status of the primary DB instance becomes **available**.
- Multi-AZ deployments maintain all databases on the same node. If a database on the primary host fails over, all your SQL Server databases fail over as one atomic unit to your standby host. Amazon RDS provisions a new healthy host, and replaces the unhealthy host.
- Multi-AZ with Mirroring or Always On supports a single standby replica.
- Users, logins, and permissions are automatically replicated for you on the secondary. You don't need to recreate them. User-defined server roles (a SQL Server 2012 feature) are only replicated in Multi-AZ instances for Always On instances.
- If you have SQL Server Agent jobs, recreate them on the secondary. You do so because these jobs are stored in the msdb database, and you can't replicate this database by using Mirroring or Always On. Create the jobs first in the original primary, then fail over, and create the same jobs in the new primary.
- You might observe elevated latencies compared to a standard DB instance deployment (in a single Availability Zone) because of the synchronous data replication.
- Failover times are affected by the time it takes to complete the recovery process. Large transactions increase the failover time.

The following are some recommendations for working with Multi-AZ deployments for Microsoft SQL Server DB instances:

- For databases used in production or preproduction, we recommend Multi-AZ deployments for high availability, Provisioned IOPS for fast, consistent performance, and instance classes (m3.large and larger, m4.large and larger) that are optimized for Provisioned IOPS.
- You can't select the Availability Zone (AZ) for the secondary instance, so when you deploy application hosts, take this into account. Your database might fail over to another AZ, and the application hosts might not be in the same AZ as the database. For this reason, we recommend that you balance your application hosts across all AZs in the given AWS Region.

- For best performance, don't enable Mirroring or Always On during a large data load operation. If you want your data load to be as fast as possible, finish loading data before you convert your DB instance to a Multi-AZ deployment.
- Applications that access the SQL Server databases should have exception handling that catches connection errors. The following code sample shows a try/catch block that catches a communication error.

```

for (int iRetryCount = 0; (iRetryCount < RetryMaxAttempts && keepInserting); iRetryCount++)
{
    using (SqlConnection connection = new SqlConnection(DatabaseConnectionString))
    {
        using (SqlCommand command = connection.CreateCommand())
        {
            command.CommandText = "INSERT INTO SOME_TABLE VALUES ('SomeValue');";

            try
            {
                connection.Open();

                while (keepInserting)
                {
                    command.ExecuteNonQuery();
                    intervalCount++;
                }
                connection.Close();
            }

            catch (Exception ex)
            {
                Logger(ex.Message);
            }
        }
    }

    if (iRetryCount < RetryMaxAttempts && keepInserting)
    {
        Thread.Sleep(RetryIntervalPeriodInSeconds * 1000);
    }
}

```

- Don't use the `Set Partner Off` command when working with Multi-AZ instances. For example, don't do the following.

```
--Don't do this
ALTER DATABASE db1 SET PARTNER off
```

- Don't set the recovery mode to `simple`. For example, don't do the following.

```
--Don't do this
ALTER DATABASE db1 SET RECOVERY simple
```

- Don't use the `DEFAULT_DATABASE` parameter when creating new logins on Multi-AZ DB instances, because these settings can't be applied to the standby mirror. For example, don't do the following.

```
--Don't do this
CREATE LOGIN [test_dba] WITH PASSWORD=foo, DEFAULT_DATABASE=[db2]
```

Also, don't do the following.

```
--Don't do this
ALTER LOGIN [test_dba] SET DEFAULT_DATABASE=[db3]
```

Determining the Location of the Secondary

You can determine the location of the secondary replica by using the AWS Management Console. You need to know the location of the secondary if you are setting up your primary DB instance in a VPC.

Endpoint: [sg-sqlsvr08r2.c6c8mntzhgv0.us-west-2.rds.amazonaws.com:8200](#) (authoritative)

Configuration Details		Security and Network	
DB Name:		Availability Zone:	us-west-2
Engine:	sqlserver-se(10.50.2789.0.v1)	VPC ID:	
Username:	sgawsuser	Subnet Group:	
Option Group(s):	default:sqlserver-se-10-50 (in-sync)	Subnets:	None
Parameter Group:	default.sqlserver-se-10.5 (in-sync)	Security Groups:	sg-db-secondary
Availability and Durability		Maintenance Details	
DB Instance Status:	available	Auto Minor Version Upgrade:	Enabled
Replication State:	-	Maintenance Window:	2014-05-19T00:00:00Z - 2014-05-19T01:00:00Z
Replication Error:	-	Backup Window:	2014-05-19T00:00:00Z - 2014-05-19T01:00:00Z
Multi AZ:	Yes		
Secondary Zone:	us-west-2c		
Automated Backups:	Enabled (1 Day)		
Latest Restore Time:	May 19, 2014 7:15:01 AM UTC-7		

You can also view the Availability Zone of the secondary using the AWS CLI command `describe-db-instances` or RDS API action `DescribeDBInstances`. The output shows the secondary AZ where the standby mirror is located.

Migrating from Mirroring to Always On

To migrate from Mirroring to Always On, first check your version. If you are using a DB instance with a version prior to 13.00.5216.0, modify the instance to patch it to 13.00.5216.0.

To upgrade to Always On, modify the instance to remove Multi-AZ, and then modify it again to add Multi-AZ. This converts your instance to use AlwaysOn.

Using SSL with a Microsoft SQL Server DB Instance

You can use Secure Sockets Layer (SSL) to encrypt connections between your client applications and your Amazon RDS DB instances running Microsoft SQL Server. SSL support is available in all AWS regions for all supported SQL Server editions.

When you create a SQL Server DB instance, Amazon RDS creates an SSL certificate for it. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks.

There are 2 ways to use SSL to connect to your SQL Server DB instance:

- Force SSL for all connections — this happens transparently to the client, and the client doesn't have to do any work to use SSL.
- Encrypt specific connections — this sets up an SSL connection from a specific client computer, and you must do work on the client to encrypt connections.

For information about Transport Layer Security (TLS) support for SQL Server, see [TLS 1.2 support for Microsoft SQL Server](#).

Forcing Connections to Your DB Instance to Use SSL

You can force all connections to your DB instance to use SSL. If you force connections to use SSL, it happens transparently to the client, and the client doesn't have to do any work to use SSL.

If you want to force SSL, use the `rds.force_ssl` parameter. By default, the `rds.force_ssl` parameter is set to `false`. Set the `rds.force_ssl` parameter to `true` to force connections to use SSL. The `rds.force_ssl` parameter is static, so after you change the value, you must reboot your DB instance for the change to take effect.

To force all connections to your DB instance to use SSL

1. Determine the parameter group that is attached to your DB instance:
 - a. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
 - b. In the top right corner of the Amazon RDS console, choose the AWS Region of your DB instance.
 - c. In the navigation pane, choose **Databases**, and then choose the name of your DB instance to show its details.
 - d. Choose the **Configuration** tab. Find the **Parameter group** in the section.
2. If necessary, create a new parameter group. If your DB instance uses the default parameter group, you must create a new parameter group. If your DB instance uses a nondefault parameter group, you can choose to edit the existing parameter group or to create a new parameter group. If you edit an existing parameter group, the change affects all DB instances that use that parameter group.

To create a new parameter group, follow the instructions in [Creating a DB Parameter Group \(p. 170\)](#).

3. Edit your new or existing parameter group to set the `rds.force_ssl` parameter to `true`. To edit the parameter group, follow the instructions in [Modifying Parameters in a DB Parameter Group \(p. 171\)](#).
4. If you created a new parameter group, modify your DB instance to attach the new parameter group. Modify the **DB Parameter Group** setting of the DB instance. For more information, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).
5. Reboot your DB instance. For more information, see [Rebooting a DB Instance \(p. 129\)](#).

Encrypting Specific Connections

You can force all connections to your DB instance to use SSL, or you can encrypt connections from specific client computers only. To use SSL from a specific client, you must obtain certificates for the client computer, import certificates on the client computer, and then encrypt the connections from the client computer.

Note

All SQL Server instances created after August 5, 2014, use the DB instance endpoint in the Common Name (CN) field of the SSL certificate. Prior to August 5, 2014, SSL certificate verification was not available for VPC-based SQL Server instances. If you have a VPC-based SQL Server DB instance that was created before August 5, 2014, and you want to use SSL certificate verification and ensure that the instance endpoint is included as the CN for the SSL certificate for that DB instance, then rename the instance. When you rename a DB instance, a new certificate is deployed and the instance is rebooted to enable the new certificate.

Obtaining Certificates for Client Computers

To encrypt connections from a client computer to an Amazon RDS DB instance running Microsoft SQL Server, you need a certificate on your client computer.

To obtain that certificate, download the certificate to your client computer. You can download a root certificate that works for all regions from <https://s3.amazonaws.com/rds-downloads/rds-ca-2015-root.pem>. You can download a certificate bundle that contains both the old and new root certificates from <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>. For region-specific intermediate certificates, and more information, see [Using SSL to Encrypt a Connection to a DB Instance \(p. 392\)](#).

After you have downloaded the appropriate certificate, import the certificate into your Microsoft Windows operating system by following the procedure in the section following.

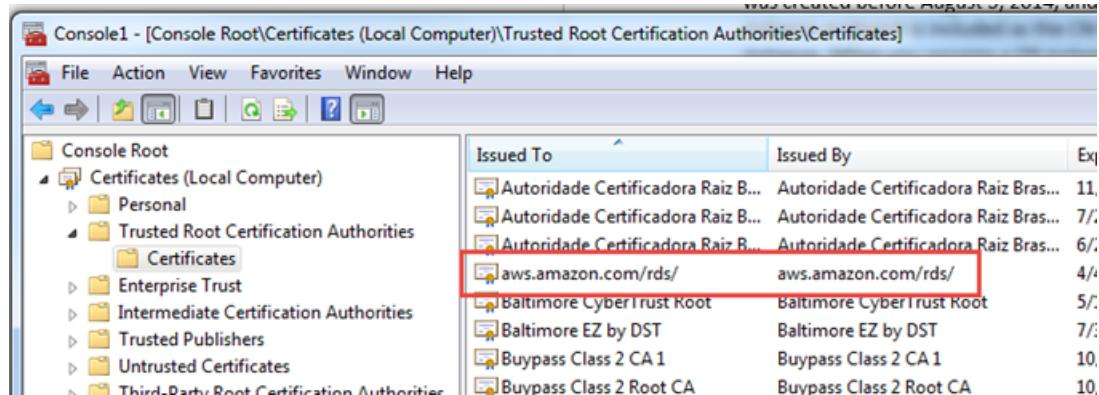
Importing Certificates on Client Computers

You can use the following procedure to import your certificate into the Microsoft Windows operating system on your client computer.

To import the certificate into your Windows operating system:

1. On the **Start** menu, type **Run** in the search box and press **Enter**.
2. In the **Open** box, type **MMC** and then choose **OK**.
3. In the MMC console, on the **File** menu, choose **Add/Remove Snap-in**.
4. In the **Add or Remove Snap-ins** dialog box, for **Available snap-ins**, select **Certificates**, and then choose **Add**.
5. In the MMC console, on the **File** menu, choose **Add/Remove Snap-in**.
6. In the **Certificates snap-in** dialog box, choose **Computer account**, and then choose **Next**.
7. In the **Select computer** dialog box, choose **Finish**.
8. In the **Add or Remove Snap-ins** dialog box, choose **OK**.
9. In the MMC console, expand **Certificates**, open the context (right-click) menu for **Trusted Root Certification Authorities**, choose **All Tasks**, and then choose **Import**.
10. On the first page of the Certificate Import Wizard, choose **Next**.
11. On the second page of the Certificate Import Wizard, choose **Browse**. In the browse window, change the file type to **All files (*.*)** because .pem is not a standard certificate extension. Locate the .pem file that you downloaded previously.
12. Choose **Open** to select the certificate file, and then choose **Next**.

13. On the third page of the Certificate Import Wizard, choose **Next**.
14. On the fourth page of the Certificate Import Wizard, choose **Finish**. A dialog box appears indicating that the import was successful.
15. In the MMC console, expand **Certificates**, expand **Trusted Root Certification Authorities**, and then choose **Certificates**. Locate the certificate to confirm it exists, as shown following.



16. Restart your computer.

Encrypting Connections to an Amazon RDS DB Instance Running Microsoft SQL Server

After you have imported a certificate into your client computer, you can encrypt connections from the client computer to an Amazon RDS DB instance running Microsoft SQL Server.

For SQL Server Management Studio, use the following procedure. For more information about SQL Server Management Studio, see [Use SQL Server Management Studio](#).

To encrypt connections from SQL Server Management Studio

1. Launch SQL Server Management Studio.
2. For **Connect to server**, type the server information, login user name, and password.
3. Choose **Options**.
4. Select **Encrypt connection**.
5. Choose **Connect**.
6. Confirm that your connection is encrypted by running the following query. Verify that the query returns `true` for `encrypt_option`.

```
select ENCRYPT_OPTION from SYS.DM_EXEC_CONNECTIONS where SESSION_ID = @@SPID
```

For any other SQL client, use the following procedure.

To encrypt connections from other SQL clients

1. Append `encrypt=true` to your connection string. This string might be available as an option, or as a property on the connection page in GUI tools.

Note

To enable SSL encryption for clients that connect using JDBC, you might need to add the Amazon RDS SQL certificate to the Java CA certificate (cacerts) store. You can do this by using the [keytool](#) utility.

2. Confirm that your connection is encrypted by running the following query. Verify that the query returns true for encrypt_option.

```
select ENCRYPT_OPTION from SYS.DM_EXEC_CONNECTIONS where SESSION_ID = @@SPID
```

Options for the Microsoft SQL Server Database Engine

This section describes options, or additional features, that are available for Amazon RDS instances running the Microsoft SQL Server DB engine. To enable these options, you add them to an option group, and then associate the option group with your DB instance. For more information, see [Working with Option Groups \(p. 156\)](#).

Amazon RDS supports the following options for Microsoft SQL Server DB instances.

Option	Option ID	Engine Editions
Native Backup and Restore (p. 559)	SQLSERVER_BACKUP_RESTORE	SQL Server Enterprise Edition SQL Server Standard Edition SQL Server Web Edition SQL Server Express Edition
Transparent Data Encryption (p. 561)	TRANSPARENT_DATA_ENCRYPTION	SQL Server Enterprise Edition

Microsoft SQL Server Native Backup and Restore Support

Amazon RDS supports native backup and restore for Microsoft SQL Server databases using full backup files (.bak files). You can import and export SQL Server databases in a single, easily portable file. You can create a full backup of your on-premises database, store it on Amazon Simple Storage Service (Amazon S3), and then restore the backup file onto an existing Amazon RDS DB instance running SQL Server. You can back up an Amazon RDS SQL Server database, store it on Amazon S3, and then restore the backup file onto an on-premises server, or a different Amazon RDS DB instance running SQL Server. For more information, see [Importing and Exporting SQL Server Databases \(p. 533\)](#).

Native Backup and Restore Option Settings

Amazon RDS supports the following settings for the Native Backup and Restore option.

Option Setting	Valid Values	Description
IAM_ROLE_ARN	A valid Amazon Resource Name (ARN) in the format <code>arn:aws:iam::account-id:role/role-name</code> .	The ARN for an AWS Identity and Access Management (IAM) role to access the Amazon S3 bucket that contains your backup files. For more information, see AWS Identity and Access Management (IAM) .

Adding the Native Backup and Restore Option

The general process for adding the Native Backup and Restore option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the Native Backup and Restore option, you don't need to restart your DB instance. As soon as the option group is active, you can begin backing up and restoring immediately.

To add the Native Backup and Restore option

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group. For more information, see [Creating an Option Group \(p. 157\)](#).
2. Add the **SQLSERVER_BACKUP_RESTORE** option to the option group, and configure the option settings. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).
 - a. For **IAM Role**, select an existing IAM role. Alternatively, you can choose to have a new IAM role created for you by choosing [Create a New Role](#).
 - b. For **Select S3 Bucket**, select an existing bucket. Alternatively, you can choose to have a new Amazon S3 bucket created for you by choosing [Create a New S3 Bucket](#).
 - c. For **Enable Encryption**, choose **Yes** to encrypt the backup file. If you choose **Yes**, for **Master Key** you must also choose an encryption key. For more information about encryption keys, see [Getting Started](#) in the AWS Key Management Service (AWS KMS) documentation.
3. Apply the option group to a new or existing DB instance.
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Microsoft SQL Server Database Engine \(p. 504\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).

Modifying Native Backup and Restore Option Settings

After you enable the Native Backup and Restore option, you can modify the settings for the option. For more information about how to modify option settings, see [Modifying an Option Setting \(p. 164\)](#). For more information about each setting, see [Native Backup and Restore Option Settings \(p. 559\)](#).

Removing the Native Backup and Restore Option

You can turn off the native backup and restore feature by removing the option from your DB instance. After you remove the Native Backup and Restore option, you don't need to restart your DB instance.

To remove the Native Backup and Restore option from a DB instance, do one of the following:

- Remove the Native Backup and Restore option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#)
- Modify the DB instance and specify a different option group that doesn't include the Native Backup and Restore option. This change affects a single DB instance. You can specify the default (empty)

option group, or a different custom option group. For more information, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).

Microsoft SQL Server Transparent Data Encryption Support

Amazon RDS supports using Transparent Data Encryption (TDE) to encrypt stored data on your DB instances running Microsoft SQL Server. TDE automatically encrypts data before it is written to storage, and automatically decrypts data when the data is read from storage.

Amazon RDS supports TDE for the following SQL Server versions and editions:

- SQL Server 2017 Enterprise Edition
- SQL Server 2016 Enterprise Edition
- SQL Server 2014 Enterprise Edition
- SQL Server 2012 Enterprise Edition
- SQL Server 2008 R2 Enterprise Edition

To enable transparent data encryption for a DB instance that is running SQL Server, specify the **TDE** option in an Amazon RDS option group that is associated with that DB instance.

Transparent data encryption for SQL Server provides encryption key management by using a two-tier key architecture. A certificate, which is generated from the database master key, is used to protect the data encryption keys. The database encryption key performs the actual encryption and decryption of data on the user database. Amazon RDS backs up and manages the database master key and the TDE certificate. To comply with several security standards, Amazon RDS is working to implement automatic periodic master key rotation.

Transparent data encryption is used in scenarios where you need to encrypt sensitive data in case data files and backups are obtained by a third party or when you need to address security-related regulatory compliance issues. Note that you cannot encrypt the system databases for SQL Server, such as the Model or Master databases.

A detailed discussion of transparent data encryption is beyond the scope of this guide, but you should understand the security strengths and weaknesses of each encryption algorithm and key. For information about transparent data encryption for SQL Server, see [Transparent Data Encryption \(TDE\)](#) on the Microsoft website.

You should determine if your DB instance is already associated with an option group that has the **TDE** option. To view the option group that a DB instance is associated with, you can use the RDS console, the `describe-db-instance` AWS CLI command, or the API action [DescribeDBInstances](#).

The process for enabling transparent data encryption on a SQL Server DB instance is as follows:

1. If the DB instance is not associated with an option group that has the **TDE** option enabled, you must either create an option group and add the **TDE** option or modify the associated option group to add the **TDE** option. For information about creating or modifying an option group, see [Working with Option Groups \(p. 156\)](#). For information about adding an option to an option group, see [Adding an Option to an Option Group \(p. 160\)](#).
2. Associate the DB instance with the option group with the **TDE** option. For information about associating a DB instance with an option group, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).

When the **TDE** option is added to an option group, Amazon RDS generates a certificate that is used in the encryption process. You can then use the certificate to run SQL statements that will encrypt data in a database on the DB instance. The following example uses the RDS-created certificate called RDSTDECertificateName to encrypt a database called customerDatabase.

```
----- Enabling TDE -----  
  
-- Find a RDSTDECertificate to use  
USE [master]  
GO  
SELECT name FROM sys.certificates WHERE name LIKE 'RDSTDECertificate%'  
GO  
  
USE [customerDatabase]  
GO  
-- Create DEK using one of the certificates from the previous step  
CREATE DATABASE ENCRYPTION KEY  
WITH ALGORITHM = AES_128  
ENCRYPTION BY SERVER CERTIFICATE [RDSTDECertificateName]  
GO  
  
-- Enable encryption on the database  
ALTER DATABASE [customerDatabase]  
SET ENCRYPTION ON  
GO  
  
-- Verify that the database is encrypted  
USE [master]  
GO  
SELECT name FROM sys.databases WHERE is_encrypted = 1  
GO  
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryption_keys  
GO
```

The time it takes to encrypt a SQL Server database using TDE depends on several factors, including the size of the DB instance, whether PIOPS is enabled for the instance, the amount of data, and other factors.

The **TDE** option is a persistent option than cannot be removed from an option group unless all DB instances and backups are disassociated from the option group. Once you add the **TDE** option to an option group, the option group can only be associated with DB instances that use TDE. For more information about persistent options in an option group, see [Option Groups Overview \(p. 156\)](#).

Because the **TDE** option is a persistent option, you can have a conflict between the option group and an associated DB instance. You can have a conflict between the option group and an associated DB instance in the following situations:

- The current option group has the **TDE** option, and you replace it with an option group that does not have the **TDE** option.
- You restore from a DB snapshot to a new DB instance that does not have an option group that contains the TDE option. For more information about this scenario, see [Option Group Considerations \(p. 222\)](#).

To disable TDE for a DB instance, first ensure that there are no encrypted objects left on the DB instance by either unencrypting the objects or by dropping them. If any encrypted objects exist on the DB instance, you will not be allowed to disable TDE for the DB instance. When you use the AWS Management Console to remove the **TDE** option from an option group, the console indicates that it is processing, and an event is created indicating an error if the option group is associated with an encrypted DB instance or DB snapshot.

The following example removes the TDE encryption from a database called customerDatabase.

```
----- Removing TDE -----  
  
USE [customerDatabase]  
GO  
  
-- Disable encryption on the database  
ALTER DATABASE [customerDatabase]  
SET ENCRYPTION OFF  
GO  
  
-- Wait until the encryption state of the database becomes 1. The state is 5 (Decryption in  
progress) for a while  
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryption_keys  
GO  
  
-- Drop the DEK used for encryption  
DROP DATABASE ENCRYPTION KEY  
GO  
  
-- Alter to SIMPLE Recovery mode so that your encrypted log gets truncated  
USE [master]  
GO  
ALTER DATABASE [customerDatabase] SET RECOVERY SIMPLE  
GO
```

When all objects are unencrypted, you can modify the DB instance to be associated with an option group without the **TDE** option or you can remove the **TDE** option from the option group.

Performance Considerations

The performance of a SQL Server DB instance can be impacted by using transparent data encryption.

Performance for unencrypted databases can also be degraded if the databases are on a DB instance that has at least one encrypted database. As a result, we recommend that you keep encrypted and unencrypted databases on separate DB instances.

Because of the nature of encryption, the database size and the size of the transaction log is larger than for an unencrypted database. You could run over your allocation of free backup space. The nature of TDE will cause an unavoidable performance hit. If you need high performance and TDE, measure the impact and make sure it meets your needs. There is less of an impact on performance if you use Provisioned IOPS and at least an M3.Large DB instance class.

Common DBA Tasks for Microsoft SQL Server

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances that are running the Microsoft SQL Server database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.

Note

When working with a SQL Server DB instance, you can run scripts to modify a newly created database, but you cannot modify the [model] database, the database used as the model for new databases.

Topics

- [Accessing the tempdb Database on Microsoft SQL Server DB Instances on Amazon RDS \(p. 565\)](#)
- [Analyzing Your Database Workload on an Amazon RDS DB Instance with SQL Server Tuning Advisor \(p. 567\)](#)
- [Collations and Character Sets for Microsoft SQL Server \(p. 569\)](#)
- [Determining a Recovery Model for Your Microsoft SQL Server Database \(p. 570\)](#)
- [Dropping a Microsoft SQL Server Database That Is Multi-AZ \(p. 570\)](#)
- [Using Change Data Capture \(p. 570\)](#)
- [Renaming a Microsoft SQL Server Database in a Multi-AZ Deployment \(p. 572\)](#)
- [Resetting the db_owner Role Password \(p. 573\)](#)
- [Restoring License-Terminated DB Instances \(p. 573\)](#)
- [Transitioning a Microsoft SQL Server Database from OFFLINE to ONLINE \(p. 574\)](#)
- [Using SQL Server Agent \(p. 574\)](#)
- [Working with Microsoft SQL Server Logs \(p. 575\)](#)
- [Working with Trace and Dump Files \(p. 576\)](#)

Accessing the tempdb Database on Microsoft SQL Server DB Instances on Amazon RDS

You can access the tempdb database on your Microsoft SQL Server DB instances on Amazon RDS. You can run code on tempdb by using Transact-SQL through Microsoft SQL Server Management Studio (SSMS), or any other standard SQL client application. For more information about connecting to your DB instance, see [Connecting to a DB Instance Running the Microsoft SQL Server Database Engine \(p. 514\)](#).

The master user for your DB instance is granted CONTROL access to tempdb so that this user can modify the tempdb database options. The master user isn't the database owner of the tempdb database. If necessary, the master user can grant CONTROL access to other users so that they can also modify the tempdb database options.

Note

You can't run Database Console Commands (DBCC) on the tempdb database.

Modifying tempdb Database Options

You can modify the database options on the tempdb database on your Amazon RDS DB instances. For more information about which options can be modified, see [tempdb Database](#) in the Microsoft documentation.

Database options such as the maximum file size options are persistent after you restart your DB instance. You can modify the database options to optimize performance when importing data, and to prevent running out of storage.

Optimizing Performance when Importing Data

To optimize performance when importing large amounts of data into your DB instance, set the SIZE and FILEGROWTH properties of the tempdb database to large numbers. For more information about how to optimize tempdb, see [Optimizing tempdb Performance](#) in the Microsoft documentation.

The following example demonstrates setting the size to 100 GB and file growth to 10 percent.

```
alter database[tempdb] modify file (NAME = N'templog', SIZE=100GB, FILEGROWTH = 10%)
```

Preventing Storage Problems

To prevent the tempdb database from using all available disk space, set the MAXSIZE property. The following example demonstrates setting the property to 2048 MB.

```
alter database [tempdb] modify file (NAME = N'templog', MAXSIZE = 2048MB)
```

Shrinking the tempdb Database

There are two ways to shrink the tempdb database on your Amazon RDS DB instance. You can use the `rds_shrink_tempdbfile` procedure, or you can set the SIZE property,

Using the `rds_shrink_tempdbfile` Procedure

You can use the Amazon RDS procedure `msdb.dbo.rds_shrink_tempdbfile` to shrink the tempdb database. You can only call `rds_shrink_tempdbfile` if you have CONTROL access to tempdb. When you call `rds_shrink_tempdbfile`, there is no downtime for your DB instance.

The `rds_shrink_tempdbfile` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>@temp_filename</code>	SYSNAME	—	required	The logical name of the file to shrink.
<code>@target_size</code>	int	null	optional	The new size for the file, in megabytes.

The following example gets the names of the files for the tempdb database.

```
use tempdb;
GO

select name, * from sys.sysfiles;
GO
```

The following example shrinks a tempdb database file named `test_file`, and requests a new size of 10 megabytes:

```
exec msdb.dbo.rds_shrink_tempdbfile @temp_filename = N'test_file', @target_size = 10;
```

Setting the SIZE Property

You can also shrink the tempdb database by setting the `SIZE` property and then restarting your DB instance. For more information about restarting your DB instance, see [Rebooting a DB Instance \(p. 129\)](#).

The following example demonstrates setting the `SIZE` property to 1024 MB.

```
alter database [tempdb] modify file (NAME = N'templog', SIZE = 1024MB)
```

Considerations for Multi-AZ Deployments

If your Amazon RDS DB instance is in a Multi-AZ Deployment for Microsoft SQL Server with Database Mirroring or Always On, there are some things to consider.

The tempdb database can't be replicated. No data that you store on your primary instance is replicated to your secondary instance.

If you modify any database options on the tempdb database, you can capture those changes on the secondary by using one of the following methods:

- First modify your DB instance and turn Multi-AZ off, then modify tempdb, and finally turn Multi-AZ back on. This method doesn't involve any downtime.

For more information, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).

- First modify tempdb in the original primary instance, then fail over manually, and finally modify tempdb in the new primary instance. This method involves downtime.

For more information, see [Rebooting a DB Instance \(p. 129\)](#).

Analyzing Your Database Workload on an Amazon RDS DB Instance with SQL Server Tuning Advisor

The Database Engine Tuning Advisor is a client application provided by Microsoft that analyzes database workload and recommends an optimal set of indexes for your Microsoft SQL Server databases based on the kinds of queries you run. Like SQL Server Management Studio, you run Tuning Advisor from a client computer that connects to your Amazon RDS DB instance that is running SQL Server. The client computer can be a local computer that you run on premises within your own network or it can be an Amazon EC2 Windows instance that is running in the same region as your Amazon RDS DB instance.

This section shows how to capture a workload for Tuning Advisor to analyze. This is the preferred process for capturing a workload because Amazon RDS restricts host access to the SQL Server instance. The full documentation on Tuning Advisor can be found on [MSDN](#).

To use Tuning Advisor, you must provide what is called a workload to the advisor. A workload is a set of Transact-SQL statements that execute against a database or databases that you want to tune. Database Engine Tuning Advisor uses trace files, trace tables, Transact-SQL scripts, or XML files as workload input when tuning databases. When working with Amazon RDS, a workload can be a file on a client computer or a database table on an Amazon RDS SQL Server DB accessible to your client computer. The file or the table must contain queries against the databases you want to tune in a format suitable for replay.

For Tuning Advisor to be most effective, a workload should be as realistic as possible. You can generate a workload file or table by performing a trace against your DB instance. While a trace is running, you can either simulate a load on your DB instance or run your applications with a normal load.

There are two types of traces: client-side and server-side. A client-side trace is easier to set up and you can watch trace events being captured in real-time in SQL Server Profiler. A server-side trace is more complex to set up and requires some Transact-SQL scripting. In addition, because the trace is written to a file on the Amazon RDS DB instance, storage space is consumed by the trace. It is important to track of how much storage space a running server-side trace uses because the DB instance could enter a storage-full state and would no longer be available if it runs out of storage space.

For a client-side trace, when a sufficient amount of trace data has been captured in the SQL Server Profiler, you can then generate the workload file by saving the trace to either a file on your local computer or in a database table on a DB instance that is available to your client computer. The main disadvantage of using a client-side trace is that the trace may not capture all queries when under heavy loads. This could weaken the effectiveness of the analysis performed by the Database Engine Tuning Advisor. If you need to run a trace under heavy loads and you want to ensure that it captures every query during a trace session, you should use a server-side trace.

For a server-side trace, you must get the trace files on the DB instance into a suitable workload file or you can save the trace to a table on the DB instance after the trace completes. You can use the SQL Server Profiler to save the trace to a file on your local computer or have the Tuning Advisor read from the trace table on the DB instance.

Running a Client-Side Trace on a SQL Server DB Instance

To run a client-side trace on a SQL Server DB instance

1. Start SQL Server Profiler. It is installed in the Performance Tools folder of your SQL Server instance folder. You must load or define a trace definition template to start a client-side trace.
2. In the SQL Server Profiler File menu, click **New Trace**. In the **Connect to Server** dialog box, enter the DB instance endpoint, port, master user name, and password of the database you would like to run a trace on.
3. In the **Trace Properties** dialog box, enter a trace name and choose a trace definition template. A default template, **TSQL_Replay**, ships with the application. You can edit this template to define your trace. Edit events and event information under the **Events Selection** tab of the **Trace Properties**

dialog box. For more information about trace definition templates and using the SQL Server Profiler to specify a client-side trace see the documentation in [MSDN](#).

4. Start the client-side trace and watch SQL queries in real-time as they execute against your DB instance.
5. Select **Stop Trace** from the **File** menu when you have completed the trace. Save the results as a file or as a trace table on your DB instance.

Running a Server-Side Trace on a SQL Server DB Instance

Writing scripts to create a server-side trace can be complex and is beyond the scope of this document. This section contains sample scripts that you can use as examples. As with a client-side trace, the goal is to create a workload file or trace table that you can open using the Database Engine Tuning Advisor.

The following is an abridged example script that starts a server-side trace and captures details to a workload file. The trace initially saves to the file RDSTrace.trc in the D:\RDSDBDATA\Log directory and rolls-over every 100 MB so subsequent trace files are named RDSTrace_1.trc, RDSTrace_2.trc, etc.

```
DECLARE @file_name NVARCHAR(245) = 'D:\RDSDBDATA\Log\RDSTrace';
DECLARE @max_file_size BIGINT = 100;
DECLARE @on BIT = 1
DECLARE @rc INT
DECLARE @traceid INT

EXEC @rc = sp_trace_create @traceid OUTPUT, 2, @file_name, @max_file_size
IF (@rc = 0) BEGIN
    EXEC sp_trace_setevent @traceid, 10, 1, @on
    EXEC sp_trace_setevent @traceid, 10, 2, @on
    EXEC sp_trace_setevent @traceid, 10, 3, @on
    .
    .
    EXEC sp_trace_setfilter @traceid, 10, 0, 7, N'SQL Profiler'
    EXEC sp_trace_setstatus @traceid, 1
END
```

The following example is a script that stops a trace. Note that a trace created by the previous script continues to run until you explicitly stop the trace or the process runs out of disk space.

```
DECLARE @traceid INT
SELECT @traceid = traceid FROM ::fn_trace_getinfo(default)
WHERE property = 5 AND value = 1 AND traceid <> 1

IF @traceid IS NOT NULL BEGIN
    EXEC sp_trace_setstatus @traceid, 0
    EXEC sp_trace_setstatus @traceid, 2
END
```

You can save server-side trace results to a database table and use the database table as the workload for the Tuning Advisor by using the fn_trace_gettable function. The following commands load the results of all files named RDSTrace.trc in the D:\rdsdbdata\Log directory, including all rollover files like RDSTrace_1.trc, into a table named RDSTrace in the current database.

```
SELECT * INTO RDSTrace
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace.trc', default);
```

To save a specific rollover file to a table, for example the RDSTrace_1.trc file, specify the name of the rollover file and substitute 1 instead of default as the last parameter to fn_trace_gettable.

```
SELECT * INTO RDSTrace_1
```

```
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace_1.trc', 1);
```

Running Tuning Advisor with a Trace

Once you create a trace, either as a local file or as a database table, you can then run Tuning Advisor against your DB instance. Microsoft includes documentation on using the Database Engine Tuning Advisor in [MSDN](#). Using Tuning Advisor with Amazon RDS is the same process as when working with a standalone, remote SQL Server instance. You can either use the Tuning Advisor UI on your client machine or use the dta.exe utility from the command line. In both cases, you must connect to the Amazon RDS DB instance using the endpoint for the DB instance and provide your master user name and master user password when using Tuning Advisor.

The following code example demonstrates using the dta.exe command line utility against an Amazon RDS DB instance with an endpoint of **dta.cnazcmklsdei.us-east-1.rds.amazonaws.com**. The example includes the master user name **admin** and the master user password **test**, the example database to tune is named **RDSDTA** and the input workload is a trace file on the local machine named **C:\RDSTrace.trc**. The example command line code also specifies a trace session named **RDSTrace1** and specifies output files to the local machine named **RDSTrace.sql** for the SQL output script, **RDSTrace.txt** for a result file, and **RDSTrace.xml** for an XML file of the analysis. There is also an error table specified on the RDSDTA database named **RDSTraceErrors**.

```
dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -P test -D RDSDTA -if C:\RDSTrace.trc -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\ RDSTrace.txt -ox C:\ RDSTrace.xml -e RDSDTA.dbo.RDSTraceErrors
```

Here is the same example command line code except the input workload is a table on the remote Amazon RDS instance named **RDSTrace** which is on the **RDSDTA** database.

```
dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -P test -D RDSDTA -it RDSDTA.dbo.RDSTrace -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\ RDSTrace.txt -ox C:\ RDSTrace.xml -e RDSDTA.dbo.RDSTraceErrors
```

A full list of dta utility command-line parameters can be found in [MSDN](#).

Collations and Character Sets for Microsoft SQL Server

Amazon RDS creates a default server collation for character sets when a Microsoft SQL Server DB instance is created. This default server collation is currently English (United States), or more precisely, **SQL_Latin1_General_CI_AS**. You can change the default collation at the database, table, or column level by overriding the collation when creating a new database or database object. For example, you can change from the default collation **SQL_Latin1_General_CI_AS** to **Japanese_CI_AS** for Japanese collation support. Even arguments in a query can be type-cast to use a different collation if necessary.

For example, the following query would change the default collation for the **AccountName** column to **Japanese_CI_AS**:

```
CREATE TABLE [dbo].[Account]
(
    [AccountID] [nvarchar](10) NOT NULL,
    [AccountName] [nvarchar](100) COLLATE Japanese_CI_AS NOT NULL
) ON [PRIMARY];
```

The Microsoft SQL Server DB engine supports Unicode by the built-in **NCHAR**, **NVARCHAR**, and **NTEXT** data types. For example, if you need CJK support, use these Unicode data types for character storage and

override the default server collation when creating your databases and tables. Here are several links from Microsoft covering collation and Unicode support for SQL Server:

- [Working with Collations](#)
- [Collation and International Terminology](#)
- [Using SQL Server Collations](#)
- [International Considerations for Databases and Database Engine Applications](#)

Determining a Recovery Model for Your Microsoft SQL Server Database

In Amazon RDS, the recovery model, retention period, and database status are linked. Changes to one can affect the other settings, for example:

- Changing a database's recovery model to "Simple" while backup retention is enabled will result in Amazon RDS setting the recovery model to "Full" within five minutes of the setting change. This will also result in Amazon RDS taking a snapshot of the DB instance.
- Setting the backup retention to "0" days results in Amazon RDS setting the recovery mode to "Simple."
- Changing a database's recovery model from "Simple" to any other option while backup retention is set to "0" days results in Amazon RDS setting the recovery model back to "Simple."

Dropping a Microsoft SQL Server Database That Is Multi-AZ

You can drop a database on an Amazon RDS DB instance running Microsoft SQL Server in a Multi-AZ deployment. To drop the database, use the following command:

```
--replace your-database-name with the name of the database you want to drop
EXECUTE msdb.dbo.rds_drop_database N'your-database-name'
```

Note

After you use this procedure to drop the database, Amazon RDS drops all existing connections to the database and removes the database's backup history.

Using Change Data Capture

Amazon RDS supports change data capture (CDC) for your DB instances running Microsoft SQL Server. CDC captures changes that are made to the data in your tables. It stores metadata about each change, which you can access later. For more information about how CDC works, see [Change Data Capture](#) in the Microsoft documentation.

Before you use CDC with your Amazon RDS DB instances, enable it in the database by running `msdb.dbo.rds_cdc_enable_db`. After CDC is enabled, any user who is `db_owner` of that database can enable or disable CDC on tables in that database.

Important

During restores, CDC will be disabled. All of the related metadata is automatically removed from the database. This applies to snapshot restores, point-in-time restores, and SQL Server Native restores from S3. After performing one of these types of restores, you can re-enable CDC and re-specify tables to track.

```
--Enable CDC for RDS DB Instance
```

```
exec msdb.dbo.rds_cdc_enable_db '<database name>'
```

To disable CDC, `msdb.dbo.rds_cdc_disable_db` run .

```
--Disable CDC for RDS DB Instance
exec msdb.dbo.rds_cdc_disable_db '<database name>'
```

Topics

- [Tracking Tables with Change Data Capture \(p. 571\)](#)
- [Change Data Capture Jobs \(p. 571\)](#)
- [Change Data Capture for Multi-AZ Instances \(p. 572\)](#)

Tracking Tables with Change Data Capture

After CDC is enabled on the database, you can start tracking specific tables. You can choose the tables to track by running [sys.sp_cdc_enable_table](#).

```
--Begin tracking a table
exec sys.sp_cdc_enable_table
    @source_schema      = N'<source_schema>'
    , @source_name       = N'<source_name>'
    , @role_name         = N'<role name>'

--The following parameters are optional:

--, @capture_instance      = '<capture_instance>'
--, @supports_net_changes  = '<supports_net_changes>'
--, @index_name             = '<index_name>'
--, @captured_column_list   = '<captured_column_list>'
--, @filegroup_name         = '<filegroup_name>'
--, @allow_partition_switch = '<allow_partition_switch>'
;
```

To view the CDC configuration for your tables, run [sys.sp_cdc_help_change_data_capture](#).

```
--View CDC configuration
exec sys.sp_cdc_help_change_data_capture

--The following parameters are optional and must be used together.
-- '<schema name>', '<table name>'
;
```

For more information on CDC tables, functions, and stored procedures in SQL Server documentation, see the following:

- [Change Data Capture Stored Procedures \(Transact-SQL\)](#)
- [Change Data Capture Functions \(Transact-SQL\)](#)
- [Change Data Capture Tables \(Transact-SQL\)](#)

Change Data Capture Jobs

When you enable CDC, SQL Server creates the CDC jobs. Database owners (`db_owner`) can view, create, modify, and delete the CDC jobs. However, the RDS system account owns them. Therefore, the jobs aren't visible from native views, procedures, or in SQL Server Management Studio.

To control behavior of CDC in a database, use native SQL Server procedures such as `sp_cdc_enable_table` and `sp_cdc_start_job`. To change CDC job parameters, like `maxtrans` and `maxscans`, you can use `sp_cdc_change_jobs`.

To get more information regarding the CDC jobs, you can query the following dynamic management views:

- `sys.dm_cdc_errors`
- `sys.dm_cdc_log_scan_sessions`
- `sysjobs`
- `sysjobhistory`

Change Data Capture for Multi-AZ Instances

If you use CDC on a Multi-AZ instance, make sure the mirror's CDC job configuration matches the one on the principal. CDC jobs are mapped to the `database_id`. If the database IDs on the secondary are different from the principal, then the jobs won't be associated with the correct database. To try to prevent errors after failover, RDS drops and recreates the jobs on the new principal. The recreated jobs use the parameters that the principal recorded before failover.

Although this process runs quickly, it's still possible that the CDC jobs might run before RDS can correct them. Here are three ways to force parameters to be consistent between primary and secondary replicas:

- Use the same job parameters for all the databases that have CDC enabled.
- Before you change the CDC job configuration, convert the Multi-AZ instance to Single-AZ.
- Manually transfer the parameters whenever you change them on the principal.

To view and define the CDC parameters that are used to recreate the CDC jobs after a failover, use `rds_show_configuration` and `rds_set_configuration`.

The following example returns the value set for `cdc_capture_maxtrans`. For any parameter that is set to `RDS_DEFAULT`, RDS automatically configures the value.

```
-- Show configuration for each parameter on either primary and secondary replicas.  
exec rdsadmin.dbo.rds_show_configuration 'cdc_capture_maxtrans'
```

To set the configuration on the secondary, run `rdsadmin.dbo.rds_set_configuration`. This procedure sets the parameter values for all of the databases on the secondary server. These settings are used only after a failover. The following example sets the `maxtrans` for all CDC capture jobs to `1000`:

```
--To set values on secondary. These are used after failover.  
exec rdsadmin..rds_set_configuration 'cdc_capture_maxtrans' , 1000
```

To set the CDC job parameters on the principal, use `sys.sp_cdc_change_jobs` instead.

Renaming a Microsoft SQL Server Database in a Multi-AZ Deployment

To rename a Microsoft SQL Server database instance that uses Multi-AZ, use the following procedure:

1. First, turn off Multi-AZ for the DB instance.
2. Rename the database by running `rdsadmin.dbo.rds_modify_db_name`.
3. Then, turn on Multi-AZ Mirroring or Always On for the DB instance, to return it to its original state.

For more information, see [Adding Multi-AZ to a Microsoft SQL Server DB Instance \(p. 551\)](#).

Note

If your instance doesn't use Multi-AZ, you don't need to change any settings before or after running `rdsadmin.dbo.rds_modify_db_name`.

Example: In the following example, the `rdsadmin.dbo.rds_modify_db_name` stored procedure renames a database from `MOO` to `ZAR`. This is similar to running the statement `DDL ALTER DATABASE [MOO] MODIFY NAME = [ZAR]`.

```
EXEC rdsadmin.dbo.rds_modify_db_name N'MOO', N'ZAR'  
GO
```

Resetting the db_owner Role Password

If you lock yourself out of the `db_owner` role on your Microsoft SQL Server database, you can reset the `db_owner` role password by modifying the DB instance master password. By changing the DB instance master password, you can regain access to the DB instance, access databases using the modified password for the `db_owner`, and restore privileges for the `db_owner` role that may have been accidentally revoked. You can change the DB instance password by using the Amazon RDS console, the AWS CLI command [modify-db-instance](#), or by using the [ModifyDBInstance](#) action. For more information about modifying a SQL Server DB instance, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).

Restoring License-Terminated DB Instances

Microsoft has requested that some Amazon RDS customers who did not report their Microsoft License Mobility information terminate their DB instance. Amazon RDS takes snapshots of these DB instances, and you can restore from the snapshot to a new DB instance that has the License Included model.

You can restore from a snapshot of Standard Edition to either Standard Edition or Enterprise Edition.

You can restore from a snapshot of Enterprise Edition to either Standard Edition or Enterprise Edition.

To restore from a SQL Server snapshot after Amazon RDS has created a final snapshot of your instance:

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
 2. In the navigation pane, choose **Snapshots**.
 3. Choose the snapshot of your SQL Server DB instance. Amazon RDS creates a final snapshot of your DB instance. The name of the terminated instance snapshot is in the format '`<name of instance>-final-snapshot`'. For example, if your DB instance name is `mytest.cdxgahs1ksma.us-east-1.rds.com`, the final snapshot is called `mytest-final-snapshot` and is located in the same AWS Region as the original DB instance.
 4. For **Actions**, choose **Restore Snapshot**.
- The **Restore DB Instance** window appears.
5. For **License Model**, choose **license-included**.
 6. Choose the SQL Server DB engine that you want to use.
 7. For **DB Instance Identifier**, enter the name for the restored DB instance.
 8. Choose **Restore DB Instance**.

For more information about restoring from a snapshot, see [Restoring from a DB Snapshot \(p. 218\)](#).

Transitioning a Microsoft SQL Server Database from OFFLINE to ONLINE

You can transition your Microsoft SQL Server database on an Amazon RDS DB instance from OFFLINE to ONLINE.

SQL Server method	Amazon RDS method
ALTER DATABASE <i>name</i> SET ONLINE;	EXEC rdsadmin.dbo.rds_set_database_online <i>name</i>

Using SQL Server Agent

With Amazon RDS, you can use SQL Server Agent on a DB instance running Microsoft SQL Server Standard, Web Edition, or Enterprise Edition. SQL Server Agent is a Microsoft Windows service that executes scheduled administrative tasks, which are called jobs. You can use SQL Server Agent to run T-SQL jobs to rebuild indexes, run corruption checks, and aggregate data in a SQL Server DB instance.

SQL Server Agent can run a job on a schedule, in response to a specific event, or on demand. For more information, see [SQL Server Agent](#) in the SQL Server documentation. You should avoid scheduling jobs to run during the maintenance and backup windows for your DB instance because these maintenance and backup processes that are launched by AWS could interrupt the job or cause it to be cancelled. Because Amazon RDS backs up your DB instance, you do not use SQL Server Agent to create backups.

To view the history of an individual SQL Server Agent job in the SQL Server Management Studio, you open Object Explorer, right-click the job, and then click **View History**.

Because SQL Server Agent is running on a managed host in a DB instance, there are some actions that are not supported. Running replication jobs and running command-line scripts by using ActiveX, Windows command shell, or Windows PowerShell are not supported. In addition, you cannot manually start, stop, or restart SQL Server Agent because its operation is managed by the host. Email notifications through SQL Server Agent are not available from a DB instance.

When you create a SQL Server DB instance, the master user name is enrolled in the SQLAgentUserRole role. To allow an additional login/user to use SQL Server Agent, you must log in as the master user and do the following.

1. Create another server-level login by using the `CREATE LOGIN` command.
2. Create a user in msdb using `CREATE USER` command, and then link this user to the login that you created in the previous step.
3. Add the user to the SQLAgentUserRole using the `sp_addrolemember` system stored procedure.

For example, suppose your master user name is `myawsmaster` and you want to give access to SQL Server Agent to a user named `theirname` with a password `theirpassword`. You would log in using the master user name and run the following commands.

```
--Initially set context to master database
USE [master];
GO
--Create a server-level login named theirname with password theirpassword
CREATE LOGIN [theirname] WITH PASSWORD = 'theirpassword';
GO
--Set context to msdb database
USE [msdb];
```

```
GO
--Create a database user named theirname and link it to server-level login theirname
CREATE USER [theirname] FOR LOGIN [theirname];
GO
--Added database user theirname in msdb to SQLAgentUserRole in msdb
EXEC sp_addrolemember [SQLAgentUserRole], [theirname];
```

To delete a SQL Server Agent job, run the following T-SQL statement.

```
EXEC msdb..sp_delete_job @job_name = '<job-name>';
```

Note

Don't use the UI in SQL Server Management Console (SSMS) to delete a SQL Server Agent job. If you do, you get an error message similar to the following:

```
The EXECUTE permission was denied on the object 'xp_regrid', database
'mssqlsystemresource', schema 'sys'.
```

This error occurs because, as a managed service, RDS is restricted from running procedures that access the Windows registry. When you use SSMS to delete the job, it tries to run a process that RDS isn't authorized to do.

Working with Microsoft SQL Server Logs

You can use the Amazon RDS console to view, watch, and download SQL Server Agent logs and Microsoft SQL Server error logs.

Watching Log Files

If you view a log in the Amazon RDS console, you can see its contents as they exist at that moment. Watching a log in the console opens it in a dynamic state so that you can see updates to it in near real time.

Only the latest log is active for watching. For example, suppose you have the logs shown following:

Name	Last Written	Size	view	watch	download
log/ERROR	January 14, 2015 at 5:17:35 AM UTC-8	6.1 kB	view	watch	download
log/ERROR.1	January 13, 2015 at 3:59:00 PM UTC-8	53.3 kB	view	watch	download
log/ERROR.2	January 12, 2015 at 3:59:00 PM UTC-8	5.9 kB	view	watch	download
log/ERROR.3	January 11, 2015 at 3:59:00 PM UTC-8	5.9 kB	view	watch	download
log/ERROR.4	January 10, 2015 at 3:59:00 PM UTC-8	5.9 kB	view	watch	download

Only log/ERROR, as the most recent log, is being actively updated. You can choose to watch others, but they are static and will not update.

Archiving Log Files

The Amazon RDS console shows logs for the past week through the current day. You can download and archive logs to keep them for reference past that time. One way to archive logs is to load them into an Amazon S3 instance. For instructions on how to set up an Amazon S3 instance and upload a file, see [Amazon S3 Basics](#) in the *Amazon Simple Storage Service Getting Started Guide* and click **Get Started**.

Using the rds_read_error_log Procedure

To view Microsoft SQL server error and agent logs, use the Amazon RDS stored procedure `rds_read_error_log` with the following parameters:

- **@index** – the version of the log to retrieve. The default value is 0, which retrieves the current error log. Specify 1 to retrieve the previous log, specify 2 to retrieve the one before that, and so on.
- **@type** – the type of log to retrieve. Specify 1 to retrieve an error log. Specify 2 to retrieve an agent log.

Example

The following example requests the current error log.

```
EXEC rdsadmin.dbo.rds_read_error_log @index = 0, @type = 1;
```

Working with Trace and Dump Files

This section describes working with trace files and dump files for your Amazon RDS DB instances running Microsoft SQL Server.

Generating a Trace SQL Query

```
declare @rc int
declare @TraceID int
declare @maxfilesize bigint

set @maxfilesize = 5

exec @rc = sp_trace_create @TraceID output, 0, N'D:\rdsdbdata\log\rdstest', @maxfilesize,
NULL
```

Viewing an Open Trace

```
select * from ::fn_trace_getinfo(default)
```

Viewing Trace Contents

```
select * from ::fn_trace_gettable('D:\rdsdbdata\log\rdstest.trc', default)
```

Setting the Retention Period for Trace and Dump Files

Trace and dump files can accumulate and consume disk space. By default, Amazon RDS purges trace and dump files that are older than seven days.

To view the current trace and dump file retention period, use the `rds_show_configuration` procedure, as shown in the following example.

```
exec rdsadmin..rds_show_configuration;
```

To modify the retention period for trace files, use the `rds_set_configuration` procedure and set the `tracefile retention` in minutes. The following example sets the trace file retention period to 24 hours.

```
exec rdsadmin..rds_set_configuration 'tracefile retention', 1440;
```

To modify the retention period for dump files, use the `rds_set_configuration` procedure and set the `dumpfile retention` in minutes. The following example sets the dump file retention period to 3 days.

```
exec rdsadmin..rds_set_configuration 'dumpfile retention', 4320;
```

For security reasons, you cannot delete a specific trace or dump file on a SQL Server DB instance. To delete all unused trace or dump files, set the retention period for the files to 0.

Advanced Administrative Tasks and Concepts for Microsoft SQL Server DB Instances

This section provides information about advanced administrative tasks and concepts for Microsoft SQL Server DB instances on Amazon RDS.

Topics

- [Using Windows Authentication with a Microsoft SQL Server DB Instance \(p. 578\)](#)

Using Windows Authentication with a Microsoft SQL Server DB Instance

You can use Windows Authentication to authenticate users when they connect to your Amazon RDS DB instance running Microsoft SQL Server. The DB instance works with AWS Directory Service for Microsoft Active Directory, also called **AWS Managed Microsoft AD**, to enable Windows Authentication. When users authenticate with a SQL Server DB instance joined to the trusting domain, authentication requests are forwarded to the domain directory that you create with AWS Directory Service.

Amazon RDS supports Windows Authentication for SQL Server in all AWS Regions except the following:

- US West (N. California)
- Asia Pacific (Mumbai)
- South America (São Paulo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Amazon RDS uses Mixed Mode for Windows Authentication. This approach means that the *master user* (the name and password used to create your SQL Server DB instance) uses SQL Authentication. Because the master user account is a privileged credential, you should restrict access to this account.

To get Windows Authentication using an on-premises or self-hosted Microsoft Active Directory, you need to create a forest trust. For more information on setting up forest trusts using AWS Directory Service, see [Create a Trust Relationship \(AWS Managed Microsoft AD\)](#).

To set up Windows authentication for a SQL Server DB instance, do the following steps (explained in greater detail in this section):

1. Use the AWS Directory Service for Microsoft Active Directory, also called AWS Managed Microsoft AD, either from the AWS console or AWS Directory Service API to create a AWS Managed Microsoft AD directory.
2. If you use the AWS CLI or Amazon RDS API to create your SQL Server DB instance, you need to create an IAM role that uses the managed IAM policy `AmazonRDSDirectoryServiceAccess`. The role allows Amazon RDS to make calls to your directory. If you use the AWS console to create your SQL Server DB instance, AWS creates the IAM role for you.
3. Create and configure users and groups in the *AWS Managed Microsoft AD* directory using the Microsoft Active Directory tools. For more information about creating users and groups in your Active Directory, see [Add Users and Groups \(Simple AD and AWS Managed Microsoft AD\)](#) in the AWS Directory Service documentation. [Add Users and Groups \(Simple AD and AWS Managed Microsoft AD\)](#).
4. Use Amazon RDS to create a new SQL Server DB instance either from the AWS console, AWS CLI, or Amazon RDS API. In the create request, you provide the domain identifier ("d-*" identifier) that was generated when you created your directory and the name of the role you created. You can also modify an existing SQL Server DB instance to use Windows Authentication by setting the *domain* and *IAM role* parameters for the DB instance, and locating the DB instance in the same VPC as the domain directory.
5. Use the Amazon RDS *master user* credentials to connect to the SQL Server DB instance as you would any other DB instance. Because the DB instance is joined to the *AWS Managed Microsoft AD* domain, you can provision SQL Server logins and users from the Active Directory users and groups in their domain (known as SQL Server "Windows" logins). Database permissions are managed through standard SQL Server permissions granted and revoked to these windows logins.

Creating the Endpoint for Kerberos Authentication

Kerberos-based authentication requires that the endpoint be the customer-specified host name, a period, and then the fully qualified domain name (FQDN). For example, the following is an example of an endpoint you would use with Kerberos-based authentication. In this example, the SQL Server DB instance host name is `ad-test` and the domain name is `corp-ad.company.com`:

```
ad-test.corp-ad.company.com
```

If you want to check to make sure your connection is using Kerberos, you can run the following query:

```
SELECT net_transport, auth_scheme
  FROM sys.dm_exec_connections
 WHERE session_id = @@SPID;
```

Setting Up Windows Authentication for SQL Server DB Instances

You use AWS Directory Service for Microsoft Active Directory, also called **AWS Managed Microsoft AD**, to set up Windows Authentication for a SQL Server DB instance. To set up Windows Authentication, you take the following steps:

Step 1: Create a Directory Using the AWS Directory Service for Microsoft Active Directory

AWS Directory Service creates a fully managed, Microsoft Active Directory in the AWS cloud. When you create a AWS Managed Microsoft AD directory, AWS Directory Service creates two domain controllers and DNS servers on your behalf. The directory servers are created in different subnets in a VPC; this redundancy helps ensure that your directory remains accessible even if a failure occurs.

When you create a *AWS Managed Microsoft AD* directory, AWS Directory Service performs the following tasks on your behalf:

- Sets up a Microsoft Active Directory within the VPC.
- Creates a directory administrator account with the user name Admin and the specified password. You use this account to manage your directory.

Note

Be sure to save this password. AWS Directory Service does not store this password and it cannot be retrieved or reset.

- Creates a security group for the directory controllers.

When you launch an AWS Directory Service for Microsoft Active Directory, AWS creates an Organizational Unit (OU) that contains all your directory's objects. This OU, which has the NetBIOS name that you typed when you created your directory, is located in the domain root. The domain root is owned and managed by AWS.

The `admin` account that was created with your *AWS Managed Microsoft AD* directory has permissions for the most common administrative activities for your OU:

- Create update, or delete users, groups, and computers
- Add resources to your domain such as file or print servers, and then assign permissions for those resources to users and groups in your OU
- Create additional OUs and containers

- Delegate authority
- Create and link group policies
- Restore deleted objects from the Active Directory Recycle Bin
- Run AD and DNS Windows PowerShell modules on the Active Directory Web Service

The *admin* account also has rights to perform the following domain-wide activities:

- Manage DNS configurations (Add, remove, or update records, zones, and forwarders)
- View DNS event logs
- View security event logs

To create a directory with AWS Directory Service for Microsoft Active Directory

1. In the [AWS Directory Service console](#) navigation pane, select **Directories** and choose **Set up Directory**.
2. Choose **Create AWS Managed Microsoft AD**. AWS Managed Microsoft AD is the only option currently supported for use with Amazon RDS.
3. Provide the following information:

Directory DNS

The fully qualified name for the directory, such as corp.example.com.

NetBIOS name

The short name for the directory, such as CORP.

Administrator password

The password for the directory administrator. The directory creation process creates an administrator account with the user name Admin and this password.

The directory administrator password and cannot include the word "admin." The password is case-sensitive and must be between 8 and 64 characters in length, inclusive. It must also contain at least one character from three of the following four categories:

- Lowercase letters (a-z)
- Uppercase letters (A-Z)
- Numbers (0-9)
- Non-alphanumeric characters (~!@#\$%^&*_+=`|\{}{};:"";<>,?./)

Confirm password

Retype the administrator password.

Description

An optional description for the directory.

4. Provide the following information in the **VPC Details** section and choose **Next Step**.

VPC

The VPC for the directory. Note that the SQL Server DB instance must be created in this same VPC.

Subnets

Select the subnets for the directory servers. The two subnets must be in different Availability Zones.

5. Review the directory information and make any necessary changes. When the information is correct, choose **Create AWS Managed Microsoft AD**.

It takes several minutes for the directory to be created. When it has been successfully created, the **Status** value changes to **Active**.

To see information about your directory, select the directory in the directory listing. Note the Directory ID; you will need this value when you create or modify your SQL Server DB instance.

Directories > AmazonRDS.com (d-90673c663e)	
▼ Details	
Directory type	Microsoft AD
Directory ID	d-90673c663e
Directory name	AmazonRDS.com
NetBIOS name	RDS
Description	test active directory
DNS Address	172.30.4.100, 172.30.5.4
Status	Creating
Status last updated	Thu Feb 25 12:57:26 GMT-800 2016
Launch time	Thu Feb 25 12:57:23 GMT-800 2016
Availability zones	us-east-1e, us-east-1a
VPC	vpc-fd8c1b99
Subnets	subnet-b38b9f98, subnet-4d802a3b
Enabled apps & services	

Step 2: Create the IAM role for Use by Amazon RDS

If you use the AWS console to create your SQL Server DB instance, you can skip this step. If you used the AWS CLI or Amazon RDS API to create your SQL Server DB instance, you must create an IAM role that uses the managed IAM policy **AmazonRDSDirectoryServiceAccess**. This role allows Amazon RDS to make calls to the AWS Directory Service for you.

The following IAM policy, **AmazonRDSDirectoryServiceAccess**, provides access to AWS Directory Service:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "ds:DescribeDirectories",  
                "ds:AuthorizeApplication",  
                "ds:UnauthorizeApplication"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        }  
    ]  
}
```

Create an IAM role using this policy. For more information about creating IAM roles, see [Creating Customer Managed Policies](#).

Step 3: Create and Configure Users and Groups

You can create users and groups with the Active Directory Users and Computers tool, which is part of the Active Directory Domain Services and Active Directory Lightweight Directory Services tools. Users represent individual people or entities that have access to your directory. Groups are very useful for giving or denying privileges to groups of users, rather than having to apply those privileges to each individual user.

To create users and groups in an AWS Directory Service directory, you must be connected to a Windows EC2 instance that is a member of the AWS Directory Service directory, and be logged in as a user that has privileges to create users and groups. For more information, see [Add Users and Groups \(Simple AD and AWS Managed Microsoft AD\)](#).

Step 4: Create or Modify a SQL Server DB Instance

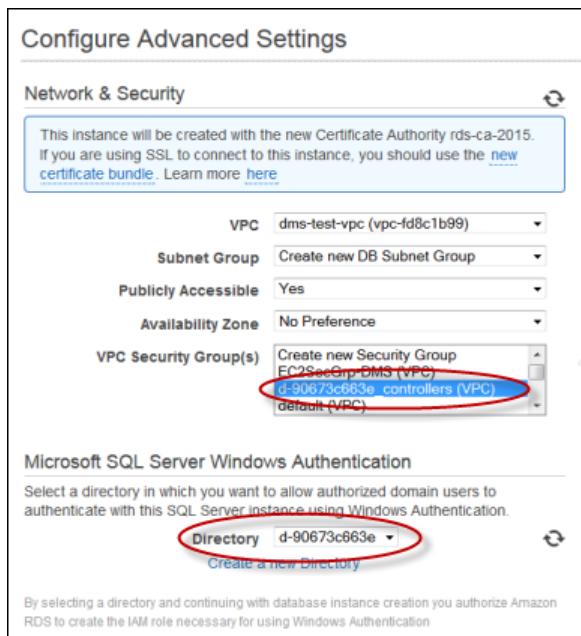
Next, you create or modify a Microsoft SQL Server DB instance for use with the directory. You can do this in one of the following ways:

- Create a new SQL Server DB instance
- Modify an existing SQL Server DB instance
- Restore a SQL Server DB instance from a DB Snapshot
- Restore a SQL Server DB instance from a Point-in-Time Restore

Windows Authentication is only supported for SQL Server DB instances in a VPC, and the DB instance must be in the same VPC as the directory.

Several parameters are required for the DB instance to be able to use the domain directory you created:

- For the **domain** parameter, you must enter the domain identifier ("d-*" identifier) generated when you created the directory.
- Use the same VPC that was used when you created the directory.
- Use a security group that allows egress within the VPC so the DB instance can communicate with the directory.



Step 5: Create Windows Authentication SQL Server Logins

Use the Amazon RDS *master user* credentials to connect to the SQL Server DB instance as you would any other DB instance. Because the DB instance is joined to the AWS *Managed Microsoft AD* domain, you can provision SQL Server logins and users from the Active Directory users and groups in your domain. Database permissions are managed through standard SQL Server permissions granted and revoked to these windows logins.

To allow an Active Directory user to authenticate with SQL Server, a SQL Server Windows login must exist for the user or a group that the user is a member of. Fine-grained access control is handled through granting and revoking permissions on these SQL Server logins. If a user does not have a corresponding SQL Server login and is not a member of a group with a corresponding SQL Server login, that user cannot access the SQL Server DB instance.

The ALTER ANY LOGIN permission is required to create an Active Directory SQL Server login. If you have not yet created any logins with this permission, connect as the DB instance's *master user* using SQL Server Authentication. Run the following data definition language (DDL) command to create a SQL Server login for an Active Directory user or group:

```
CREATE LOGIN [<user or group>] FROM WINDOWS WITH DEFAULT_DATABASE = [master],  
DEFAULT_LANGUAGE = [us_english];
```

Users or groups must be specified using the pre-Windows 2000 login name in the format *domainName\login_name*. You cannot use a User Principle Name (UPN) in the format *login_name@DomainName*. For more information about CREATE LOGIN, go to <https://msdn.microsoft.com/en-us/library/ms189751.aspx> in the Microsoft Developer Network documentation.

Users (both humans and applications) from your domain can now connect to the RDS SQL Server instance from a domain joined client machine using Windows authentication.

Managing a DB Instance in a Domain

You can use the AWS console, AWS CLI, or the Amazon RDS API to manage your DB instance and its relationship with your domain, such as moving the DB instance into, out of, or between domains.

For example, using the Amazon RDS API, you can do the following:

- To re-attempt a domain join for a failed membership, use the *ModifyDBInstance* API action and specify the current membership's directory ID.
- To update the IAM role name for membership, use the *ModifyDBInstance* API action and specify the current membership's directory ID and the new IAM role.
- To remove a DB instance from a domain, use the *ModifyDBInstance* API action and specify 'none' as the domain parameter.
- To move a DB instance from one domain to another, use the *ModifyDBInstance* API action and specify the domain identifier of the new domain as the domain parameter.
- To list membership for each DB instance, use the *DescribeDBInstances* API action.

Understanding Domain Membership

After you create or modify your DB instance, the instance becomes a member of the domain. The AWS console indicates the status of the domain membership for the DB instance. The status of the DB instance can be one of the following:

- joined - The instance is a member of the domain.
- joining - The instance is in the process of becoming a member of the domain.
- pending-join - The instance membership is pending .
- pending-maintenance-join - AWS will attempt to make the instance a member of the domain during the next scheduled maintenance window.
- pending-removal - The removal of the instance from the domain is pending.
- pending-maintenance-removal - AWS will attempt to remove the instance from the domain during the next scheduled maintenance window.
- failed - A configuration problem has prevented the instance from joining the domain. Check and fix your configuration before re-issuing the instance modify command.
- removing - The instance is being removed from the domain.

A request to become a member of a domain can fail because of a network connectivity issue or an incorrect IAM role. If you create a DB instance or modify an existing instance and the attempt to become a member of a domain fails, you should re-issue the modify command or modify the newly created instance to join the domain.

Connecting to SQL Server with Windows Authentication

To connect to SQL Server with Windows Authentication, you must be logged into a domain-joined computer as a domain user. After launching SQL Server Management Studio, choose **Windows Authentication** as the authentication type, as shown following.



Restoring a SQL Server DB Instance and then Adding It to a Domain

You can restore a DB snapshot or do a point-in-time restore for a SQL Server DB instance and then add it to a domain. Once the DB instance is restored, modify the instance using the process explained in the section [Step 4: Create or Modify a SQL Server DB Instance \(p. 582\)](#) to add the DB instance to a domain.

Related Topics

- [Microsoft SQL Server on Amazon RDS \(p. 488\)](#)
- [Configuring Security in Amazon RDS \(p. 342\)](#)

MySQL on Amazon RDS

Amazon RDS supports DB instances running several versions of MySQL. You can use the following major versions:

- MySQL 8.0
- MySQL 5.7
- MySQL 5.6
- MySQL 5.5

For more information about minor version support, see [MySQL on Amazon RDS Versions \(p. 589\)](#).

You first use the Amazon RDS management tools or interfaces to create an Amazon RDS MySQL DB instance. You can then resize the DB instance, authorize connections to the DB instance, create and restore from backups or snapshots, create Multi-AZ secondaries, create Read Replicas, and monitor the performance of the DB instance. You use standard MySQL utilities and applications to store and access the data in the DB instance.

Amazon RDS for MySQL is compliant with many industry standards. For example, you can use Amazon RDS for MySQL databases to build HIPAA-compliant applications and to store healthcare related information, including protected health information (PHI) under an executed Business Associate Agreement (BAA) with AWS. Amazon RDS for MySQL also meets Federal Risk and Authorization Management Program (FedRAMP) security requirements and has received a FedRAMP Joint Authorization Board (JAB) Provisional Authority to Operate (P-ATO) at the FedRAMP HIGH Baseline within the AWS GovCloud (US-West) Region. For more information on supported compliance standards, see [AWS Cloud Compliance](#).

For information about the features in each version of MySQL, see [The Main Features of MySQL](#) in the MySQL documentation.

Common Management Tasks for MySQL on Amazon RDS

The following are the common management tasks you perform with an Amazon RDS MySQL DB instance, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Understanding Amazon RDS Understand key Amazon RDS components, including DB instances, regions, Availability Zones, security groups, parameter groups, and option groups.	What Is Amazon Relational Database Service (Amazon RDS)? (p. 1)
Setting up Amazon RDS for first time use Set up Amazon RDS so that you can create MySQL DB instances in Amazon Web Services (AWS).	Setting Up for Amazon RDS (p. 5)

Task Area	Relevant Documentation
Understanding Amazon RDS DB instances Create virtual MySQL server instances that run in AWS. Because DB instances are the building blocks of Amazon RDS, we recommend that you understand their principles.	Amazon RDS DB Instances (p. 80)
Creating a DB instance for production Create a DB instance for production purposes. Creating an instance includes choosing a DB instance class with appropriate processing power and memory capacity and choosing a storage type that supports the way you expect to use your database.	DB Instance Class (p. 82) Amazon RDS Storage Types (p. 103) Creating a DB Instance Running the MySQL Database Engine (p. 597)
Managing security for your DB instance By default, DB instances are created with a firewall that prevents access to them. You must create a security group with the correct IP addresses and network configuration to access the DB instance. You can also use AWS Identity and Access Management (IAM) policies to assign permissions that determine who is allowed to manage RDS resources.	Configuring Security in Amazon RDS (p. 342) Overview of Managing Access Permissions to Your Amazon RDS Resources (p. 344) Controlling Access with Security Groups (p. 394) Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412)
Connecting to your DB instance Connect to your DB instance using a standard SQL client application such as the MySQL command line utility or MySQL Workbench.	Connecting to a DB Instance Running the MySQL Database Engine (p. 606)
Configuring high availability for a production DB instance Provide high availability with synchronous standby replication in a different Availability Zone, automatic failover, fault tolerance for DB instances using Multi-AZ deployments, and Read Replicas.	High Availability (Multi-AZ) for Amazon RDS (p. 109)
Configuring a DB instance in an Amazon Virtual Private Cloud Configure a virtual private cloud (VPC) in the Amazon VPC service. An Amazon VPC is a virtual network logically isolated from other virtual networks in AWS.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412) Working with an Amazon RDS DB Instance in a VPC (p. 420)

Task Area	Relevant Documentation
Configuring specific MySQL database parameters and features Configure specific MySQL database parameters with a parameter group that can be associated with many DB instances. You can also configure specific MySQL database features with an option group that can be associated with many DB instances.	Working with DB Parameter Groups (p. 169) Working with Option Groups (p. 156) Options for MySQL DB Instances (p. 677)
Modifying a DB instance running the MySQL database engine Change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class.	Modifying a DB Instance Running the MySQL Database Engine (p. 610) Modifying an Amazon RDS DB Instance (p. 115)
Configuring database backup and restore Configure your DB instance to take automated backups. You can also back up and restore your databases manually by using full backup files.	Working With Backups (p. 208) Backing Up and Restoring Amazon RDS DB Instances (p. 207)
Importing and exporting data Import data from other RDS MySQL DB instances, MySQL instances running external to Amazon RDS, and other types of data sources, and export data to MySQL instances running external to Amazon RDS.	Restoring a Backup into an Amazon RDS MySQL DB Instance (p. 631)
Monitoring a MySQL DB instance Monitor your RDS MySQL DB instance by using Amazon CloudWatch RDS metrics, events, and Enhanced Monitoring. View log files for your RDS MySQL DB instance.	Monitoring Amazon RDS (p. 244) Viewing DB Instance Metrics (p. 254) Viewing Amazon RDS Events (p. 305) Amazon RDS Database Log Files (p. 307) MySQL Database Log Files (p. 320)
Replicating your data Create a MySQL Read Replica, in the same AWS Region or a different one. You can use Read Replicas for load balancing, disaster recovery, and processing read-heavy database workloads, such as for analysis and reporting.	Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances (p. 143) Replication with a MySQL or MariaDB Instance Running External to Amazon RDS (p. 667)

There are also several sections with useful information about working with Amazon RDS MySQL DB instances:

- [Common DBA Tasks for MySQL DB Instances \(p. 685\)](#)
- [Options for MySQL DB Instances \(p. 677\)](#)
- [MySQL on Amazon RDS SQL Reference \(p. 692\)](#)

MySQL on Amazon RDS Versions

For MySQL, version numbers are organized as version = X.Y.Z. In Amazon RDS terminology, X.Y denotes the major version, and Z is the minor version number. For Amazon RDS implementations, a version change is considered major if the major version number changes—for example, going from version 5.6 to 5.7. A version change is considered minor if only the minor version number changes—for example, going from version 5.7.16 to 5.7.21.

Amazon RDS currently supports the following versions of MySQL:

Major Version	Minor Version
MySQL 8.0	<ul style="list-style-type: none">8.0.11
MySQL 5.7	<ul style="list-style-type: none">5.7.235.7.225.7.215.7.195.7.175.7.16
MySQL 5.6	<ul style="list-style-type: none">5.6.415.6.405.6.395.6.375.6.355.6.34
MySQL 5.5	<ul style="list-style-type: none">5.5.615.5.595.5.575.5.545.5.535.5.46

You can specify any currently supported MySQL version when creating a new DB instance. You can specify the MySQL 8.0, 5.7, 5.6, or 5.5 major versions, and any supported minor version for the specified major version. If no version is specified, Amazon RDS will default to a supported version, typically the most recent version. If a major version (for example, MySQL 5.7) is specified but a minor version is not, Amazon RDS will default to a recent release of the major version you have specified. To see a list of supported versions, as well as defaults for newly created DB instances, use the [DescribeDBEngineVersions](#) API action.

With Amazon RDS, you control when to upgrade your MySQL instance to a new version supported by Amazon RDS. You can maintain compatibility with specific MySQL versions, test new versions with your application before deploying in production, and perform version upgrades at times that best fit your schedule.

Unless you specify otherwise, your DB instance will automatically be upgraded to new MySQL minor versions as they are supported by Amazon RDS. This patching occurs during your scheduled maintenance window. You can modify a DB instance to turn off automatic minor version upgrades.

If you opt out of automatically scheduled upgrades, you can manually upgrade to a supported minor version release by following the same procedure as you would for a major version update. For information, see [Upgrading a DB Instance Engine Version \(p. 123\)](#).

Amazon RDS currently supports the major version upgrades from MySQL version 5.5 to version 5.6, from MySQL version 5.6 to version 5.7, and from MySQL version 5.7 to version 8.0. Because major version upgrades involve some compatibility risk, they do not occur automatically; you must make a request to modify the DB instance. You should thoroughly test any upgrade before upgrading your production instances. For information about upgrading a MySQL DB instance, see [Upgrading the MySQL DB Engine \(p. 618\)](#).

You can test a DB instance against a new version before upgrading by creating a DB snapshot of your existing DB instance, restoring from the DB snapshot to create a new DB instance, and then initiating a version upgrade for the new DB instance. You can then experiment safely on the upgraded clone of your DB instance before deciding whether or not to upgrade your original DB instance.

For information about the Amazon RDS deprecation policy for MySQL, see [Amazon RDS FAQs](#).

MySQL Features Not Supported By Amazon RDS

Amazon RDS doesn't currently support the following MySQL features:

- Authentication Plugin
- Error Logging to the System Log
- Group Replication Plugin
- InnoDB Tablespace Encryption
- MariaDB Audit Plugin (not supported for Amazon RDS MySQL version 8.0 only)

The MariaDB Audit Plugin is supported for Amazon RDS MySQL version 5.5, 5.6, and 5.7.

- Password Strength Plugin
- Persisted system variables
- Replication filters
- Semisynchronous replication
- Transportable tablespace
- X Plugin

Note

Global transaction IDs are supported for MySQL 5.7.23 and later MySQL 5.7 versions. Global transaction IDs are not supported for Amazon RDS MySQL 5.5, 5.6, or 8.0.

IAM database authentication is supported for MySQL 5.6 and 5.7. IAM database authentication is not supported for MySQL 5.5 or 8.0.

Amazon RDS Performance Insights is supported for MySQL 5.6 and 5.7. Amazon RDS Performance Insights is not supported for MySQL 5.5 or 8.0.

To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances. It also restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application. Amazon RDS doesn't allow direct host access to a DB instance by using Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection. When you create a DB instance, you are assigned to the *db_owner* role for all databases on that instance, and you have all database-level permissions except for those used for backups. Amazon RDS manages backups for you.

Supported Storage Engines for MySQL on Amazon RDS

While MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for recovery and data durability. Amazon RDS fully supports the InnoDB storage engine for MySQL DB instances. Amazon RDS features such as Point-In-Time restore and snapshot restore require a recoverable storage engine and are supported for the InnoDB storage engine only. You must be running an instance of MySQL 5.6 or later to use the InnoDB memcached interface. For more information, see [MySQL memcached Support \(p. 681\)](#).

The Federated Storage Engine is currently not supported by Amazon RDS for MySQL.

For user-created schemas, the MyISAM storage engine does not support reliable recovery and can result in lost or corrupt data when MySQL is restarted after a recovery, preventing Point-In-Time restore or snapshot restore from working as intended. However, if you still choose to use MyISAM with Amazon RDS, snapshots can be helpful under some conditions.

Note

System tables in the mysql schema can be in MyISAM storage.

If you want to convert existing MyISAM tables to InnoDB tables, you can use the `ALTER TABLE` command (for example, `alter table TABLE_NAME engine=innodb;`). Bear in mind that MyISAM and InnoDB have different strengths and weaknesses, so you should fully evaluate the impact of making this switch on your applications before doing so.

MySQL 5.1 is no longer supported in Amazon RDS. However, you can restore existing MySQL 5.1 snapshots. When you restore a MySQL 5.1 snapshot, the instance is automatically upgraded to MySQL 5.5.

MySQL Security on Amazon RDS

Security for Amazon RDS MySQL DB instances is managed at three levels:

- AWS Identity and Access Management controls who can perform Amazon RDS management actions on DB instances. When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations. For more information, see [Authentication and Access Control \(p. 342\)](#).
- When you create a DB instance, you use either a VPC security group or a DB security group to control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance. These connections can be made using Secure Sockets Layer (SSL). In addition, firewall rules at your company can control whether devices running at your company can open connections to the DB instance.
- To authenticate login and permissions for a MySQL DB instance, you can take either of the following approaches, or a combination of them.

You can take the same approach as with a stand-alone instance of MySQL. Commands such as `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE`, and `SET PASSWORD` work just as they do in on-premises databases, as does directly modifying database schema tables. For information, see [MySQL User Account Management](#) in the MySQL documentation.

You can also use IAM database authentication. With IAM database authentication, you authenticate to your DB instance by using an IAM user or IAM role and an authentication token. An *authentication token* is a unique value that is generated using the Signature Version 4 signing process. By using IAM database authentication, you can use the same credentials to control access to your AWS

resources and your databases. For more information, see [IAM Database Authentication for MySQL and PostgreSQL \(p. 372\)](#).

When you create an Amazon RDS DB instance, the master user has the following default privileges:

- alter
- alter routine
- create
- create routine
- create temporary tables
- create user
- create view
- delete
- drop
- event
- execute
- grant option
- index
- insert
- lock tables
- process
- references
- replication client
- replication slave (MySQL 5.6 and later)
- select
- show databases
- show view
- trigger
- update

Note

Although it is possible to delete the master user on the DB instance, it is not recommended. To recreate the master user, use the [ModifyDBInstance](#) RDS API action or the [modify-db-instance](#) AWS CLI command and specify a new master user password with the appropriate parameter. If the master user does not exist in the instance, the master user is created with the specified password.

To provide management services for each DB instance, the `rdsadmin` user is created when the DB instance is created. Attempting to drop, rename, change the password, or change privileges for the `rdsadmin` account will result in an error.

To allow management of the DB instance, the standard `kill` and `kill_query` commands have been restricted. The Amazon RDS commands `rds_kill` and `rds_kill_query` are provided to allow you to terminate user sessions or queries on DB instances.

Using SSL with a MySQL DB Instance

Amazon RDS supports Secure Sockets Layer (SSL) connections with DB instances running the MySQL database engine.

Amazon RDS creates an SSL certificate and installs the certificate on the DB instance when Amazon RDS provisions the instance. These certificates are signed by a certificate authority. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. The public key is stored at <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>.

An SSL certificate created by Amazon RDS is the trusted root entity and should work in most cases but might fail if your application does not accept certificate chains. If your application does not accept certificate chains, you might need to use an intermediate certificate to connect to your region. For example, you must use an intermediate certificate to connect to the AWS GovCloud (US-West) region using SSL. For a list of regional intermediate certificates that you can download, see [Intermediate Certificates \(p. 393\)](#).

MySQL uses yaSSL for secure connections in the following versions:

- MySQL version 5.7.19 and earlier 5.7 versions
- MySQL version 5.6.37 and earlier 5.6 versions
- MySQL version 5.5.57 and earlier 5.5 versions

MySQL uses OpenSSL for secure connections in the following versions:

- MySQL version 8.0
- MySQL version 5.7.21 and later 5.7 versions
- MySQL version 5.6.39 and later 5.6 versions
- MySQL version 5.5.59 and later 5.5 versions

Amazon RDS for MySQL supports Transport Layer Security (TLS) versions 1.0, 1.1, and 1.2. The following table shows the TLS support for MySQL versions.

MySQL Version	TLS 1.0	TLS 1.1	TLS 1.2
MySQL 8.0	Supported	Supported	Supported
MySQL 5.7	Supported	Supported	Supported for MySQL 5.7.21 and later
MySQL 5.6	Supported	Not supported	Not supported
MySQL 5.5	Supported	Not supported	Not supported

To encrypt connections using the default `mysql` client, launch the `mysql` client using the `--ssl-ca` parameter to reference the public key, as shown in the examples following.

The following example shows how to launch the client using the `--ssl-ca` parameter for MySQL 5.7 and later.

```
mysql -h myinstance.c9akciq32.rds-us-east-1.amazonaws.com  
--ssl-ca=[full path]rds-combined-ca-bundle.pem --ssl-mode=VERIFY_IDENTITY
```

The following example shows how to launch the client using the `--ssl-ca` parameter for MySQL 5.6 and earlier.

```
mysql -h myinstance.c9akciq32.rds-us-east-1.amazonaws.com  
--ssl-ca=[full path]rds-combined-ca-bundle.pem --ssl-verify-server-cert
```

You can require SSL connections for specific users accounts. For example, you can use one of the following statements, depending on your MySQL version, to require SSL connections on the user account `encrypted_user`.

For MySQL 5.7 and later, use the following statement.

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

For MySQL 5.6 and earlier, use the following statement.

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL;
```

For more information on SSL connections with MySQL, see the [Using Encrypted Connections in the MySQL documentation](#).

Using memcached and Other Options with MySQL

Most Amazon RDS DB engines support option groups that allow you to select additional features for your DB instance. DB instances on MySQL version 5.6 and later support the `memcached` option, a simple, key-based cache. For more information about `memcached` and other options, see [Options for MySQL DB Instances \(p. 677\)](#). For more information about working with option groups, see [Working with Option Groups \(p. 156\)](#).

InnoDB Cache Warming

InnoDB cache warming can provide performance gains for your MySQL DB instance by saving the current state of the buffer pool when the DB instance is shut down, and then reloading the buffer pool from the saved information when the DB instance starts up. This bypasses the need for the buffer pool to "warm up" from normal database use and instead preloads the buffer pool with the pages for known common queries. The file that stores the saved buffer pool information only stores metadata for the pages that are in the buffer pool, and not the pages themselves. As a result, the file does not require much storage space. The file size is about 0.2 percent of the cache size. For example, for a 64 GiB cache, the cache warming file size is 128 MiB. For more information on InnoDB cache warming, see [Saving and Restoring the Buffer Pool State in the MySQL documentation](#).

MySQL on Amazon RDS supports InnoDB cache warming for MySQL version 5.6 and later. To enable InnoDB cache warming, set the `innodb_buffer_pool_dump_at_shutdown` and `innodb_buffer_pool_load_at_startup` parameters to 1 in the parameter group for your DB instance. Changing these parameter values in a parameter group will affect all MySQL DB instances that use that parameter group. To enable InnoDB cache warming for specific MySQL DB instances, you might need to create a new parameter group for those instances. For information on parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

InnoDB cache warming primarily provides a performance benefit for DB instances that use standard storage. If you use PIOPS storage, you do not commonly see a significant performance benefit.

Important

If your MySQL DB instance does not shut down normally, such as during a failover, then the buffer pool state will not be saved to disk. In this case, MySQL loads whatever buffer pool file is available when the DB instance is restarted. No harm is done, but the restored buffer pool might not reflect the most recent state of the buffer pool prior to the restart. To ensure that you have

a recent state of the buffer pool available to warm the InnoDB cache on startup, we recommend that you periodically dump the buffer pool "on demand." You can dump or load the buffer pool on demand if your DB instance is running MySQL version 5.6.19 or later.

You can create an event to dump the buffer pool automatically and on a regular interval. For example, the following statement creates an event named `periodic_buffer_pool_dump` that dumps the buffer pool every hour.

```
CREATE EVENT periodic_buffer_pool_dump
ON SCHEDULE EVERY 1 HOUR
DO CALL mysql.rds_innodb_buffer_pool_dump_now();
```

For more information on MySQL events, see [Event Syntax](#) in the MySQL documentation.

Dumping and Loading the Buffer Pool on Demand

For MySQL version 5.6.19 and later, you can save and load the InnoDB cache "on demand."

- To dump the current state of the buffer pool to disk, call the [mysql.rds_innodb_buffer_pool_dump_now \(p. 710\)](#) stored procedure.
- To load the saved state of the buffer pool from disk, call the [mysql.rds_innodb_buffer_pool_load_now \(p. 710\)](#) stored procedure.
- To cancel a load operation in progress, call the [mysql.rds_innodb_buffer_pool_load_abort \(p. 710\)](#) stored procedure.

Local Time Zone for MySQL DB Instances

By default, the time zone for an RDS MySQL DB instance is Universal Time Coordinated (UTC). You can set the time zone for your DB instance to the local time zone for your application instead.

To set the local time zone for a DB instance, set the `time_zone` parameter in the parameter group for your DB instance to one of the supported values listed later in this section. When you set the `time_zone` parameter for a parameter group, all DB instances and Read Replicas that are using that parameter group change to use the new local time zone. For information on setting parameters in a parameter group, see [Working with DB Parameter Groups \(p. 169\)](#).

After you set the local time zone, all new connections to the database reflect the change. If you have any open connections to your database when you change the local time zone, you won't see the local time zone update until after you close the connection and open a new connection.

You can set a different local time zone for a DB instance and one or more of its Read Replicas. To do this, use a different parameter group for the DB instance and the replica or replicas and set the `time_zone` parameter in each parameter group to a different local time zone.

If you are replicating across regions, then the replication master DB instance and the Read Replica use different parameter groups (parameter groups are unique to a region). To use the same local time zone for each instance, you must set the `time_zone` parameter in the instance's and Read Replica's parameter groups.

When you restore a DB instance from a DB snapshot, the local time zone is set to UTC. You can update the time zone to your local time zone after the restore is complete. If you restore a DB instance to a point in time, then the local time zone for the restored DB instance is the time zone setting from the parameter group of the restored DB instance.

You can set your local time zone to one of the following values.

Africa/Cairo	Asia/Bangkok	Australia/Darwin
--------------	--------------	------------------

Africa/Casablanca	Asia/Beirut	Australia/Hobart
Africa/Harare	Asia/Calcutta	Australia/Perth
Africa/Monrovia	Asia/Damascus	Australia/Sydney
Africa/Nairobi	Asia/Dhaka	Brazil/East
Africa/Tripoli	Asia/Irkutsk	Canada/Newfoundland
Africa/Windhoek	Asia/Jerusalem	Canada/Saskatchewan
America/Araguaina	Asia/Kabul	Europe/Amsterdam
America/Asuncion	Asia/Karachi	Europe/Athens
America/Bogota	Asia/Kathmandu	Europe/Dublin
America/Caracas	Asia/Krasnoyarsk	Europe/Helsinki
America/Chihuahua	Asia/Magadan	Europe/Istanbul
America/Cuiaba	Asia/Muscat	Europe/Kaliningrad
America/Denver	Asia/Novosibirsk	Europe/Moscow
America/Fortaleza	Asia/Riyadh	Europe/Paris
America/Guatemala	Asia/Seoul	Europe/Prague
America/Halifax	Asia/Shanghai	Europe/Sarajevo
America/Manaus	Asia/Singapore	Pacific/Auckland
America/Matamoros	Asia/Taipei	Pacific/Fiji
America/Monterrey	Asia/Tehran	Pacific/Guam
America/Montevideo	Asia/Tokyo	Pacific/Honolulu
America/Phoenix	Asia/Ulaanbaatar	Pacific/Samoa
America/Santiago	Asia/Vladivostok	US/Alaska
America/Tijuana	Asia/Yakutsk	US/Central
Asia/Amman	Asia/Yerevan	US/Eastern
Asia/Ashgabat	Atlantic/Azores	US/East-Indiana
Asia/Baghdad	Australia/Adelaide	US/Pacific
Asia/Baku	Australia/Brisbane	UTC

Known Issues and Limitations for MySQL on Amazon RDS

There are some known issues and limitations for working with MySQL on Amazon RDS. For more information, see [Known Issues and Limitations for MySQL on Amazon RDS \(p. 689\)](#).

Creating a DB Instance Running the MySQL Database Engine

The basic building block of Amazon RDS is the DB instance. The DB instance is where you create your MySQL databases.

Important

You must complete the tasks in the [Setting Up for Amazon RDS \(p. 5\)](#) section before you can create or connect to a DB instance.

For an example that walks you through the process of creating and connecting to a sample DB instance, see [Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance \(p. 28\)](#).

AWS Management Console

To launch a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the AWS Management Console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
4. Choose **Create database** to open the **Select engine** page.

Select engine

Engine options

Amazon Aurora

Amazon
Aurora

MySQL



MariaDB



PostgreSQL



Oracle

ORACLE

Microsoft SQL Server



MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 16 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.

Only enable options eligible for RDS Free Usage Tier [info](#)

Cancel

Next

5. In the **Select engine** window, choose **MySQL**, and then choose **Next**.
6. The **Choose use case** page asks if you are planning to use the DB instance you are creating for production. If you are, choose **Production - MySQL**. If you choose **Production - MySQL**, the following are preselected in a later step:
 - **Multi-AZ failover option**
 - **Provisioned IOPS storage option**
 - **Enable deletion protection option**

We recommend these features for any production environment.

7. Choose **Next** to continue. The **Specify DB details** page appears.

On the **Specify DB details** page, specify your DB instance information. For information about each setting, see [Settings for MySQL DB Instances \(p. 601\)](#).

Specify DB details

Instance specifications

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#)

DB engine

MySQL Community Edition

License model [Info](#)

general-public-license

DB engine version [Info](#)

MySQL 5.6.40



Known Issues/Limitations

Review the [Known Issues/Limitations](#) to learn about potential compatibility issues with specific database versions.

DB instance class [Info](#)

db.r4.xlarge — 4 vCPU, 30.5 GiB RAM

Multi-AZ deployment [Info](#)

Create replica in different zone

Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

No

Storage type [Info](#)

Provisioned IOPS (SSD)

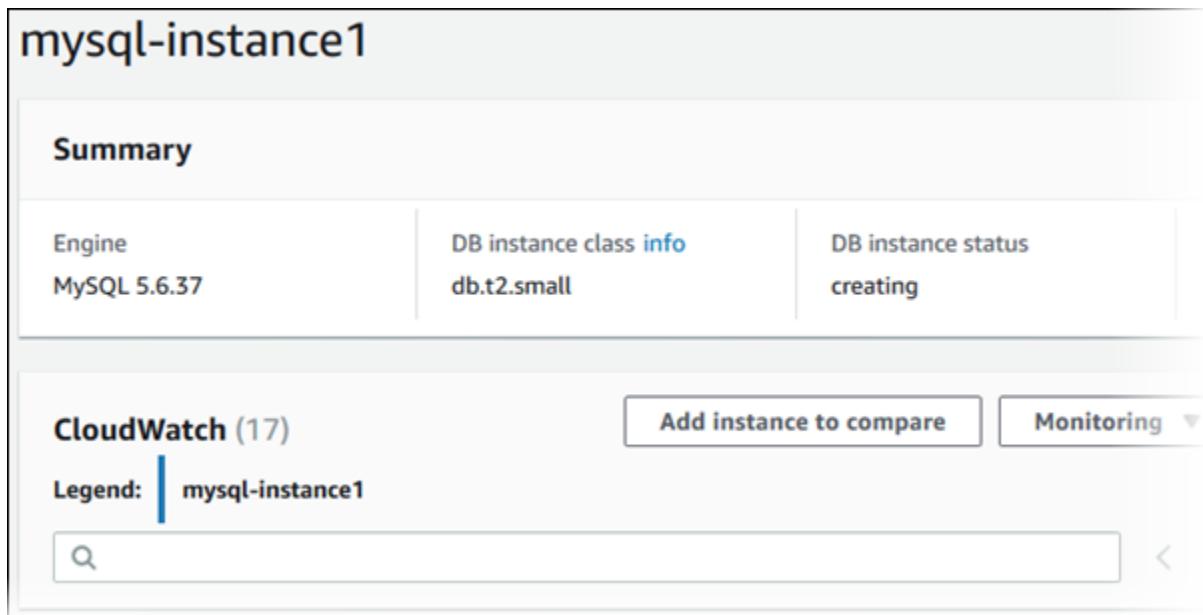
- Choose **Next** to continue. The **Configure advanced settings** page appears.

On the **Configure advanced settings** page, provide additional information that Amazon RDS needs to launch the DB instance. For information about each setting, see [Settings for MySQL DB Instances \(p. 601\)](#).

- Choose **Create database**.

- On the final page, choose **View DB instance details**.

On the RDS console, the details for the new DB instance appear. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and storage allocated, it could take several minutes for the new instance to be available.



CLI

To create a MySQL DB instance by using the AWS CLI, call the [create-db-instance](#) command with the parameters below. For information about each setting, see [Settings for MySQL DB Instances \(p. 601\)](#).

- `--db-instance-identifier`
- `--db-instance-class`
- `--db-security-groups`
- `--db-subnet-group`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--allocated-storage`
- `--backup-retention-period`

Example

The following example creates a MySQL DB instance named `mydbinstance`.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
--db-instance-identifier mydbinstance \
--db-instance-class db.m1.small \
--engine MySQL \
--allocated-storage 20 \
--master-username masterawsuser \
--master-user-password masteruserpassword \
--backup-retention-period 3
```

For Windows:

```
aws rds create-db-instance ^
```

```
--db-instance-identifier mydbinstance ^
--db-instance-class db.m3.medium ^
--engine MySQL ^
--allocated-storage 20 ^
--master-username masterawsuser ^
--master-user-password masteruserpassword ^
--backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE mydbinstance db.m3.medium mysql 20 sa creating 3 **** n 5.6.40
SECGROUP default active
PARAMGRP default.mysql5.6 in-sync
```

API

To create a MySQL DB instance by using the Amazon RDS API, call the [CreateDBInstance](#) action with the parameters below. For information about each setting, see [Settings for MySQL DB Instances \(p. 601\)](#).

- AllocatedStorage
- BackupRetentionPeriod
- DBInstanceClass
- DBInstanceIdentifier
- DBSecurityGroups
- DBSubnetGroup
- Engine
- MasterUsername
- MasterUserPassword

Example

The following example creates a MySQL DB instance named mydbinstance.

```
https://rds.us-west-2.amazonaws.com/
?Action/CreateDBInstance
&AllocatedStorage=20
&BackupRetentionPeriod=3
&DBInstanceClass=db.m3.medium
&DBInstanceIdentifier=mydbinstance
&DBName=mydatabase
&DBSecurityGroups.member.1=mymysecuritygroup
&DBSubnetGroup=mymydbsubnetgroup
&Engine=mysql
&MasterUserPassword=masteruserpassword
&MasterUsername=masterawsuser
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140213/us-west-2/rds/aws4_request
&X-Amz-Date=20140213T162136Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=8052a76dfb18469393c5f0182cdab0ebc224a9c7c5c949155376c1c250fc7ec3
```

Settings for MySQL DB Instances

The following table contains details about settings that you choose when you create a MySQL DB instance.

Setting	Setting Description
Allocated storage	The amount of storage to allocate for your DB instance (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information, see DB instance storage (p. 103) .
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.
Availability zone	The availability zone for your DB instance. Use the default value of No Preference unless you want to specify an Availability Zone. For more information, see Regions and Availability Zones (p. 101) .
Backup retention period	The number of days that you want automatic backups of your DB instance to be retained. For any non-trivial DB instance, you should set this value to 1 or greater. For more information, see Working With Backups (p. 208) .
Backup window	The time period during which Amazon RDS automatically takes a backup of your DB instance. Unless you have a specific time that you want to have your database backup, use the default of No Preference . For more information, see Working With Backups (p. 208) .
Copy tags to snapshots	Select this option to copy any DB instance tags to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .
Database name	The name for the database on your DB instance. The name must contain 1 to 64 alpha-numeric characters. If you do not provide a name, Amazon RDS does not create a database on the DB instance you are creating. To create additional databases on your DB instance, connect to your DB instance and use the SQL command <code>CREATE DATABASE</code> . For more information, see Connecting to a DB Instance Running the MySQL Database Engine (p. 606) .
Database port	The port that you want to access the DB instance through. MySQL installations default to port 3306. If you use a DB security group with your DB instance, this must be the same port value you provided when creating the DB security group.

Setting	Setting Description
	The firewalls at some companies block connections to the default MySQL port. If your company firewall blocks the default port, choose another port for your DB instance.
DB engine version	The version of MySQL that you want to use.
DB instance class	<p>The configuration for your DB instance. For example, a db.m1.small instance class equates to 1.7 GiB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.</p> <p>If possible, choose an instance class large enough that a typical query working set can be held in memory. When working sets are held in memory the system can avoid writing to disk, and this improves performance.</p> <p>For more information, see DB Instance Class (p. 82).</p>
DB instance identifier	The name for your DB instance. Your DB instance identifier can contain up to 63 alphanumeric characters, and must be unique for your account in the AWS Region you chose. You can add some intelligence to the name, such as including the AWS Region you chose, for example mysql-instance1 .
DB parameter group	<p>A parameter group for your DB instance. You can choose the default parameter group or you can create a custom parameter group.</p> <p>For more information, see Working with DB Parameter Groups (p. 169).</p>
Deletion protection	Enable deletion protection to prevent your DB instance from being deleted. If you create a production DB instance with the AWS Management Console, deletion protection is enabled by default. For more information, see Deleting a DB Instance (p. 135) .
Encryption	Enable Encryption to enable encryption at rest for this DB instance. <p>For more information, see Encrypting Amazon RDS Resources (p. 389).</p>
Enhanced monitoring	Enable enhanced monitoring to gather metrics in real time for the operating system that your DB instance runs on. <p>For more information, see Enhanced Monitoring (p. 256).</p>
IAM DB authentication	Enable IAM DB authentication to enable IAM database authentication for this DB instance. <p>For more information, see IAM Database Authentication for MySQL and PostgreSQL (p. 372).</p>
License model	MySQL has only one license model, general-public-license the general license agreement for MySQL.

Setting	Setting Description
Log exports	Select the types of MySQL database log files to generate. For more information, see MySQL Database Log Files (p. 320) .
Maintenance window	The 30 minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose No Preference . For more information, see The Amazon RDS Maintenance Window (p. 120) .
Master password	The password for your master user account. The password must contain from 8 to 16 printable ASCII characters (excluding /, ", a space, and @).
Master username	The name that you use as the master user name to log on to your DB instance. For more information, and a list of the default privileges for the master user, see MySQL Security on Amazon RDS (p. 591) .
Multi-AZ deployment	Create replica in different zone to create a passive secondary replica of your DB instance in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose No . For more information, see High Availability (Multi-AZ) for Amazon RDS (p. 109) .
Option group	An option group for your DB instance. You can choose the default option group or you can create a custom option group. For more information, see Working with Option Groups (p. 156) .
Public accessibility	Yes to give your DB instance a public IP address. This means that it is accessible outside the VPC (the DB instance also needs to be in a public subnet in the VPC). Choose No if you want the DB instance to only be accessible from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 422) .
Storage type	The storage type for your DB instance. For more information, see Amazon RDS Storage Types (p. 103) .
Subnet group	This setting depends on the platform you are on. If you are a new customer to AWS, choose default , which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC.

Setting	Setting Description
Virtual Private Cloud (VPC)	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, choose Not in VPC . For more information, see Amazon Virtual Private Cloud (VP Cs) and Amazon RDS (p. 412) .
VPC security groups	If you are a new customer to AWS, choose Create new VPC security group . Otherwise, choose Select existing VPC security groups , and select security groups you previously created. When you choose Create new VPC security group in the RDS console, a new security group is created with an inbound rule that allows access to the DB instance from the IP address detected in your browser. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 399) .

Related Topics

- [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#)
- [Connecting to a DB Instance Running the MySQL Database Engine \(p. 606\)](#)
- [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#)
- [Deleting a DB Instance \(p. 135\)](#)

Connecting to a DB Instance Running the MySQL Database Engine

Before you can connect to a DB instance running the MySQL database engine, you must create a DB instance. For information, see [Creating a DB Instance Running the MySQL Database Engine \(p. 597\)](#). Once Amazon RDS provisions your DB instance, you can use any standard MySQL client application or utility to connect to the instance. In the connection string, you specify the DNS address from the DB instance endpoint as the host parameter, and specify the port number from the DB instance endpoint as the port parameter.

To authenticate to your RDS DB instance, you can use one of the authentication methods for MySQL and IAM database authentication.

- To learn how to authenticate to MySQL using one of the authentication methods for MySQL, see [Authentication Method](#) in the MySQL documentation.
- To learn how to authenticate to MySQL using IAM database authentication, see [IAM Database Authentication for MySQL and PostgreSQL \(p. 372\)](#).

You can use the AWS Management Console, the AWS CLI `describe-db-instances` command, or the Amazon RDS API `DescribeDBInstances` action to list the details of an Amazon RDS DB instance, including its endpoint.

To find the endpoint for a MySQL DB instance in the AWS Management Console:

1. Open the RDS console and then choose **Databases** to display a list of your DB instances.
2. Choose the MySQL DB instance name to display its details.
3. On the **Connectivity** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.

mysql-instance1

Summary

DB Name	CPU
mysql-instance1	
Role	Curren
Master	

Connectivity [Monitoring](#) [Logs & events](#) [Configuration](#) [Maintenance](#)

Connectivity

Endpoint & port

Endpoint

mysql-instance1  rds.amazonaws.com

Port

3306

If an endpoint value is mysql-instance1.123456789012.us-east-1.rds.amazonaws.com and the port value is 3306, then you would specify the following values in a MySQL connection string:

- For host or host name, specify mysql-instance1.123456789012.us-east-1.rds.amazonaws.com
- For port, specify 3306

You can connect to an Amazon RDS MySQL DB instance by using tools like the MySQL command line utility. For more information on using the MySQL utility, go to [mysql - The MySQL Command Line Tool](#) in the MySQL documentation. One GUI-based application you can use to connect is MySQL Workbench. For more information, go to the [Download MySQL Workbench](#) page.

Two common causes of connection failures to a new DB instance are:

- The DB instance was created using a security group that does not authorize connections from the device or Amazon EC2 instance where the MySQL application or utility is running. If the DB instance was created in a VPC, it must have a VPC security group that authorizes the connections. If the DB instance was created outside of a VPC, it must have a DB security group that authorizes the connections.
- The DB instance was created using the default port of 3306, and your company has firewall rules blocking connections to that port from devices in your company network. To fix this failure, recreate the instance with a different port.

You can use SSL encryption on connections to an Amazon RDS MySQL DB instance. For information, see [Using SSL with a MySQL DB Instance \(p. 592\)](#). If you are using IAM database authentication, you must use an SSL connection. For information, see [IAM Database Authentication for MySQL and PostgreSQL \(p. 372\)](#).

For information on connecting to a MariaDB DB instance, see [Connecting to a DB Instance Running the MariaDB Database Engine \(p. 452\)](#).

Connecting from the MySQL Utility

To connect to a DB instance using the MySQL utility, type the following command at a command prompt to connect to a DB instance using the MySQL utility. For the -h parameter, substitute the DNS name (endpoint) for your DB instance. For the -P parameter, substitute the port for your DB instance. Enter the master user password when prompted.

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com -P 3306 -u mymasteruser -p
```

After you enter the password for the user, you will see output similar to the following.

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.6.40-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Connecting with SSL

Amazon RDS creates an SSL certificate for your DB instance when the instance is created. If you enable SSL certificate verification, then the SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. To connect to your DB instance using SSL, you can use native password authentication or IAM database authentication. To connect to your DB instance using IAM database authentication, see [IAM Database Authentication for MySQL and PostgreSQL \(p. 372\)](#). To connect to your DB instance using native password authentication, you can follow these steps:

To connect to a DB instance with SSL using the MySQL utility

1. A root certificate that works for all regions can be downloaded [here](#).
2. Enter the following command at a command prompt to connect to a DB instance with SSL using the MySQL utility. For the -h parameter, substitute the DNS name for your DB instance. For the --ssl-ca parameter, substitute the SSL certificate file name as appropriate.

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-ca-2015-root.pem -p
```

3. You can require that the SSL connection verifies the DB instance endpoint against the endpoint in the SSL certificate.

For MySQL 5.7 and later:

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-ca-2015-root.pem --ssl-mode=VERIFY_IDENTITY -p
```

For MySQL 5.6 and earlier:

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-ca-2015-root.pem --ssl-verify-server-cert -p
```

4. Enter the master user password when prompted.

You will see output similar to the following.

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.6.40-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Maximum MySQL connections

The maximum number of connections allowed to an Amazon RDS MySQL DB instance is based on the amount of memory available for the DB instance class of the DB instance. A DB instance class with more memory available will result in a larger amount of connections available. For more information on DB instance classes, see [DB Instance Class \(p. 82\)](#).

The connection limit for a DB instance is set by default to the maximum for the DB instance class for the DB instance. You can limit the number of concurrent connections to any value up to the maximum number of connections allowed using the `max_connections` parameter in the parameter group for the DB instance. For more information, see [Working with DB Parameter Groups \(p. 169\)](#).

You can retrieve the maximum number of connections allowed for an Amazon RDS MySQL DB instance by executing the following query on your DB instance:

```
SELECT @@max_connections;
```

You can retrieve the number of active connections to an Amazon RDS MySQL DB instance by executing the following query on your DB instance:

```
SHOW STATUS WHERE `variable_name` = 'Threads_connected';
```

Modifying a DB Instance Running the MySQL Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS MySQL DB instance, and describes the settings for MySQL instances.

We recommend that you test any changes on a test instance before modifying a production instance, so that you fully understand the impact of each change. This is especially important when upgrading database versions.

After you modify your DB instance settings, you can apply the changes immediately, or apply them during the next maintenance window for the DB instance. Some modifications cause an interruption by restarting the DB instance.

Note

When you modify a DB instance, Amazon RDS will reboot the instance if both of the following are true:

- You change the DB instance class.
- You specify a custom parameter group.

AWS Management Console

To modify a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want. For information about each setting, see [Settings for MySQL DB Instances \(p. 611\)](#).
5. When all the changes are as you want them, choose **Continue** and check the summary of modifications.
6. To apply the changes immediately, choose **Apply immediately**. Choosing this option can cause an outage in some cases. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify a MySQL DB instance by using the AWS CLI, call the `modify-db-instance` command. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see [Settings for MySQL DB Instances \(p. 611\)](#).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days). The code enables automatic minor version upgrades by using `--auto-minor-version-upgrade`. To disable automatic minor version upgrades, use `--no-auto-minor-version-upgrade`. The

changes are applied during the next maintenance window by using `--no-apply-immediately`. Use `--apply-immediately` to apply the changes immediately. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --backup-retention-period 7 \
    --auto-minor-version-upgrade \
    --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --backup-retention-period 7 ^
    --auto-minor-version-upgrade ^
    --no-apply-immediately
```

API

To modify a MySQL instance by using the Amazon RDS API, call the [ModifyDBInstance](#) action. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see [Settings for MySQL DB Instances \(p. 611\)](#).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days) and enabling automatic minor version upgrades. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
?Action=ModifyDBInstance
&ApplyImmediately=false
&AutoMinorVersionUpgrade=true
&BackupRetentionPeriod=7
&DBInstanceIdentifier=mydbinstance
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
&X-Amz-Date=20131016T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for MySQL DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated storage	The storage, in gigabytes, that you want to allocate for your DB instance.	If Apply immediately is set to true, the change occurs immediately.	No downtime. Performance may be

Setting	Setting Description	When the Change Occurs	Downtime Notes
	For more information, see DB instance storage (p. 103) .	If Apply immediately is set to false, the change occurs during the next maintenance window.	degraded during the change.
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.	–	–
Backup retention period	The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0. For more information, see Working With Backups (p. 208) .	If Apply immediately is set to true, the change occurs immediately. If Apply immediately is set to false and you change the setting from a non-zero value to another non-zero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.	An outage occurs if you change from 0 to a non-zero value, or from a non-zero value to 0.
Backup window	The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours. For more information, see Working With Backups (p. 208) .	The change is applied asynchronously, as soon as possible.	–
Certificate Authority	The certificate that you want to use.	–	–
Copy tags to snapshots	If you have any DB instance tags, this option copies them when you create a DB snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .	The change occurs immediately. This setting ignores the Apply immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Database port	<p>The port that you want to use to access the database.</p> <p>The port value must not match any of the port values specified for options in the option group for the DB instance.</p>	The change occurs immediately. This setting ignores the Apply immediately setting.	The DB instance is rebooted immediately.
DB engine version	<p>The version of the MySQL database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications.</p> <p>For more information, see Upgrading the MySQL DB Engine (p. 618).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change.
DB instance class	<p>The DB instance class that you want to use.</p> <p>For more information, see DB Instance Class (p. 82).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change.
DB instance identifier	<p>The DB instance identifier. This value is stored as a lowercase string.</p> <p>For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 126).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change. The DB instance is rebooted.
DB parameter group	<p>The parameter group that you want associated with the DB instance.</p> <p>For more information, see Working with DB Parameter Groups (p. 169).</p>	The parameter group change occurs immediately.	<p>An outage doesn't occur during this change. When you change the parameter group, changes to some parameters are applied to the DB instance immediately without a reboot. Changes to other parameters are applied only after the DB instance is rebooted.</p> <p>For more information, see Rebooting a DB Instance (p. 129).</p>

Setting	Setting Description	When the Change Occurs	Downtime Notes
Deletion protection	Enable deletion protection to prevent your DB instance from being deleted. For more information, see Deleting a DB Instance (p. 135) .	–	–
Enhanced monitoring	Enable enhanced monitoring to enable gathering metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 256) .	–	–
IAM DB authentication	Enable IAM DB authentication to enable IAM database authentication for this DB instance. For more information, see IAM Database Authentication for MySQL and PostgreSQL (p. 372) .	–	–
Log exports	Select the types of MySQL database log files to publish to Amazon CloudWatch Logs. For more information, see MySQL Database Log Files (p. 320) .	The change occurs immediately. This setting ignores the Apply immediately setting.	–
Maintenance window	The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours. If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied. For more information, see The Amazon RDS Maintenance Window (p. 120) .	The change occurs immediately. This setting ignores the Apply immediately setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.
Multi-AZ deployment	Yes to deploy your DB instance in multiple Availability Zones; otherwise, No . For more information, see Regions and Availability Zones (p. 101) .	If Apply immediately is set to true, the change occurs immediately. If Apply immediately is set to false, the change occurs during the next maintenance window.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
New master password	The password for your master user. The password must contain from 8 to 41 alphanumeric characters.	The change is applied asynchronously, as soon as possible. This setting ignores the Apply immediately setting.	–
Option group	The option group that you want associated with the DB instance. For more information, see Working with Option Groups (p. 156) .	If Apply immediately is set to true, the change occurs immediately. If Apply immediately is set to false, the change occurs during the next maintenance window.	–
Public accessibility	Yes to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. No to make the DB instance accessible only from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 422) .	The change occurs immediately. This setting ignores the Apply immediately setting.	–
Security group	The security group you want associated with the DB instance. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 399) .	The change is applied asynchronously, as soon as possible. This setting ignores the Apply immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Storage type	<p>The storage type that you want to use.</p> <p>For more information, see Amazon RDS Storage Types (p. 103).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	<p>The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place.</p> <ul style="list-style-type: none"> From General Purpose (SSD) to Magnetic. From General Purpose (SSD) to Provisioned IOPS (SSD), if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ DB instance or for the source DB instance of a read replica. From Magnetic to General Purpose (SSD). From Magnetic to Provisioned IOPS (SSD). From Provisioned IOPS (SSD) to Magnetic. From Provisioned IOPS (SSD) to General Purpose (SSD), if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ

Setting	Setting Description	When the Change Occurs	Downtime Notes
			DB instance or for the source DB instance of a read replica.
Subnet group	<p>The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC.</p> <p>For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 426).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change. The DB instance is rebooted.

Related Topics

- [Rebooting a DB Instance \(p. 129\)](#)
- [Connecting to a DB Instance Running the MySQL Database Engine \(p. 606\)](#)
- [Upgrading the MySQL DB Engine \(p. 618\)](#)
- [Deleting a DB Instance \(p. 135\)](#)

Upgrading the MySQL DB Engine

When Amazon RDS supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades: major version upgrades and minor version upgrades. In general, a major engine version upgrade can introduce changes that are not compatible with existing applications. In contrast, a minor version upgrade includes only changes that are backward-compatible with existing applications.

You must modify the DB instance manually to perform a major version upgrade. Minor version upgrades occur automatically if you enable auto minor version upgrades on your DB instance. In all other cases, you must modify the DB instance manually to perform a minor version upgrade.

Topics

- [Overview of Upgrading \(p. 618\)](#)
- [Major Version Upgrades for MySQL \(p. 618\)](#)
- [Testing an Upgrade \(p. 620\)](#)
- [Upgrading a MySQL DB Instance \(p. 621\)](#)
- [Upgrading a MySQL Database with Reduced Downtime \(p. 621\)](#)

Overview of Upgrading

Amazon RDS takes two DB snapshots during the upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken when the upgrade completes.

Note

Amazon RDS only takes DB snapshots if you have set the backup retention period for your DB instance to a number greater than 0. To change your backup retention period, see [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#).

After the upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the first DB snapshot taken to create a new DB instance.

You control when to upgrade your DB instance to a new version supported by Amazon RDS. This level of control helps you maintain compatibility with specific database versions and test new versions with your application before deploying in production. When you are ready, you can perform version upgrades at the times that best fit your schedule.

If your DB instance is using read replication, you must upgrade all of the Read Replicas before upgrading the source instance.

If your DB instance is in a Multi-AZ deployment, both the primary and standby DB instances are upgraded. The primary and standby DB instances are upgraded at the same time and you will experience an outage until the upgrade is complete. The time for the outage varies based on the size of your DB instance.

Major Version Upgrades for MySQL

Amazon RDS supports the following in-place upgrades for major versions of the MySQL database engine:

- MySQL 5.5 to MySQL 5.6
- MySQL 5.6 to MySQL 5.7

- MySQL 5.7 to MySQL 8.0

Note

You can only create MySQL version 5.7 and 8.0 DB instances with latest-generation and current-generation DB instance classes, in addition to the db.m3 previous-generation DB instance class. If you want to upgrade a MySQL version 5.6 DB instance running on a previous-generation DB instance class (other than db.m3) to a MySQL version 5.7 DB instance, you must first modify the DB instance to use a latest-generation or current-generation DB instance class. After the DB instance has been modified to use a latest-generation or current-generation DB instance class, you can then modify the DB instance to use the MySQL version 5.7 database engine. For information on Amazon RDS DB instance classes, see [DB Instance Class \(p. 82\)](#).

Major version upgrades can contain database changes that are not backward-compatible with existing applications. As a result, Amazon RDS doesn't apply major version upgrades automatically; you must manually modify your DB instance. You should thoroughly test any upgrade before applying it to your production instances.

To perform a major version upgrade for a MySQL version 5.5 DB instance on Amazon RDS to MySQL version 5.6 or later, you should first perform any available OS updates. After OS updates are complete, you must upgrade to each major version: 5.5 to 5.6, then 5.6 to 5.7, and then 5.7 to 8.0. MySQL DB instances created before April 24, 2014, show an available OS update until the update has been applied. For more information on OS updates, see [Applying Updates for a DB Instance \(p. 118\)](#).

During a major version upgrade of MySQL, Amazon RDS runs the MySQL binary `mysql_upgrade` to upgrade tables, if required. Also, Amazon RDS empties the `slow_log` and `general_log` tables during a major version upgrade. To preserve log information, save the log contents before the major version upgrade.

MySQL major version upgrades typically complete in about 10 minutes. Some upgrades might take longer because of the DB instance class size or because the instance doesn't follow certain operational guidelines in [Best Practices for Amazon RDS \(p. 70\)](#). If you upgrade a DB instance from the Amazon RDS console, the status of the DB instance indicates when the upgrade is complete. If you upgrade using the AWS Command Line Interface (AWS CLI), use the `describe-db-instances` command and check the `Status` value.

If you are using a custom parameter group, you must specify either a default parameter group for the new DB engine version or create your own custom parameter group for the new DB engine version. Associating the new parameter group with the DB instance requires a customer-initiated database reboot after the upgrade completes. The DB instance's parameter group status shows `pending-reboot` if the DB instance needs to be rebooted to apply the parameter group changes. A DB instance's parameter group status can be viewed in the AWS console or by using a "describe" call such as `describe-db-instances`.

Upgrades to MySQL Version 5.7 Might Be Slow

MySQL version 5.6.4 introduced a new date and time format for the `datetime`, `time`, and `timestamp` columns that allows fractional components in date and time values. When upgrading a DB instance to MySQL version 5.7, MySQL will force the conversion of all date and time column types to the new format. Because this conversion rebuilds your tables, it might take a considerable amount of time to complete the DB instance upgrade. The forced conversion will occur for any DB instances that are running a version prior to MySQL version 5.6.4, and also any DB instances that were upgraded from a version prior to MySQL version 5.6.4 to a version other than 5.7.

If your DB instance is running a version prior to MySQL version 5.6.4, or was upgraded from a version prior to MySQL version 5.6.4, then we recommend that you convert the `datetime`, `time`, and `timestamp` columns in your database before upgrading your DB instance to MySQL version 5.7. This conversion can significantly reduce the amount of time required to upgrade the DB instance to MySQL

version 5.7. To upgrade your date and time columns to the new format, issue the `ALTER TABLE <table_name> FORCE;` command for each table that contains date or time columns. Because altering a table locks the table as read-only, we recommend that you perform this update during a maintenance window.

You can use the following query to find all tables in your database that have columns of type datetime, time, or timestamp and to create an `ALTER TABLE <table_name> FORCE;` command for each table:

```
SELECT DISTINCT CONCAT('ALTER TABLE `',
    REPLACE(is_tables.TABLE_SCHEMA, '`', ``), ``,
    REPLACE(is_tables.TABLE_NAME, '`', ``), `` FORCE;`)
FROM information_schema.TABLES is_tables
INNER JOIN information_schema.COLUMNS col ON col.TABLE_SCHEMA =
is_tables.TABLE_SCHEMA
    AND col.TABLE_NAME = is_tables.TABLE_NAME
LEFT OUTER JOIN information_schema.INNODB_SYS_TABLES systables ON
    SUBSTRING_INDEX(systables.NAME, '#', 1) =
CONCAT(is_tables.TABLE_SCHEMA,'/',is_tables.TABLE_NAME)
    LEFT OUTER JOIN information_schema.INNODB_SYS_COLUMNS syscolumns ON
        syscolumns.TABLE_ID = systables.TABLE_ID AND syscolumns.NAME = col.COLUMN_NAME
WHERE col.COLUMN_TYPE IN ('time','timestamp','datetime')
    AND is_tables.TABLE_TYPE = 'BASE TABLE'
    AND is_tables.TABLE_SCHEMA NOT IN ('mysql','information_schema','performance_schema')
    AND (is_tables.ENGINE = 'InnoDB' AND syscolumns.MTYPE = 6);
```

Testing an Upgrade

Before you perform a major version upgrade on your DB instance, you should thoroughly test your database, and all applications that access the database, for compatibility with the new version. We recommend that you use the following procedure.

To test a major version upgrade

1. Review the upgrade documentation for the new version of the database engine to see if there are compatibility issues that might affect your database or applications:
 - [Changes in MySQL 5.6](#)
 - [Changes in MySQL 5.7](#)
 - [Changes in MySQL 8.0](#)
2. If your DB instance is a member of a custom DB parameter group, you need to create a new DB parameter group with your existing settings that is compatible with the new major version. Specify the new DB parameter group when you upgrade your test instance, so that your upgrade testing ensures that it works correctly. For more information about creating a DB parameter group, see [Working with DB Parameter Groups \(p. 169\)](#).
3. Create a DB snapshot of the DB instance to be upgraded. For more information, see [Creating a DB Snapshot \(p. 216\)](#).
4. Restore the DB snapshot to create a new test DB instance. For more information, see [Restoring from a DB Snapshot \(p. 218\)](#).
5. Modify this new test DB instance to upgrade it to the new version, using one of the methods detailed following. If you created a new parameter group in step 2, specify that parameter group.
6. Evaluate the storage used by the upgraded instance to determine if the upgrade requires additional storage.
7. Run as many of your quality assurance tests against the upgraded DB instance as needed to ensure that your database and application work correctly with the new version. Implement any new tests needed to evaluate the impact of any compatibility issues you identified in step 1. Test all stored procedures and functions. Direct test versions of your applications to the upgraded DB instance.

8. If all tests pass, then perform the upgrade on your production DB instance. We recommend that you do not allow write operations to the DB instance until you confirm that everything is working correctly.

Upgrading a MySQL DB Instance

For information about manually or automatically upgrading a MySQL DB instance, see [Upgrading a DB Instance Engine Version \(p. 123\)](#).

Upgrading a MySQL Database with Reduced Downtime

If your MySQL DB instance is currently in use with a production application, you can use the following procedure to upgrade the database version for your DB instance and reduce the amount of downtime for your application. This procedure shows an example of upgrading from MySQL version 5.5 to MySQL version 5.6. You can use the same general steps for upgrades to other major versions.

To upgrade an MySQL database while a DB instance is in use

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. Create a Read Replica of your MySQL 5.5 DB instance. This process creates an upgradable copy of your database.
 - a. On the console, choose **Databases**, and then choose the DB instance that you want to upgrade.
 - b. For **Actions**, choose **Create read replica**.
 - c. Provide a value for **DB instance identifier** for your Read Replica and ensure that the **DB instance class** and other settings match your MySQL 5.5 DB instance.
 - d. Choose **Create read replica**.
3. When the Read Replica has been created and **Status** shows **available**, upgrade the Read Replica to MySQL 5.6:
 - a. On the console, choose **Databases**, and then choose the Read Replica that you just created.
 - b. Choose **Modify**.
 - c. For **DB engine version**, choose the MySQL 5.6 version to upgrade to, and then choose **Continue**.
 - d. For **Scheduling of Modifications**, choose **Apply immediately**.
 - e. Choose **Modify DB instance** to start the upgrade.
4. When the upgrade is complete and **Status** shows **available**, verify that the upgraded Read Replica is up to date with the master MySQL 5.5 DB instance. You can do this by connecting to the Read Replica and issuing the `SHOW SLAVE STATUS` command. If the `Seconds_Behind_Master` field is 0, then replication is up to date.
5. Make your MySQL 5.6 Read Replica a master DB instance.

Important

When you promote your MySQL 5.6 Read Replica to a standalone, single-AZ DB instance, it no longer is a replication replica to your MySQL 5.5 DB instance. We recommend that you promote your MySQL 5.6 Read Replica during a maintenance window when your source MySQL 5.5 DB instance is in read-only mode and all write operations are suspended. When the promotion is completed, you can direct your write operations to the upgraded MySQL 5.6 DB instance to ensure that no write operations are lost.

In addition, we recommend that before promoting your MySQL 5.6 Read Replica you perform all necessary data definition language (DDL) operations, such as creating indexes,

on the MySQL 5.6 Read Replica. This approach avoids negative effects on the performance of the MySQL 5.6 Read Replica after it has been promoted. To promote a Read Replica, use the following procedure.

- a. On the console, choose **Databases**, and then choose the Read Replica that you just upgraded.
- b. Choose **Actions**, and then choose **Promote read replica**.
- c. Choose **Yes** to enable automated backups for the Read Replica instance. For more information, see [Working With Backups \(p. 208\)](#).

Choose **Continue**.

- d. Choose **Promote Read Replica**.
6. You now have an upgraded version of your MySQL database. At this point, you can direct your applications to the new MySQL 5.6 DB instance, add Read Replicas, set up Multi-AZ support, and so on.

Upgrading a MySQL DB Snapshot

With Amazon RDS, you can create a storage volume DB snapshot of your MySQL DB instance. When you create a DB snapshot, the snapshot is based on the engine version used by your Amazon RDS instance. In addition to upgrading the DB engine version of your DB instance, you can also upgrade the engine version for your DB snapshots. For example, you can upgrade DB snapshots created from the MySQL 5.1 engine to DB snapshots for the MySQL 5.5 engine. After restoring a DB snapshot upgraded to a new engine version, you should test that the upgrade was successful. To learn how to test a major version upgrade, see [Testing an Upgrade \(p. 620\)](#). To learn how to restore a DB snapshot, see [Restoring from a DB Snapshot \(p. 218\)](#).

Amazon RDS supports upgrading a MySQL DB snapshot from MySQL 5.1 to MySQL 5.5.

Upgrading a MySQL DB Snapshot

You can upgrade manual DB snapshots, which can be encrypted or not encrypted, from MySQL 5.1 to MySQL 5.5 within the same AWS Region. You can't upgrade automated DB snapshots that are created during the automated backup process.

AWS Management Console

To upgrade a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Snapshots**.
3. For **Actions**, choose **Modify Snapshot**. The **Modify DB Snapshot** page appears.
4. Choose **Modify Snapshot** to upgrade the snapshot. During the upgrade process, all snapshot actions are disabled. Also, the DB snapshot status changes from **available** to **upgrading**, and then changes to **active** upon completion. If the DB snapshot can't be upgraded because of snapshot corruption issues, the status changes to **unavailable**. You can't recover the snapshot from this state.

AWS CLI

To upgrade a DB snapshot to a new database engine version, use the AWS CLI `modify-db-snapshot` command.

Parameters

- `--db-snapshot-identifier` – The identifier of the DB snapshot to upgrade. The identifier must be a unique Amazon Resource Name (ARN). For more information, see [Working with Amazon Resource Names \(ARNs\) in Amazon RDS \(p. 181\)](#).
- `--engine-version` – The engine version to upgrade the DB snapshot to.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-snapshot \
--db-snapshot-identifier <mydbsnapshot> \
--engine-version <new_version>
```

For Windows:

```
aws rds modify-db-snapshot ^
--db-snapshot-identifier <mydbsnapshot> ^
--engine-version <new_version>
```

API

To upgrade a DB snapshot to a new database engine version, call the Amazon RDS API [ModifyDBSnapshot](#) action.

- **DBSnapshotIdentifier** – The identifier of the DB snapshot to upgrade. The identifier must be a unique Amazon Resource Name (ARN). For more information, see [Working with Amazon Resource Names \(ARNs\) in Amazon RDS \(p. 181\)](#).
- **EngineVersion** – The engine version to upgrade the DB snapshot to.

Example

```
https://rds.us-west-2.amazonaws.com/
?Action=ModifyDBSnapshot
&DBSnapshotIdentifier=mydbsnapshot
&EngineVersion=newversion
&SignatureVersion=4
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20161222/us-west-1/rds/aws4_request
&X-Amz-Date=20161222T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=8052a76dfb18469393c5f0182cdab0ebc224a9c7c5c949155376c1c250fc7ec3
```

Importing Data into a MySQL DB Instance

You can use several different techniques to import data into an Amazon RDS for MySQL DB instance. The best approach depends on the source of the data, the amount of data, and whether the import is done one time or is ongoing. If you are migrating an application along with the data, also consider the amount of downtime that you are willing to experience.

Overview

Find techniques to import data into an Amazon RDS for MySQL DB instance in the following table.

Source	Amount of Data	One Time or Ongoing	Application Downtime	Technique	More Information
Existing MySQL database on premises or on Amazon EC2	Any	One time	Some	Create a backup of your on-premises database, store it on Amazon S3, and then restore the backup file to a new Amazon RDS DB instance running MySQL.	Restoring a Backup into an Amazon RDS MySQL DB Instance (p. 631)
Any existing database	Any	One time or ongoing	Minimal	Use AWS Database Migration Service to migrate the database with minimal downtime and, for many database DB engines, continue ongoing replication.	What is AWS Database Migration Service in the AWS Database Migration Service User Guide
Existing Amazon RDS MySQL DB instance	Any	One time	Minimal	Create a Read Replica, and then promote the Read Replica.	Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances (p. 143)
Existing MySQL or MariaDB database	Small	One time	Some	Copy the data directly to your Amazon RDS MySQL DB instance using a command-line utility.	Importing Data from a MySQL or MariaDB DB to an Amazon

Source	Amount of Data	One Time or Ongoing	Application Downtime	Technique	More Information
					RDS MySQL or MariaDB DB Instance (p. 638)
Data not stored in an existing database	Medium	One time	Some	Create flat files and import them using the <code>mysqlimport</code> utility.	Importing Data From Any Source to a MySQL or MariaDB DB Instance (p. 652)
Existing MySQL or MariaDB database on premises or on Amazon EC2	Any	Ongoing	Minimal	Configure replication with an existing MySQL database as the replication source.	Replication with a MySQL or MariaDB Instance Running External to Amazon RDS (p. 667) or Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 640)

Note

The '`mysql`' system database contains authentication and authorization information required to log in to your DB instance and access your data. Dropping, altering, renaming, or truncating tables, data, or other contents of the '`mysql`' database in your DB instance can result in error and might render the DB instance and your data inaccessible. If this occurs, you can restore the DB instance from a snapshot using the AWS CLI `restore-db-instance-from-db-snapshot` command. You can recover the DB instance using the AWS CLI `restore-db-instance-to-point-in-time` command.

Importing Data Considerations

Following, you can find additional technical information related to loading data into MySQL. This information is intended for advanced users who are familiar with the MySQL server architecture. All comments related to LOAD DATA LOCAL INFILE also apply to `mysqlimport`.

Binary Log

Data loads incur a performance penalty and require additional free disk space (up to four times more) when binary logging is enabled versus loading the same data with binary logging turned off. The severity of the performance penalty and the amount of free disk space required is directly proportional to the size of the transactions used to load the data.

Transaction Size

Transaction size plays an important role in MySQL data loads. It has a major influence on resource consumption, disk space utilization, resume process, time to recover, and input format (flat files or SQL). This section describes how transaction size affects binary logging and makes the case for disabling binary logging during large data loads. As noted earlier, binary logging is enabled and disabled by setting the Amazon RDS automated backup retention period. Non-zero values enable binary logging, and zero disables it. We also describe the impact of large transactions on InnoDB and why it's important to keep transaction sizes small.

Small Transactions

For small transactions, binary logging doubles the number of disk writes required to load the data. This effect can severely degrade performance for other database sessions and increase the time required to load the data. The degradation experienced depends in part upon the upload rate, other database activity taking place during the load, and the capacity of your Amazon RDS DB instance.

The binary logs also consume disk space roughly equal to the amount of data loaded until they are backed up and removed. Fortunately, Amazon RDS minimizes this by backing up and removing binary logs on a frequent basis.

Large Transactions

Large transactions incur a 3X penalty for IOPS and disk consumption with binary logging enabled. This is due to the binary log cache spilling to disk, consuming disk space and incurring additional IO for each write. The cache cannot be written to the binlog until the transaction commits or rolls back, so it consumes disk space in proportion to the amount of data loaded. When the transaction commits, the cache must be copied to the binlog, creating a third copy of the data on disk.

Because of this, there must be at least three times as much free disk space available to load the data compared to loading with binary logging disabled. For example, 10 GiB of data loaded as a single transaction consumes at least 30 GiB disk space during the load. It consumes 10 GiB for the table + 10 GiB for the binary log cache + 10 GiB for the binary log itself. The cache file remains on disk until the session that created it terminates or the session fills its binary log cache again during another transaction. The binary log must remain on disk until backed up, so it might be some time before the extra 20 GiB is freed.

If the data was loaded using LOAD DATA LOCAL INFILE, yet another copy of the data is created if the database has to be recovered from a backup made before the load. During recovery, MySQL extracts the data from the binary log into a flat file. MySQL then executes LOAD DATA LOCAL INFILE, just as in the original transaction. However, this time the input file is local to the database server. Continuing with the example preceding, recovery fails unless there is at least 40 GiB free disk space available.

Disable Binary Logging

Whenever possible, disable binary logging during large data loads to avoid the resource overhead and addition disk space requirements. In Amazon RDS, disabling binary logging is as simple as setting the backup retention period to zero. If you do this, we recommend that you take a DB snapshot of the database instance immediately before the load. By doing this, you can quickly and easily undo changes made during loading if you need to.

After the load, set the backup retention period back to an appropriate (no zero) value.

You can't set the backup retention period to zero if the DB instance is a source DB instance for Read Replicas.

InnoDB

The information in this section provides a strong argument for keeping transaction sizes small when using InnoDB.

Undo

InnoDB generates undo to support features such as transaction rollback and MVCC. Undo is stored in the InnoDB system tablespace (usually `ibdata1`) and is retained until removed by the purge thread. The purge thread cannot advance beyond the undo of the oldest active transaction, so it is effectively blocked until the transaction commits or completes a rollback. If the database is processing other transactions during the load, their undo also accumulates in the system tablespace and cannot be removed even if they commit and no other transaction needs the undo for MVCC. In this situation, all transactions (including read-only transactions) that access any of the rows changed by any transaction (not just the load transaction) slow down. The slowdown occurs because transactions scan through undo that could have been purged if not for the long-running load transaction.

Undo is stored in the system tablespace, and the system tablespace never shrinks in size. Thus, large data load transactions can cause the system tablespace to become quite large, consuming disk space that you can't reclaim without recreating the database from scratch.

Rollback

InnoDB is optimized for commits. Rolling back a large transaction can take a very, very long time. In some cases, it might be faster to perform a point-in-time recovery or restore a DB snapshot.

Input Data Format

MySQL can accept incoming data in one of two forms: flat files and SQL. This section points out some key advantages and disadvantages of each.

Flat Files

Loading flat files with `LOAD DATA LOCAL INFILE` can be the fastest and least costly method of loading data as long as transactions are kept relatively small. Compared to loading the same data with SQL, flat files usually require less network traffic, lowering transmission costs and load much faster due to the reduced overhead in the database.

One Big Transaction

`LOAD DATA LOCAL INFILE` loads the entire flat file as one transaction. This isn't necessarily a bad thing. If the size of the individual files can be kept small, this has a number of advantages:

- Resume capability – Keeping track of which files have been loaded is easy. If a problem arises during the load, you can pick up where you left off with little effort. Some data might have to be retransmitted to Amazon RDS, but with small files, the amount retransmitted is minimal.

- Load data in parallel – If you've got IOPS and network bandwidth to spare with a single file load, loading in parallel might save time.
- Throttle the load rate – Data load having a negative impact on other processes? Throttle the load by increasing the interval between files.

Be Careful

The advantages of LOAD DATA LOCAL INFILE diminish rapidly as transaction size increases. If breaking up a large set of data into smaller ones isn't an option, SQL might be the better choice.

SQL

SQL has one main advantage over flat files: it's easy to keep transaction sizes small. However, SQL can take significantly longer to load than flat files and it can be difficult to determine where to resume the load after a failure. For example, mysqldump files are not restartable. If a failure occurs while loading a mysqldump file, the file requires modification or replacement before the load can resume. The alternative is to restore to the point in time before the load and replay the file after the cause of the failure has been corrected.

Take Checkpoints Using Amazon RDS Snapshots

If you have a load that's going to take several hours or even days, loading without binary logging isn't a very attractive prospect unless you can take periodic checkpoints. This is where the Amazon RDS DB snapshot feature comes in very handy. A DB snapshot creates a point-in-time consistent copy of your database instance which can be used to restore the database to that point in time after a crash or other mishap.

To create a checkpoint, simply take a DB snapshot. Any previous DB snapshots taken for checkpoints can be removed without affecting durability or restore time.

Snapshots are fast too, so frequent checkpointing doesn't add significantly to load time.

Decreasing Load Time

Here are some additional tips to reduce load times:

- Create all secondary indexes before loading. This is counter-intuitive for those familiar with other databases. Adding or modifying a secondary index causes MySQL to create a new table with the index changes, copy the data from the existing table to the new table, and drop the original table.
- Load data in PK order. This is particularly helpful for InnoDB tables, where load times can be reduced by 75–80 percent and data file size cut in half.
- Disable foreign key constraints `foreign_key_checks=0`. For flat files loaded with LOAD DATA LOCAL INFILE, this is required in many cases. For any load, disabling FK checks provides significant performance gains. Just be sure to enable the constraints and verify the data after the load.
- Load in parallel unless already near a resource limit. Use partitioned tables when appropriate.
- Use multi-value inserts when loading with SQL to minimize statement execution overhead. When using mysqldump, this is done automatically.
- Reduce InnoDB log IO `innodb_flush_log_at_trx_commit=0`
- If you are loading data into a DB instance that does not have Read Replicas, set the `sync_binlog` parameter to 0 while loading data. When data loading is complete, set the `sync_binlog` parameter back to 1.
- Load data before converting the DB instance to a Multi-AZ deployment. However, if the DB instance already uses a Multi-AZ deployment, switching to a Single-AZ deployment for data loading is not recommended, because doing so only provides marginal improvements.

Note

Using `innodb_flush_log_at_trx_commit=0` causes InnoDB to flush its logs every second instead of at each commit. This provides a significant speed advantage, but can lead to data loss during a crash. Use with caution.

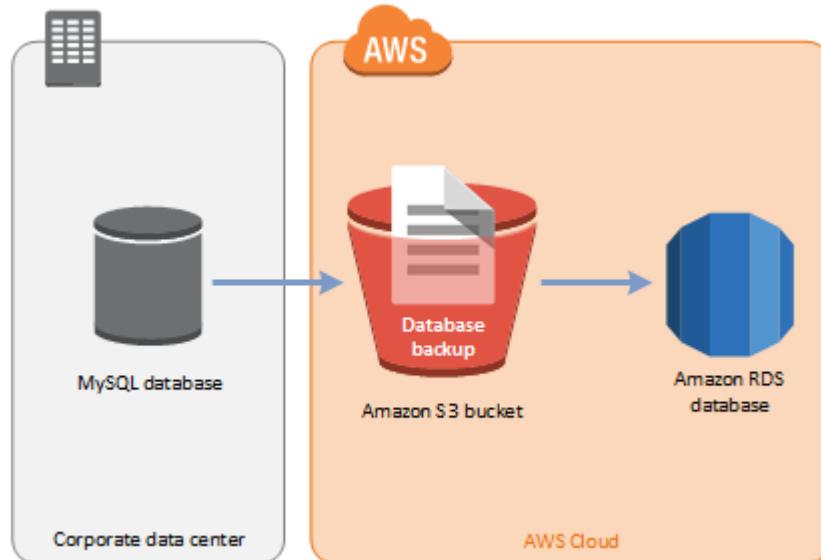
Topics

- [Restoring a Backup into an Amazon RDS MySQL DB Instance \(p. 631\)](#)
- [Importing Data from a MySQL or MariaDB DB to an Amazon RDS MySQL or MariaDB DB Instance \(p. 638\)](#)
- [Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime \(p. 640\)](#)
- [Importing Data From Any Source to a MySQL or MariaDB DB Instance \(p. 652\)](#)

Restoring a Backup into an Amazon RDS MySQL DB Instance

Amazon RDS supports importing MySQL databases by using backup files. You can create a backup of your on-premises database, store it on Amazon S3, and then restore the backup file onto a new Amazon RDS DB instance running MySQL.

You can find the supported scenario in the following diagram.



Importing backup files from Amazon S3 is supported for MySQL version 5.6. Importing backup files from Amazon S3 is available in all AWS Regions.

We recommend that you import your database to Amazon RDS by using backup files if your database can be offline while the backup file is created, copied, and restored. If your on-premises database can't be offline, you can use binlog replication to update your database after you have migrated to Amazon RDS through Amazon S3 as explained in this topic. For more information, see [Replication with a MySQL or MariaDB Instance Running External to Amazon RDS \(p. 667\)](#). You can also use the AWS Database Migration Service to migrate your database to Amazon RDS. For more information, see [What Is AWS Database Migration Service?](#)

Limitations and Recommendations for Importing Backup Files from Amazon S3 to Amazon RDS

The following are some limitations and recommendations for importing backup files from Amazon S3:

- You can only import your data to a new DB instance, not an existing DB instance.
- You must use Percona XtraBackup to create the backup of your on-premises database.
- You can't migrate from a source database that has tables defined outside of the default MySQL data directory.
- You can't import a MySQL 5.5, 5.7, or 8.0 database.
- You can't import an on-premises MySQL 5.6 database to an Amazon RDS MySQL 5.7 or 8.0 database. You can upgrade your DB instance after you complete the import.
- You can't restore databases larger than 6 TB in size.
- You can't restore from an encrypted source database, but you can restore to an encrypted Amazon RDS DB instance.

- You can't restore from an Amazon S3 bucket in a different AWS Region than your Amazon RDS DB instance.
- Importing from Amazon S3 is not supported on the db.t2.micro DB instance class. However, you can restore to a different DB instance class, and then change the instance class later. For more information about instance classes, see [Specifications for All Available DB Instance Classes \(p. 82\)](#).
- Amazon S3 limits the size of a file uploaded to an Amazon S3 bucket to 5 TB. If a backup file exceeds 5 TB, then you must split the backup file into smaller files.
- Amazon RDS limits the number of files uploaded to an Amazon S3 bucket to 1 million. If the backup data for your database, including all full and incremental backups, exceeds 1 million files, use a tarball (.tar.gz) file to store full and incremental backup files in the Amazon S3 bucket.
- User accounts are not imported automatically. Save your user accounts from your source database and add them to your new DB instance later.
- Functions are not imported automatically. Save your functions from your source database and add them to your new DB instance later.
- Stored procedures are not imported automatically. Save your stored procedures from your source database and add them to your new DB instance later.
- Time zone information is not imported automatically. Record the time zone information for your source database, and set the time zone of your new DB instance later. For more information, see [Local Time Zone for MySQL DB Instances \(p. 595\)](#).
- Backward migration is not supported for both major versions and minor versions. For example, you can't migrate from version 5.7 to version 5.6, and you can't migrate from version 5.6.39 to version 5.6.37.

Overview of Setting Up to Import Backup Files from Amazon S3 to Amazon RDS

These are the components you need to set up to import backup files from Amazon S3 to Amazon RDS:

- An Amazon S3 bucket to store your backup files.
- A backup of your on-premises database created by Percona XtraBackup.
- An AWS Identity and Access Management (IAM) role to allow Amazon RDS to access the bucket.

If you already have an Amazon S3 bucket, you can use that. If you don't have an Amazon S3 bucket, you can create a new one. If you want to create a new bucket, see [Creating a Bucket](#).

Use the Percona XtraBackup tool to create your backup. For more information, see [Creating Your Database Backup \(p. 632\)](#).

If you already have an IAM role, you can use that. If you don't have an IAM role, you can create a new one manually. Alternatively, you can choose to have a new IAM role created for you in your account by the wizard when you restore the database by using the AWS Management Console. If you want to create a new IAM role manually, or attach trust and permissions policies to an existing IAM role, see [Creating an IAM Role Manually \(p. 634\)](#). If you want to have a new IAM role created for you, follow the procedure in [AWS Management Console \(p. 635\)](#).

Creating Your Database Backup

Use the Percona XtraBackup software to create your backup. For MySQL 5.6, Amazon RDS supports backup files created with Percona XtraBackup version 2.3.

We recommend that if you don't already have Percona XtraBackup installed, you use the latest version of the software available. You can download Percona XtraBackup from [Download Percona XtraBackup](#).

You can create a full backup of your MySQL database files using Percona XtraBackup. Alternatively, if you already use Percona XtraBackup to back up your MySQL database files, you can upload your existing full and incremental backup directories and files.

For more information about backing up your database with Percona XtraBackup, see [Percona XtraBackup - Documentation](#) and [The xtrabackup Binary](#) on the Percona website.

Creating a Full Backup With Percona XtraBackup

To create a full backup of your MySQL database files that can be restored from Amazon S3, use the Percona XtraBackup utility (`xtrabackup`) to back up your database.

For example, the following command creates a backup of a MySQL database and stores the files in the folder `/on-premises/s3-restore/backup`.

```
xtrabackup --backup --user=<myuser> --password=<password> --target-dir=</on-premises/s3-restore/backup>
```

If you want to compress your backup into a single file (which can be split later, if needed), you can save your backup in one of the following formats:

- Gzip (.gz)
- tar (.tar)
- Percona xbstream (.xbstream)

The following command creates a backup of your MySQL database split into multiple Gzip files.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \  
--target-dir=</on-premises/s3-restore/backup> | gzip - | split -d --bytes=500MB \  
- </on-premises/s3-restore/backup/backup>.tar.gz
```

The following command creates a backup of your MySQL database split into multiple tar files.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \  
--target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \  
- </on-premises/s3-restore/backup/backup>.tar
```

The following command creates a backup of your MySQL database split into multiple xbstream files.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=xbstream \  
--target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \  
- </on-premises/s3-restore/backup/backup>.xbstream
```

Using Incremental Backups With Percona XtraBackup

If you already use Percona XtraBackup to perform full and incremental backups of your MySQL database files, you don't need to create a full backup and upload the backup files to Amazon S3. Instead, you can save a significant amount of time by copying your existing backup directories and files to your Amazon S3 bucket. For more information about creating incremental backups using Percona XtraBackup, see [Incremental Backup](#).

When copying your existing full and incremental backup files to an Amazon S3 bucket, you must recursively copy the contents of the base directory. Those contents include the full backup and also all incremental backup directories and files. This copy must preserve the directory structure in the Amazon

S3 bucket. Amazon RDS iterates through all files and directories. Amazon RDS uses the `xtrabackup-checkpoints` file that is included with each incremental backup to identify the base directory, and to order incremental backups by log sequence number (LSN) range.

Backup Considerations for Percona XtraBackup

Amazon RDS consumes your backup files based on the file name. Name your backup files with the appropriate file extension based on the file format—for example, `.xbstream` for files stored using the Percona xbstream format.

Amazon RDS consumes your backup files in alphabetical order and also in natural number order. Use the `split` option when you issue the `xtrabackup` command to ensure that your backup files are written and named in the proper order.

Amazon RDS doesn't support partial backups created using Percona XtraBackup. You can't use the following options to create a partial backup when you back up the source files for your database: `--tables`, `--tables-exclude`, `--tables-file`, `--databases`, `--databases-exclude`, or `--databases-file`.

Amazon RDS supports incremental backups created using Percona XtraBackup. For more information about creating incremental backups using Percona XtraBackup, see [Incremental Backup](#).

Creating an IAM Role Manually

If you don't have an IAM role, you can create a new one manually. Alternatively, you can choose to have a new IAM role created for you by the wizard when you restore the database by using the AWS Management Console. If you want to have a new IAM role created for you, follow the procedure in [AWS Management Console \(p. 635\)](#).

To manually create a new IAM role for importing your database from Amazon S3, create a role to delegate permissions from Amazon RDS to your Amazon S3 bucket. When you create an IAM role, you attach trust and permissions policies. To import your backup files from Amazon S3, use trust and permissions policies similar to the examples following. For more information about creating the role, see [Creating a Role to Delegate Permissions to an AWS Service](#).

Alternatively, you can choose to have a new IAM role created for you by the wizard when you restore the database by using the AWS Management Console. If you want to have a new IAM role created for you, follow the procedure in [AWS Management Console \(p. 635\)](#).

The trust and permissions policies require that you provide an Amazon Resource Name (ARN). For more information about ARN formatting, see [Amazon Resource Names \(ARNs\) and AWS Service Namespaces](#).

Example Trust Policy for Importing from Amazon S3

```
{  
    "Version": "2012-10-17",  
    "Statement":  
    [ {  
        "Effect": "Allow",  
        "Principal": {"Service": "rds.amazonaws.com"},  
        "Action": "sts:AssumeRole"  
    } ]  
}
```

Example Permissions Policy for Importing from Amazon S3 — IAM User Permissions

```
{
```

```
"Version": "2012-10-17",
"Statement":
[
    {
        "Sid": "AllowS3AccessRole",
        "Effect": "Allow",
        "Action": "iam:PassRole",
        "Resource": "arn:aws:iam::IAM User ID:role/S3Access"
    }
]
```

Example Permissions Policy for Importing from Amazon S3 — Role Permissions

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action":
            [
                "s3>ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::bucket_name"
        },
        {
            "Effect": "Allow",
            "Action":
            [
                "s3.GetObject"
            ],
            "Resource": "arn:aws:s3:::bucket_name/prefix*"
        }
    ]
}
```

Note

If you include a file name prefix, include the asterisk (*) after the prefix. If you don't want to specify a prefix, specify only an asterisk.

AWS Management Console

To import data from Amazon S3 to a new MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the Amazon RDS console, choose the AWS Region in which to create your DB instance. Choose the same AWS Region as the Amazon S3 bucket that contains your database backup.
3. In the navigation pane, choose **Databases**.
4. Choose **Restore from S3** to launch the wizard.

The wizard opens on the **Select engine** page.

5. On the **Select engine** page, choose the MySQL icon, and then choose **Next**.

The **Specify source backup details** page appears.

Specify source backup details

Source database specifications

Source engine

mysql

Source engine version

5.6

S3 bucket

Refresh

S3 bucket

- Select one -

S3 folder path prefix (optional) [info](#)

IAM role

Refresh

Create a new role

- Yes
 No

IAM role name

Cancel

Previous

Next

6. On the **Specify source backup details** page, specify your backup information.
 - a. For **Source engine**, choose **mysql**.
 - b. For **Source engine version**, choose the MySQL version of your source database.
 - c. For **S3 bucket**, choose your Amazon S3 bucket.
 - d. (Optional) For **S3 folder path prefix**, enter a file path prefix for the files stored in your Amazon S3 bucket. If you don't specify a prefix, then RDS creates your DB instance using all of the files and folders in the root folder of the S3 bucket. If you do specify a prefix, then RDS creates your DB instance using the files and folders in the S3 bucket where the path for the file begins with the specified prefix. For example, suppose that you store your backup files on S3 in a subfolder named `backups`, and you have multiple sets of backup files, each in its own

directory (gzip_backup1, gzip_backup2, and so on). In this case, you specify a prefix of backups/gzip_backup1 to restore from the files in the gzip_backup1 folder.

- e. For **Create a new role**, choose **Yes** to create a new IAM role in your account, or choose **No** to select an existing IAM role.
 - f. For **IAM role**, select an existing IAM role, or specify the name for a new IAM role. You can choose to have a new IAM role created for you by choosing **Yes** for **Create a New Role**.
7. Choose **Next** to continue. The **Specify DB Details** page appears.

On the **Specify DB details** page, specify your DB instance information. For information about each setting, see [Settings for MySQL DB Instances \(p. 601\)](#).

Note

Be sure to allocate enough memory for your new DB instance so that the restore can succeed. You can also allocate additional memory for future growth.

8. Choose **Next** to continue. The **Configure advanced settings** page appears.

Provide additional information that Amazon RDS needs to launch the DB instance. For information about each setting, see [Settings for MySQL DB Instances \(p. 601\)](#).

9. Choose **Create database**.

CLI

To import data from Amazon S3 to a new MySQL DB instance by using the AWS CLI, call the `restore-db-instance-from-s3` command with the parameters following. For information about each setting, see [Settings for MySQL DB Instances \(p. 601\)](#).

Note

Be sure to allocate enough memory for your new DB instance so that the restore can succeed. You can also allocate additional memory for future growth.

- `--allocated-storage`
- `--db-instance-identifier`
- `--db-instance-class`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--s3-bucket-name`
- `--s3-ingestion-role-arn`
- `--s3-prefix`
- `--source-engine`
- `--source-engine-version`

Example

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-from-s3 \
--allocated-storage 250 \
--db-instance-identifier myidentifier \
--db-instance-class db.m4.large \
--engine mysql \
```

```
--master-user-name masterawsuser \
--master-user-password masteruserpassword \
--s3-bucket-name mybucket \
--s3-ingestion-role-arn arn:aws:iam::account-number:role/rolename \
--s3-prefix bucketprefix \
--source-engine mysql \
--source-engine-version 5.6.40
```

For Windows:

```
aws rds restore-db-instance-from-s3 ^
--allocated-storage 250 ^
--db-instance-identifier myidentifier ^
--db-instance-class db.m4.large ^
--engine mysql ^
--master-user-name masterawsuser ^
--master-user-password masteruserpassword ^
--s3-bucket-name mybucket ^
--s3-ingestion-role-arn arn:aws:iam::account-number:role/rolename ^
--s3-prefix bucketprefix ^
--source-engine mysql ^
--source-engine-version 5.6.40
```

API

To import data from Amazon S3 to a new MySQL DB instance by using the Amazon RDS API, call the [RestoreDBInstanceFromS3](#) action.

Related Topics

- [Importing Data into a MySQL DB Instance \(p. 625\)](#)
- [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#)

Importing Data from a MySQL or MariaDB DB to an Amazon RDS MySQL or MariaDB DB Instance

If your scenario supports it, it is easier to move data in and out of Amazon RDS by using backup files and Amazon S3. For more information, see [Restoring a Backup into an Amazon RDS MySQL DB Instance \(p. 631\)](#).

You can also import data from an existing MySQL or MariaDB database to an Amazon RDS MySQL or MariaDB DB instance. You do so by copying the database with [mysqldump](#) and piping it directly into the Amazon RDS MySQL or MariaDB DB instance. The [mysqldump](#) command-line utility is commonly used to make backups and transfer data from one MySQL or MariaDB server to another. It is included with MySQL and MariaDB client software.

A typical [mysqldump](#) command to move data from an external database to an Amazon RDS DB instance looks similar to the following:

```
mysqldump -u <local_user> \
--databases <database_name> \
--single-transaction \
--compress \
--order-by-primary \
-p<local_password> | mysql -u <RDS_user> \
```

```
--port=<port_number> \
--host=<host_name> \
-p<RDS_password>
```

Important

Make sure not to leave a space between the `-p` option and the entered password.

Note

- Exclude the following schemas from the dump file: `sys`, `performance_schema`, and `information_schema`. The `mysqldump` utility excludes these schemas by default.
- If you need to migrate users and privileges, consider using a tool that generates the data control language (DCL) for recreating them, such as the [pt-show-grants](#) utility.

The parameters used are as follows:

- `-u <local_user>` – Use to specify a user name. In the first usage of this parameter, you specify the name of a user account on the local MySQL or MariaDB database identified by the `--databases` parameter.
- `--databases <database_name>` – Use to specify the name of the database on the local MySQL or MariaDB instance that you want to import into Amazon RDS.
- `--single-transaction` – Use to ensure that all of the data loaded from the local database is consistent with a single point in time. If there are other processes changing the data while `mysqldump` is reading it, using this option helps maintain data integrity.
- `--compress` – Use to reduce network bandwidth consumption by compressing the data from the local database before sending it to Amazon RDS.
- `--order-by-primary` – Use to reduce load time by sorting each table's data by its primary key.
- `-p<local_password>` – Use to specify a password. In the first usage of this parameter, you specify the password for the user account identified by the first `-u` parameter.
- `-u <RDS_user>` – Use to specify a user name. In the second usage of this parameter, you specify the name of a user account on the default database for the Amazon RDS MySQL or MariaDB DB instance identified by the `--host` parameter.
- `--port <port_number>` – Use to specify the port for your Amazon RDS MySQL or MariaDB DB instance. By default, this is 3306 unless you changed the value when creating the instance.
- `--host <host_name>` – Use to specify the DNS name from the Amazon RDS DB instance endpoint, for example, `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.
- `-p<RDS_password>` – Use to specify a password. In the second usage of this parameter, you specify the password for the user account identified by the second `-u` parameter.

You must create any stored procedures, triggers, functions, or events manually in your Amazon RDS database. If you have any of these objects in the database that you are copying, then exclude them when you run `mysqldump` by including the following parameters with your `mysqldump` command: `--routines=0 --triggers=0 --events=0`.

The following example copies the `world` sample database on the local host to an Amazon RDS MySQL DB instance.

For Linux, OS X, or Unix:

```
sudo mysqldump -u localuser \
--databases world \
--single-transaction \
```

```
--compress \
--order-by-primary \
-plocalpassword | mysql -u rdsuser
--port=3306 \
--host=myinstance.123456789012.us-east-1.rds.amazonaws.com \
-prdspassword
```

For Windows, the following command needs to be run in a command prompt that has been opened by right-clicking **Command Prompt** on the Windows programs menu and choosing **Run as administrator**:

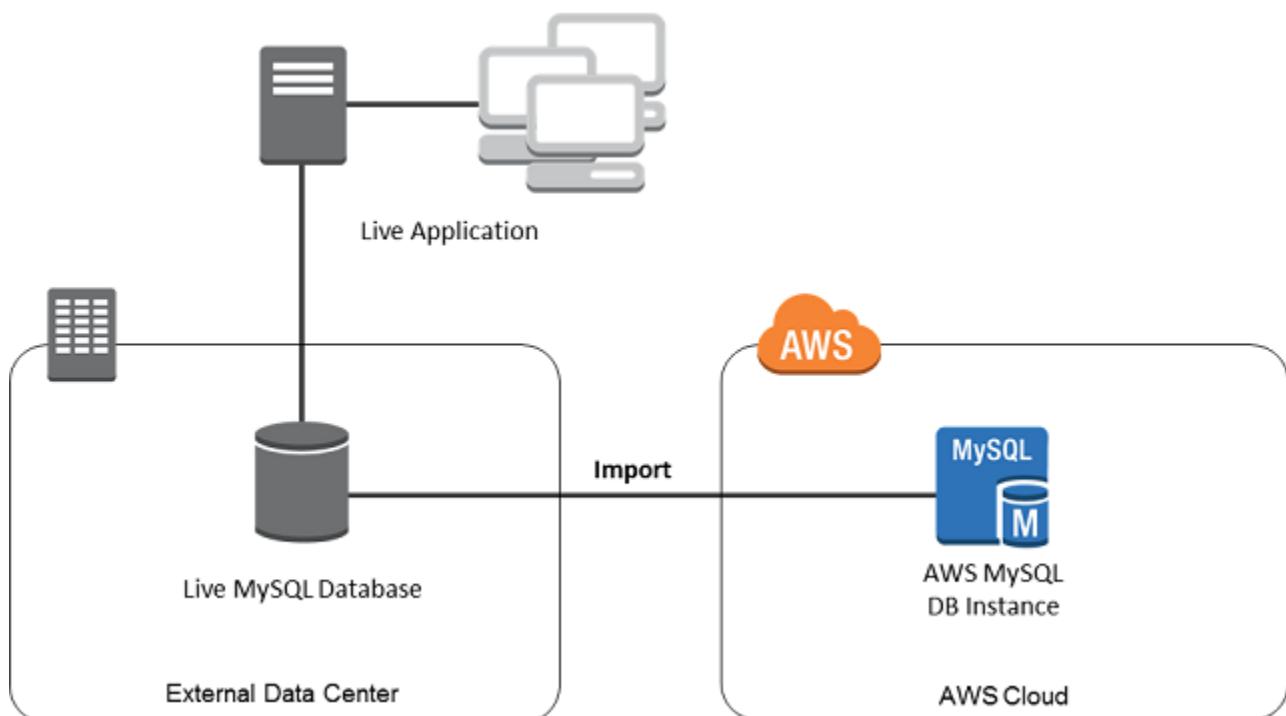
```
mysqldump -u localuser ^
--databases world ^
--single-transaction ^
--compress ^
--order-by-primary ^
-plocalpassword | mysql -u rdsuser ^
--port=3306 ^
--host=myinstance.123456789012.us-east-1.rds.amazonaws.com ^
-prdspassword
```

Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime

If your scenario supports it, it is easier to move data in and out of Amazon RDS by using backup files and Amazon S3. For more information, see [Restoring a Backup into an Amazon RDS MySQL DB Instance \(p. 631\)](#).

In some cases, you might need to import data from an external MySQL or MariaDB database that supports a live application to an Amazon RDS MySQL or MariaDB DB instance. In these cases, you can use the following procedure to minimize the impact on application availability. This procedure can also help if you are working with a very large database. Here, the procedure helps because you can reduce the cost of the import by reducing the amount of data that is passed across the network to AWS.

In this procedure, you transfer a copy of your database data to an Amazon EC2 instance and import the data into a new Amazon RDS DB instance. You then use replication to bring the Amazon RDS DB instance up-to-date with your live external instance, before redirecting your application to the Amazon RDS DB instance. You configure MariaDB replication based on global transaction identifiers (GTIDs) if the external instance is MariaDB 10.0.2 or greater and the target instance is Amazon RDS MariaDB; otherwise, you configure replication based on binary log coordinates. We recommend GTID-based replication if your external database supports it due to its enhanced crash-safety features. For more information, see [Global Transaction ID](#) in the MariaDB documentation.

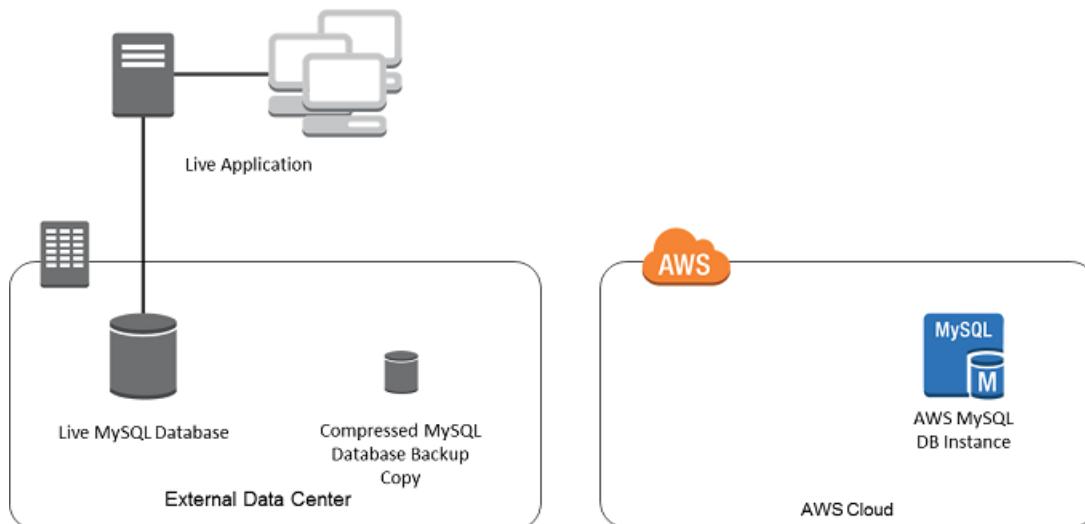


Note

We don't recommend that you use this procedure with source MySQL databases from MySQL versions earlier than version 5.1, due to potential replication issues. For more information, see [Replication Compatibility Between MySQL Versions](#) in the MySQL documentation.

Create a Copy of Your Existing Database

The first step in the process of migrating a large amount of data to an Amazon RDS MySQL or MariaDB DB instance with minimal downtime is to create a copy of the source data.



You can use the `mysqldump` utility to create a database backup in either SQL or delimited-text format. You should do a test run with each format in a nonproduction environment to see which method minimizes the amount of time that `mysqldump` runs.

You should also weigh `mysqldump` performance against the benefit offered by using the delimited-text format for loading. A backup using delimited-text format creates a tab-separated text file for each table being dumped. You can load these files in parallel using the `LOAD DATA LOCAL INFILE` command to reduce the amount of time required to import your database. For more information about choosing a `mysqldump` format and then loading the data, see [Using mysqldump For Backups](#) in the MySQL documentation.

Before you start the backup operation, you must set the replication options on the MySQL or MariaDB database that you are copying to Amazon RDS. The replication options include enabling binary logging and setting a unique server ID. Setting these options causes your server to start logging database transactions and prepares it to be a replication master later in this process.

Note

- Your database needs to be stopped to set the replication options and be in read-only mode while the backup copy is created, so you need to schedule a maintenance window for these operations.
- Exclude the following schemas from the dump file: `sys`, `performance_schema`, and `information_schema`. The `mysqldump` utility excludes these schemas by default.
- If you need to migrate users and privileges, consider using a tool that generates the data control language (DCL) for recreating them, such as the [pt-show-grants](#) utility.

To Set Replication Options

1. Edit the `my.cnf` file (this file is usually under `/etc`).

```
sudo vi /etc/my.cnf
```

Add the `log_bin` and `server_id` options to the `[mysqld]` section. The `log_bin` option provides a file name identifier for binary log files. The `server_id` option provides a unique identifier for the server in master-replica relationships.

The following example shows the updated `[mysqld]` section of a `my.cnf` file:

```
[mysqld]
log-bin=mysql-bin
server-id=1
```

For more information, see [Setting the Replication Master Configuration](#) in the MySQL documentation.

2. Restart the `mysql` service.

```
sudo service mysqld restart
```

To Create a Backup Copy of Your Existing Database

1. Create a backup of your data using the `mysqldump` utility, specifying either SQL or delimited-text format.

You must specify `--master-data=2` in order to create a backup file that can be used to start replication between servers. For more information, see the [mysqldump](#) documentation.

To improve performance and ensure data integrity, use the `--order-by-primary` and `--single-transaction` options of `mysqldump`.

To avoid including the MySQL system database in the backup, do not use the `--all-databases` option with `mysqldump`. For more information, see [Creating a Dump Snapshot Using mysqldump](#) in the MySQL documentation.

Use `chmod` if necessary to make sure that the directory where the backup file is being created is writeable.

Important

On Windows, run the command window as an administrator.

- To produce SQL output, use the following command.

For Linux, OS X, or Unix:

```
sudo mysqldump \  
    --databases <database_name> \  
    --master-data=2 \  
    --single-transaction \  
    --order-by-primary \  
    -r backup.sql \  
    -u <local_user> \  
    -p <password>
```

For Windows:

```
mysqldump ^  
    --databases <database_name> ^  
    --master-data=2 ^  
    --single-transaction ^  
    --order-by-primary ^  
    -r backup.sql ^  
    -u <local_user> ^  
    -p <password>
```

- To produce delimited-text output, use the following command.

For Linux, OS X, or Unix:

```
sudo mysqldump \  
    --tab=<target_directory> \  
    --fields-terminated-by ',' \  
    --fields-enclosed-by '"' \  
    --lines-terminated-by 0x0d0a \  
    <database_name> \  
    --master-data=2 \  
    --single-transaction \  
    --order-by-primary \  
    -p <password>
```

For Windows:

```
mysqldump ^  
    --tab=<target_directory> ^  
    --fields-terminated-by ',' ^  
    --fields-enclosed-by '"' ^  
    --lines-terminated-by 0x0d0a ^  
    <database_name> ^  
    --master-data=2 ^  
    --single-transaction ^  
    --order-by-primary ^
```

```
-p <password>
```

Note

You must create any stored procedures, triggers, functions, or events manually in your Amazon RDS database. If you have any of these objects in the database that you are copying, exclude them when you run `mysqldump` by including the following arguments with your `mysqldump` command: `--routines=0 --triggers=0 --events=0`.

When using the delimited-text format, a CHANGE MASTER TO comment is returned when you run `mysqldump`. This comment contains the master log file name and position. If the external instance is other than MariaDB version 10.0.2 or greater, note the values for `MASTER_LOG_FILE` and `MASTER_LOG_POS`; you need these values when setting up replication.

```
-- Position to start replication or point-in-time recovery from
--
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;
```

If you are using SQL format, you can get the master log file name and position in step 4 of the procedure at [Replicate Between Your External Database and New Amazon RDS DB Instance \(p. 649\)](#). If the external instance is MariaDB version 10.0.2 or greater, you can get the GTID in the next step.

2. If the external instance you are using is MariaDB version 10.0.2 or greater, you use GTID-based replication. Run `SHOW MASTER STATUS` on the external MariaDB instance to get the binary log file name and position, then convert them to a GTID by running `BINLOG_GTID_POS` on the external MariaDB instance.

```
SELECT BINLOG_GTID_POS('<binary log file name>', <binary log file position>);
```

Note the GTID returned; you need it to configure replication.

3. Compress the copied data to reduce the amount of network resources needed to copy your data to the Amazon RDS DB instance. Take note of the size of the backup file; you need this information when determining how large an Amazon EC2 instance to create. When you are done, compress the backup file using GZIP or your preferred compression utility.
 - To compress SQL output, use the following command.

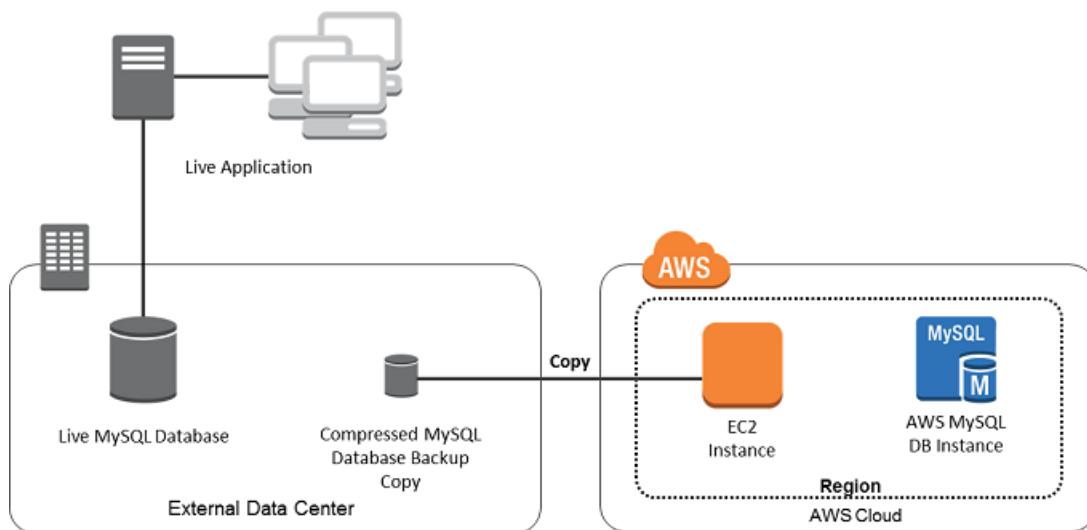
```
gzip backup.sql
```

- To compress delimited-text output, use the following command.

```
tar -zcvf backup.tar.gz <target_directory>
```

Create an Amazon EC2 Instance and Copy the Compressed Database

Copying your compressed database backup file to an Amazon EC2 instance takes fewer network resources than doing a direct copy of uncompressed data between database instances. After your data is in Amazon EC2, you can copy it from there directly to your Amazon RDS MySQL or MariaDB DB instance. For you to save on the cost of network resources, your Amazon EC2 instance must be in the same AWS Region as your Amazon RDS DB instance. Having the Amazon EC2 instance in the same AWS Region as your Amazon RDS DB instance also reduces network latency during the import.



To Create an Amazon EC2 Instance and Copy Your Data

1. In the AWS Region where you plan to create the RDS DB instance to run your MySQL database engine, create a VPC, a VPC security group, and a VPC subnet. Ensure that the inbound rules for your VPC security group allow the IP addresses required for your application to connect to AWS. This can be a range of IP addresses (for example, 203.0.113.0/24), or another VPC security group. You can use the [Amazon VPC Management Console](#) to create and manage VPCs, subnets, and security groups. For more information, see [Getting Started with Amazon VPC](#) in the *Amazon Virtual Private Cloud Getting Started Guide*.

Note

Older AWS accounts can also launch instances in Amazon EC2-Classic mode. In this case, make sure that the inbound rules in the DB security group for your Amazon RDS instance allow access for your EC2-Classic instance using the Amazon EC2 private IP address. For more information, see [Working with DB Security Groups \(EC2-Classic Platform\) \(p. 399\)](#).

2. Open the [Amazon EC2 Management Console](#) and select the AWS Region to contain both your Amazon EC2 instance and your Amazon RDS DB instance. Launch an Amazon EC2 instance using the VPC, subnet, and security group that you created in Step 1. Ensure that you select an instance type with enough storage for your database backup file when it is uncompressed. For details on Amazon EC2 instances, see [Getting Started with Amazon EC2 Linux Instances](#) in the *Amazon Elastic Compute Cloud User Guide for Linux*.
3. To connect to your Amazon RDS DB instance from your Amazon EC2 instance, you need to edit your VPC security group, and add an inbound rule specifying the private IP address of your EC2 instance. You can find the private IP address on the **Details** tab of the **Instance** pane in the EC2 console window. To edit the VPC security group and add an inbound rule, choose **Security Groups** in the EC2 console navigation pane, choose your security group, and then add an inbound rule for MySQL/Aurora specifying the private IP address of your EC2 instance. To learn how to add an inbound rule to a VPC security group, see [Adding and Removing Rules](#).
4. Copy your compressed database backup file from your local system to your Amazon EC2 instance. Use `chmod` if necessary to make sure you have write permission for the target directory of the Amazon EC2 instance. You can use `scp` or an SSH client to copy the file. The following is an example:

```
$ scp -r -i <key pair>.pem backup.sql.gz ec2-user@<EC2 DNS>:<target_directory>/  
backup.sql.gz
```

Important

Be sure to copy sensitive data using a secure network transfer protocol.

5. Connect to your Amazon EC2 instance and install the latest updates and the MySQL client tools using the following commands:

```
sudo yum update -y  
sudo yum install mysql-server -y
```

For more information, see [Connect to Your Instance](#) in the *Amazon Elastic Compute Cloud User Guide for Linux*.

6. While connected to your Amazon EC2 instance, decompress your database backup file. For example:

- To decompress SQL output, use the following command:

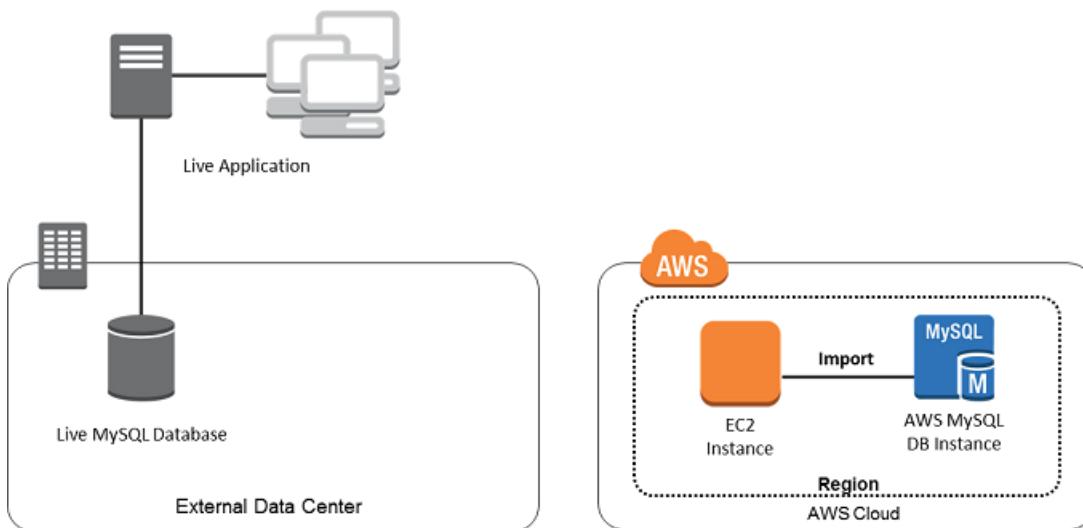
```
gzip backup.sql.gz -d
```

- To decompress delimited-text output, use the following command:

```
tar xzvf backup.tar.gz
```

Create an Amazon RDS MySQL or MariaDB DB instance and Import Data from Your Amazon EC2 Instance

By creating an Amazon RDS MySQL or MariaDB DB instance in the same AWS Region as your Amazon EC2 instance, you can import the database backup file from EC2 faster than over the internet.



To Create an Amazon RDS MySQL or MariaDB DB Instance and Import Your Data

1. Determine which DB instance class and what amount of storage space is required to support the expected workload for this Amazon RDS DB instance. This process should include deciding what is sufficient space and processing capacity for your data load procedures, and also what is required to handle the production workload. You can estimate this based on the size and resources of the source MySQL or MariaDB database. For more information, see [DB Instance Class \(p. 82\)](#).
2. Determine if Amazon RDS provisioned input/output operations per second (IOPS) is required to support the workloads. Provisioned IOPS storage delivers fast throughput for online transaction processing (OLTP) workloads, which are I/O intensive. For more information, see [Provisioned IOPS SSD Storage \(p. 105\)](#).

3. Open the [Amazon RDS console](#). In the upper-right corner, choose the AWS Region that contains your Amazon EC2 instance.
4. In the navigation pane, choose **Databases**.
5. Choose **Create database**, and then go through the steps to select options for your DB instance:
 - a. On the **Select engine** page, choose **MySQL** or **MariaDB**, as appropriate, and then choose **Next**.
 - b. On the **Choose use case** page, choose **Dev/Test – MySQL** to skip configuring Multi-AZ deployment and provisioned IOPS storage.
 - c. In the **Instance specifications** section of the **Specify DB details** page, specify the DB instance class and allocated storage size that you have determined are appropriate. Choose **No** for **Multi-AZ deployment**. For **Storage type**, specify whether or not to use **Provisioned IOPS (SSD)** as you determined in Step 2. For **DB engine version**, choose the version that is compatible with your source MySQL instance, as follows:
 - If your source instance is MySQL 5.1.x, the Amazon RDS DB instance must be MySQL 5.5.x.
 - If your source instance is MySQL 5.5.x, the Amazon RDS DB instance must be MySQL 5.5.x or greater.
 - If your source instance is MySQL 5.6.x, the Amazon RDS DB instance must be MySQL 5.6.x or MariaDB.
 - If your source instance is MySQL 5.7.x, the Amazon RDS DB instance must be MySQL 5.7.x, 5.6.x, or MariaDB.
 - If your source instance is MySQL 8.0.x, the Amazon RDS DB instance must be MySQL 8.0.x.
 - If your source instance is MariaDB 5.1, 5.2, or 5.3, the Amazon RDS DB instance must be MySQL 5.1.x.
 - If your source instance is MariaDB 5.5 or greater, the Amazon RDS DB instance must be MariaDB.

Accept the default values for all other boxes in this section.

In the **Settings** section, specify the requested database and user information. Choose **Next** when you are done.

- d. In the **Network & Security** section of the **Configure advanced settings** page, choose the same VPC and VPC security group as for your Amazon EC2 instance. This approach ensures that your Amazon EC2 instance and your Amazon RDS instance are visible to each other over the network. Set **Public accessibility** to **Yes**. Your DB instance must be publicly accessible to set up replication with your source database as described later in this topic. Accept the default values for all other boxes in this section.

In the **Database options** section, specify a database name. Accept the default values for all other boxes in this section.

In the **Backup** section, set the backup retention period to **0 days**. Accept the default values for all other boxes in this section.

Accept the default values for the remaining options. Choose **Create database** when you are done.

Do not configure multiple Availability Zones, backup retention, or Read Replicas until after you have imported the database backup. When that import is done, you can set Multi-AZ and backup retention the way you want them for the production instance. For a detailed walkthrough of creating an Amazon RDS MySQL DB instance, see [Creating a DB Instance Running the MySQL Database Engine \(p. 597\)](#). For a detailed walkthrough of creating an Amazon RDS MariaDB DB instance, see [Creating a DB Instance Running the MariaDB Database Engine \(p. 443\)](#).

6. Review the default configuration options for the Amazon RDS DB instance. In the left navigation pane of the Amazon RDS Management Console, choose **Parameter groups**, and then choose the magnifying glass icon next to the **default.mysqlx.x** or **default.mariadb.x** parameter group. If this parameter group does not have the configuration options that you want, find a different one that does, or create a new parameter group. For more information on creating a parameter group, see [Creating a Parameter Group](#).

[Working with DB Parameter Groups \(p. 169\)](#). If you decide to use a different parameter group than the default, associate it with your Amazon RDS DB instance. For more information, see [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#) or [Modifying a DB Instance Running the MariaDB Database Engine \(p. 456\)](#).

7. Connect to the new Amazon RDS DB instance as the master user, and create the users required to support the administrators, applications, and services that need to access the instance. The host name for the Amazon RDS DB instance is the **Endpoint** value for this instance without including the port number, for example `mysampledb.claxc2oy9ak1.us-west-2.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.
8. Connect to your Amazon EC2 instance. For more information, see [Connect to Your Instance](#) in the *Amazon Elastic Compute Cloud User Guide for Linux*.
9. Connect to your Amazon RDS DB instance as a remote host from your Amazon EC2 instance using the `mysql` command. The following is an example:

```
mysql -h <host_name> -P 3306 -u <db_master_user> -p
```

The host name is the DNS name from the Amazon RDS DB instance endpoint.

- 10At the `mysql` prompt, run the `source` command and pass it the name of your database dump file to load the data into the Amazon RDS DB instance.

- For SQL format, use the following command.

```
mysql> source backup.sql;
```

- For delimited-text format, first create the database (if it isn't the default database you created when setting up the Amazon RDS DB instance).

```
mysql> create database <database_name>;  
$ mysql> use <database_name>;
```

Then create the tables.

```
mysql> source <table1>.sql  
$ mysql> source <table2>.sql  
etc...
```

Then import the data.

```
mysql> LOAD DATA LOCAL INFILE 'table1.txt' INTO TABLE table1 FIELDS TERMINATED BY ','  
ENCLOSED BY ''' LINES TERMINATED BY '0x0d0a';  
$ mysql> LOAD DATA LOCAL INFILE 'table2.txt' INTO TABLE table2 FIELDS TERMINATED BY ','  
ENCLOSED BY ''' LINES TERMINATED BY '0x0d0a';  
etc...
```

To improve performance, you can perform these operations in parallel from multiple connections so that all of your tables get created and then loaded at the same time.

Note

If you used any data-formatting options with `mysqldump` when you initially dumped the table, you must use the same options with `mysqlimport` or `LOAD DATA LOCAL INFILE` to ensure proper interpretation of the data file contents.

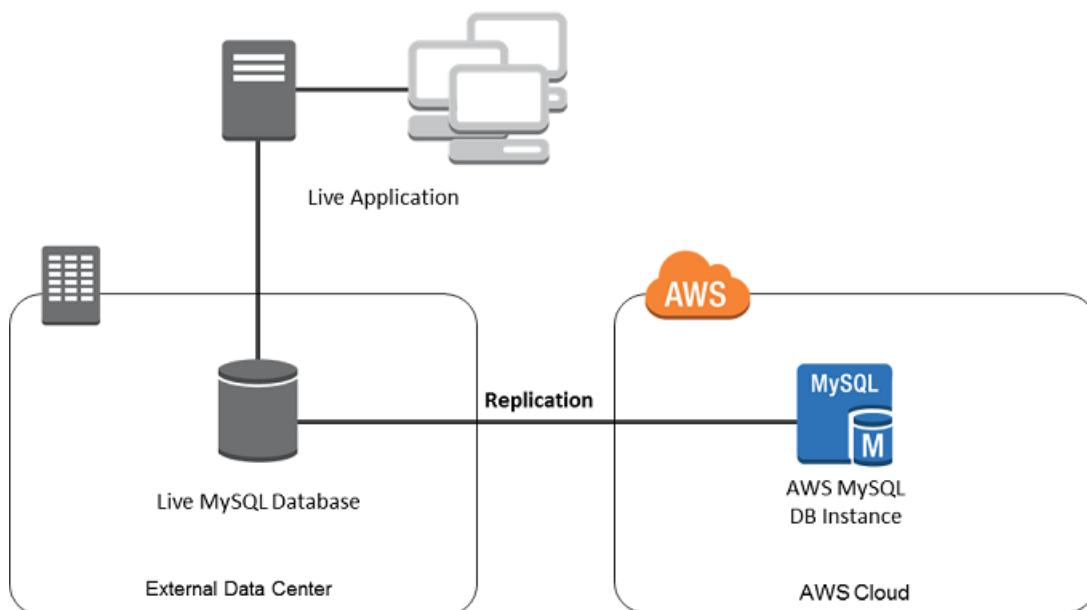
- 11Run a simple `SELECT` query against one or two of the tables in the imported database to verify that the import was successful.

Note

If you no longer need the Amazon EC2 instance used in this procedure, you should terminate the EC2 instance to reduce your Amazon AWS resource usage. To terminate an EC2 instance, see [Terminating an Instance](#).

Replicate Between Your External Database and New Amazon RDS DB Instance

Your source database was likely updated during the time that it took to copy and transfer the data to the Amazon RDS MySQL or MariaDB DB instance. That being the case, you can use replication to bring the copied database up-to-date with the source database.



Note

The permissions required to start replication on an Amazon RDS DB instance are restricted and not available to your Amazon RDS master user. Because of this, you must use either the Amazon RDS [mysql.rds_set_external_master \(p. 694\)](#) command or the [mysql.rds_set_external_master_gtid \(p. 485\)](#) command to configure replication, and the [mysql.rds_start_replication \(p. 704\)](#) command to start replication between your live database and your Amazon RDS database.

To Start Replication

Earlier, you enabled binary logging and set a unique server ID for your source database. Now you can set up your Amazon RDS DB instance as a replica with your live database as the replication master.

1. In the Amazon RDS Management Console, add the IP address of the server that hosts the source database to the VPC security group for the Amazon RDS DB instance. For more information on modifying a VPC security group, see [Security Groups for Your VPC](#) in the *Amazon Virtual Private Cloud User Guide*.

You might also need to configure your local network to permit connections from the IP address of your Amazon RDS DB instance, so that it can communicate with your source instance. To find the IP address of the Amazon RDS DB instance, use the host command.

```
host <RDS_MySQL_DB_host_name>
```

The host name is the DNS name from the Amazon RDS DB instance endpoint, for example `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.

2. Using the client of your choice, connect to the source instance and create a user to be used for replication. This account is used solely for replication and must be restricted to your domain to improve security. The following is an example.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>;'
```

3. For the source instance, grant REPLICATION CLIENT and REPLICATION SLAVE privileges to your replication user. For example, to grant the REPLICATION CLIENT and REPLICATION SLAVE privileges on all databases for the 'repl_user' user for your domain, issue the following command.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'  
IDENTIFIED BY '<password>;'
```

4. If you used SQL format to create your backup file and the external instance is not MariaDB 10.0.2 or greater, look at the contents of that file.

```
cat backup.sql
```

The file includes a CHANGE MASTER TO comment that contains the master log file name and position. This comment is included in the backup file when you use the `--master-data` option with `mysqldump`. Note the values for `MASTER_LOG_FILE` and `MASTER_LOG_POS`.

```
--  
-- Position to start replication or point-in-time recovery from  
--  
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;
```

If you used delimited text format to create your backup file and the external instance is not MariaDB 10.0.2 or greater, you should already have binary log coordinates from step 1 of the procedure at [To Create a Backup Copy of Your Existing Database \(p. 642\)](#).

If the external instance is MariaDB 10.0.2 or greater, you should already have the GTID from which to start replication from step 2 of the procedure at [To Create a Backup Copy of Your Existing Database \(p. 642\)](#).

5. Make the Amazon RDS DB instance the replica. If the external instance is not MariaDB 10.0.2 or greater, connect to the Amazon RDS DB instance as the master user and identify the source database as the replication master by using the [mysql.rds_set_external_master \(p. 694\)](#) command. Use the master log file name and master log position that you determined in the previous step if you have a SQL format backup file. Alternatively, use the name and position that you determined when creating the backup files if you used delimited-text format. The following is an example.

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,  
'repl_user', '<password>', 'mysql-bin-changelog.000031', 107, 0);
```

If the external instance is MariaDB 10.0.2 or greater, connect to the Amazon RDS DB instance as the master user and identify the source database as the replication master by using the [mysql.rds_set_external_master_gtid \(p. 485\)](#) command. Use the GTID that you determined in step 2 of the procedure at [To Create a Backup Copy of Your Existing Database \(p. 642\)](#). The following is an example.

```
CALL mysql.rds_set_external_master_gtid ('<master_server_ip_address>', 3306,  
'ReplicationUser', '<password>', '<GTID>', 0);
```

The `master_server_ip_address` is the IP address of master MySQL instance. An EC2 private DNS address is currently not supported.

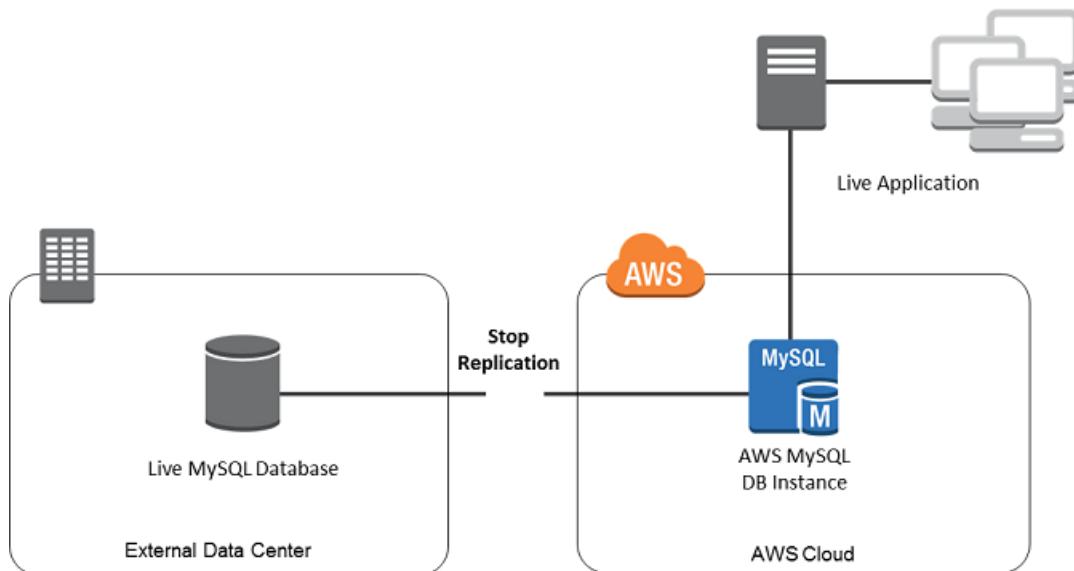
6. On the Amazon RDS DB instance, issue the [mysql.rds_start_replication \(p. 704\)](#) command to start replication.

```
CALL mysql.rds_start_replication;
```

7. On the Amazon RDS DB instance, run the [SHOW SLAVE STATUS](#) command to determine when the replica is up-to-date with the replication master. The results of the [SHOW SLAVE STATUS](#) command include the `Seconds_Behind_Master` field. When the `Seconds_Behind_Master` field returns 0, then the replica is up-to-date with the master.
8. After the Amazon RDS DB instance is up-to-date, enable automated backups so you can restore that database if needed. You can enable or modify automated backups for your Amazon RDS DB instance using the [Amazon RDS Management Console](#). For more information, see [Working With Backups \(p. 208\)](#).

Redirect Your Live Application to Your Amazon RDS Instance

After the Amazon RDS MySQL or MariaDB DB instance is up-to-date with the replication master, you can now update your live application to use the Amazon RDS instance.



To Redirect Your Live Application to Your Amazon RDS MySQL or MariaDB DB Instance and Stop Replication

1. To add the VPC security group for the Amazon RDS DB instance, add the IP address of the server that hosts the application. For more information on modifying a VPC security group, see [Security Groups for Your VPC](#) in the *Amazon Virtual Private Cloud User Guide*.
2. Verify that the `Seconds_Behind_Master` field in the [SHOW SLAVE STATUS](#) command results is 0, which indicates that the replica is up-to-date with the replication master.

```
SHOW SLAVE STATUS;
```

3. Close all connections to the source when their transactions complete.
4. Update your application to use the Amazon RDS DB instance. This update typically involves changing the connection settings to identify the host name and port of the Amazon RDS DB instance, the user account and password to connect with, and the database to use.
5. Stop replication for the Amazon RDS instance using the [mysql.rds_stop_replication \(p. 706\)](#) command.

```
CALL mysql.rds_stop_replication;
```

6. Run the [mysql.rds_reset_external_master \(p. 700\)](#) command on your Amazon RDS DB instance to reset the replication configuration so this instance is no longer identified as a replica.

```
CALL mysql.rds_reset_external_master;
```

7. Enable additional Amazon RDS features such as Multi-AZ support and Read Replicas. For more information, see [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#) and [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

Note

If you no longer need the Amazon RDS instance used in this procedure, you should delete the RDS instance to reduce your Amazon AWS resource usage. To delete an RDS instance, see [Deleting a DB Instance \(p. 135\)](#).

Importing Data From Any Source to a MySQL or MariaDB DB Instance

If you have more than 1 GiB of data to load, or if your data is coming from somewhere other than a MySQL or MariaDB database, we recommend creating flat files and loading them with `mysqldump`. `mysqldump` is another command line utility bundled with the MySQL and MariaDB client software whose purpose is to load flat files into MySQL or MariaDB. For information about `mysqldump`, see [mysqldump - A Data Import Program](#) in the MySQL documentation.

We also recommend creating DB snapshots of the target Amazon RDS DB instance before and after the data load. Amazon RDS DB snapshots are complete backups of your DB instance that can be used to restore your DB instance to a known state. When you initiate a DB snapshot, I/O operations to your database instance are momentarily suspended while your database is backed up.

Creating a DB snapshot immediately before the load lets you restore the database to its state before the load, if you need to. A DB snapshot taken immediately after the load protects you from having to load the data again in case of a mishap and can also be used to seed new database instances.

The following list shows the steps to take. Each step is discussed in more detail below.

1. Create flat files containing the data to be loaded.
2. Stop any applications accessing the target DB instance.
3. Create a DB snapshot.
4. Consider disabling Amazon RDS automated backups.
5. Load the data using `mysqldump`.
6. Enable automated backups again.

Step 1: Create Flat Files Containing the Data to be Loaded

Use a common format, such as CSV (Comma-Separated Values), to store the data to be loaded. Each table must have its own file; data for multiple tables cannot be combined in the same file. Give each file the same name as the table it corresponds to. The file extension can be anything you like. For example, if the table name is "sales", the file name could be "sales.csv" or "sales.txt", but not "sales_01.csv".

Whenever possible, order the data by the primary key of the table being loaded. This drastically improves load times and minimizes disk storage requirements.

The speed and efficiency of this procedure is dependent upon keeping the size of the files small. If the uncompressed size of any individual file is larger than 1 GiB, split it into multiple files and load each one separately.

On Unix-like systems (including Linux), use the 'split' command. For example, the following command splits the sales.csv file into multiple files of less than 1 GiB, splitting only at line breaks (-C 1024m). The new files are named sales.part_00, sales.part_01, and so on.

```
split -C 1024m -d sales.csv sales.part_
```

Similar utilities are available on other operating systems.

Step 2: Stop Any Applications Accessing the Target DB Instance

Before starting a large load, stop all application activity accessing the target DB instance that you plan to load to. We recommend this particularly if other sessions will be modifying the tables being loaded or tables they reference. Doing this reduces the risk of constraint violations occurring during the load and improves load performance. It also makes it possible to restore the database instance to the point just before the load without losing changes made by processes not involved in the load.

Of course, this might not be possible or practical. If you are unable to stop applications from accessing the DB instance before the load, take steps to ensure the availability and integrity of your data. The specific steps required vary greatly depending upon specific use cases and site requirements.

Step 3: Create a DB Snapshot

If you plan to load data into a new DB instance that contains no data, you can skip this step. Otherwise, creating a DB snapshot of your DB instance allows you to restore the DB instance to the point just before the load, if it becomes necessary. As previously mentioned, when you initiate a DB snapshot, I/O operations to your database instance are suspended for a few minutes while the database is backed up.

In the example below, we use the AWS CLI `create-db-snapshot` command to create a DB snapshot of our AcmeRDS instance and give the DB snapshot the identifier "preload".

For Linux, OS X, or Unix:

```
aws rds create-db-snapshot \
--db-instance-identifier AcmeRDS \
--db-snapshot-identifier preload
```

For Windows:

```
aws rds create-db-snapshot ^
--db-instance-identifier AcmeRDS ^
--db-snapshot-identifier preload
```

You can also use the restore from DB snapshot functionality in order to create test database instances for dry runs or to "undo" changes made during the load.

Keep in mind that restoring a database from a DB snapshot creates a new DB instance that, like all DB instances, has a unique identifier and endpoint. If you need to restore the database instance without changing the endpoint, you must first delete the DB instance so that the endpoint can be reused.

For example, to create a DB instance for dry runs or other testing, you would give the DB instance its own identifier. In the example, "AcmeRDS-2" is the identifier and we would connect to the database instance using the endpoint associated with AcmeRDS-2.

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-from-db-snapshot \  
  --db-instance-identifier AcmeRDS-2 \  
  --db-snapshot-identifier preload
```

For Windows:

```
aws rds restore-db-instance-from-db-snapshot ^  
  --db-instance-identifier AcmeRDS-2 ^  
  --db-snapshot-identifier preload
```

To reuse the existing endpoint, we must first delete the database instance and then give the restored database the same identifier.

For Linux, OS X, or Unix:

```
aws rds delete-db-instance \  
  --db-instance-identifier AcmeRDS \  
  --final-db-snapshot-identifier AcmeRDS-Final  
  
aws rds restore-db-instance-from-db-snapshot \  
  --db-instance-identifier AcmeRDS \  
  --db-snapshot-identifier preload
```

For Windows:

```
aws rds delete-db-instance ^  
  --db-instance-identifier AcmeRDS ^  
  --final-db-snapshot-identifier AcmeRDS-Final  
  
aws rds restore-db-instance-from-db-snapshot ^  
  --db-instance-identifier AcmeRDS ^  
  --db-snapshot-identifier preload
```

The example takes a final DB snapshot of the database instance before deleting it. This is optional, but recommended.

Step 4: Consider Disabling Amazon RDS Automated Backups

Warning

Do not disable automated backups if you need the ability to perform point-in-time recovery.

Disabling automated backups erases all existing backups, so point-in-time recovery is not possible after automated backups have been disabled. Disabling automated backups is a performance optimization and is not required for data loads. DB snapshots are not affected by disabling automated backups. All existing DB snapshots are still available for restore.

Disabling automated backups reduces load time by about 25 percent and reduce the amount of storage space required during the load. If you plan to load data into a new DB instance that contains no data, disabling backups is an easy way to speed up the load and avoid using the additional storage needed for backups. However, if you plan to load into a DB instance that already contains data, weigh the benefits of disabling backups against the impact of losing the ability to perform point-in-time-recovery.

DB instances have automated backups enabled by default (with a one day retention period). In order to disable automated backups, you must set the backup retention period to zero. After the load, you can re-enable backups by setting the backup retention period to a non-zero value. In order to enable or disable backups, Amazon RDS must shut the DB instance down and restart it in order to turn MySQL or MariaDB logging on or off.

Use the AWS CLI `modify-db-instance` command to set the backup retention to zero and apply the change immediately. Setting the retention period to zero requires a DB instance restart, so wait until the restart has completed before proceeding.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier AcmeRDS \
--apply-immediately \
--backup-retention-period 0
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier AcmeRDS ^
--apply-immediately ^
--backup-retention-period 0
```

You can check the status of your DB instance with the AWS CLI `describe-db-instances` command. The example displays the status of the AcmeRDS database instance and includes the `--headers` option to show column headings.

For Linux, OS X, or Unix:

```
aws rds describe-db-instances \
--db-instance-identifier AcmeRDS \
--headers
```

For Windows:

```
aws rds describe-db-instances ^
--db-instance-identifier AcmeRDS ^
--headers
```

When the Status column shows that the database is available, you're ready to proceed.

Step 5: Load the Data

Use the `mysqlimport` utility to load the flat files into Amazon RDS. In the example we tell `mysqlimport` to load all of the files named "sales" with an extension starting with "part_". This is a convenient way to load all of the files created in the "split" example. Use the `--compress` option to minimize network traffic. The `--fields-terminated-by=,` option is used for CSV files and the `--local` option specifies that the incoming data is located on the client. Without the `--local` option, the Amazon RDS DB instance looks for the data on the database host, so always specify the `--local` option.

For Linux, OS X, or Unix:

```
mysqlimport --local \
--compress \
--user=username \
--password \
--host=hostname \
--fields-terminated-by=',' Acme sales.part_*
```

For Windows:

```
mysqlimport --local ^
--compress ^
--user=username ^
--password ^
--host=hostname ^
--fields-terminated-by=',' Acme sales.part_*
```

For very large data loads, take additional DB snapshots periodically between loading files and note which files have been loaded. If a problem occurs, you can easily resume from the point of the last DB snapshot, avoiding lengthy reloads.

Step 6: Enable Amazon RDS Automated Backups

After the load is finished, re-enable Amazon RDS automated backups by setting the backup retention period back to its pre-load value. As noted earlier, Amazon RDS restarts the DB instance, so be prepared for a brief outage.

In the example, we use the AWS CLI modify-db-instance command to enable automated backups for the AcmeRDS DB instance and set the retention period to 1 day.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier AcmeRDS \
--backup-retention-period 1 \
--apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier AcmeRDS ^
--backup-retention-period 1 ^
--apply-immediately
```

Working with MySQL Replication

You usually use Read Replicas to configure replication between Amazon RDS DB instances. For general information about Read Replicas, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#). For specific information about working with Read Replicas on Amazon RDS MySQL, see [Working with MySQL Read Replicas \(p. 657\)](#).

You can use global transaction identifiers (GTIDs) for replication with Amazon RDS MySQL. For more information, see [Using GTID-Based Replication for Amazon RDS MySQL \(p. 663\)](#).

You can also set up replication between an Amazon RDS MySQL DB instance and a MySQL or MariaDB instance that is external to Amazon RDS. For information about configuring replication with an external source, see [Replication with a MySQL or MariaDB Instance Running External to Amazon RDS \(p. 667\)](#).

For any of these replication options, you can use either row-based replication, statement-based, or mixed replication. Row-based replication only replicates the changed rows that result from a SQL statement. Statement-based replication replicates the entire SQL statement. Mixed replication uses statement-based replication when possible, but switches to row-based replication when SQL statements that are unsafe for statement-based replication are executed. In most cases, mixed replication is recommended. The binary log format of the DB instance determines whether replication is row-based, statement-based, or mixed. For information about setting the binary log format, see [Binary Logging Format \(p. 325\)](#).

Note

You can configure replication to import databases from a MySQL or MariaDB instance that is external to Amazon RDS, or to export databases to such instances. For more information, see [Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime \(p. 640\)](#) and [Exporting Data from a MySQL DB Instance by Using Replication \(p. 673\)](#).

Topics

- [Working with MySQL Read Replicas \(p. 657\)](#)
- [Using GTID-Based Replication for Amazon RDS MySQL \(p. 663\)](#)
- [Replication with a MySQL or MariaDB Instance Running External to Amazon RDS \(p. 667\)](#)

Working with MySQL Read Replicas

This section contains specific information about working with Read Replicas on Amazon RDS MySQL. For general information about Read Replicas and instructions for using them, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

Topics

- [Read Replica Configuration with MySQL \(p. 658\)](#)
- [Configuring Delayed Replication with MySQL \(p. 658\)](#)
- [Read Replica Updates with MySQL \(p. 660\)](#)
- [Multi-AZ Read Replica Deployments with MySQL \(p. 660\)](#)
- [Monitoring MySQL Read Replicas \(p. 661\)](#)
- [Starting and Stopping Replication with MySQL Read Replicas \(p. 661\)](#)
- [Deleting Read Replicas with MySQL \(p. 661\)](#)
- [Troubleshooting a MySQL Read Replica Problem \(p. 661\)](#)

Read Replica Configuration with MySQL

Before a MySQL DB instance can serve as a replication source, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0. This requirement also applies to a Read Replica that is the source DB instance for another Read Replica. Automatic backups are supported only for Read Replicas running any version of MySQL 5.6 and later. You can configure replication based on binary log coordinates for a MySQL DB instance.

On Amazon RDS MySQL version 5.7.23 and later MySQL 5.7 versions, you can configure replication using global transaction identifiers (GTIDs). For more information, see [Using GTID-Based Replication for Amazon RDS MySQL \(p. 663\)](#).

You can create up to five Read Replicas from one DB instance. For replication to operate effectively, each Read Replica should have as the same amount of compute and storage resources as the source DB instance. If you scale the source DB instance, you should also scale the Read Replicas.

If a Read Replica is running any version of MySQL 5.6 and later, you can specify it as the source DB instance for another Read Replica. For example, you can create ReadReplica1 from MyDBInstance, and then create ReadReplica2 from ReadReplica1. Updates made to MyDBInstance are replicated to ReadReplica1 and then replicated from ReadReplica1 to ReadReplica2. You can't have more than four instances involved in a replication chain. For example, you can create ReadReplica1 from MySourceDBInstance, and then create ReadReplica2 from ReadReplica1, and then create ReadReplica3 from ReadReplica2, but you can't create a ReadReplica4 from ReadReplica3.

If you promote a MySQL Read Replica that is in turn replicating to other Read Replicas, those Read Replicas remain active. Consider an example where MyDBInstance1 replicates to MyDBInstance2, and MyDBInstance2 replicates to MyDBInstance3. If you promote MyDBInstance2, replication from MyDBInstance1 to MyDBInstance2 no longer occurs, but MyDBInstance2 still replicates to MyDBInstance3.

To enable automatic backups on a Read Replica for Amazon RDS MySQL version 5.6 and later, first create the Read Replica, then modify the Read Replica to enable automatic backups.

You can run multiple concurrent Read Replica create or delete actions that reference the same source DB instance, as long as you stay within the limit of five Read Replicas for the source instance.

Preparing MySQL DB Instances That Use MyISAM

If your MySQL DB instance uses a nontransactional engine such as MyISAM, you need to perform the following steps to successfully set up your Read Replica. These steps are required to make sure that the Read Replica has a consistent copy of your data. These steps are not required if all of your tables use a transactional engine such as InnoDB.

1. Stop all data manipulation language (DML) and data definition language (DDL) operations on non-transactional tables in the source DB instance and wait for them to complete. SELECT statements can continue running.
2. Flush and lock the tables in the source DB instance.
3. Create the Read Replica using one of the methods in the following sections.
4. Check the progress of the Read Replica creation using, for example, the `DescribeDBInstances` API operation. Once the Read Replica is available, unlock the tables of the source DB instance and resume normal database operations.

Configuring Delayed Replication with MySQL

You can use delayed replication as a strategy for disaster recovery. With delayed replication, you specify the minimum amount of time, in seconds, to delay replication from the master to the Read Replica. In

the event of a disaster, such as a table deleted unintentionally, you complete the following steps to recover from the disaster quickly:

- Stop replication to the Read Replica before the change that caused the disaster is sent to it.
Use the [mysql.rds_stop_replication \(p. 706\)](#) stored procedure to stop replication.
- Start replication and specify that replication stops automatically at a log file location.
You specify a location just before the disaster using the [mysql.rds_start_replication_until \(p. 704\)](#) stored procedure.
- Promote the Read Replica to be the new master DB instance by using the instructions in [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

Note

- On Amazon RDS MySQL 5.7, delayed replication is supported for MySQL 5.7.22 and later. On Amazon RDS MySQL 5.6, delayed replication is supported for MySQL 5.6.40 and later. Delayed replication is not supported on Amazon RDS MySQL 8.0.
- You must use stored procedures to configure delayed replication. You can't configure delayed replication with the AWS Management Console, the AWS CLI, or the Amazon RDS API.
- On Amazon RDS MySQL 5.7.23 and later MySQL 5.7 versions, you can use GTID-based replication in a delayed replication configuration. If you use GTID-based replication, use the [mysql.rds_start_replication_until_gtid \(p. 705\)](#) stored procedure instead of the [mysql.rds_start_replication_until \(p. 704\)](#) stored procedure. For more information about GTID-based replication, see [Using GTID-Based Replication for Amazon RDS MySQL \(p. 663\)](#).

Topics

- [Configuring Delayed Replication During Read Replica Creation \(p. 659\)](#)
- [Modifying Delayed Replication for an Existing Read Replica \(p. 660\)](#)
- [Setting a Location to Stop Replication to a Read Replica \(p. 660\)](#)

Configuring Delayed Replication During Read Replica Creation

To configure delayed replication for any future Read Replica created from a DB instance, run the [mysql.rds_set_configuration \(p. 711\)](#) stored procedure with the `target_delay` parameter.

To configure delayed replication during Read Replica creation

1. Using a MySQL client, connect to the MySQL DB instance that will be the source for Read Replicas as the master user.
2. Run the [mysql.rds_set_configuration \(p. 711\)](#) stored procedure with the `target_delay` parameter.

For example, run the following stored procedure to specify that replication is delayed by at least one hour (3600 seconds) for any Read Replica created from the current DB instance.

```
call mysql.rds_set_configuration('target delay', 3600);
```

Note

After running this stored procedure, any Read Replica you create using the AWS CLI or Amazon RDS API is configured with replication delayed by the specified number of seconds.

Modifying Delayed Replication for an Existing Read Replica

To modify delayed replication for an existing Read Replica, run the [mysql.rds_set_source_delay \(p. 703\)](#) stored procedure.

To modify delayed replication for an existing Read Replica

1. Using a MySQL client, connect to the Read Replica as the master user.
2. Use the [mysql.rds_stop_replication \(p. 706\)](#) stored procedure to stop replication.
3. Run the [mysql.rds_set_source_delay \(p. 703\)](#) stored procedure.

For example, run the following stored procedure to specify that replication to the Read Replica is delayed by at least one hour (3600 seconds).

```
call mysql.rds_set_source_delay(3600);
```

4. Use the [mysql.rds_start_replication \(p. 704\)](#) stored procedure to start replication.

Setting a Location to Stop Replication to a Read Replica

After stopping replication to the Read Replica, you can start replication and then stop it at a specified binary log file location using the [mysql.rds_start_replication_until \(p. 704\)](#) stored procedure.

To start replication to a Read Replica and stop replication at a specific location

1. Using a MySQL client, connect to the source MySQL DB instance as the master user.
2. Run the [mysql.rds_start_replication_until \(p. 704\)](#) stored procedure.

The following example initiates replication and replicates changes until it reaches location 120 in the `mysql-bin-changelog.000777` binary log file. In a disaster recovery scenario, assume that location 120 is just before the disaster.

```
call mysql.rds_start_replication_until(
    'mysql-bin-changelog.000777',
    120);
```

Replication stops automatically when the stop point is reached. The following RDS event is generated:
`Replication has been stopped since the replica reached the stop point specified by the rds_start_replication_until stored procedure.`

After replication is stopped, in a disaster recovery scenario, you can [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#) promote the Read Replica to be the new master DB instance. For information about promoting the Read Replica, see [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

Read Replica Updates with MySQL

Read Replicas are designed to support read queries, but you might need occasional updates. For example, you might need to add an index to speed the specific types of queries accessing the replica. You can enable updates by setting the `read_only` parameter to `0` in the DB parameter group for the Read Replica.

Multi-AZ Read Replica Deployments with MySQL

You can create a Read Replica from either single-AZ or Multi-AZ DB instance deployments. You use Multi-AZ deployments to improve the durability and availability of critical data, but you can't use the Multi-AZ

secondary to serve read-only queries. Instead, you can create Read Replicas from high-traffic Multi-AZ DB instances to offload read-only queries. If the source instance of a Multi-AZ deployment fails over to the secondary, any associated Read Replicas automatically switch to use the secondary (now primary) as their replication source. For more information, see [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#).

You can create a Read Replica as a Multi-AZ DB instance. Amazon RDS creates a standby of your replica in another Availability Zone for failover support for the replica. Creating your Read Replica as a Multi-AZ DB instance is independent of whether the source database is a Multi-AZ DB instance.

Monitoring MySQL Read Replicas

For MySQL Read Replicas, you can monitor replication lag in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric. The `ReplicaLag` metric reports the value of the `Seconds_Behind_Master` field of the `SHOW SLAVE STATUS` command.

Common causes for replication lag for MySQL are the following:

- A network outage.
- Writing to tables with indexes on a Read Replica. If the `read_only` parameter is not set to 0 on the Read Replica, it can break replication.
- Using a nontransactional storage engine such as MyISAM. Replication is only supported for the InnoDB storage engine on MySQL.

When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance. If the `ReplicaLag` metric returns -1, then replication is currently not active. `ReplicaLag = -1` is equivalent to `Seconds_Behind_Master = NULL`.

Starting and Stopping Replication with MySQL Read Replicas

You can stop and restart the replication process on an Amazon RDS DB instance by calling the system stored procedures [mysql.rds_stop_replication \(p. 706\)](#) and [mysql.rds_start_replication \(p. 704\)](#). You can do this when replicating between two Amazon RDS instances for long-running operations such as creating large indexes. You also need to stop and start replication when importing or exporting databases. For more information, see [Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime \(p. 640\)](#) and [Exporting Data from a MySQL DB Instance by Using Replication \(p. 673\)](#).

If replication is stopped for more than 30 consecutive days, either manually or due to a replication error, Amazon RDS terminates replication between the master DB instance and all Read Replicas. It does so to prevent increased storage requirements on the master DB instance and long failover times. The Read Replica DB instance is still available. However, replication can't be resumed because the binary logs required by the Read Replica are deleted from the master DB instance after replication is terminated. You can create a new Read Replica for the master DB instance to reestablish replication.

Deleting Read Replicas with MySQL

You must explicitly delete Read Replicas, using the same mechanisms for deleting a DB instance. If you delete the source DB instance without deleting the replicas, each replica is promoted to a standalone DB instance.

Troubleshooting a MySQL Read Replica Problem

For MySQL DB instances, in some cases Read Replicas can't be switched to the secondary if some binlog events aren't flushed during the failure. In these cases, you must manually delete and recreate the Read Replicas. You can reduce the chance of this happening by setting the following dynamic variable

values: `sync_binlog=1`, `innodb_flush_log_at_trx_commit=1`, and `innodb_support_xa=1`. These settings might reduce performance, so test their impact before implementing the changes in a production environment. For MySQL 5.5, `sync_binlog` defaults to 0, but in MySQL 5.6 and later, problems are less likely to occur because these parameters are all set to the recommended values by default.

The replication technologies for MySQL are asynchronous. Because they are asynchronous, occasional `BinLogDiskUsage` increases on the source DB instance and `ReplicaLag` on the Read Replica are to be expected. For example, a high volume of write operations to the source DB instance can occur in parallel. In contrast, write operations to the Read Replica are serialized using a single I/O thread, which can lead to a lag between the source instance and Read Replica. For more information about read-only replicas in the MySQL documentation, see [Replication Implementation Details](#).

You can do several things to reduce the lag between updates to a source DB instance and the subsequent updates to the Read Replica, such as the following:

- Sizing a Read Replica to have a storage size and DB instance class comparable to the source DB instance.
- Ensuring that parameter settings in the DB parameter groups used by the source DB instance and the Read Replica are compatible. For more information and an example, see the discussion of the `max_allowed_packet` parameter later in this section.

Amazon RDS monitors the replication status of your Read Replicas and updates the `Replication State` field of the Read Replica instance to `Error` if replication stops for any reason. An example might be if DML queries run on your Read Replica conflict with the updates made on the source DB instance.

You can review the details of the associated error thrown by the MySQL engine by viewing the `Replication Error` field. Events that indicate the status of the Read Replica are also generated, including [RDS-EVENT-0045 \(p. 291\)](#), [RDS-EVENT-0046 \(p. 292\)](#), and [RDS-EVENT-0047 \(p. 291\)](#). For more information about events and subscribing to events, see [Using Amazon RDS Event Notification \(p. 287\)](#). If a MySQL error message is returned, review the error number in the [MySQL error message documentation](#).

One common issue that can cause replication errors is when the value for the `max_allowed_packet` parameter for a Read Replica is less than the `max_allowed_packet` parameter for the source DB instance. The `max_allowed_packet` parameter is a custom parameter that you can set in a DB parameter group that is used to specify the maximum size of DML code that can be executed on the database. In some cases, the `max_allowed_packet` parameter value in the DB parameter group associated with a source DB instance is smaller than the `max_allowed_packet` parameter value in the DB parameter group associated with the source's Read Replica. In these cases, the replication process can throw an error (Packet bigger than '`max_allowed_packet`' bytes) and stop replication. You can fix the error by having the source and Read Replica use DB parameter groups with the same `max_allowed_packet` parameter values.

Other common situations that can cause replication errors include the following:

- Writing to tables on a Read Replica. If you are creating indexes on a Read Replica, you need to have the `read_only` parameter set to `0` to create the indexes. If you are writing to tables on the Read Replica, it might break replication.
- Using a non-transactional storage engine such as MyISAM. Read replicas require a transactional storage engine. Replication is only supported for the InnoDB storage engine on MySQL.
- Using unsafe nondeterministic queries such as `SYSDATE()`. For more information, see [Determination of Safe and Unsafe Statements in Binary Logging](#).

If you decide that you can safely skip an error, you can follow the steps described in the section [Skipping the Current Replication Error \(p. 685\)](#). Otherwise, you can delete the Read Replica and create an

instance using the same DB instance identifier so that the endpoint remains the same as that of your old Read Replica. If a replication error is fixed, the `Replication State` changes to *replicating*.

Using GTID-Based Replication for Amazon RDS MySQL

Global transaction identifiers (GTIDs) are unique identifiers generated for committed MySQL transactions. MySQL uses two different types of transactions for replication:

- *GTID transactions* – Transactions that are identified by a GTID.
- *Anonymous transactions* – Transactions that do not have a GTID assigned.

In a replication configuration, GTIDs are unique across all DB instances. GTIDs simplify replication configuration because when you use them, you don't have to refer to log file positions. GTIDs also make it easier to track replicated transactions and determine whether masters and replicas are consistent.

You can use GTID-based replication to replicate data with Amazon RDS MySQL Read Replicas or with an external MySQL database. For Amazon RDS MySQL Read Replicas, you can configure GTID-based replication when you are creating new Read Replicas, or you can convert existing Read Replicas to use GTID-based replication.

You can also use GTID-based replication in a delayed replication configuration with Amazon RDS MySQL. For more information, see [Configuring Delayed Replication with MySQL \(p. 658\)](#).

For more information about GTID-based replication with MySQL, see [Replication with Global Transaction Identifiers](#) in the MySQL documentation.

Note

GTID-based replication is supported for Amazon RDS MySQL version 5.7.23 and later MySQL 5.7 versions. All Amazon RDS MySQL DB instances in a replication configuration must meet this requirement. GTID-based replication is not supported for Amazon RDS MySQL 5.5, 5.6, or 8.0.

Topics

- [Parameters for GTID-Based Replication \(p. 663\)](#)
- [Configuring GTID-Based Replication for New Read Replicas \(p. 664\)](#)
- [Configuring GTID-Based Replication for Existing Read Replicas \(p. 665\)](#)
- [Disabling GTID-Based Replication for an Amazon RDS MySQL DB Instance with Read Replicas \(p. 666\)](#)

Note

For information about configuring GTID-based replication with an external database, see [Replication with a MySQL or MariaDB Instance Running External to Amazon RDS \(p. 667\)](#).

Parameters for GTID-Based Replication

Use the following parameters to configure GTID-based replication.

Parameter	Valid Values	Description
gtid_mode	OFF, OFF_PERMISSIVE, ON_PERMISSIVE, ON	OFF specifies that new transactions are anonymous transactions (that is, don't have GTIDs), and a transaction must be anonymous to be replicated.

Parameter	Valid Values	Description
		<p>OFF_PERMISSIVE specifies that new transactions are anonymous transactions, but all transactions can be replicated.</p> <p>ON_PERMISSIVE specifies that new transactions are GTID transactions, but all transactions can be replicated.</p> <p>ON specifies that new transactions are GTID transactions, and a transaction must be a GTID transaction to be replicated.</p>
enforce_gtid_consistency	OFF, ON, WARN	<p>OFF allows transactions to violate GTID consistency.</p> <p>ON prevents transactions from violating GTID consistency.</p> <p>WARN allows transactions to violate GTID consistency but generates a warning when a violation occurs.</p>

To use GTID-based replication with an Amazon RDS MySQL DB instance or Read Replica, make sure that parameter group for the DB instance or Read Replica has these parameters set to enable GTID-based replication. For more information about parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

Note

In the AWS Management Console, the `gtid_mode` parameter appears as `gtid-mode`.

Configuring GTID-Based Replication for New Read Replicas

When GTID-based replication is enabled for an Amazon RDS MySQL DB instance, GTID-based replication is configured automatically for Read Replicas of the DB instance.

To enable GTID-based replication for new Read Replicas

1. Make sure that the parameter group associated with the DB instance has the following parameter settings:
 - `gtid_mode` – ON or ON_PERMISSIVE
 - `enforce_gtid_consistency` – ON

For more information about parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

2. If you changed the parameter group of the DB instance, reboot the DB instance. For more information on how to do so, see [Rebooting a DB Instance \(p. 129\)](#).
3. Create one or more Read Replicas of the DB instance. For more information on how to do so, see [Creating a Read Replica \(p. 145\)](#).

Amazon RDS attempts to establish GTID-based replication between the MySQL DB instance and the Read Replicas using the `MASTER_AUTO_POSITION`. If the attempt fails, Amazon RDS uses log file positions for replication with the Read Replicas. For more information about the `MASTER_AUTO_POSITION`, see [GTID Auto-Positioning](#) in the MySQL documentation.

Configuring GTID-Based Replication for Existing Read Replicas

If you have an Amazon RDS MySQL DB instance with Read Replicas, and data is not being replicated using GTID-based replication, you can configure GTID-based replication between the DB instance and the Read Replicas.

To enable GTID-based replication for existing Read Replicas

1. If the DB instance or any Read Replica is using Amazon RDS MySQL version 5.7.22 or lower, upgrade the DB instance or Read Replica to Amazon RDS MySQL version 5.7.23 or later MySQL 5.7 version.

For more information, see [Upgrading the MySQL DB Engine \(p. 618\)](#).

2. (Optional) Reset the GTID parameters and test the behavior of the DB instance and Read Replicas:

- a. Make sure that the parameter group associated with the DB instance and each Read Replica has the `enforce_gtid_consistency` parameter set to `WARN`.

For more information about parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

- b. If you changed the parameter group of the DB instance, reboot the DB instance. If you changed the parameter group for a Read Replica, reboot the Read Replica.

For more information, see [Rebooting a DB Instance \(p. 129\)](#).

- c. Run your DB instance and Read Replicas with your normal workload and monitor the log files.

If you see warnings about GTID-incompatible transactions, adjust your application so that it only uses GTID-compatible features. Make sure that the DB instance is not generating any warnings about GTID-incompatible transactions before proceeding to the next step.

3. Reset the GTID parameters for GTID-based replication that allows anonymous transactions until the Read Replicas have processed all of them.

- a. Make sure that the parameter group associated with the DB instance and each Read Replica has the following parameter settings:

- `gtid_mode` – `ON_PERMISSIVE`
- `enforce_gtid_consistency` – `ON`

- b. If you changed the parameter group of the DB instance, reboot the DB instance. If you changed the parameter group for a Read Replica, reboot the Read Replica.

4. Wait for all of your anonymous transactions to be replicated. To check that these are replicated, do the following:

- a. Run the following statement on your primary DB instance.

```
SHOW MASTER STATUS;
```

Note the values in the `File` and `Position` columns.

- b. On each Read Replica, use the file and position information from its master in the previous step to run the following query.

```
SELECT MASTER_POS_WAIT(file, position);
```

For example, if the file name is `mysql-bin-changelog.000031` and the position is `107`, run the following statement.

```
SELECT MASTER_POS_WAIT(mysql-bin-changelog.000031, 107);
```

If the Read Replica is past the specified position, the query returns immediately. Otherwise, the function waits. Proceed to the next step when the query returns for all Read Replicas.

5. Reset the GTID parameters for GTID-based replication only.
 - a. Make sure that the parameter group associated with the DB instance and each Read Replica has the following parameter settings:
 - `gtid_mode` – ON
 - `enforce_gtid_consistency` – ON
 - b. Reboot the DB instance and each Read Replica.
6. On each Read Replica, run the following procedure.

```
CALL mysql.rds_set_master_auto_position(1);
```

Disabling GTID-Based Replication for an Amazon RDS MySQL DB Instance with Read Replicas

You can disable GTID-based replication for an Amazon RDS MySQL DB instance with Read Replicas.

To disable GTID-based replication for an RDS MySQL DB instance with Read Replicas

1. On each Read Replica, run the following procedure.

```
CALL mysql.rds_set_master_auto_position(0);
```

2. Reset the `gtid_mode` to `ON_PERMISSIVE`.

- a. Make sure that the parameter group associated with the Amazon RDS MySQL DB instance and each Read Replica has `gtid_mode` set to `ON_PERMISSIVE`.

For more information about parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

- b. Reboot the Amazon RDS MySQL DB instance and each Read Replica. For more information about rebooting, see [Rebooting a DB Instance \(p. 129\)](#).

3. Reset the `gtid_mode` to `OFF_PERMISSIVE`:

- a. Make sure that the parameter group associated with the Amazon RDS MySQL DB instance and each Read Replica has `gtid_mode` set to `OFF_PERMISSIVE`.
- b. Reboot the Amazon RDS MySQL DB instance and each Read Replica.

4. Wait for all of the GTID transactions to be applied on all of the Read Replicas. To check that these are applied, do the following:

- a. On the Amazon RDS MySQL DB instance, run the `SHOW MASTER STATUS` command.

Your output is similar to the following.

File	Position
mysql-bin-changelog.000031	107

Note the file and position in your output.

- b. On each Read Replica, use the file and position information from its master in the previous step to run the following query.

```
SELECT MASTER_POS_WAIT(file, position);
```

For example, if the file name is `mysql-bin-changelog.000031` and the position is `107`, run the following statement.

```
SELECT MASTER_POS_WAIT(mysql-bin-changelog.000031, 107);
```

If the Read Replica is past the specified position, the query returns immediately. Otherwise, the function waits. Proceed to the next step when the query returns for all Read Replicas.

5. Reset the GTID parameters to disable GTID-based replication.
 - a. Make sure that the parameter group associated with the Amazon RDS MySQL DB instance and each Read Replica has the following parameter settings:
 - `gtid_mode` – OFF
 - `enforce_gtid_consistency` – OFF
 - b. Reboot the Amazon RDS MySQL DB instance and each Read Replica.

Replication with a MySQL or MariaDB Instance Running External to Amazon RDS

You can set up replication between an Amazon RDS MySQL or MariaDB DB instance and a MySQL or MariaDB instance that is external to Amazon RDS.

Topics

- [Before You Begin \(p. 667\)](#)
- [Configuring Binary Log File Position Replication with an External Master Instance \(p. 668\)](#)
- [Configuring GTID-Based Replication with an External Master Instance \(p. 670\)](#)

Before You Begin

You can configure replication using the binary log file position of replicated transactions. On Amazon RDS MySQL 5.7.23 and later MySQL 5.7 versions, you can also configure replication using global transaction identifiers (GTIDs).

The permissions required to start replication on an Amazon RDS DB instance are restricted and not available to your Amazon RDS master user. Because of this, you must use the Amazon RDS [mysql.rds_set_external_master \(p. 694\)](#) and [mysql.rds_start_replication \(p. 704\)](#) commands to set up replication between your live database and your Amazon RDS database.

To set the binary logging format for a MySQL or MariaDB database, update the `binlog_format` parameter. If your DB instance uses the default DB instance parameter group, create a new DB parameter group to modify `binlog_format` settings. We recommend that you use the default setting for `binlog_format`, which is `MIXED`. However, you can also set `binlog_format` to `ROW` or `STATEMENT` if you need a specific binlog format. Reboot your DB instance for the change to take effect.

For information about setting the `binlog_format` parameter, see [Binary Logging Format \(p. 325\)](#). For information about the implications of different MySQL replication types, see [Advantages and Disadvantages of Statement-Based and Row-Based Replication](#) in the MySQL documentation.

Note

Use the procedure in this topic to configure replication in all cases except when the external instance is MariaDB version 10.0.2 or greater and the Amazon RDS instance is MariaDB. In that case, use the procedure at [Configuring GTID-Based Replication into an Amazon RDS MariaDB DB instance \(p. 473\)](#) to set up GTID-based replication.

Configuring Binary Log File Position Replication with an External Master Instance

Follow these guidelines when you set up an external replication master and a replica on Amazon RDS:

- Monitor failover events for the Amazon RDS DB instance that is your replica. If a failover occurs, then the DB instance that is your replica might be recreated on a new host with a different network address. For information on how to monitor failover events, see [Using Amazon RDS Event Notification \(p. 287\)](#).
- Maintain the binary logs (binlogs) on your master instance until you have verified that they have been applied to the replica. This maintenance makes sure that you can restore your master instance in the event of a failure.
- Turn on automated backups on your Amazon RDS DB instance. Turning on automated backups makes sure that you can restore your replica to a particular point in time if you need to re-synchronize your master and replica. For information on backups and point-in-time restore, see [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#).

To configure binary log file replication with an external master instance

1. Make the source MySQL or MariaDB instance read-only.

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Run the `SHOW MASTER STATUS` command on the source MySQL or MariaDB instance to determine the binlog location.

You receive output similar to the following example.

File	Position
mysql-bin-changelog.000031	107

3. Copy the database from the external instance to the Amazon RDS DB instance using `mysqldump`. For very large databases, you might want to use the procedure in [Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime \(p. 640\)](#).

For Linux, OS X, or Unix:

```
mysqldump --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u <local_user> \
    -p<local_password> | mysql \
        --host=hostname \
        --port=3306 \
        -u <RDS_user_name> \
        -p<RDS_password>
```

For Windows:

```
mysqldump --databases <database_name> ^
    --single-transaction ^
    --compress ^
    --order-by-primary ^
    -u <local_user> ^
    -p<local_password> | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u <RDS_user_name> ^
        -p<RDS_password>
```

Note

Make sure that there isn't a space between the `-p` option and the entered password.

To specify the host name, user name, port, and password to connect to your Amazon RDS DB instance, use the `--host`, `--user` (`-u`), `--port` and `-p` options in the `mysql` command. The host name is the Domain Name Service (DNS) name from the Amazon RDS DB instance endpoint, for example, `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the AWS Management Console.

4. Make the source MySQL or MariaDB instance writeable again.

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

For more information on making backups for use with replication, see [Backing Up a Master or Slave by Making It Read Only](#) in the MySQL documentation.

5. In the AWS Management Console, add the IP address of the server that hosts the external database to the VPC security group for the Amazon RDS DB instance. For more information on modifying a VPC security group, see [Security Groups for Your VPC](#) in the *Amazon Virtual Private Cloud User Guide*.

You might also need to configure your local network to permit connections from the IP address of your Amazon RDS DB instance, so that it can communicate with your external MySQL or MariaDB instance. To find the IP address of the Amazon RDS DB instance, use the `host` command.

```
host <RDS_MySQL_DB_host_name>
```

The host name is the DNS name from the Amazon RDS DB instance endpoint.

6. Using the client of your choice, connect to the external instance and create a user to use for replication. Use this account solely for replication. and restrict it to your domain to improve security. The following is an example.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>;'
```

7. For the external instance, grant REPLICATION CLIENT and REPLICATION SLAVE privileges to your replication user. For example, to grant the REPLICATION CLIENT and REPLICATION SLAVE privileges on all databases for the 'repl_user' user for your domain, issue the following command.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'  
IDENTIFIED BY '<password>;'
```

8. Make the Amazon RDS DB instance the replica. To do so, connect to the Amazon RDS DB instance as the master user and identify the external MySQL or MariaDB database as the replication master by using the [mysql.rds_set_external_master \(p. 694\)](#) command. Use the master log file name and master log position that you determined in step 2. The following is an example.

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306, 'repl_user',  
'<password>', 'mysql-bin-changelog.000031', 107, 0);
```

Note

On Amazon RDS MySQL, you can choose to use delayed replication by running the [mysql.rds_set_external_master_with_delay \(p. 696\)](#) stored procedure instead.

One reason to use delayed replication is to enable disaster recovery with the [mysql.rds_start_replication_until \(p. 704\)](#) stored procedure. Currently, delayed replication is not supported on Amazon RDS MariaDB.

9. On the Amazon RDS DB instance, issue the [mysql.rds_start_replication \(p. 704\)](#) command to start replication:

```
CALL mysql.rds_start_replication;
```

Configuring GTID-Based Replication with an External Master Instance

When you set up an external replication master and a replica on Amazon RDS, monitor failover events for the Amazon RDS DB instance that is your replica. If a failover occurs, then the DB instance that is your replica might be recreated on a new host with a different network address. For information on how to monitor failover events, see [Using Amazon RDS Event Notification \(p. 287\)](#).

Important

GTID-based replication is only supported on Amazon RDS MySQL version 5.7.23 and later MySQL 5.7 versions. GTID-based replication is not supported for Amazon RDS MySQL 5.5, 5.6, or 8.0.

To configure GTID-based replication with an external master instance

1. Prepare for GTID-based replication:
 - a. Make sure that the external MySQL or MariaDB database has GTID-based replication enabled. To do so, make sure that the external database has the following parameters set to the specified values:

```
gtid_mode - ON
```

```
enforce_gtid_consistency - ON
```

For more information, see [Replication with Global Transaction Identifiers in the MySQL documentation](#) or [Global Transaction ID in the MariaDB documentation](#).

- b. Make sure that the parameter group associated with the DB instance has the following parameter settings:

- `gtid_mode` – ON, ON_PERMISSIVE, or OFF_PERMISSIVE
- `enforce_gtid_consistency` – ON

For more information about parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

- c. If you changed the parameter group of the DB instance, reboot the DB instance. For more information, see [Rebooting a DB Instance \(p. 129\)](#).

2. Make the source MySQL or MariaDB instance read-only.

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

3. Copy the database from the external instance to the Amazon RDS DB instance using `mysqldump`.

For very large databases, you might want to use the procedure in [Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime \(p. 640\)](#).

For Linux, OS X, or Unix:

```
mysqldump --databases <database_name> \
--single-transaction \
--compress \
--order-by-primary \
-u <local_user> \
-p<local_password> | mysql \
--host=hostname \
--port=3306 \
-u <RDS_user_name> \
-p<RDS_password>
```

For Windows:

```
mysqldump --databases <database_name> ^
--single-transaction ^
--compress ^
--order-by-primary ^
-u <local_user> ^
-p<local_password> | mysql ^
--host=hostname ^
--port=3306 ^
-u <RDS_user_name> ^
-p<RDS_password>
```

Note

Make sure that there is not a space between the `-p` option and the entered password.

To specify the host name, user name, port, and password to connect to your Amazon RDS DB instance, use the `--host`, `--user` (`-u`), `--port` and `-p` options in the `mysql` command.

The host name is the DNS name from the Amazon RDS DB instance endpoint, for example, `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the AWS Management Console.

4. Make the source MySQL or MariaDB instance writeable again.

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

For more information on making backups for use with replication, see [Backing Up a Master or Slave by Making It Read Only](#) in the MySQL documentation.

5. In the AWS Management Console, add the IP address of the server that hosts the external database to the VPC security group for the Amazon RDS DB instance. For more information on modifying a VPC security group, see [Security Groups for Your VPC](#) in the *Amazon Virtual Private Cloud User Guide*.

You might also need to configure your local network to permit connections from the IP address of your Amazon RDS DB instance, so that it can communicate with your external MySQL or MariaDB instance. To find the IP address of the Amazon RDS DB instance, use the host command.

```
host <RDS_MySQL_DB_host_name>
```

The host name is the DNS name from the Amazon RDS DB instance endpoint.

6. Using the client of your choice, connect to the external instance and create a user to use for replication. Use this account solely for replication. and restrict it to your domain to improve security. The following is an example.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';
```

7. For the external instance, grant REPLICATION CLIENT and REPLICATION SLAVE privileges to your replication user. For example, to grant the REPLICATION CLIENT and REPLICATION SLAVE privileges on all databases for the 'repl_user' user for your domain, issue the following command.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
IDENTIFIED BY '<password>';
```

8. Make the Amazon RDS DB instance the replica. To do so, connect to the Amazon RDS DB instance as the master user and identify the external MySQL or MariaDB database as the replication master by using the [mysql.rds_set_external_master_with_auto_position \(p. 698\)](#) command. The following is an example.

```
CALL mysql.rds_set_external_master_with_auto_position ('mymasterserver.mydomain.com',
3306, 'repl_user', '<password>', 0, 0);
```

Note

On Amazon RDS MySQL, you can choose to use delayed replication by running the [mysql.rds_set_external_master_with_delay \(p. 696\)](#) stored procedure instead.

One reason to use delayed replication is to enable disaster recovery with the [mysql.rds_start_replication_until_gtid \(p. 705\)](#) stored procedure. Currently, delayed replication is not supported on Amazon RDS MariaDB.

9. On the Amazon RDS DB instance, issue the [mysql.rds_start_replication \(p. 704\)](#) command to start replication.

```
CALL mysql.rds_start_replication;
```

Exporting Data from a MySQL DB Instance by Using Replication

You can use replication to export data from a MySQL 5.6 or later DB instance to a MySQL instance running external to Amazon RDS. The MySQL instance external to Amazon RDS can be running either on-premises in your data center, or on an Amazon EC2 instance. The MySQL DB instance must be running version 5.6.13 or later. The MySQL instance external to Amazon RDS must be running the same version as the Amazon RDS instance, or a later version.

Replication to an instance of MySQL running external to Amazon RDS is only supported during the time it takes to export a database from a MySQL DB instance. The replication should be terminated when the data has been exported and applications can start accessing the external instance.

The following list shows the steps to take. Each step is discussed in more detail in later sections.

1. Prepare an instance of MySQL running external to Amazon RDS.
2. Configure the MySQL DB instance to be the replication source.
3. Use `mysqldump` to transfer the database from the Amazon RDS instance to the instance external to Amazon RDS.
4. Start replication to the instance running external to Amazon RDS.
5. After the export completes, stop replication.

Prepare an Instance of MySQL External to Amazon RDS

Install an instance of MySQL external to Amazon RDS.

Connect to the instance as the master user, and create the users required to support the administrators, applications, and services that access the instance.

Follow the directions in the MySQL documentation to prepare the instance of MySQL running external to Amazon RDS as a replica. For more information, see [Setting the Replication Slave Configuration](#).

Configure an egress rule for the external instance to operate as a Read Replica during the export. The egress rule will allow the MySQL Read Replica to connect to the MySQL DB instance during replication. Specify an egress rule that allows TCP connections to the port and IP address of the source Amazon RDS MySQL DB instance.

If the Read Replica is running in an Amazon EC2 instance in an Amazon VPC, specify the egress rules in a VPC security group. If the Read Replica is running in an Amazon EC2 instance that is not in a VPC, specify the egress rule in an Amazon EC2 security group. If the Read Replica is installed on-premises, specify the egress rule in a firewall.

If the Read Replica is running in a VPC, configure VPC ACL rules in addition to the security group egress rule. For more information about Amazon VPC network ACLs, see [Network ACLs](#).

- ACL ingress rule allowing TCP traffic to ports 1024-65535 from the IP address of the source MySQL DB instance.
- ACL egress rule: allowing outbound TCP traffic to the port and IP address of the source MySQL DB instance.

Prepare the Replication Source

Prepare the MySQL DB instance as the replication source.

Ensure your client computer has enough disk space available to save the binary logs while setting up replication.

Create a replication account by following the directions in [Creating a User For Replication](#).

Configure ingress rules on the system running the replication source MySQL DB instance that will allow the external MySQL Read Replica to connect during replication. Specify an ingress rule that allows TCP connections to the port used by the Amazon RDS instance from the IP address of the MySQL Read Replica running external to Amazon RDS.

If the Amazon RDS instance is running in a VPC, specify the ingress rules in a VPC security group. If the Amazon RDS instance is not running in an in a VPC, specify the ingress rules in a database security group.

If the Amazon RDS instance is running in a VPC, configure VPC ACL rules in addition to the security group ingress rule. For more information about Amazon VPC network ACLs, see [Network ACLs](#).

- ACL ingress rule: allow TCP connections to the port used by the Amazon RDS instance from the IP address of the external MySQL Read Replica.
- ACL egress rule: allow TCP connections from ports 1024-65535 to the IP address of the external MySQL Read Replica.

Ensure that the backup retention period is set long enough that no binary logs are purged during the export. If any of the logs are purged before the export is complete, you must restart replication from the beginning. For more information about setting the backup retention period, see [Working With Backups \(p. 208\)](#).

Use the `mysql.rds_set_configuration` stored procedure to set the binary log retention period long enough that the binary logs are not purged during the export. For more information, see [Accessing MySQL Binary Logs \(p. 326\)](#).

To further ensure that the binary logs of the source instance are not purged, create an Amazon RDS Read Replica from the source instance. For more information, see [Creating a Read Replica \(p. 145\)](#). After the Amazon RDS Read Replica has been created, call the `mysql.rds_stop_replication` stored procedure to stop the replication process. The source instance will no longer purge its binary log files, so they will be available for the replication process.

Copy the Database

Run the MySQL `SHOW SLAVE STATUS` statement on the RDS read replica, and note the values for the following:

- `master_host`
- `master_port`
- `master_log_file`
- `exec_master_log_pos`

Use the `mysqldump` utility to create a snapshot, which copies the data from Amazon RDS to your local client computer. Then run another utility to load the data into the MySQL instance running external to RDS. Ensure your client computer has enough space to hold the `mysqldump` files from the databases to be replicated. This process can take several hours for very large databases. Follow the directions in [Creating a Dump Snapshot Using mysqldump](#).

The following example shows how to run `mysqldump` on a client, and then pipe the dump into the `mysql` client utility, which loads the data into the external MySQL instance.

For Linux, OS X, or Unix:

```
mysqldump -h RDS instance endpoint \
-u user \
-p password \
--port=3306 \
--single-transaction \
--routines \
--triggers \
--databases database database2 \
--compress \
--compact | mysql \
-h MySQL host \
-u master user \
-p password \
--port 3306
```

For Windows:

```
mysqldump -h RDS instance endpoint ^
-u user ^
-p password ^
--port=3306 ^
--single-transaction ^
--routines ^
--triggers ^
--databases database database2 ^
--compress ^
--compact | mysql ^
-h MySQL host ^
-u master user ^
-p password ^
--port 3306
```

The following example shows how to run `mysqldump` on a client and write the dump to a file.

For Linux, OS X, or Unix:

```
mysqldump -h RDS instance endpoint \
-u user \
-p password \
--port=3306 \
--single-transaction \
--routines \
--triggers \
--databases database database2 > path/rds-dump.sql
```

For Windows:

```
mysqldump -h RDS instance endpoint ^
-u user ^
-p password ^
--port=3306 ^
--single-transaction ^
--routines ^
--triggers ^
--databases database database2 > path\rds-dump.sql
```

Complete the Export

After you have loaded the `mysqldump` files to create the databases on the MySQL instance running external to Amazon RDS, start replication from the source MySQL DB instance to export all source changes that have occurred after you stopped replication from the Amazon RDS Read Replica.

Use the MySQL `CHANGE MASTER` statement to configure the external MySQL instance. Specify the ID and password of the user granted `REPLICATION SLAVE` permissions. Specify the `master_host`, `master_port`, `relay_master_log_file` and `exec_master_log_pos` values you got from the `Mysql SHOW SLAVE STATUS` statement you ran on the RDS Read Replica. For more information, see [Setting the Master Configuration on the Slave](#).

Use the MySQL `START SLAVE` command to initiate replication from the source MySQL DB instance and the MySQL replica.

Run the MySQL `SHOW SLAVE STATUS` command on the Amazon RDS instance to verify that it is operating as a Read Replica. For more information about interpreting the results, see [SHOW SLAVE STATUS Syntax](#).

After replication on the MySQL instance has caught up with the Amazon RDS source, use the MySQL `STOP SLAVE` command to terminate replication from the source MySQL DB instance.

On the Amazon RDS Read Replica, call the `mysql.rds_start_replication` stored procedure. This will allow Amazon RDS to start purging the binary log files from the source MySQL DB instance.

Related Topics

- [Restoring a Backup into an Amazon RDS MySQL DB Instance \(p. 631\)](#)
- [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#)

Options for MySQL DB Instances

This appendix describes options, or additional features, that are available for Amazon RDS instances running the MySQL DB engine. To enable these options, you can add them to a custom option group, and then associate the option group with your DB instance. For more information about working with option groups, see [Working with Option Groups \(p. 156\)](#).

Amazon RDS supports the following options for MySQL:

Option	Option ID	Engine Versions
MariaDB Audit Plugin Support (p. 678)	MARIADB_AUDIT_PLUGIN	All MySQL 5.6 versions
		MySQL 5.7.16 and later 5.7 versions
MySQL memcached Support (p. 681)	MEMCACHED	All MySQL 5.6, 5.7, and 8.0 versions

MariaDB Audit Plugin Support

Amazon RDS supports using the MariaDB Audit Plugin on MySQL database instances. The MariaDB Audit Plugin records database activity such as users logging on to the database, queries run against the database, and more. The record of database activity is stored in a log file.

Note

Currently, the MariaDB Audit Plugin is only supported for the following Amazon RDS MySQL versions:

- All 5.6 versions
 - MySQL 5.7.16 and later 5.7 versions

Audit Plugin Option Settings

Amazon RDS supports the following settings for the MariaDB Audit Plugin option.

Option Setting	Valid Values	Default Value	Description
SERVER_AUDIT	/rdsdbdata/ log/audit/	/rdsdbdata/ log/audit/	The location of the log file. The log file contains the record of the activity specified in SERVER_AUDIT_EVENTS. For more information, see Viewing and Listing Database Log Files (p. 307) and MySQL Database Log Files (p. 320) .
SERVER_AUDIT_SIZE	1000000000000000000000000		The size in bytes that when reached, causes the file to rotate. For more information, see Log File Size (p. 324) .
SERVER_AUDIT_ROTATIONS	100		The number of log rotations to save. For more information, see Log File Size (p. 324) and Downloading a Database Log File (p. 307) .
SERVER_AUDIT_EVENTS	CONNECT, QUERY	CONNECT, QUERY	<p>The types of activity to record in the log. Installing the MariaDB Audit Plugin is itself logged.</p> <ul style="list-style-type: none">• CONNECT: Log successful and unsuccessful connections to the database, and disconnections from the database.• QUERY: Log the text of all queries run against the database.• TABLE: Log tables affected by queries when the queries are run against the database. <p>For MariaDB, CONNECT, QUERY, and TABLE are supported.</p> <p>For MySQL, CONNECT and QUERY are supported.</p>
SERVER_AUDIT_USERS	Multiple comma-separated values	None	Include only activity from the specified users. By default, activity is recorded for all users. If a user is specified in both SERVER_AUDIT_EXCL_USERS and SERVER_AUDIT_INCL_USERS, then activity is recorded for the user.

Option Setting	Valid Values	Default Value	Description
SERVER_AUDIT_EXCL_USERS	Multiple comma-separated values	None	<p>Exclude activity from the specified users. By default, activity is recorded for all users. If a user is specified in both SERVER_AUDIT_EXCL_USERS and SERVER_AUDIT_INCL_USERS, then activity is recorded for the user.</p> <p>The rdsadmin user queries the database every second to check the health of the database. Depending on your other settings, this activity can possibly cause the size of your log file to grow very large, very quickly. If you don't need to record this activity, add the rdsadmin user to the SERVER_AUDIT_EXCL_USERS list.</p> <p>Note CONNECT activity is always recorded for all users, even if the user is specified for this option setting.</p>
SERVER_AUDIT_LOGGING	ON	ON	Logging is active. The only valid value is ON. Amazon RDS does not support deactivating logging. If you want to deactivate logging, remove the MariaDB Audit Plugin. For more information, see Removing the MariaDB Audit Plugin (p. 680) .

Adding the MariaDB Audit Plugin

The general process for adding the MariaDB Audit Plugin to a DB instance is the following:

- Create a new option group, or copy or modify an existing option group
- Add the option to the option group
- Associate the option group with the DB instance

After you add the MariaDB Audit Plugin, you don't need to restart your DB instance. As soon as the option group is active, auditing begins immediately.

To add the MariaDB Audit Plugin

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group. Choose **mysql** for **Engine**, and choose **5.6** or **5.7** for **Major engine version**. For more information, see [Creating an Option Group \(p. 157\)](#).
2. Add the **MARIADB_AUDIT_PLUGIN** option to the option group, and configure the option settings. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#). For more information about each setting, see [Audit Plugin Option Settings \(p. 678\)](#).
3. Apply the option group to a new or existing DB instance.
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the MySQL Database Engine \(p. 597\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#).

Viewing and Downloading the MariaDB Audit Plugin Log

After you enable the MariaDB Audit Plugin, you access the results in the log files the same way you access any other text-based log files. The audit log files are located at `/rdsdbdata/log/audit/`. For information about viewing the log file in the console, see [Viewing and Listing Database Log Files \(p. 307\)](#). For information about downloading the log file, see [Downloading a Database Log File \(p. 307\)](#).

Modifying MariaDB Audit Plugin Settings

After you enable the MariaDB Audit Plugin, you can modify the settings. For more information about how to modify option settings, see [Modifying an Option Setting \(p. 164\)](#). For more information about each setting, see [Audit Plugin Option Settings \(p. 678\)](#).

Removing the MariaDB Audit Plugin

Amazon RDS doesn't support turning off logging in the MariaDB Audit Plugin. However, you can remove the plugin from a DB instance. When you remove the MariaDB Audit Plugin, the DB instance is restarted automatically to stop auditing.

To remove the MariaDB Audit Plugin from a DB instance, do one of the following:

- Remove the MariaDB Audit Plugin option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#)
- Modify the DB instance and specify a different option group that doesn't include the plugin. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#).

MySQL memcached Support

Amazon RDS supports using the memcached interface to InnoDB tables that was introduced in MySQL 5.6. The memcached API enables applications to use InnoDB tables in a manner similar to NoSQL key-value data stores.

The memcached interface is a simple, key-based cache. Applications use memcached to insert, manipulate, and retrieve key-value data pairs from the cache. MySQL 5.6 introduced a plugin that implements a daemon service that exposes data from InnoDB tables through the memcached protocol. For more information about the MySQL memcached plugin, see [InnoDB Integration with memcached](#).

To enable memcached support for an Amazon RDS MySQL 5.6 or later instance

1. Determine the security group to use for controlling access to the memcached interface. If the set of applications already using the SQL interface are the same set that will access the memcached interface, you can use the existing VPC or DB security group used by the SQL interface. If a different set of applications will access the memcached interface, define a new VPC or DB security group. For more information about managing security groups, see [Controlling Access with Security Groups \(p. 394\)](#).
2. Create a custom DB option group, selecting MySQL as the engine type and a 5.6 or later version. For more information about creating an option group, see [Creating an Option Group \(p. 157\)](#).
3. Add the MEMCACHED option to the option group. Specify the port that the memcached interface will use, and the security group to use in controlling access to the interface. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).
4. Modify the option settings to configure the memcached parameters, if necessary. For more information about how to modify option settings, see [Modifying an Option Setting \(p. 164\)](#).
5. Apply the option group to an instance. Amazon RDS enables memcached support for that instance when the option group is applied:
 - You enable memcached support for a new instance by specifying the custom option group when you launch the instance. For more information about launching a MySQL instance, see [Creating a DB Instance Running the MySQL Database Engine \(p. 597\)](#).
 - You enable memcached support for an existing instance by specifying the custom option group when you modify the instance. For more information about modifying a MySQL instance, see [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#).
6. Specify which columns in your MySQL tables can be accessed through the memcached interface. The memcached plug-in creates a catalog table named `containers` in a dedicated database named `innodb_memcache`. You insert a row into the `containers` table to map an InnoDB table for access through memcached. You specify a column in the InnoDB table that is used to store the memcached key values, and one or more columns that are used to store the data values associated with the key. You also specify a name that a memcached application uses to refer to that set of columns. For details on inserting rows in the `containers` table, see [Internals of the InnoDB memcached Plugin](#). For an example of mapping an InnoDB table and accessing it through memcached, see [Specifying the Table and Column Mappings for an InnoDB + memcached Application](#).
7. If the applications accessing the memcached interface are on different computers or EC2 instances than the applications using the SQL interface, add the connection information for those computers to the VPC or DB security group associated with the MySQL instance. For more information about managing security groups, see [Controlling Access with Security Groups \(p. 394\)](#).

You turn off the memcached support for an instance by modifying the instance and specifying the default option group for your MySQL version. For more information about modifying a MySQL instance, see [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#).

MySQL memcached Security Considerations

The memcached protocol does not support user authentication. For more information about MySQL memcached security considerations, see [memcached Deployment](#) and [Using memcached as a MySQL Caching Layer](#).

You can take the following actions to help increase the security of the memcached interface:

- Specify a different port than the default of 11211 when adding the **MEMCACHED** option to the option group.
- Ensure that you associate the memcached interface with either a VPC or DB security group that limits access to known, trusted client addresses or EC2 instances. For more information about managing security groups, see [Controlling Access with Security Groups \(p. 394\)](#).

MySQL memcached Connection Information

To access the memcached interface, an application must specify both the DNS name of the Amazon RDS instance and the memcached port number. For example, if an instance has a DNS name of `my-cache-instance.cg034hpkmmt.region.rds.amazonaws.com` and the memcached interface is using port 11212, the connection information specified in PHP would be:

```
<?php  
  
$cache = new Memcache;  
$cache->connect('my-cache-instance.cg034hpkmmt.region.rds.amazonaws.com', 11212);  
?>
```

To find the DNS name and memcached port of an Amazon RDS MySQL instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the AWS Management Console, select the region that contains the DB instance.
3. In the navigation pane, choose **Databases**.
4. Choose the MySQL DB instance name to display its details.
5. In the **Connect** section, note the value of the **Endpoint** field. The DNS name is the same as the endpoint. Also, note that the port in the **Connect** section is not used to access the memcached interface.
6. In the **Details** section, note the name listed in the **Option Group** field.
7. In the navigation pane, choose **Option groups**.
8. Choose the name of the option group used by the MySQL DB instance to show the option group details. In the **Options** section, note the value of the **Port** setting for the **MEMCACHED** option.

MySQL memcached Option Settings

Amazon RDS exposes the MySQL memcached parameters as option settings in the Amazon RDS **MEMCACHED** option.

MySQL memcached Parameters

- **DAEMON_MEMCACHED_R_BATCH_SIZE** – an integer that specifies how many memcached read operations (get) to perform before doing a COMMIT to start a new transaction. The allowed values are 1 to 4294967295; the default is 1. The option does not take effect until the instance is restarted.

- **DAEMON_MEMCACHED_W_BATCH_SIZE** – an integer that specifies how many memcached write operations, such as add, set, or incr, to perform before doing a COMMIT to start a new transaction. The allowed values are 1 to 4294967295; the default is 1. The option does not take effect until the instance is restarted.
- **INNODB_API_BK_COMMIT_INTERVAL** – an integer that specifies how often to auto-commit idle connections that use the InnoDB memcached interface. The allowed values are 1 to 1073741824; the default is 5. The option takes effect immediately, without requiring that you restart the instance.
- **INNODB_API_DISABLE_ROWLOCK** – a Boolean that disables (1 (true)) or enables (0 (false)) the use of row locks when using the InnoDB memcached interface. The default is 0 (false). The option does not take effect until the instance is restarted.
- **INNODB_API_ENABLE_MDL** – a Boolean that when set to 0 (false) locks the table used by the InnoDB memcached plugin, so that it cannot be dropped or altered by DDL through the SQL interface. The default is 0 (false). The option does not take effect until the instance is restarted.
- **INNODB_API_TRX_LEVEL** – an integer that specifies the transaction isolation level for queries processed by the memcached interface. The allowed values are 0 to 3. The default is 0. The option does not take effect until the instance is restarted.

Amazon RDS configures these MySQL memcached parameters, and they cannot be modified: DAEMON_MEMCACHED_LIB_NAME, DAEMON_MEMCACHED_LIB_PATH, and INNODB_API_ENABLE_BINLOG. The parameters that MySQL administrators set by using `daemon_memcached_options` are available as individual MEMCACHED option settings in Amazon RDS.

MySQL `daemon_memcached_options` Parameters

- **BINDING_PROTOCOL** – a string that specifies the binding protocol to use. The allowed values are auto, ascii, or binary. The default is auto, which means the server automatically negotiates the protocol with the client. The option does not take effect until the instance is restarted.
- **BACKLOG_QUEUE_LIMIT** – an integer that specifies how many network connections can be waiting to be processed by memcached. Increasing this limit may reduce errors received by a client that is not able to connect to the memcached instance, but does not improve the performance of the server. The allowed values are 1 to 2048; the default is 1024. The option does not take effect until the instance is restarted.
- **CAS_DISABLED** – a Boolean that enables (1 (true)) or disables (0 (false)) the use of compare and swap (CAS), which reduces the per-item size by 8 bytes. The default is 0 (false). The option does not take effect until the instance is restarted.
- **CHUNK_SIZE** – an integer that specifies the minimum chunk size, in bytes, to allocate for the smallest item's key, value, and flags. The allowed values are 1 to 48. The default is 48 and you can significantly improve memory efficiency with a lower value. The option does not take effect until the instance is restarted.
- **CHUNK_SIZE_GROWTH_FACTOR** – a float that controls the size of new chunks. The size of a new chunk is the size of the previous chunk times CHUNK_SIZE_GROWTH_FACTOR. The allowed values are 1 to 2; the default is 1.25. The option does not take effect until the instance is restarted.
- **ERROR_ON_MEMORY_EXHAUSTED** – a Boolean that when set to 1 (true) specifies that memcached will return an error rather than evicting items when there is no more memory to store items. If set to 0 (false), memcached will evict items if there is no more memory. The default is 0 (false). The option does not take effect until the instance is restarted.
- **MAX_SIMULTANEOUS_CONNECTIONS** – an integer that specifies the maximum number of concurrent connections. Setting this value to anything under 10 prevents MySQL from starting. The allowed values are 10 to 1024; the default is 1024. The option does not take effect until the instance is restarted.
- **VERBOSITY** – a string that specifies the level of information logged in the MySQL error log by the memcached service. The default is v. The option does not take effect until the instance is restarted. The allowed values are:

- **v** – Logs errors and warnings while executing the main event loop.
- **vv** – In addition to the information logged by v, also logs each client command and the response.
- **vvv** – In addition to the information logged by vv, also logs internal state transitions.

Amazon RDS configures these MySQL DAEMON_MEMCACHED_OPTIONS parameters, they cannot be modified: DAEMON_PROCESS, LARGE_MEMORY_PAGES, MAXIMUM_CORE_FILE_LIMIT, MAX_ITEM_SIZE, LOCK_DOWN_PAGE_MEMORY, MASK, IDFILE, REQUESTS_PER_EVENT, SOCKET, and USER.

Common DBA Tasks for MySQL DB Instances

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the MySQL database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.

For information about working with MySQL log files on Amazon RDS, see [MySQL Database Log Files \(p. 320\)](#)

Topics

- [Killing a Session or Query \(p. 685\)](#)
- [Skipping the Current Replication Error \(p. 685\)](#)
- [Working with InnoDB Tablespaces to Improve Crash Recovery Times \(p. 686\)](#)
- [Managing the Global Status History \(p. 687\)](#)

Killing a Session or Query

You can terminate user sessions or queries on DB instances by using the `rds_kill` and `rds_kill_query` commands. First connect to your MySQL database instance, then issue the appropriate command as shown following. For more information, see [Connecting to a DB Instance Running the MySQL Database Engine \(p. 606\)](#).

```
CALL mysql.rds_kill(thread-ID)
CALL mysql.rds_kill_query(thread-ID)
```

For example, to kill the session that is running on thread 99, you would type the following:

```
CALL mysql.rds_kill(99);
```

To kill the query that is running on thread 99, you would type the following:

```
CALL mysql.rds_kill_query(99);
```

Skipping the Current Replication Error

Amazon RDS provides a mechanism for you to skip an error on your Read Replicas if the error is causing your Read Replica to hang and the error doesn't affect the integrity of your data. First connect to your MySQL database instance, then issue the appropriate commands as shown following. For more information, see [Connecting to a DB Instance Running the MySQL Database Engine \(p. 606\)](#).

Note

You should first verify that the error can be safely skipped. In a MySQL utility, connect to the Read Replica and run the following MySQL command:

```
SHOW SLAVE STATUS\G
```

For information about the values returned, go to [SHOW SLAVE STATUS Syntax](#) in the MySQL documentation.

To skip the error, you can issue the following command:

```
CALL mysql.rds_skip_repl_error;
```

This command has no effect if you run it on the source DB instance, or on a Read Replica that has not encountered a replication error.

For more information, such as the versions of MySQL that support `mysql.rds_skip_repl_error`, see [mysql.rds_skip_repl_error \(p. 707\)](#).

Important

If you attempt to call `mysql.rds_skip_repl_error` and encounter the following error: `ERROR 1305 (42000): PROCEDURE mysql.rds_skip_repl_error does not exist`, then upgrade your MySQL DB instance to the latest minor version or one of the minimum minor versions listed in [mysql.rds_skip_repl_error \(p. 707\)](#).

Working with InnoDB Tablespaces to Improve Crash Recovery Times

Every table in MySQL consists of a table definition, data, and indexes. The MySQL storage engine InnoDB stores table data and indexes in a *tablespace*. InnoDB creates a global shared tablespace that contains a data dictionary and other relevant metadata, and it can contain table data and indexes. InnoDB can also create separate tablespaces for each table and partition. These separate tablespaces are stored in files with a .ibd extension and the header of each tablespace contains a number that uniquely identifies it.

Amazon RDS provides a parameter in a MySQL parameter group called `innodb_file_per_table`. This parameter controls whether InnoDB adds new table data and indexes to the shared tablespace (by setting the parameter value to 0) or to individual tablespaces (by setting the parameter value to 1). Amazon RDS sets the default value for `innodb_file_per_table` parameter to 1, which allows you to drop individual InnoDB tables and reclaim storage used by those tables for the DB instance. In most use cases, setting the `innodb_file_per_table` parameter to 1 is the recommended setting.

You should set the `innodb_file_per_table` parameter to 0 when you have a large number of tables, such as over 1000 tables when you use standard (magnetic) or general purpose SSD storage or over 10,000 tables when you use Provisioned IOPS storage. When you set this parameter to 0, individual tablespaces are not created and this can improve the time it takes for database crash recovery.

MySQL processes each metadata file, which includes tablespaces, during the crash recovery cycle. The time it takes MySQL to process the metadata information in the shared tablespace is negligible compared to the time it takes to process thousands of tablespace files when there are multiple tablespaces. Because the tablespace number is stored within the header of each file, the aggregate time to read all the tablespace files can take up to several hours. For example, a million InnoDB tablespaces on standard storage can take from five to eight hours to process during a crash recovery cycle. In some cases, InnoDB can determine that it needs additional cleanup after a crash recovery cycle so it will begin another crash recovery cycle, which will extend the recovery time. Keep in mind that a crash recovery cycle also entails rolling-back transactions, fixing broken pages, and other operations in addition to the processing of tablespace information.

Since the `innodb_file_per_table` parameter resides in a parameter group, you can change the parameter value by editing the parameter group used by your DB instance without having to reboot the DB instance. After the setting is changed, for example, from 1 (create individual tables) to 0 (use shared tablespace), new InnoDB tables will be added to the shared tablespace while existing tables continue to have individual tablespaces. To move an InnoDB table to the shared tablespace, you must use the `ALTER TABLE` command.

Migrating Multiple Tablespaces to the Shared Tablespace

You can move an InnoDB table's metadata from its own tablespace to the shared tablespace, which will rebuild the table metadata according to the `innodb_file_per_table` parameter setting. First connect

to your MySQL database instance, then issue the appropriate commands as shown following. For more information, see [Connecting to a DB Instance Running the MySQL Database Engine \(p. 606\)](#).

```
ALTER TABLE table_name ENGINE = InnoDB, ALGORITHM=COPY;
```

For example, the following query returns an `ALTER TABLE` statement for every InnoDB table that is not in the shared tablespace.

```
SELECT CONCAT('ALTER TABLE `',
REPLACE(LEFT(NAME , INSTR((NAME), '/') - 1), ``, ``), ``.``,
REPLACE(SUBSTR(NAME FROM INSTR(NAME, '/') + 1), ``, ``), `` ENGINE=InnoDB,
ALGORITHM=COPY;') AS Query
FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES
WHERE SPACE <> 0 AND LEFT(NAME, INSTR((NAME), '/') - 1) NOT IN ('mysql','');
```

Note

This query is supported on MySQL 5.6 and later.

Rebuilding a MySQL table to move the table's metadata to the shared tablespace requires additional storage space temporarily to rebuild the table, so the DB instance must have storage space available. During rebuilding, the table is locked and inaccessible to queries. For small tables or tables not frequently accessed, this may not be an issue; for large tables or tables frequently accessed in a heavily concurrent environment, you can rebuild tables on a Read Replica.

You can create a Read Replica and migrate table metadata to the shared tablespace on the Read Replica. While the `ALTER TABLE` statement blocks access on the Read Replica, the source DB instance is not affected. The source DB instance will continue to generate its binary logs while the Read Replica lags during the table rebuilding process. Because the rebuilding requires additional storage space and the replay log file can become large, you should create a Read Replica with storage allocated that is larger than the source DB instance.

The following steps should be followed to create a Read Replica and rebuild InnoDB tables to use the shared tablespace:

1. Ensure that backup retention is enabled on the source DB instance so that binary logging is enabled
2. Use the AWS Console or AWS CLI to create a Read Replica for the source DB instance. Since the creation of a Read Replica involves many of the same processes as crash recovery, the creation process may take some time if there are a large number of InnoDB tablespaces. Allocate more storage space on the Read Replica than is currently used on the source DB instance.
3. When the Read Replica has been created, create a parameter group with the parameter settings `read_only = 0` and `innodb_file_per_table = 0`, and then associate the parameter group with the Read Replica.
4. Issue `ALTER TABLE <name> ENGINE = InnoDB` against all tables you want migrated on the replica.
5. When all of your `ALTER TABLE` statements have completed on the Read Replica, verify that the Read Replica is connected to the source DB instance and that the two instances are in-sync.
6. When ready, use the AWS Console or AWS CLI to promote the Read Replica to be the master instance. Make sure that the parameter group used for the new master has the `innodb_file_per_table` parameter set to 0. Change the name of the new master, and point any applications to the new master instance.

Managing the Global Status History

MySQL maintains many status variables that provide information about its operation. Their value can help you detect locking or memory issues on a DB instance . The values of these status variables are

cumulative since last time the DB instance was started. You can reset most status variables to 0 by using the `FLUSH STATUS` command.

To allow for monitoring of these values over time, Amazon RDS provides a set of procedures that will snapshot the values of these status variables over time and write them to a table, along with any changes since the last snapshot. This infrastructure, called Global Status History (GoSH), is installed on all MySQL DB instances starting with versions 5.5.23. GoSH is disabled by default.

To enable GoSH, you first enable the event scheduler from a DB parameter group by setting the parameter `event_scheduler` to ON. For information about creating and modifying a DB parameter group, see [Working with DB Parameter Groups \(p. 169\)](#).

You can then use the procedures in the following table to enable and configure GoSH. First connect to your MySQL database instance, then issue the appropriate commands as shown following. For more information, see [Connecting to a DB Instance Running the MySQL Database Engine \(p. 606\)](#). For each procedure, type the following:

```
CALL procedure-name;
```

Where *procedure-name* is one of the procedures in the table.

Procedure	Description
<code>rds_enable_gsh_collector</code>	Enables GoSH to take default snapshots at intervals specified by <code>rds_set_gsh_collector</code> .
<code>rds_set_gsh_collector</code>	Specifies the interval, in minutes, between snapshots. Default value is 5.
<code>rds_disable_gsh_collector</code>	Disables snapshots.
<code>rds_collect_global_status_history</code>	Takes a snapshot on demand.
<code>rds_enable_gsh_rotation</code>	Enables rotation of the contents of the <code>mysql.rds_global_status_history</code> table to <code>mysql.rds_global_status_history_old</code> at intervals specified by <code>rds_set_gsh_rotation</code> .
<code>rds_set_gsh_rotation</code>	Specifies the interval, in days, between table rotations. Default value is 7.
<code>rds_disable_gsh_rotation</code>	Disables table rotation.
<code>rds_rotate_global_status_history</code>	Rotates the contents of the <code>mysql.rds_global_status_history</code> table to <code>mysql.rds_global_status_history_old</code> on demand.

When GoSH is running, you can query the tables that it writes to. For example, to query the hit ratio of the Innodb buffer pool, you would issue the following query:

```
select a.collection_end, a.collection_start, (( a.variable_Delta-b.variable_delta)/
a.variable_delta)*100 as "HitRatio"
  from mysql.rds_global_status_history as a join mysql.rds_global_status_history as b on
a.collection_end = b.collection_end
    where a.variable_name = 'Innodb_buffer_pool_read_requests' and b.variable_name =
'Innodb_buffer_pool_reads'
```

Known Issues and Limitations for MySQL on Amazon RDS

Known issues and limitations for working with MySQL on Amazon RDS are as follows.

Inconsistent InnoDB Buffer Pool Size

For MySQL 5.7, there is currently a bug in the way that the InnoDB buffer pool size is managed. MySQL 5.7 might adjust the value of the `innodb_buffer_pool_size` parameter to a large value that can result in the InnoDB buffer pool growing too large and using up too much memory. This effect can cause the MySQL database engine to stop running or can prevent the MySQL database engine from starting. This issue is more common for DB instance classes that have less memory available.

To resolve this issue, set the value of the `innodb_buffer_pool_size` parameter to a multiple of the product of the `innodb_buffer_pool_instances` parameter value and the `innodb_buffer_pool_chunk_size` parameter value. For example, you might set the `innodb_buffer_pool_size` parameter value to a multiple of eight times the product of the `innodb_buffer_pool_instances` and `innodb_buffer_pool_chunk_size` parameter values, as shown in the following example.

```
innodb_buffer_pool_chunk_size = 536870912
innodb_buffer_pool_instances = 4
innodb_buffer_pool_size = (536870912 * 4) * 8 = 17179869184
```

For details on this MySQL 5.7 bug, go to <https://bugs.mysql.com/bug.php?id=79379> in the MySQL documentation.

Index Merge Optimization Returns Wrong Results

Queries that use index merge optimization might return wrong results due to a bug in the MySQL query optimizer that was introduced in MySQL 5.5.37. When you issue a query against a table with multiple indexes the optimizer scans ranges of rows based on the multiple indexes, but does not merge the results together correctly. For more information on the query optimizer bug, go to <http://bugs.mysql.com/bug.php?id=72745> and <http://bugs.mysql.com/bug.php?id=68194> in the MySQL bug database.

For example, consider a query on a table with two indexes where the search arguments reference the indexed columns.

```
SELECT * FROM table1
WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

In this case, the search engine will search both indexes. However, due to the bug, the merged results are incorrect.

To resolve this issue, you can do one of the following:

- Set the `optimizer_switch` parameter to `index_merge=off` in the DB parameter group for your MySQL DB instance. For information on setting DB parameter group parameters, see [Working with DB Parameter Groups \(p. 169\)](#).
- Upgrade your MySQL DB instance to MySQL version 5.6, 5.7, or 8.0. For more information, see [Upgrading a MySQL DB Snapshot \(p. 623\)](#).
- If you cannot upgrade your instance or change the `optimizer_switch` parameter, you can work around the bug by explicitly identifying an index for the query, for example:

```
SELECT * FROM table1
USE INDEX covering_index
WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

For more information, go to [Index Merge Optimization](#).

Log File Size

For MySQL, there is a size limit on BLOBS written to the redo log. To account for this limit, ensure that the `innodb_log_file_size` parameter for your MySQL DB instance is 10 times larger than the largest BLOB data size found in your tables, plus the length of other variable length fields (VARCHAR, VARBINARY, TEXT) in the same tables. For information on how to set parameter values, see [Working with DB Parameter Groups \(p. 169\)](#). For information on the redo log BLOB size limit, go to [Changes in MySQL 5.6.20](#).

MySQL Parameter Exceptions for Amazon RDS DB Instances

Some MySQL parameters require special considerations when used with an Amazon RDS DB instance.

[lower_case_table_names](#)

Because Amazon RDS uses a case-sensitive file system, setting the value of the `lower_case_table_names` server parameter to 2 ("names stored as given but compared in lowercase") is not supported. Supported values for Amazon RDS DB instances are 0 ("names stored as given and comparisons are case-sensitive"), which is the default, or 1 ("names stored in lowercase and comparisons are not case-sensitive").

The `lower_case_table_names` parameter should be set as part of a custom DB parameter group before creating a DB instance. You should avoid changing the `lower_case_table_names` parameter for existing database instances because doing so could cause inconsistencies with point-in-time recovery backups and Read Replica DB instances.

Read Replicas should always use the same `lower_case_table_names` parameter value as the master DB instance.

[long_query_time](#)

You can set the `long_query_time` parameter to a floating point value which allows you to log slow queries to the MySQL slow query log with microsecond resolution. You can set a value such as 0.1 seconds, which would be 100 milliseconds, to help when debugging slow transactions that take less than one second.

MySQL File Size Limits

For Amazon RDS MySQL DB instances, the maximum provisioned storage limit constrains the size of a table to a maximum size of 16 TB when using InnoDB file-per-table tablespaces. This limit also constrains the system tablespace to a maximum size of 16 TB. InnoDB file-per-table tablespaces (with tables each in their own tablespace) is set by default for Amazon RDS MySQL DB instances.

Note

Some existing DB instances have a lower limit. For example, MySQL DB instances created prior to April 2014 have a file and table size limit of 2 TB. This 2 TB file size limit also applies to DB

instances or Read Replicas created from DB snapshots taken prior to April 2014, regardless of when the DB instance was created.

There are advantages and disadvantages to using InnoDB file-per-table tablespaces, depending on your application. To determine the best approach for your application, go to [InnoDB File-Per-Table Mode](#) in the MySQL documentation.

We don't recommend allowing tables to grow to the maximum file size. In general, a better practice is to partition data into smaller tables, which can improve performance and recovery times.

One option that you can use for breaking a large table up into smaller tables is partitioning. Partitioning distributes portions of your large table into separate files based on rules that you specify. For example, if you store transactions by date, you can create partitioning rules that distribute older transactions into separate files using partitioning. Then periodically, you can archive the historical transaction data that doesn't need to be readily available to your application. For more information, go to <https://dev.mysql.com/doc/refman/5.6/en/partitioning.html> in the MySQL documentation.

To determine the file size of a table

- Use the following SQL command to determine if any of your tables are too large and are candidates for partitioning.

```
SELECT TABLE_SCHEMA, TABLE_NAME,
round(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) As "Approximate size (MB)"
FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema');
```

To enable InnoDB file-per-table tablespaces

- To enable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to 1 in the parameter group for the DB instance.

To disable InnoDB file-per-table tablespaces

- To disable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to 0 in the parameter group for the DB instance.

For information on updating a parameter group, see [Working with DB Parameter Groups \(p. 169\)](#).

When you have enabled or disabled InnoDB file-per-table tablespaces, you can issue an `ALTER TABLE` command to move a table from the global tablespace to its own tablespace, or from its own tablespace to the global tablespace as shown in the following example:

```
ALTER TABLE table_name ENGINE=InnoDB;
```

MySQL on Amazon RDS SQL Reference

This appendix describes system stored procedures that are available for Amazon RDS instances running the MySQL DB engine.

Overview

The following system stored procedures are supported for Amazon RDS DB instances running MySQL.

Replication

- [mysql.rds_set_master_auto_position \(p. 693\)](#)
- [mysql.rds_set_external_master \(p. 694\)](#)
- [mysql.rds_set_external_master_with_delay \(p. 696\)](#)
- [mysql.rds_set_external_master_with_auto_position \(p. 698\)](#)
- [mysql.rds_reset_external_master \(p. 700\)](#)
- [mysql.rds_import_binlog_ssl_material \(p. 701\)](#)
- [mysql.rds_remove_binlog_ssl_material \(p. 702\)](#)
- [mysql.rds_set_source_delay \(p. 703\)](#)
- [mysql.rds_start_replication \(p. 704\)](#)
- [mysql.rds_start_replication_until \(p. 704\)](#)
- [mysql.rds_start_replication_until_gtid \(p. 705\)](#)
- [mysql.rds_stop_replication \(p. 706\)](#)
- [mysql.rds_skip_transaction_with_gtid \(p. 707\)](#)
- [mysql.rds_skip_repl_error \(p. 707\)](#)
- [mysql.rds_next_master_log \(p. 708\)](#)

InnoDB cache warming

- [mysql.rds_innodb_buffer_pool_dump_now \(p. 710\)](#)
- [mysql.rds_innodb_buffer_pool_load_now \(p. 710\)](#)
- [mysql.rds_innodb_buffer_pool_load_abort \(p. 710\)](#)

Managing additional configuration (for example, binlog file retention)

- [mysql.rds_set_configuration \(p. 711\)](#)
- [mysql.rds_show_configuration \(p. 712\)](#)

Terminating a session or query

- [mysql.rds_kill \(p. 713\)](#)
- [mysql.rds_kill_query \(p. 713\)](#)

Logging

- [mysql.rds_rotate_general_log \(p. 714\)](#)
- [mysql.rds_rotate_slow_log \(p. 714\)](#)

Managing the global status history

- [mysql.rds_enable_gsh_collector \(p. 714\)](#)
- [mysql.rds_set_gsh_collector \(p. 715\)](#)
- [mysql.rds_disable_gsh_collector \(p. 715\)](#)
- [mysql.rds_collect_global_status_history \(p. 715\)](#)
- [mysql.rds_enable_gsh_rotation \(p. 715\)](#)
- [mysql.rds_set_gsh_rotation \(p. 716\)](#)
- [mysql.rds_disable_gsh_rotation \(p. 716\)](#)
- [mysql.rds_rotate_global_status_history \(p. 716\)](#)

SQL Reference Conventions

Following, you can find explanations for the conventions that are used to describe the syntax of the system stored procedures and tables described in the SQL reference section.

Character	Description
UPPERCASE	Words in uppercase are keywords.
[]	Square brackets indicate optional arguments.
{ }	Braces indicate that you are required to choose one of the arguments inside the braces.
	Pipes separate arguments that you can choose.
<i>italics</i>	Words in italics indicate placeholders. You must insert the appropriate value in place of the word in <i>italics</i> .
...	An ellipsis indicates that you can repeat the preceding element.
'	Words in single quotes indicate that you must type the quotes.

mysql.rds_set_master_auto_position

Sets the replication mode to be based on either binary log file positions or on global transaction identifiers (GTIDs).

Syntax

```
CALL mysql.rds_set_master_auto_position (
    auto_position_mode
);
```

Parameters

auto_position_mode

A value that indicates whether to use log file position replication or GTID-based replication:

- 0 – Use the replication method based on binary log file position. The default is 0.

- 1 – Use the GTID-based replication method.

Usage Notes

The master user must run the `mysql.rds_set_master_auto_position` procedure.

For Amazon RDS MySQL 5.7, this procedure is supported for MySQL 5.7.23 and later MySQL 5.7 versions. This procedure is not supported for Amazon RDS MySQL 5.5, 5.6, or 8.0.

mysql.rds_set_external_master

Configures a MySQL DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS.

Note

You can use the [mysql.rds_set_external_master_with_delay \(p. 696\)](#) stored procedure to configure an external master and delayed replication.

Syntax

```
CALL mysql.rds_set_external_master (
    host_name
    , host_port
    , replication_user_name
    , replication_user_password
    , mysql_binary_log_file_name
    , mysql_binary_log_file_location
    , ssl_encryption
);
```

Parameters

host_name

The host name or IP address of the MySQL instance running external to Amazon RDS to become the replication master.

host_port

The port used by the MySQL instance running external to Amazon RDS to be configured as the replication master. If your network configuration includes Secure Shell (SSH) port replication that converts the port number, specify the port number that is exposed by SSH.

replication_user_name

The ID of a user with REPLICATION CLIENT and REPLICATION SLAVE permissions on the MySQL instance running external to Amazon RDS. We recommend that you provide an account that is used solely for replication with the external instance.

replication_user_password

The password of the user ID specified in `replication_user_name`.

mysql_binary_log_file_name

The name of the binary log on the replication master that contains the replication information.

mysql_binary_log_file_location

The location in the `mysql_binary_log_file_name` binary log at which replication starts reading the replication information.

ssl_encryption

A value that specifies whether Secure Socket Layer (SSL) encryption is used on the replication connection. 1 specifies to use SSL encryption, 0 specifies to not use encryption. The default is 0.

Note

This parameter currently is only implemented for Amazon Aurora with MySQL compatibility. On MySQL DB instances, only the default is allowed.

Usage Notes

The master user must run the `mysql.rds_set_external_master` procedure. This procedure must be run on the MySQL DB instance to be configured as the Read Replica of a MySQL instance running external to Amazon RDS.

Before you run `mysql.rds_set_external_master`, you must configure the instance of MySQL running external to Amazon RDS to be a replication master. To connect to the MySQL instance running external to Amazon RDS, you must specify `replication_user_name` and `replication_user_password` values that indicate a replication user that has `REPLICATION CLIENT` and `REPLICATION SLAVE` permissions on the external instance of MySQL.

To configure an external instance of MySQL as a replication master

1. Using the MySQL client of your choice, connect to the external instance of MySQL and create a user account to be used for replication. The following is an example.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password'
```

2. On the external instance of MySQL, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. The following example grants `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the 'repl_user' user for your domain.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'  
IDENTIFIED BY 'password'
```

For more information, see [Replication with a MySQL or MariaDB Instance Running External to Amazon RDS \(p. 667\)](#).

To use encrypted replication, configure the master to use SSL connections. Also, import the certificate authority certificate, client certificate, and client key into the DB instance or DB cluster using the `mysql.rds_import_binlog_ssl_material` (p. 701) procedure.

Note

We recommend that you use Read Replicas to manage replication between two Amazon RDS DB instances when possible. When you do so, we recommend that you use only this and other replication-related stored procedures. These practices enable more complex replication topologies between Amazon RDS DB instances. We offer these stored procedures primarily to enable replication with MySQL instances running external to Amazon RDS. For information about managing replication between Amazon RDS DB instances, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

After calling `mysql.rds_set_external_master` to configure an Amazon RDS DB instance as a Read Replica, you can call [mysql.rds_start_replication](#) (p. 704) on the Read Replica to start the replication process. You can call [mysql.rds_reset_external_master](#) (p. 700) to remove the Read Replica configuration.

When `mysql.rds_set_external_master` is called, Amazon RDS records the time, user, and an action of "set master" in the `mysql.rds_history` and `mysql.rds_replication_status` tables.

Examples

When run on a MySQL DB instance, the following example configures the DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS.

```
call mysql.rds_set_external_master(
    'Externaldb.some.com',
    3306,
    'repl_user'@'mydomain.com',
    'password',
    'mysql-bin-changelog.0777',
    120,
    0);
```

mysql.rds_set_external_master_with_delay

Configures an Amazon RDS MySQL DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS and configures delayed replication.

Syntax

```
CALL mysql.rds_set_external_master_with_delay (
    host_name
    , host_port
    , replication_user_name
    , replication_user_password
    , mysql_binary_log_file_name
    , mysql_binary_log_file_location
    , ssl_encryption
    , delay
    );
);
```

Parameters

host_name

The host name or IP address of the MySQL instance running external to Amazon RDS that will become the replication master.

host_port

The port used by the MySQL instance running external to Amazon RDS to be configured as the replication master. If your network configuration includes SSH port replication that converts the port number, specify the port number that is exposed by SSH.

replication_user_name

The ID of a user with REPLICATION CLIENT and REPLICATION SLAVE permissions on the MySQL instance running external to Amazon RDS. We recommend that you provide an account that is used solely for replication with the external instance.

replication_user_password

The password of the user ID specified in *replication_user_name*.

mysql_binary_log_file_name

The name of the binary log on the replication master contains the replication information.

mysql_binary_log_file_location

The location in the `mysql_binary_log_file_name` binary log at which replication will start reading the replication information.

ssl_encryption

This option is not currently implemented. The default is 0.

delay

The minimum number of seconds to delay replication from the master.

The limit for this parameter is one day (86400 seconds).

Usage Notes

The master user must run the `mysql.rds_set_external_master_with_delay` procedure. This procedure must be run on the MySQL DB instance to be configured as the Read Replica of a MySQL instance running external to Amazon RDS.

Before you run `mysql.rds_set_external_master_with_delay`, you must configure the instance of MySQL running external to Amazon RDS to be a replication master. To connect to the MySQL instance running external to Amazon RDS, you must specify values for `replication_user_name` and `replication_user_password`. These values must indicate a replication user that has `REPLICATION CLIENT` and `REPLICATION SLAVE` permissions on the external instance of MySQL.

To configure an external instance of MySQL as a replication master

1. Using the MySQL client of your choice, connect to the external instance of MySQL and create a user account to be used for replication. The following is an example.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
```

2. On the external instance of MySQL, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. The following example grants `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the '`repl_user`' user for your domain.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'  
IDENTIFIED BY 'SomePassWord'
```

For more information, see [Replication with a MySQL or MariaDB Instance Running External to Amazon RDS \(p. 667\)](#).

Note

We recommend that you use Read Replicas to manage replication between two Amazon RDS DB instances when possible. When you do so, we recommend that you use only this and other replication-related stored procedures. These practices enable more complex replication topologies between Amazon RDS DB instances. We offer these stored procedures primarily to enable replication with MySQL instances running external to Amazon RDS. For information about managing replication between Amazon RDS DB instances, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

After calling `mysql.rds_set_external_master_with_delay` to configure an Amazon RDS DB instance as a Read Replica, you can call [mysql.rds_start_replication \(p. 704\)](#) on the Read Replica to start the replication process. You can call [mysql.rds_reset_external_master \(p. 700\)](#) to remove the Read Replica configuration.

When you call `mysql.rds_set_external_master_with_delay`, Amazon RDS records the time, the user, and an action of "set master" in the `mysql.rds_history` and `mysql.rds_replication_status` tables.

For disaster recovery, you can use this procedure with the [mysql.rds_start_replication_until \(p. 704\)](#) or [mysql.rds_start_replication_until_gtid \(p. 705\)](#) stored procedure. To roll forward changes to a delayed Read Replica to the time just before a disaster, you can run the `mysql.rds_set_external_master_with_delay` procedure. After the `mysql.rds_start_replication_until` procedure stops replication, you can promote the Read Replica to be the new master DB instance by using the instructions in [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

To use the `mysql.rds_start_replication_until_gtid` procedure, GTID-based replication must be enabled. To skip a specific GTID-based transaction that is known to cause disaster, you can use the [mysql.rds_skip_transaction_with_gtid \(p. 707\)](#) stored procedure. For more information about working with GTID-based replication, see [Using GTID-Based Replication for Amazon RDS MySQL \(p. 663\)](#).

The `mysql.rds_set_external_master_with_delay` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6.40 and later 5.6 versions
- MySQL 5.7.22 and later 5.7 versions

Examples

When run on a MySQL DB instance, the following example configures the DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS. It sets the minimum replication delay to one hour (3,600 seconds) on the MySQL DB instance. A change from the MySQL master running external to Amazon RDS is not applied on the MySQL DB instance Read Replica for at least one hour.

```
call mysql.rds_set_external_master_with_delay(
    'Externaldb.some.com',
    3306,
    'repl_user'@'mydomain.com',
    'SomePassW0rd',
    'mysql-bin-changelog.000777',
    120,
    0,
    3600);
```

mysql.rds_set_external_master_with_auto_position

Configures an Amazon RDS MySQL DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS. This procedure also configures delayed replication and replication based on global transaction identifiers (GTIDs).

Syntax

```
CALL mysql.rds_set_external_master_with_auto_position (
    host_name
    , host_port
    , replication_user_name
    , replication_user_password
    , ssl_encryption
    , delay
);
```

Parameters

host_name

The host name or IP address of the MySQL instance running external to Amazon RDS to become the replication master.

host_port

The port used by the MySQL instance running external to Amazon RDS to be configured as the replication master. If your network configuration includes Secure Shell (SSH) port replication that converts the port number, specify the port number that is exposed by SSH.

replication_user_name

The ID of a user with REPLICATION CLIENT and REPLICATION SLAVE permissions on the MySQL instance running external to Amazon RDS. We recommend that you provide an account that is used solely for replication with the external instance.

replication_user_password

The password of the user ID specified in *replication_user_name*.

ssl_encryption

This option is not currently implemented. The default is 0.

delay

The minimum number of seconds to delay replication from the master.

The limit for this parameter is one day (86,400 seconds).

Usage Notes

The master user must run the `mysql.rds_set_external_master_with_auto_position` procedure. This procedure must be run on the MySQL DB instance to be configured as the Read Replica of a MySQL instance running external to Amazon RDS.

For Amazon RDS MySQL 5.7, this procedure is supported for MySQL 5.7.23 and later MySQL 5.7 versions. This procedure is not supported for Amazon RDS MySQL 5.5, 5.6, or 8.0.

Before you run `mysql.rds_set_external_master_with_auto_position`, you must configure the instance of MySQL running external to Amazon RDS to be a replication master. To connect to the MySQL instance running external to Amazon RDS, you must specify values for *replication_user_name* and *replication_user_password*. These values must indicate a replication user that has REPLICATION CLIENT and REPLICATION SLAVE permissions on the external instance of MySQL.

To configure an external instance of MySQL as a replication master

1. Using the MySQL client of your choice, connect to the external instance of MySQL and create a user account to be used for replication. The following is an example.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
```

2. On the external instance of MySQL, grant REPLICATION CLIENT and REPLICATION SLAVE privileges to your replication user. The following example grants REPLICATION CLIENT and REPLICATION SLAVE privileges on all databases for the 'repl_user' user for your domain.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'  
IDENTIFIED BY 'SomePassW0rd'
```

For more information, see [Replication with a MySQL or MariaDB Instance Running External to Amazon RDS \(p. 667\)](#).

Note

We recommend that you use Read Replicas to manage replication between two Amazon RDS DB instances when possible. When you do so, we recommend that you use only this and other replication-related stored procedures. These practices enable more complex replication topologies between Amazon RDS DB instances. We offer these stored procedures primarily to enable replication with MySQL instances running external to Amazon RDS. For information about managing replication between Amazon RDS DB instances, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

After calling `mysql.rds_set_external_master_with_auto_position` to configure an Amazon RDS DB instance as a Read Replica, you can call [mysql.rds_start_replication \(p. 704\)](#) on the Read Replica to start the replication process. You can call [mysql.rds_reset_external_master \(p. 700\)](#) to remove the Read Replica configuration.

When you call `mysql.rds_set_external_master_with_auto_position`, Amazon RDS records the time, the user, and an action of "set master" in the `mysql.rds_history` and `mysql.rds_replication_status` tables.

For disaster recovery, you can use this procedure with the [mysql.rds_start_replication_until \(p. 704\)](#) or [mysql.rds_start_replication_until_gtid \(p. 705\)](#) stored procedure. To roll forward changes to a delayed Read Replica to the time just before a disaster, you can run the `mysql.rds_set_external_master_with_auto_position` procedure. After the `mysql.rds_start_replication_until_gtid` procedure stops replication, you can promote the Read Replica to be the new master DB instance by using the instructions in [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

To use the `mysql.rds_start_replication_until_gtid` procedure, GTID-based replication must be enabled. To skip a specific GTID-based transaction that is known to cause disaster, you can use the [mysql.rds_skip_transaction_with_gtid \(p. 707\)](#) stored procedure. For more information about working with GTID-based replication, see [Using GTID-Based Replication for Amazon RDS MySQL \(p. 663\)](#).

Examples

When run on a MySQL DB instance, the following example configures the DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS. It sets the minimum replication delay to one hour (3,600 seconds) on the MySQL DB instance. A change from the MySQL master running external to Amazon RDS is not applied on the MySQL DB instance Read Replica for at least one hour.

```
call mysql.rds_set_external_master_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd',
  0,
  3600);
```

mysql.rds_reset_external_master

Reconfigures a MySQL DB instance to no longer be a Read Replica of an instance of MySQL running external to Amazon RDS.

Syntax

```
CALL mysql.rds_reset_external_master;
```

Usage Notes

The master user must run the `mysql.rds_reset_external_master` procedure. This procedure must be run on the MySQL DB instance to be removed as a Read Replica of a MySQL instance running external to Amazon RDS.

Note

We recommend that you use Read Replicas to manage replication between two Amazon RDS DB instances when possible. When you do so, we recommend that you use only this and other replication-related stored procedures. These practices enable more complex replication topologies between Amazon RDS DB instances. We offer these stored procedures primarily to enable replication with MySQL instances running external to Amazon RDS. For information about managing replication between Amazon RDS DB instances, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

For more information about using replication to import data from an instance of MySQL running external to Amazon RDS, see [Restoring a Backup into an Amazon RDS MySQL DB Instance \(p. 631\)](#).

mysql.rds_import_binlog_ssl_material

Imports the certificate authority certificate, client certificate, and client key into an Aurora MySQL DB cluster. The information is required for SSL communication and encrypted replication.

Note

Currently, this procedure is only supported for Aurora MySQL version 5.6.

Syntax

```
CALL mysql.rds_import_binlog_ssl_material (
    ssl_material
);
```

Parameters

ssl_material

JSON payload that contains the contents of the following .pem format files for a MySQL client:

- "ssl_ca": "*Certificate authority certificate*"
- "ssl_cert": "*Client certificate*"
- "ssl_key": "*Client key*"

Usage Notes

Prepare for encrypted replication before you run this procedure:

- If you don't have SSL enabled on the external MySQL master database and don't have a client key and client certificate prepared, enable SSL on the MySQL database server and generate the required client key and client certificate.
- If SSL is enabled on the external master, supply a client key and certificate for the Aurora MySQL DB cluster. If you don't have these, generate a new key and certificate for the Aurora MySQL DB cluster. To sign the client certificate, you must have the certificate authority key you used to configure SSL on the external MySQL master database.

For more information, see [Creating SSL Certificates and Keys Using openssl](#) in the MySQL documentation.

Important

After you prepare for encrypted replication, use an SSL connection to run this procedure. The client key must not be transferred across an insecure connection.

This procedure imports SSL information from an external MySQL database into an Aurora MySQL DB cluster. The SSL information is in .pem format files that contain the SSL information for the Aurora MySQL DB cluster. During encrypted replication, the Aurora MySQL DB cluster acts a client to the MySQL database server. The certificates and keys for the Aurora MySQL client are in files in .pem format.

You can copy the information from these files into the `ssl_material` parameter in the correct JSON payload. To support encrypted replication, import this SSL information into the Aurora MySQL DB cluster.

The JSON payload must be in the following format.

```
'{"ssl_ca":"-----BEGIN CERTIFICATE-----  
ssl_ca_pem_body_code  
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----  
ssl_cert_pem_body_code  
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----  
ssl_key_pem_body_code  
-----END RSA PRIVATE KEY-----\n"}'
```

Examples

The following example imports SSL information into an Aurora MySQL DB cluster. In .pem format files, the body code typically is longer than the body code shown in the example.

```
call mysql.rds_import_binlog_ssl_material(  
'{"ssl_ca":"-----BEGIN CERTIFICATE-----  
AAAAB3NzaC1yc2EAAAQABAAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V  
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4xyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr  
lsLnB1tntckij7fbtxJMxLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0Fzz  
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUzofz221CBt5IMucxXPkX4rWi+z7wB3Rb  
BQoQzd8v7yeb7Oz1PnWOyN0qFU0XA246RA8QFYiCNYwi3f05p6KLxEXAMPLE  
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----  
AAAAB3NzaC1yc2EAAAQABAAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V  
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4xyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr  
lsLnB1tntckij7fbtxJMxLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0Fzz  
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUzofz221CBt5IMucxXPkX4rWi+z7wB3Rb  
BQoQzd8v7yeb7Oz1PnWOyN0qFU0XA246RA8QFYiCNYwi3f05p6KLxEXAMPLE  
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----  
AAAAB3NzaC1yc2EAAAQABAAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V  
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4xyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr  
lsLnB1tntckij7fbtxJMxLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0Fzz  
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUzofz221CBt5IMucxXPkX4rWi+z7wB3Rb  
BQoQzd8v7yeb7Oz1PnWOyN0qFU0XA246RA8QFYiCNYwi3f05p6KLxEXAMPLE  
-----END RSA PRIVATE KEY-----\n"}');
```

Note

For information about using Amazon Aurora, see the [Amazon Aurora User Guide](#).

mysql.rds_remove_binlog_ssl_material

Removes the certificate authority certificate, client certificate, and client key for SSL communication and encrypted replication. This information is imported by using [mysql.rds_import_binlog_ssl_material \(p. 701\)](#).

Note

Currently, this procedure is only supported for Aurora MySQL version 5.6.

Syntax

```
CALL mysql.rds_remove_binlog_ssl_material;
```

mysql.rds_set_source_delay

Sets the minimum number of seconds to delay replication from the master to the current Read Replica. Use this procedure when you are connected to a Read Replica to delay replication from its master.

Syntax

```
CALL mysql.rds_set_source_delay(  
    delay  
)
```

Parameters

delay

The minimum number of seconds to delay replication from the master.

The limit for this parameter is one day (86400 seconds).

Usage Notes

The master user must run the `mysql.rds_set_source_delay` procedure.

For disaster recovery, you can use this procedure with the [mysql.rds_start_replication_until \(p. 704\)](#) stored procedure or the [mysql.rds_start_replication_until_gtid \(p. 705\)](#) stored procedure. To roll forward changes to a delayed Read Replica to the time just before a disaster, you can run the `mysql.rds_set_source_delay` procedure. After the `mysql.rds_start_replication_until` or `mysql.rds_start_replication_until_gtid` procedure stops replication, you can promote the Read Replica to be the new master DB instance by using the instructions in [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

To use the `mysql.rds_start_replication_until_gtid` procedure, GTID-based replication must be enabled. To skip a specific GTID-based transaction that is known to cause disaster, you can use the [mysql.rds_skip_transaction_with_gtid \(p. 707\)](#) stored procedure. For more information on GTID-based replication, see [Using GTID-Based Replication for Amazon RDS MySQL \(p. 663\)](#).

The `mysql.rds_set_source_delay` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6.40 and later 5.6 versions
- MySQL 5.7.22 and later 5.7 versions

Examples

To delay replication from the master to the current Read Replica for at least one hour (3,600 seconds), you can call `mysql.rds_set_source_delay` with the following parameter:

```
CALL mysql.rds_set_source_delay(3600);
```

mysql.rds_start_replication

Initiates replication from a MySQL DB instance.

Note

You can use the [mysql.rds_start_replication_until \(p. 704\)](#) or [mysql.rds_start_replication_until_gtid \(p. 705\)](#) stored procedure to initiate replication from an Amazon RDS MySQL DB instance and stop replication at the specified binary log file location.

Syntax

```
CALL mysql.rds_start_replication;
```

Usage Notes

The master user must run the `mysql.rds_start_replication` procedure.

If you are configuring replication to import data from an instance of MySQL running external to Amazon RDS, you call `mysql.rds_start_replication` on the Read Replica to start the replication process after you have called [mysql.rds_set_external_master \(p. 694\)](#) to build the replication configuration. For more information, see [Restoring a Backup into an Amazon RDS MySQL DB Instance \(p. 631\)](#).

If you are configuring replication to export data to an instance of MySQL external to Amazon RDS, you call `mysql.rds_start_replication` and `mysql.rds_stop_replication` on the Read Replica to control some replication actions, such as purging binary logs. For more information, see [Exporting Data from a MySQL DB Instance by Using Replication \(p. 673\)](#).

You can also call `mysql.rds_start_replication` on the Read Replica to restart any replication process that you previously stopped by calling [mysql.rds_stop_replication \(p. 706\)](#). For more information, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

mysql.rds_start_replication_until

Initiates replication from an Amazon RDS MySQL DB instance and stops replication at the specified binary log file location.

Syntax

```
CALL mysql.rds_start_replication_until (
  replication_log_file
  , replication_stop_point
);
```

Parameters

replication_log_file

The name of the binary log on the replication master contains the replication information.

replication_stop_point

The location in the `replication_log_file` binary log at which replication will stop.

Usage Notes

The master user must run the `mysql.rds_start_replication_until` procedure.

You can use this procedure with delayed replication for disaster recovery. If you have delayed replication configured, you can use this procedure to roll forward changes to a delayed Read Replica to the time just before a disaster. After this procedure stops replication, you can promote the Read Replica to be the new master DB instance by using the instructions in [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

You can configure delayed replication using the following stored procedures:

- [mysql.rds_set_configuration \(p. 711\)](#)
- [mysql.rds_set_external_master_with_delay \(p. 696\)](#)
- [mysql.rds_set_source_delay \(p. 703\)](#)

The file name specified for the `replication_log_file` parameter must match the master binlog file name.

When the `replication_stop_point` parameter specifies a stop location that is in the past, replication is stopped immediately.

The `mysql.rds_start_replication_until` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6.40 and later 5.6 versions
- MySQL 5.7.22 and later 5.7 versions

Examples

The following example initiates replication and replicates changes until it reaches location 120 in the `mysql-bin-changelog.000777` binary log file.

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

mysql.rds_start_replication_until_gtid

Initiates replication from an Amazon RDS MySQL DB instance and stops replication immediately after the specified global transaction identifier (GTID).

Syntax

```
CALL mysql.rds_start_replication_until_gtid (
  gtid
);
```

Parameters

gtid

The GTID after which replication is to stop.

Usage Notes

The master user must run the `mysql.rds_start_replication_until_gtid` procedure.

For Amazon RDS MySQL 5.7, this procedure is supported for MySQL 5.7.23 and later MySQL 5.7 versions. This procedure is not supported for Amazon RDS MySQL 5.5, 5.6, or 8.0.

You can use this procedure with delayed replication for disaster recovery. If you have delayed replication configured, you can use this procedure to roll forward changes to a delayed Read Replica to the time just before a disaster. After this procedure stops replication, you can promote the Read Replica to be the new master DB instance by using the instructions in [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

You can configure delayed replication using the following stored procedures:

- [mysql.rds_set_configuration \(p. 711\)](#)
- [mysql.rds_set_external_master_with_auto_position \(p. 698\)](#)
- [mysql.rds_set_source_delay \(p. 703\)](#)

When the `gtid` parameter specifies a transaction that has already been executed by the replica, replication is stopped immediately.

Examples

The following example initiates replication and replicates changes until it reaches GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`.

```
call mysql.rds_start_replication_until_gtid(
    '3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_stop_replication

Terminates replication from a MySQL DB instance.

Syntax

```
CALL mysql.rds_stop_replication;
```

Usage Notes

The master user must run the `mysql.rds_stop_replication` procedure.

If you are configuring replication to import data from an instance of MySQL running external to Amazon RDS, you call `mysql.rds_stop_replication` on the Read Replica to stop the replication process after the import has completed. For more information, see [Restoring a Backup into an Amazon RDS MySQL DB Instance \(p. 631\)](#).

If you are configuring replication to export data to an instance of MySQL external to Amazon RDS, you call `mysql.rds_start_replication` and `mysql.rds_stop_replication` on the Read Replica to control some replication actions, such as purging binary logs. For more information, see [Exporting Data from a MySQL DB Instance by Using Replication \(p. 673\)](#).

You can also use `mysql.rds_stop_replication` to stop replication between two Amazon RDS DB instances. You typically stop replication to perform a long running operation on the Read Replica, such

as creating a large index on the Read Replica. You can restart any replication process that you stopped by calling [mysql.rds_start_replication \(p. 704\)](#) on the Read Replica. For more information, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

mysql.rds_skip_transaction_with_gtid

Skips replication of a transaction with the specified global transaction identifier (GTID) on a MySQL DB instance.

You can use this procedure for disaster recovery when a specific GTID transaction is known to cause a problem. Use this stored procedure to skip the problematic transaction. Examples of problematic transactions include transactions that disable replication, delete important data, or cause the DB instance to become unavailable.

Syntax

```
CALL mysql.rds_skip_transaction_with_gtid (
    gtid_to_skip
);
```

Parameters

gtid_to_skip

The GTID of the replication transaction to skip.

Usage Notes

The master user must run the `mysql.rds_skip_transaction_with_gtid` procedure.

For Amazon RDS MySQL 5.7, this procedure is supported for MySQL 5.7.23 and later MySQL 5.7 versions. This procedure is not supported for Amazon RDS MySQL 5.5, 5.6, or 8.0.

mysql.rds_skip_repl_error

Skips and deletes a replication error on a MySQL DB instance.

Syntax

```
CALL mysql.rds_skip_repl_error;
```

Usage Notes

The master user must run the `mysql.rds_skip_repl_error` procedure.

To determine if there are errors, run the MySQL `show slave status\G` command. If a replication error isn't critical, you can run `mysql.rds_skip_repl_error` to skip the error. If there are multiple errors, `mysql.rds_skip_repl_error` deletes the first error, then warns that others are present. You can then use `show slave status\G` to determine the correct course of action for the next error. For information about the values returned, see [SHOW SLAVE STATUS Syntax](#) in the MySQL documentation.

For more information about addressing replication errors with Amazon RDS, see [Troubleshooting a MySQL Read Replica Problem \(p. 661\)](#).

Important

If you try to call `mysql.rds_skip_repl_error`, you might encounter the following error:
`ERROR 1305 (42000): PROCEDURE mysql.rds_skip_repl_error does not exist.`
If you do, upgrade your MySQL DB instance to the latest minor version or one of the minimum minor versions listed in this topic.

Slave Down or Disabled Error

When you call the `mysql.rds_skip_repl_error` command, you might receive the following error message: `Slave is down or disabled.`

This error message appears because replication has stopped and could not be restarted.

If you need to skip a large number of errors, the replication lag can increase beyond the default retention period for binary log (binlog) files. In this case, you might encounter a fatal error due to binlog files being purged before they have been replayed on the Read Replica. This purge causes replication to stop, and you can no longer call the `mysql.rds_skip_repl_error` command to skip replication errors.

You can mitigate this issue by increasing the number of hours that binlog files are retained on your replication master. After you have increased the binlog retention time, you can restart replication and call the `mysql.rds_skip_repl_error` command as needed.

To set the binlog retention time, use the [mysql.rds_set_configuration \(p. 711\)](#) procedure and specify a configuration parameter of '`binlog retention hours`' along with the number of hours to retain binlog files on the DB cluster. The following example sets the retention period for binlog files to 48 hours.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

mysql.rds_next_master_log

Changes the replication master log position to the start of the next binary log on the master. Use this procedure only if you are receiving replication I/O error 1236 on a Read Replica.

Syntax

```
CALL mysql.rds_next_master_log(  
curr_master_log  
);
```

Parameters

curr_master_log

The index of the current master log file. For example, if the current file is named `mysql-bin-changelog.012345`, then the index is 12345. To determine the current master log file name, run the `SHOW SLAVE STATUS` command and view the `Master_Log_File` field.

Usage Notes

The master user must run the `mysql.rds_next_master_log` procedure.

Warning

Call `mysql.rds_next_master_log` only if replication fails after a failover of a Multi-AZ DB instance that is the replication source, and the `Last_IO_Errno` field of `SHOW SLAVE STATUS` reports I/O error 1236.

Calling `mysql.rds_next_master_log` may result in data loss in the Read Replica if transactions in the source instance were not written to the binary log on disk before the failover event occurred. You can reduce the chance of this happening by configuring the source instance parameters `sync_binlog = 1` and `innodb_support_xa = 1`, although this may reduce performance. For more information, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

Examples

Assume replication fails on an Amazon RDS Read Replica. Running `SHOW SLAVE STATUS\G` on the Read Replica returns the following result:

```
***** 1. row *****
Slave_IO_State:
    Master_Host: myhostXXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
    Master_User: MasterUser
    Master_Port: 3306
    Connect_Retry: 10
    Master_Log_File: mysql-bin-changelog.012345
    Read_Master_Log_Pos: 1219393
    Relay_Log_File: relaylog.012340
    Relay_Log_Pos: 30223388
    Relay_Master_Log_File: mysql-bin-changelog.012345
    Slave_IO_Running: No
    Slave_SQL_Running: Yes
    Replicate_Do_DB:
    Replicate_Ignore_DB:
    Replicate_Do_Table:
    Replicate_Ignore_Table:
    Replicate_Wild_Do_Table:
    Replicate_Wild_Ignore_Table:
        Last_Error:
        Skip_Counter: 0
        Exec_Master_Log_Pos: 30223232
        Relay_Log_Space: 5248928866
        Until_Condition: None
        Until_Log_File:
        Until_Log_Pos: 0
        Master_SSL_Allowed: No
        Master_SSL_CA_File:
        Master_SSL_CA_Path:
            Master_SSL_Cert:
            Master_SSL_Cipher:
            Master_SSL_Key:
        Seconds_Behind_Master: NULL
Master_SSL_Verify_Server_Cert: No
    Last_IO_Errno: 1236
    Last_IO_Error: Got fatal error 1236 from master when reading data from
binary log: 'Client requested master to start replication from impossible position; the
first event 'mysql-bin-changelog.013406' at 1219393, the last event read from '/rdsdbdata/
log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/rdsdbdata/log/
binlog/mysql-bin-changelog.012345' at 4.'
    Last_SQL_Errno: 0
    Last_SQL_Error:
Replicate_Ignore_Server_Ids:
    Master_Server_Id: 67285976
```

The `Last_IO_Errno` field shows that the instance is receiving I/O error 1236. The `Master_Log_File` field shows that the file name is `mysql-bin-changelog.012345`, which means that the log file index is 12345. To resolve the error, you can call `mysql.rds_next_master_log` with the following parameter:

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_innodb_buffer_pool_dump_now

Dumps the current state of the buffer pool to disk. For more information, see [InnoDB Cache Warming \(p. 594\)](#).

Syntax

```
CALL mysql.rds_innodb_buffer_pool_dump_now();
```

Usage Notes

The master user must run the `mysql.rds_innodb_buffer_pool_dump_now` procedure.

The `mysql.rds_innodb_buffer_pool_dump_now` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6
- MySQL 5.7
- MySQL 8.0

mysql.rds_innodb_buffer_pool_load_now

Loads the saved state of the buffer pool from disk. For more information, see [InnoDB Cache Warming \(p. 594\)](#).

Syntax

```
CALL mysql.rds_innodb_buffer_pool_load_now();
```

Usage Notes

The master user must run the `mysql.rds_innodb_buffer_pool_load_now` procedure.

The `mysql.rds_innodb_buffer_pool_load_now` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6
- MySQL 5.7
- MySQL 8.0

mysql.rds_innodb_buffer_pool_load_abort

Cancels a load of the saved buffer pool state while in progress. For more information, see [InnoDB Cache Warming \(p. 594\)](#).

Syntax

```
CALL mysql.rds_innodb_buffer_pool_load_abort();
```

Usage Notes

The master user must run the `mysql.rds_innodb_buffer_pool_load_abort` procedure.

The `mysql.rds_innodb_buffer_pool_load_abort` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6
- MySQL 5.7
- MySQL 8.0

mysql.rds_set_configuration

Specifies the number of hours to retain binary logs or the number of seconds to delay replication.

Syntax

```
CALL mysql.rds_set_configuration(name,value);
```

Parameters

name

The name of the configuration parameter to set.

value

The value of the configuration parameter.

Usage Notes

The `mysql.rds_set_configuration` stored procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6
- MySQL 5.7
- MySQL 8.0

The `mysql.rds_set_configuration` procedure supports the following configuration parameters:

- [binlog retention hours \(p. 711\)](#)
- [target delay \(p. 712\)](#)

binlog retention hours

The `binlog retention hours` parameter is used to specify the number of hours to retain binary log files. Amazon RDS normally purges a binary log as soon as possible, but the binary log might still be required for replication with a MySQL database external to Amazon RDS. The default value of `binlog retention hours` is `NULL` (do not retain binary logs).

To specify the number of hours for Amazon RDS to retain binary logs on a DB instance, use the `mysql.rds_set_configuration` stored procedure and specify a period with enough time for replication to occur, as shown in the following example.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

For MySQL DB instances, the maximum binlog retention hours value is 168 (7 days).

After you set the retention period, monitor storage usage for the DB instance to make sure that the retained binary logs don't take up too much storage.

target delay

Use the target_delay parameter to specify the number of seconds to delay replication from the master to the Read Replica. The specified delay applies to new replicas created from the current DB instance. Amazon RDS normally replicates changes as soon as possible, but some environments might want to delay replication. For example, when replication is delayed, you can roll forward a delayed Read Replica to the time just before a disaster. If a table is dropped accidentally, you can use delayed replication to recover it quickly. The default value of target_delay is 0 (don't delay replication).

For disaster recovery, you can use this configuration parameter with the [mysql.rds_start_replication_until \(p. 704\)](#) stored procedure or the [mysql.rds_start_replication_until_gtid \(p. 705\)](#) stored procedure. To roll forward changes to a delayed Read Replica to the time just before a disaster, you can run the mysql.rds_set_configuration procedure with this parameter set. After the mysql.rds_start_replication_until or mysql.rds_start_replication_until_gtid procedure stops replication, you can promote the Read Replica to be the new master DB instance by using the instructions in [Promoting a Read Replica to Be a Standalone DB Instance \(p. 146\)](#).

To use the mysql.rds_start_replication_until_gtid procedure, GTID-based replication must be enabled. To skip a specific GTID-based transaction that is known to cause disaster, you can use the [mysql.rds_skip_transaction_with_gtid \(p. 707\)](#) stored procedure. For more information about working with GTID-based replication, see [Using GTID-Based Replication for Amazon RDS MySQL \(p. 663\)](#).

To specify the number of seconds for Amazon RDS to delay replication to a Read Replica, use the mysql.rds_set_configuration stored procedure and specify the number of seconds to delay replication. The following example specifies that replication is delayed by at least one hour (3,600 seconds).

```
call mysql.rds_set_configuration('target delay', 3600);
```

The limit for the target_delay parameter is one day (86400 seconds).

Note

The target_delay parameter is only supported for Amazon RDS MySQL.

The target_delay parameter is not supported for Amazon RDS MySQL version 8.0.

mysql.rds_show_configuration

The number of hours that binary logs are retained.

Syntax

```
CALL mysql.rds_show_configuration;
```

Usage Notes

To verify the number of hours that Amazon RDS retains binary logs, use the mysql.rds_show_configuration stored procedure.

The mysql.rds_show_configuration procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6

- MySQL 5.7
- MySQL 8.0

Examples

The following example displays the retention period:

```
call mysql.rds_show_configuration;
      name          value   description
      binlog retention hours    24      binlog retention hours specifies the
duration in hours before binary logs are automatically deleted.
```

mysql.rds_kill

Terminates a connection to the MySQL server.

Syntax

```
CALL mysql.rds_kill(processID);
```

Parameters

processID

The identity of the connection thread to be terminated.

Usage Notes

Each connection to the MySQL server runs in a separate thread. To terminate a connection, use the `mysql.rds_kill` procedure and pass in the thread ID of that connection. To obtain the thread ID, use the MySQL `SHOW PROCESSLIST` command.

Examples

The following example terminates a connection with a thread ID of 4243:

```
call mysql.rds_kill(4243);
```

mysql.rds_kill_query

Terminates a query running against the MySQL server.

Syntax

```
CALL mysql.rds_kill_query(queryID);
```

Parameters

queryID

The identity of the query to be terminated.

Usage Notes

To terminate a query running against the MySQL server, use the `mysql_rds_kill_query` procedure and pass in the ID of that query. To obtain the query ID, use the MySQL [INFORMATION_SCHEMA PROCESSLIST](#) command. The connection to the MySQL server is retained.

Examples

The following example terminates a query with a thread ID of 230040:

```
call mysql.rds_kill_query(230040);
```

mysql.rds_rotate_general_log

Rotates the `mysql.general_log` table to a backup table. For more information, see [MySQL Database Log Files \(p. 320\)](#).

Syntax

```
CALL mysql.rds_rotate_general_log;
```

Usage Notes

You can rotate the `mysql.general_log` table to a backup table by calling the `mysql.rds_rotate_general_log` procedure. When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If a backup log table already exists, then it is deleted before the current log table is copied to the backup. You can query the backup log table if needed. The backup log table for the `mysql.general_log` table is named `mysql.general_log_backup`.

mysql.rds_rotate_slow_log

Rotates the `mysql.slow_log` table to a backup table. For more information, see [MySQL Database Log Files \(p. 320\)](#).

Syntax

```
CALL mysql.rds_rotate_slow_log;
```

Usage Notes

You can rotate the `mysql.slow_log` table to a backup table by calling the `mysql.rds_rotate_slow_log` procedure. When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If a backup log table already exists, then it is deleted before the current log table is copied to the backup.

You can query the backup log table if needed. The backup log table for the `mysql.slow_log` table is named `mysql.slow_log_backup`.

mysql.rds_enable_gsh_collector

Enables the Global Status History (GoSH) to take default snapshots at intervals specified by `rds_set_gsh_collector`. For more information, see [Managing the Global Status History \(p. 687\)](#).

Syntax

```
CALL mysql.rds_enable_gsh_collector;
```

mysql.rds_set_gsh_collector

Specifies the interval, in minutes, between snapshots taken by the Global Status History (GoSH). Default value is For more information, see [Managing the Global Status History \(p. 687\)](#).

Syntax

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

Parameters

intervalPeriod

The interval, in minutes, between snapshots. Default value is

mysql.rds_disable_gsh_collector

Disables snapshots taken by the Global Status History (GoSH). For more information, see [Managing the Global Status History \(p. 687\)](#).

Syntax

```
CALL mysql.rds_disable_gsh_collector;
```

mysql.rds_collect_global_status_history

Takes a snapshot on demand for the Global Status History (GoSH). For more information, see [Managing the Global Status History \(p. 687\)](#).

Syntax

```
CALL mysql.rds_collect_global_status_history;
```

mysql.rds_enable_gsh_rotation

Enables rotation of the contents of the `mysql.global_status_history` table to `mysql.global_status_history_old` at intervals specified by `rds_set_gsh_rotation`. For more information, see [Managing the Global Status History \(p. 687\)](#).

Syntax

```
CALL mysql.rds_enable_gsh_rotation;
```

mysql.rds_set_gsh_rotation

Specifies the interval, in days, between rotations of the `mysql.global_status_history` table. Default value is 7. For more information, see [Managing the Global Status History \(p. 687\)](#).

Syntax

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

Parameters

intervalPeriod

The interval, in days, between table rotations. Default value is 7.

mysql.rds_disable_gsh_rotation

Disables rotation of the `mysql.global_status_history` table. For more information, see [Managing the Global Status History \(p. 687\)](#).

Syntax

```
CALL mysql.rds_disable_gsh_rotation;
```

mysql.rds_rotate_global_status_history

Rotates the contents of the `mysql.global_status_history` table to `mysql.global_status_history_old` on demand. For more information, see [Managing the Global Status History \(p. 687\)](#).

Syntax

```
CALL mysql.rds_rotate_global_status_history;
```

Oracle on Amazon RDS

Amazon RDS supports DB instances running several versions and editions of Oracle Database. You can use the following versions and editions:

- Oracle 12c, Version 12.2.0.1
- Oracle 12c, Version 12.1.0.2
- Oracle 11g, Version 11.2.0.4

Amazon RDS also currently supports the following versions and editions that are on deprecation paths, because Oracle no longer provides patches for them:

- Oracle 12c, Version 12.1.0.1 ([Deprecation of Oracle 12.1.0.1 \(p. 736\)](#))
- Oracle 11g, Version 11.2.0.3 ([Deprecation of Oracle 11.2.0.3 \(p. 735\)](#))
- Oracle 11g, Version 11.2.0.2 ([Deprecation of Oracle 11.2.0.2 \(p. 735\)](#))

You can create DB instances and DB snapshots, point-in-time restores and automated or manual backups. DB instances running Oracle can be used inside a VPC. You can also enable various options to add additional features to your Oracle DB instance. Amazon RDS supports Multi-AZ deployments for Oracle as a high-availability, failover solution.

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application such as Oracle SQL Plus. Amazon RDS does not allow direct host access to a DB instance via Telnet or Secure Shell (SSH).

When you create a DB instance, the master account that you use to create the instance gets DBA user privileges (with some limitations). Use this account for any administrative tasks such as creating additional user accounts in the database. The SYS user, SYSTEM user, and other administrative accounts can't be used.

Before creating a DB instance, you should complete the steps in the [Setting Up for Amazon RDS \(p. 5\)](#) section of this guide.

Common Management Tasks for Oracle on Amazon RDS

The following are the common management tasks you perform with an Amazon RDS Oracle DB instance, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Instance Classes, Storage, and PIOPS If you are creating a DB instance for production purposes, you should understand how instance classes, storage types, and Provisioned IOPS work in Amazon RDS.	DB Instance Class Support for Oracle (p. 720) Amazon RDS Storage Types (p. 103)

Task Area	Relevant Documentation
Multi-AZ Deployments A production DB instance should use Multi-AZ deployments. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances.	High Availability (Multi-AZ) for Amazon RDS (p. 109)
Amazon Virtual Private Cloud (VPC) If your AWS account has a default VPC, then your DB instance is automatically created inside the default VPC. If your account does not have a default VPC, and you want the DB instance in a VPC, you must create the VPC and subnet groups before you create the DB instance.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412) Working with an Amazon RDS DB Instance in a VPC (p. 420)
Security Groups By default, DB instances are created with a firewall that prevents access to them. You therefore must create a security group with the correct IP addresses and network configuration to access the DB instance. The security group you create depends on what Amazon EC2 platform your DB instance is on, and whether you will access your DB instance from an Amazon EC2 instance. In general, if your DB instance is on the <i>EC2-Classic</i> platform, you will need to create a DB security group; if your DB instance is on the <i>EC2-VPC</i> platform, you will need to create a VPC security group.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412) Controlling Access with Security Groups (p. 394)
Parameter Groups If your DB instance is going to require specific database parameters, you should create a parameter group before you create the DB instance.	Working with DB Parameter Groups (p. 169)
Option Groups If your DB instance is going to require specific database options, you should create an option group before you create the DB instance.	Options for Oracle DB Instances (p. 787)
Connecting to Your DB Instance After creating a security group and associating it to a DB instance, you can connect to the DB instance using any standard SQL client application such as Oracle SQL Plus.	Connecting to a DB Instance Running the Oracle Database Engine (p. 751)
Backup and Restore You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots.	Backing Up and Restoring Amazon RDS DB Instances (p. 207)
Monitoring You can monitor an Oracle DB instance by using CloudWatch Amazon RDS metrics, events, and enhanced monitoring.	Viewing DB Instance Metrics (p. 254) Viewing Amazon RDS Events (p. 305)

Task Area	Relevant Documentation
Log Files You can access the log files for your Oracle DB instance.	Amazon RDS Database Log Files (p. 307)

There are also advanced tasks and optional features for working with Oracle DB instances. For more information, see the following documentation:

- For information on common DBA tasks for Oracle on Amazon RDS, see [Common DBA Tasks for Oracle DB Instances \(p. 842\)](#).
- For information on Oracle GoldenGate support, see [Using Oracle GoldenGate with Amazon RDS \(p. 902\)](#).
- For information on Siebel Customer Relationship Management (CRM) support, see [Installing a Siebel Database on Oracle on Amazon RDS \(p. 918\)](#).

Oracle Licensing

There are two licensing options available for Amazon RDS for Oracle: License Included and Bring Your Own License (BYOL). After you create an Oracle DB instance on Amazon RDS, you can change the licensing model by using the [AWS Management Console](#), the Amazon RDS API [ModifyDBInstance](#) action, or the AWS CLI [modify-db-instance](#) command.

License Included

In the License Included model, you don't need to purchase Oracle licenses separately. AWS holds the license for the Oracle database software. In this model, if you have an AWS Support account with case support, you contact AWS Support for both Amazon RDS and Oracle Database service requests.

The License Included model is supported on Amazon RDS for the following Oracle database editions:

- Oracle Database Standard Edition One (SE1)
- Oracle Database Standard Edition Two (SE2)

Bring Your Own License (BYOL)

In the Bring Your Own License model, you can use your existing Oracle Database licenses to run Oracle deployments on Amazon RDS. You must have the appropriate Oracle Database license (with Software Update License and Support) for the DB instance class and Oracle Database edition you wish to run. You must also follow Oracle's policies for licensing Oracle Database software in the cloud computing environment. For more information on Oracle's licensing policy for Amazon EC2, see [Licensing Oracle Software in the Cloud Computing Environment](#).

In this model, you continue to use your active Oracle support account, and you contact Oracle directly for Oracle Database service requests. If you have an AWS Support account with case support, you can contact AWS Support for Amazon RDS issues. Amazon Web Services and Oracle have a multi-vendor support process for cases which require assistance from both organizations.

The Bring Your Own License model is supported on Amazon RDS for the following Oracle database editions:

- Oracle Database Enterprise Edition (EE)
- Oracle Database Standard Edition (SE)

- Oracle Database Standard Edition One (SE1)
- Oracle Database Standard Edition Two (SE2)

Licensing Oracle Multi-AZ Deployments

Amazon RDS supports Multi-AZ deployments for Oracle as a high-availability, failover solution. We recommend Multi-AZ for production workloads. For more information, see [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#).

If you use the Bring Your Own License model, you must have a license for both the primary DB instance and the standby DB instance in a Multi-AZ deployment.

Migrating Between Oracle Editions

For the BYOL model, you can migrate from any Standard Edition (SE, SE1, or SE2) to Enterprise Edition (EE), assuming you have an unused Oracle license appropriate for the edition and class of DB instance you plan to run. You can't migrate from Enterprise Edition to other editions.

To change the edition and retain your data

1. Create a snapshot of the DB instance.
For more information, see [Creating a DB Snapshot \(p. 216\)](#).
2. Restore the snapshot to a new DB instance, and select the Oracle database edition you want to use.
For more information, see [Restoring from a DB Snapshot \(p. 218\)](#).
3. (Optional) Delete the old DB instance, unless you want to keep it running and have the appropriate Oracle Database licenses for it.
For more information, see [Deleting a DB Instance \(p. 135\)](#).

DB Instance Class Support for Oracle

The computation and memory capacity of a DB instance is determined by its DB instance class. The DB instance class you need depends on your processing power and memory requirements. For more information, see [DB Instance Class \(p. 82\)](#).

The following are the DB instance classes supported for Oracle.

Oracle Edition	Version 12.2.0.1 Support	Version 12.1.0.2 Support	Version 11.2.0.4 Support
Enterprise Edition (EE)	db.m5.large–db.m5.24xlarge	db.m5.large–db.m5.24xlarge	db.m5.large–db.m5.24xlarge
Bring Your Own License (BYOL)	db.m4.large–db.m4.16xlarge	db.m4.large–db.m4.16xlarge	db.m4.large–db.m4.16xlarge
	db.m3.medium–db.m3.2xlarge	db.m3.medium–db.m3.2xlarge	db.m3.medium–db.m3.2xlarge
	db.x1e.xlarge–db.x1e.32xlarge	db.x1e.xlarge–db.x1e.32xlarge	db.x1e.xlarge–db.x1e.32xlarge

Oracle Edition	Version 12.2.0.1 Support	Version 12.1.0.2 Support	Version 11.2.0.4 Support
	db.x1.16xlarge– db.x1.32xlarge db.r5.large–db.r5.24xlarge db.r4.large–db.r4.16xlarge db.r3.large–db.r3.8xlarge db.t2.small–db.t2.2xlarge	db.x1.16xlarge– db.x1.32xlarge db.r5.large–db.r5.24xlarge db.r4.large–db.r4.16xlarge db.r3.large–db.r3.8xlarge db.t2.micro–db.t2.2xlarge	db.x1.16xlarge– db.x1.32xlarge db.r5.large–db.r5.24xlarge db.r4.large–db.r4.16xlarge db.r3.large–db.r3.8xlarge db.t2.micro–db.t2.2xlarge
Standard Edition 2 (SE2)	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge	—
Bring Your Own License (BYOL)	db.m3.medium– db.m3.2xlarge db.x1e.xlarge– db.x1e.4xlarge db.r5.large–db.r5.4xlarge db.r4.large–db.r4.4xlarge db.r3.large–db.r3.4xlarge db.t2.small–db.t2.2xlarge	db.m3.medium– db.m3.2xlarge db.x1e.xlarge– db.x1e.4xlarge db.r5.large–db.r5.4xlarge db.r4.large–db.r4.4xlarge db.r3.large–db.r3.4xlarge db.t2.micro–db.t2.2xlarge	
Standard Edition 2 (SE2)	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge	—
License Included	db.m3.medium– db.m3.2xlarge db.r5.large–db.r5.4xlarge db.r4.large–db.r4.4xlarge db.r3.large–db.r3.4xlarge db.t2.small–db.t2.2xlarge	db.m3.medium– db.m3.2xlarge db.r5.large–db.r5.4xlarge db.r4.large–db.r4.4xlarge db.r3.large–db.r3.4xlarge db.t2.micro–db.t2.2xlarge	

Oracle Edition	Version 12.2.0.1 Support	Version 12.1.0.2 Support	Version 11.2.0.4 Support
Standard Edition 1 (SE1) Bring Your Own License (BYOL)	—	—	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge db.m3.medium–db.m3.2xlarge db.x1e.xlarge–db.x1e.4xlarge db.r5.large–db.r5.4xlarge db.r4.large–db.r4.4xlarge db.r3.large–db.r3.4xlarge db.t2.micro–db.t2.2xlarge
Standard Edition 1 (SE1) License Included	—	—	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge db.m3.medium–db.m3.2xlarge db.r5.large–db.r5.4xlarge db.r3.large–db.r3.4xlarge db.t2.micro–db.t2.large
Standard Edition (SE) Bring Your Own License (BYOL)	—	—	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge db.m3.medium–db.m3.2xlarge db.x1e.xlarge–db.x1e.8xlarge db.r5.large–db.r5.4xlarge db.r4.large–db.r4.8xlarge db.r3.large–db.r3.8xlarge db.t2.micro–db.t2.2xlarge

Deprecated DB Instance Classes for Oracle

The db.m1 and db.m2 DB instance classes are deprecated for Amazon RDS for Oracle. These DB instance classes have been replaced by better performing DB instance classes that are generally available at a lower cost. Starting on September 12, 2018, Amazon RDS for Oracle will automatically scale DB instances to DB instance classes that are not deprecated.

If you have DB instances that use db.m1 and db.m2 DB instance classes, Amazon RDS will modify each one automatically to use a comparable DB instance class that is not deprecated. You can change the DB instance class for a DB instance yourself by modifying the DB instance. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

If you have DB snapshots of DB instances that were using db.m1 or db.m2 DB instance classes, you can choose a DB instance class that is not deprecated when you restore the DB snapshots. For more information, see [Restoring from a DB Snapshot \(p. 218\)](#).

Oracle Security

The Oracle database engine uses role-based security. A role is a collection of privileges that can be granted to or revoked from a user. A predefined role, named *DBA*, normally allows all administrative privileges on an Oracle database engine. The following privileges are not available for the DBA role on an Amazon RDS DB instance using the Oracle engine:

- Alter database
- Alter system
- Create any directory
- Drop any directory
- Grant any privilege
- Grant any role

When you create a DB instance, the master account that you use to create the instance gets DBA user privileges (with some limitations). Use this account for any administrative tasks such as creating additional user accounts in the database. The *SYS* user, *SYSTEM* user, and other administrative accounts can't be used.

Amazon RDS Oracle supports SSL/TLS encrypted connections and also the Oracle Native Network Encryption (NNE) option to encrypt connections between your application and your Oracle DB instance. For more information about using SSL with Oracle on Amazon RDS, see [Using SSL with an Oracle DB Instance \(p. 723\)](#). For more information about the Oracle Native Network Encryption option, see [Oracle Native Network Encryption \(p. 815\)](#).

Using SSL with an Oracle DB Instance

Secure Sockets Layer (SSL) is an industry standard protocol used for securing network connections between client and server. After SSL version 3.0, the name was changed to Transport Layer Security (TLS), but it is still often referred to as SSL and we refer to the protocol as SSL. Amazon RDS supports SSL encryption for Oracle DB instances. Using SSL, you can encrypt a connection between your application client and your Oracle DB instance. SSL support is available in all AWS regions for Oracle.

You enable SSL encryption for an Oracle DB instance by adding the Oracle SSL option to the option group associated with the DB instance. Amazon RDS uses a second port, as required by Oracle, for SSL connections which allows both clear text and SSL-encrypted communication to occur at the same time between a DB instance and an Oracle client. For example, you can use the port with clear text communication to communicate with other resources inside a VPC while using the port with SSL-encrypted communication to communicate with resources outside the VPC.

For more information, see [Oracle Secure Sockets Layer \(p. 817\)](#).

Note

You can't use both SSL and Oracle native network encryption (NNE) on the same DB instance. Before you can use SSL encryption, you must disable any other connection encryption.

Oracle 12c with Amazon RDS

Amazon RDS supports Oracle version 12c, which includes Oracle Enterprise Edition and Oracle Standard Edition Two. Oracle version 12c includes two major versions:

- [Oracle 12c Version 12.2.0.1 with Amazon RDS \(p. 724\)](#)
- [Oracle 12c Version 12.1.0.2 with Amazon RDS \(p. 727\)](#)

Oracle 12c Version 12.2.0.1 with Amazon RDS

Oracle 12c version 12.2.0.1 includes many new features and updates from the previous version. In this section, you can find the features and changes important to using Oracle 12c version 12.2.0.1 on Amazon RDS. For a complete list of the changes, see the [Oracle 12c version 12.2 documentation](#). For a complete list of features supported by each Oracle 12c edition, see [Permitted Features, Options, and Management Packs by Oracle Database Offering](#) in the Oracle documentation.

Oracle 12c version 12.1.0.2 includes sixteen new parameters that impact your Amazon RDS DB instance, and also 18 new system privileges, several no longer supported packages, and several new option group settings. For provide more information on these changes, see the following sections.

Amazon RDS Parameter Changes for Oracle 12c Version 12.2.0.1

Oracle 12c version 12.2.0.1 includes 20 new parameters in addition to several parameters with new ranges and new default values.

The following table shows the new Amazon RDS parameters for Oracle 12c version 12.2.0.1.

Name	Values	Modified	Description
allow_global_dblinks	TRUE, FALSE (default)	Y	Specifies whether LDAP lookup for database links is allowed for the database.
approx_for_aggregation	TRUE, FALSE (default)	Y	Replaces exact query processing for aggregation queries with approximate query processing.
approx_for_count_distinct	TRUE, FALSE (default)	Y	Automatically replaces COUNT (DISTINCT <i>expr</i>) queries with APPROX_COUNT_DISTINCT queries.
approx_for_percentile	NONE (default), PERCENTILE_CONT, PERCENTILE_CONT DETERMINISTIC, PERCENTILE_DISC, PERCENTILE_DISC DETERMINISTIC, ALL, ALL DETERMINISTIC	Y	Converts exact percentile functions to their approximate percentile function counterparts.
cursor_invalidation	DEFERRED, IMMEDIATE (default)	Y	Controls whether deferred cursor invalidation or immediate cursor invalidation is used for DDL statements by default.

Name	Values	Modifiable	Description
data_guard_sync_latency	0 (default) to the number of seconds specified by the NET_TIMEOUT attribute for the LOG_ARCHIVE_DEST_n parameter	Y	Controls how many seconds the Log Writer (LGWR) process waits beyond the response of the first in a series of Oracle Data Guard SYNC redo transport mode connections.
data_transfer_cache_size	0 – 512M, rounded up to the next granule size	Y	Sets the size of the data transfer cache (in bytes) used to receive data blocks (typically from a primary database in an Oracle Data Guard environment) for consumption by an instance during execution of an RMAN RECOVER ... NONLOGGED BLOCK command.
inmemory_adg_enabled	TRUE (default), FALSE	Y	Indicates whether in-memory for Active Data Guard is enabled in addition to the in-memory cache size.
inmemory_expressions_usage	STATIC_ONLY, DYNAMIC_ONLY,ENABLE (default), DISABLE	Y	Controls which In-Memory Expressions (IM expressions) are populated into the In-Memory Column Store (IM column store) and are available for queries.
inmemory_virtual_columns	ENABLE, MANUAL (default), DISABLE	Y	Controls which In-Memory Expressions (IM expressions) are populated into the In-Memory Column Store (IM column store) and are available for queries.
instance_abort_delay_time	0 (default) and higher	Y	Specifies how much time to delay an internal initiated instance abort (in seconds), such as when a fatal process dies or an unrecoverable instance error occurs.
instance_mode	READ-WRITE (default), READ-ONLY, READ-MOSTLY	N	Indicates whether the instance is read-write, read-only, or read-mostly.
long_module_action	TRUE (default), FALSE	Y	Enables the use of longer lengths for modules and actions.
max_idle_time	0 (default) to the maximum integer. The value of 0 indicates that there is no limit.	Y	Specifies the maximum number of minutes that a session can be idle. After that point, the session is automatically terminated.

Name	Values	Modifiable	Description
optimizer_adaptive_plans	TRUE (default), FALSE	Y	Controls adaptive plans. Adaptive plans are execution plans built with alternative choices that are decided at run time based on statistics collected as the query executes.
optimizer_adaptive_statistics	TRUE, FALSE (default)	Y	Controls adaptive statistics. Some query shapes are too complex to rely on base table statistics alone, so the optimizer augments these statistics with adaptive statistics.
outbound_dblink_protocols	ALL (default), NONE, TCP, TCPS, IPC	Y	Specifies the network protocols allowed for communicating for outbound database links in the database.
resource_manage_goldengate	TRUE, FALSE (default)	Y	Determines whether Oracle GoldenGate apply processes in the database are resource managed.
standby_db_preserve_states	NONE (default), SESSION, ALL	N	Controls whether user sessions and other internal states of the instance are retained when a readable physical standby database is converted to a primary database.
uniform_log_timestamp_format	TRUE (default), FALSE	Y	Specifies that a uniform timestamp format be used in Oracle Database trace (.trc) files and log files (such as the alert log).

The `compatible` parameter has a new default value for Oracle 12c version 12.2.0.1 on Amazon RDS. The following table shows the new default value.

Parameter Name	Oracle 12c Version 12.2.0.1 Default Value	Oracle 12c Version 12.1.0.2 Default Value
compatible	12.2.0	12.0.0

The `optimizer_features_enable` parameter has a new value range for Oracle 12c version 12.2.0.1 on Amazon RDS. For the old and new value ranges, see the following table.

Parameter Name	12c Version 12.2.0.1 Range	12c Version 12.1.0.2 Range
optimizer_features_e8_0_0 to 12.2.0.1		8.0.0 to 12.1.0.2

The following parameters were removed in Oracle 12c Version 12.2.0.1:

- `global_context_pool_size`
- `max_enabled_roles`
- `optimizer_adaptive_features`

- parallel_automatic_tuning
- parallel_degree_level
- use_indirect_data_buffers

The following parameter is not supported in Oracle 12c Version 12.2.0.1:

- sec_case_sensitive_logon

Amazon RDS Security Changes for Oracle 12c Version 12.2.0.1

In Oracle 12c version 12.2.0.1, direct grant of the privilege ADMINISTER DATABASE TRIGGER is required for the trigger owner. During a major version upgrade to Oracle 12c version 12.2.0.1, Amazon RDS grants this privilege to any user that owns a trigger so that the trigger owner has the required privileges. For more information, see the My Oracle Support document [2275535.1](#).

Oracle 12c Version 12.1.0.2 with Amazon RDS

Oracle 12c version 12.1.0.2 brings over 500 new features and updates from the previous version. In this section, you can find the features and changes important to using Oracle 12c version 12.1.0.2 on Amazon RDS. For a complete list of the changes, see the [Oracle 12c version 12.1 documentation](#). For a complete list of features supported by each Oracle 12c edition, see [Permitted Features, Options, and Management Packs by Oracle Database Edition](#) in the Oracle documentation.

Oracle 12c version 12.1.0.2 includes 16 new parameters that impact your Amazon RDS DB instance, and also 18 new system privileges, several no longer supported packages, and several new option group settings. For more information on these changes, see the following sections.

Amazon RDS Parameter Changes for Oracle 12c Version 12.1.0.2

Oracle 12c version 12.1.0.2 includes 16 new parameters in addition to several parameters with new ranges and new default values.

The following table shows the new Amazon RDS parameters for Oracle 12c version 12.1.0.2.

Name	Values	Modifi	Description
connection_brokers	CONNECTION_BROKERSN = broker_description[,...]		Specifies connection broker types, the number of connection brokers of each type, and the maximum number of connections per broker.
db_index_compression_inheritance	TABLESPACE, TABL, ALL, NONE	Y	Displays the options that are set for table or tablespace level compression inheritance.
db_big_table_cache_percent_target	0-90	Y	Specifies the cache section target size for automatic big table caching, as a percentage of the buffer cache.
heat_map	ON, OFF	Y	Enables the database to track read and write access of all segments and modification of database blocks that are due to data manipulation language (DML) and data definition language (DDL) statements.

Name	Values	Modified	Description
inmemory_clause_default	INMEMORY, NO INMEMORY	Y	INMEMORY_CLAUSE_DEFAULT enables you to specify a default In-Memory Column Store (IM column store) clause for new tables and materialized views.
inmemory_clause_default_memory_persistence	NO MEMCOMPRESS, MEMCOMPRESS FOR DML, MEMCOMPRESS FOR QUERY, MEMCOMPRESS FOR QUERY LOW, MEMCOMPRESS FOR QUERY HIGH, MEMCOMPRESS FOR CAPACITY, MEMCOMPRESS FOR CAPACITY LOW, MEMCOMPRESS FOR CAPACITY HIGH	Y	See INMEMORY_CLAUSE_DEFAULT.
inmemory_clause_default_priority	PRIORITY LOW, PRIORITY MEDIUM, PRIORITY HIGH, PRIORITY CRITICAL, PRIORITY NONE	Y	See INMEMORY_CLAUSE_DEFAULT.
inmemory_force	DEFAULT, OFF	Y	INMEMORY_FORCE allows you to specify whether tables and materialized view that are specified as INMEMORY are populated into the In-Memory Column Store (IM column store) or not.
inmemory_max_populate_servers	Null	N	INMEMORY_MAX_POPULATE_SERVERS specifies the maximum number of background populate servers to use for In-Memory Column Store (IM column store) population, so that these servers don't overload the rest of the system.
inmemory_query	ENABLE (default), DISABLE	Y	INMEMORY_QUERY is used to enable or disable in-memory queries for the entire database at the session or system level.
inmemory_size	0, 104857600-274877906944	Y	INMEMORY_SIZE sets the size of the In-Memory Column Store (IM column store) on a database instance.

Name	Values	Modifiable	Description
inmemory_trickle_repopulate_servers_percent	0 to 50 percent	Y	INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT limits the maximum number of background populate servers used for In-Memory Column Store (IM column store) repopulation. This limit is applied because trickle repopulation is designed to use only a small percentage of the populate servers.
max_string_size	STANDARD (default), EXTENDED	N	Controls the maximum size of VARCHAR2, NVARCHAR2, and RAW. For more information, see Using Extended Data Types (p. 739) .
optimizer_adaptive_features	TRUE (default), FALSE	Y	Enables or disables all of the adaptive optimizer features.
optimizer_adaptive_reporting_only	TRUE, FALSE (default)	Y	Controls reporting-only mode for adaptive optimizations.
pdb_file_name_convert	There is no default value.	N	Maps names of existing files to new file names.
pga_aggregate_limit	1-max of memory	Y	Specifies a limit on the aggregate PGA memory consumed by the instance.
processor_group_name	There is no default value.	N	Instructs the database instance to run itself within the specified operating system processor group.
spatial_vector_acceleration	TRUE, FALSE	N	Enables or disables the spatial vector acceleration, part of spatial option.
temp_undo_enabled	TRUE, FALSE (default)	Y	Determines whether transactions within a particular session can have a temporary undo log.
threaded_execution	TRUE, FALSE	N	Enables the multithreaded Oracle model, but prevents OS authentication.
unified_audit_sga_queue_size	1 MB - 30 MB	Y	Specifies the size of the system global area (SGA) queue for unified auditing.
use_dedicated_broker	TRUE, FALSE	N	Determines how dedicated servers are spawned.

Several parameter have new value ranges for Oracle 12c version 12.1.0.2 on Amazon RDS. For the old and new value ranges, see the following table.

Parameter Name	12c Version 12.1.0.2 Range	11g Range
audit_trail	os db [, extended] xml [, extended]	os db [, extended] xml [, extended] true false

Parameter Name	12c Version 12.1.0.2 Range	11g Range
<code>compatible</code>	For DB instances upgraded from Oracle 11g, automatically set to 12.0.0 on Amazon RDS unless a lower value is explicitly provided during the upgrade (as low as 11.2.0) For new Oracle 12c version 12.1.0.2 DB instances, starts with 12.0.0 on Amazon RDS	Starts with 11.2.0 on Amazon RDS
<code>db_securefile</code>	PERMITTED PREFERRED ALWAYS IGNORE FORCE	PERMITTED ALWAYS IGNORE FORCE
<code>db_writer_processes</code>	1-100	1-36
<code>optimizer_features_e8.0.0</code>	to 12.1.0.2	8.0.0 to 11.2.0.4
<code>parallel_degree_polid</code>	MANUAL,LIMITED,AUTO,ADAPTIVE	MANUAL,LIMITED,AUTO
<code>parallel_min_server</code>	0 to parallel_max_servers	CPU_COUNT * PARALLEL_THREADS_PER_CPU * 2 to parallel_max_servers

One parameter has a new default value for Oracle 12c on Amazon RDS. The following table shows the new default value.

Parameter Name	Oracle 12c Default Value	Oracle 11g Default Value
<code>job_queue_processes</code>	50	1000

Parameters in Amazon RDS are managed using parameter groups. For more information, see [Working with DB Parameter Groups \(p. 169\)](#). To view the supported parameters for a specific Oracle edition and version, run the AWS CLI `describe-engine-default-parameters` command.

For example, to view the supported parameters for Oracle Enterprise Edition 12c, version 12.1.0.2, run the following command.

```
aws rds describe-engine-default-parameters --db-parameter-group-family oracle-ee-12.1
```

Amazon RDS System Privileges for Oracle 12c Version 12.1.0.2

Several new system privileges have been granted to the system account for Oracle 12c version 12.1.0.2. These new system privileges include the following:

- ALTER ANY CUBE BUILD PROCESS
- ALTER ANY MEASURE FOLDER
- ALTER ANY SQL TRANSLATION PROFILE
- CREATE ANY SQL TRANSLATION PROFILE
- CREATE SQL TRANSLATION PROFILE
- DROP ANY SQL TRANSLATION PROFILE

- EM EXPRESS CONNECT
- EXEMPT DDL REDACTION POLICY
- EXEMPT DML REDACTION POLICY
- EXEMPT REDACTION POLICY
- LOGMINING
- REDEFINE ANY TABLE
- SELECT ANY CUBE BUILD PROCESS
- SELECT ANY MEASURE FOLDER
- USE ANY SQL TRANSLATION PROFILE

Amazon RDS Options for Oracle 12c Version 12.1.0.2

Several Oracle options changed between Oracle 11g and Oracle 12c version 12.1.0.2, though most of the options remain the same between the two versions. The Oracle 12c version 12.1.0.2 changes include the following:

- Oracle Enterprise Manager Database Express 12c replaced Oracle Enterprise Manager 11g Database Control. For more information, see [Oracle Enterprise Manager Database Express \(p. 796\)](#).
- The option XMLDB is installed by default in Oracle 12c version 12.1.0.2. You no longer need to install this option yourself.

Amazon RDS PL/SQL Packages for Oracle 12c Version 12.1.0.2

Oracle 12c version 12.1.0.2 includes a number of new built-in PL/SQL packages. The packages included with Amazon RDS Oracle 12c version 12.1.0.2 include the following.

Package Name	Description
CTX_ANL	The CTX_ANL package is used with AUTO_LEXER and provides procedures for adding and dropping a custom dictionary from the lexer.
DBMS_APP_CONT	The DBMS_APP_CONT package provides an interface to determine if the in-flight transaction on a now unavailable session committed or not, and if the last call on that session completed or not.
DBMS_AUTO_REPORT	The DBMS_AUTO_REPORT package provides an interface to view SQL Monitoring and Real-time Automatic Database Diagnostic Monitor (ADDM) data that has been captured into Automatic Workload Repository (AWR).
DBMS_GOLDENGATE_AUTH	The DBMS_GOLDENGATE_AUTH package provides subprograms for granting privileges to and revoking privileges from GoldenGate administrators.
DBMS_HEAT_MAP	The DBMS_HEAT_MAP package provides an interface to externalize heatmaps at various levels of storage including block, extent, segment, object and tablespace.
DBMS_ILM	The DBMS_ILM package provides an interface for implementing Information Lifecycle Management (ILM) strategies using Automatic Data Optimization (ADO) policies.

Package Name	Description
DBMS_ILM_ADMIN	The DBMS_ILM_ADMIN package provides an interface to customize Automatic Data Optimization (ADO) policy execution.
DBMS_PART	The DBMS_PART package provides an interface for maintenance and management operations on partitioned objects.
DBMS_PRIVILEGE_CAPTURE	The DBMS_PRIVILEGE_CAPTURE package provides an interface to database privilege analysis.
DBMS_QOPATCH	The DBMS_QOPATCH package provides an interface to view the installed database patches.
DBMS_REDACT	The DBMS_REDACT package provides an interface to Oracle Data Redaction, which enables you to mask (redact) data that is returned from queries issued by low-privileged users or an application.
DBMS_SPD	The DBMS_SPD package provides subprograms for managing SQL plan directives (SPD).
DBMS_SQL_TRANSLATOR	The DBMS_SQL_TRANSLATOR package provides an interface for creating, configuring, and using SQL translation profiles.
DBMS_SQL_MONITOR	The DBMS_SQL_MONITOR package provides information about real-time SQL Monitoring and real-time Database Operation Monitoring.
DBMS_SYNC_REFRESH	The DBMS_SYNC_REFRESH package provides an interface to perform a synchronous refresh of materialized views.
DBMS_TSDP_MANAGE	The DBMS_TSDP_MANAGE package provides an interface to import and manage sensitive columns and sensitive column types in the database, and is used in conjunction with the DBMS_TSDP_PROTECT package with regard to transparent sensitive data protection (TSDP) policies. DBMS_TSDP_MANAGE is available with the Enterprise Edition only.
DBMS_TSDP_PROTECT	The DBMS_TSDP_PROTECT package provides an interface to configure transparent sensitive data protection (TSDP) policies in conjunction with the DBMS_TSDP_MANAGE package. DBMS_TSDP_PROTECT is available with the Enterprise Edition only.
DBMS_XDB_CONFIG	The DBMS_XDB_CONFIG package provides an interface for configuring Oracle XML DB and its repository.
DBMS_XDB_CONSTANTS	The DBMS_XDB_CONSTANTS package provides an interface to commonly used constants. Oracle recommends using constants instead of dynamic strings to avoid typographical errors.
DBMS_XDB_REPOS	The DBMS_XDB_REPOS package provides an interface to operate on the Oracle XML database Repository.
DBMS_XMLSCHEMA_ANNOTATE	The DBMS_XMLSCHEMA_ANNOTATE package provides an interface to manage and configure the structured storage model, mainly through the use of pre-registration schema annotations.
DBMS_XMLSTORAGE_MANAGE	The DBMS_XMLSTORAGE_MANAGE package provides an interface to manage and modify XML storage after schema registration has been completed.

Package Name	Description
DBMS_XSTREAM_ADMIN	The DBMS_XSTREAM_ADMIN package provides interfaces for streaming database changes between an Oracle database and other systems. XStream enables applications to stream out or stream in database changes.
DBMS_XSTREAM_AUTH	The DBMS_XSTREAM_AUTH package provides subprograms for granting privileges to and revoking privileges from XStream administrators.
UTL_CALL_STACK	The UTL_CALL_STACK package provides an interface to provide information about currently executing subprograms.

Oracle 12c Version 12.1.0.2 Features Not Supported

The following features are not supported for Oracle 12c version 12.1.0.2 on Amazon RDS:

- Automated Storage Management
- Data Guard / Active Data Guard
- Database Vault
- Multitenant Database
- Real Application Clusters (RAC)
- Real Application Testing
- Pure Unified Auditing Mode
- Mixed Unified Auditing Mode

Note

Mixed Unified Auditing Mode is supported for version 12.2.0.1 on Amazon RDS.

Several Oracle 11g PL/SQL packages are not supported in Oracle 12c version 12.1.0.2. These packages include the following:

- DBMS_AUTO_TASK_IMMEDIATE
- DBMS_CDC_PUBLISH
- DBMS_CDC_SUBSCRIBE
- DBMS_EXPFIL
- DBMS_OBFUSCATION_TOOLKIT
- DBMS_RLMGR
- SDO_NET_MEM

Oracle 11g with Amazon RDS

Oracle 11g Supported Features

The following list shows the Oracle 11g features supported by Amazon RDS.

- Total Recall
- Flashback Table, Query and Transaction Query

- Virtual Private Database
- Fine-Grained Auditing
- Comprehensive support for Microsoft .NET, OLE DB, and ODBC
- Automatic Memory Management
- Automatic Undo Management
- Advanced Compression
- Partitioning
- Star Query Optimization
- Summary Management - Materialized View Query Rewrite
- Oracle Data Redaction
- Distributed Queries/Transactions
- Text
- Materialized Views
- Import/Export and sqldr Support
- Oracle Enterprise Manager Database Control
- Oracle XML DB (without the XML DB Protocol Server)
- Oracle Application Express
- Automatic Workload Repository for Enterprise Edition (AWR). For more information, see [Working with Automatic Workload Repository \(AWR\) \(p. 858\)](#)
- Datapump (network only)
- Native network encryption
- Transparent data encryption (Oracle TDE), part of the Oracle Advanced Security feature

Oracle 11g Features Not Supported

The following features are not supported for Oracle 11g on Amazon RDS:

- Real Application Clusters (RAC)
- Real Application Testing
- Data Guard / Active Data Guard
- Oracle Enterprise Manager Grid Control
- Automated Storage Management
- Database Vault
- Streams
- Oracle Label Security
- Oracle XML DB Protocol Server

Amazon RDS Parameters for Oracle 11g

Parameters in Amazon RDS are managed using parameter groups. See [Working with DB Parameter Groups \(p. 169\)](#) for more information. To view the supported parameters for a specific Oracle edition and version, you can run the AWS CLI `describe-engine-default-parameters` command.

For example, to view the supported parameters for Oracle Enterprise Edition, version 11g, run the following command:

```
aws rds describe-engine-default-parameters --db-parameter-group-family oracle-ee-11.2
```

Oracle Engine Version Management

DB Engine Version Management is a feature of Amazon RDS that enables you to control when and how the database engine software running your DB instances is patched and upgraded. This feature gives you the flexibility to maintain compatibility with database engine patch versions, test new patch versions to ensure they work effectively with your application before deploying in production, and perform version upgrades on your own terms and timelines.

Note

Amazon RDS periodically aggregates official Oracle database patches using an Amazon RDS-specific DB Engine version. To see a list of which Oracle patches are contained in an Amazon RDS Oracle-specific engine version, go to [Oracle Database Engine Release Notes \(p. 921\)](#).

Currently, you perform all Oracle database upgrades manually. For more information about upgrading an Oracle DB instance, see [Upgrading the Oracle DB Engine \(p. 770\)](#).

Deprecation of Oracle 11.2.0.2

In 2017, Amazon RDS is deprecating support for Oracle version 11.2.0.2. Oracle is no longer providing patches for this version. Therefore, to provide the best experience for AWS customers, we are deprecating this version.

There are no longer any production DB instances running Oracle version 11.2.0.2. You might still have a snapshot of an 11.2.0.2 DB instance.

Amazon RDS is deprecating support for Oracle version 11.2.0.2 according to the following schedule.

Date	Information
August 4, 2016	You can no longer create DB instances that use Oracle version 11.2.0.2.
April 15, 2019	Any 11.2.0.2 snapshots are upgraded to 11.2.0.4. You can upgrade your snapshots yourself prior to this date. For more information, see Upgrading an Oracle DB Snapshot (p. 774) .

Deprecation of Oracle 11.2.0.3

In 2017, Amazon RDS is deprecating support for Oracle version 11.2.0.3. Oracle is no longer providing patches for this version. Therefore, to provide the best experience for AWS customers, we are deprecating this version.

There are no longer any production DB instances running Oracle version 11.2.0.3. You might still have a snapshot of an 11.2.0.3 DB instance.

Amazon RDS is deprecating support for Oracle version 11.2.0.3 according to the following schedule.

Date	Information
August 4, 2016	You can no longer create DB instances that use Oracle version 11.2.0.3.

Date	Information
March 15, 2019	Any 11.2.0.3 snapshots are upgraded to 11.2.0.4. You can upgrade your snapshots yourself prior to this date. For more information, see Upgrading an Oracle DB Snapshot (p. 774) .

Deprecation of Oracle 12.1.0.1

In 2017, Amazon RDS is deprecating support for Oracle version 12.1.0.1. Oracle is no longer providing patches for this version. Therefore, to provide the best experience for AWS customers, we are deprecating this version.

There are no longer any production DB instances running Oracle version 12.1.0.1. You might still have a snapshot of a 12.1.0.1 DB instance.

Amazon RDS will deprecate support for Oracle version 12.1.0.1 according to the following schedule.

Date	Information
February 15, 2017	You can no longer create DB instances that use Oracle version 12.1.0.1.
June 1, 2019	Any 12.1.0.1 snapshots are upgraded to 12.1.0.2. You can upgrade your snapshots yourself prior to this date. For more information, see Upgrading an Oracle DB Snapshot (p. 774) .

Using Huge Pages with an Oracle DB Instance

Amazon RDS for Oracle supports Linux kernel huge pages for increased database scalability. The use of huge pages results in smaller page tables and less CPU time spent on memory management, increasing the performance of large database instances. For more information, see [Overview of HugePages](#) in the Oracle documentation.

You can use huge pages with the following versions and editions of Oracle:

- 12.2.0.1, all editions
- 12.1.0.2, all editions
- 11.2.0.4, all editions

The `use_large_pages` parameter controls whether huge pages are enabled for a DB instance. The possible settings for this parameter are `ONLY`, `FALSE`, and `{DBInstanceClassHugePagesDefault}`. The `use_large_pages` parameter is set to `{DBInstanceClassHugePagesDefault}` in the default DB parameter group for Oracle.

To control whether huge pages are enabled for a DB instance automatically, you can use the `DBInstanceClassHugePagesDefault` formula variable in parameter groups. The value is determined as follows:

- For the DB instance classes mentioned in the table below, `DBInstanceClassHugePagesDefault` always evaluates to `FALSE` by default, and `use_large_pages` evaluates to `FALSE`. You can enable

huge pages manually if the instance class is in the db.t2, db.r3, or db.m4 family and it has at least 14 GiB of memory.

- For DB instance classes not mentioned in the table below, if the instance class has less than 100 GiB of memory, `DBInstanceClassHugePagesDefault` evaluates to `TRUE` by default, and `use_large_pages` evaluates to `ONLY`.
- For DB instance classes not mentioned in the table below, if the instance class has at least 100 GiB of memory, `DBInstanceClassHugePagesDefault` always evaluates to `TRUE`, and `use_large_pages` evaluates to `ONLY`.

Huge pages are not enabled by default for the following DB instance classes.

DB Instance Class Family	DB Instance Classes with Huge Pages Not Enabled by Default
db.m4	db.m4.large, db.m4.xlarge, db.m4.2xlarge, db.m4.4xlarge, db.m4.10xlarge
db.m3	db.m3.medium, db.m3.large, db.m3.xlarge, db.m3.2xlarge
db.r3	db.r3.large, db.r3.xlarge, db.r3.2xlarge, db.r3.4xlarge, db.r3.8xlarge
db.t2	db.t2.micro, db.t2.small, db.t2.medium, db.t2.large

For more information about DB instance classes, see [Specifications for All Available DB Instance Classes \(p. 82\)](#).

To enable huge pages for new or existing DB instances manually, set the `use_large_pages` parameter to `ONLY`. You can't use huge pages with Oracle Automatic Memory Management (AMM). If you set the parameter `use_large_pages` to `ONLY`, then you must also set both `memory_target` and `memory_max_target` to 0. For more information about setting DB parameters for your DB instance, see [Working with DB Parameter Groups \(p. 169\)](#).

You can also set the `sga_target`, `sga_max_size`, and `pga_aggregate_target` parameters. When you set system global area (SGA) and program global area (PGA) memory parameters, add the values together. Subtract this total from your available instance memory (`DBInstanceClassMemory`) to determine the free memory beyond the huge pages allocation. You must leave free memory of at least 2 GiB, or 10 percent of the total available instance memory, whichever is smaller.

After you configure your parameters, you must reboot your DB instance for the changes to take effect. For more information, see [Rebooting a DB Instance \(p. 129\)](#).

The following is a sample parameter configuration for huge pages that enables huge pages manually. You should set the values to meet your needs.

```

memory_target          = 0
memory_max_target     = 0
pga_aggregate_target = {DBInstanceClassMemory*1/8}
sga_target            = {DBInstanceClassMemory*3/4}
sga_max_size          = {DBInstanceClassMemory*3/4}
use_large_pages       = ONLY

```

Assume the following parameters values are set in a parameter group.

```

memory_target          = IF({DBInstanceClassHugePagesDefault}, 0,
{DBInstanceClassMemory*3/4})

```

```

memory_max_target      = IF({DBInstanceClassHugePagesDefault}, 0,
                           {DBInstanceClassMemory*3/4})
pga_aggregate_target  = IF({DBInstanceClassHugePagesDefault},
                           {DBInstanceClassMemory*1/8}, 0)
sga_target              = IF({DBInstanceClassHugePagesDefault},
                           {DBInstanceClassMemory*3/4}, 0)
sga_max_size            = IF({DBInstanceClassHugePagesDefault},
                           {DBInstanceClassMemory*3/4}, 0)
use_large_pages         = {DBInstanceClassHugePagesDefault}

```

The parameter group is used by a db.r4 DB instance class with less than 100 GiB of memory and a db.r3 instance with more than 100 GiB memory. With these parameter settings and `use_large_pages` set to `{DBInstanceClassHugePagesDefault}`, huge pages are enabled on the db.r4 instance, but disabled on the db.r3 instance.

Consider another example with following parameters values set in a parameter group.

```

memory_target          = IF({DBInstanceClassHugePagesDefault}, 0,
                           {DBInstanceClassMemory*3/4})
memory_max_target      = IF({DBInstanceClassHugePagesDefault}, 0,
                           {DBInstanceClassMemory*3/4})
pga_aggregate_target  = IF({DBInstanceClassHugePagesDefault},
                           {DBInstanceClassMemory*1/8}, 0)
sga_target              = IF({DBInstanceClassHugePagesDefault},
                           {DBInstanceClassMemory*3/4}, 0)
sga_max_size            = IF({DBInstanceClassHugePagesDefault},
                           {DBInstanceClassMemory*3/4}, 0)
use_large_pages         = FALSE

```

The parameter group is used by a db.r4 DB instance class with less than 100 GiB of memory and a db.r3 instance with more than 100 GiB memory. With these parameter settings, huge pages are disabled on both the db.r4 instance and the db.r3 instance.

Note

If this parameter group is used by a db.r4 DB instance class with at least 100 GiB of memory, the `FALSE` setting for `use_large_pages` is overridden and set to `ONLY`. In this case, a customer notification regarding the override is sent.

After huge pages are active on your DB instance, you can view huge pages information by enabling enhanced monitoring. For more information, see [Enhanced Monitoring \(p. 256\)](#).

Using `utl_http`, `utl_tcp`, and `utl_smtp` with an Oracle DB Instance

Amazon RDS supports outbound network access on your DB instances running Oracle. You can use `utl_http`, `utl_tcp`, and `utl_smtp` to connect from your DB instance to the network.

Note the following about working with outbound network access:

- To use `utl_http` on DB instances running Oracle 11g, you must install the XMLDB option. For more information, see [Oracle XML DB \(p. 840\)](#).
- Outbound network access with `utl_http`, `utl_tcp`, and `utl_smtp` is supported only for Oracle DB instances in a VPC. To determine whether or not your DB instance is in a VPC, see [Determining](#)

[Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#). To move a DB instance not in a VPC into a VPC, see [Moving a DB Instance Not in a VPC into a VPC \(p. 426\)](#).

- To use SMTP with the UTL_MAIL option, see [Oracle UTL_MAIL \(p. 838\)](#).
- To connect securely to remote SSL/TLS resources by creating and uploading custom Oracle wallets, follow the instructions in [Provisioning Oracle Wallets and Accessing SSL/TLS-Based Endpoints on Amazon RDS for Oracle](#).

The specific certificates that are required for your wallet vary by service. For AWS services, these can typically be found in the [Amazon Trust Services Repository](#).

- The Domain Name Server (DNS) name of the remote host can be any of the following:
 - Publicly resolvable.
 - The endpoint of an Amazon RDS DB instance.
 - Resolvable through a custom DNS server. For more information, see [Setting Up a Custom DNS Server \(p. 853\)](#).
 - The private DNS name of an Amazon EC2 instance in the same VPC or a peered VPC. In this case, make sure that the name is resolvable through a custom DNS server. Alternatively, to use the DNS provided by Amazon, you can enable the enableDnsSupport attribute in the VPC settings and enable DNS resolution support for the VPC peering connection. For more information, see [DNS Support in Your VPC](#) and [Modifying Your VPC Peering Connection](#).

Using OEM, APEX, TDE, and Other Options

Most Amazon RDS DB engines support option groups that allow you to select additional features for your DB instance. Oracle DB instances support several options, including Oracle Enterprise Manager (OEM), Transparent Data Encryption (TDE), Application Express (APEX), and Native Network Encryption. For a complete list of supported Oracle options, see [Options for Oracle DB Instances \(p. 787\)](#). For more information about working with option groups, see [Working with Option Groups \(p. 156\)](#).

Using Extended Data Types

Amazon RDS Oracle version 12c supports extended data types. With extended data types, the maximum size is 32,767 bytes for the VARCHAR2, NVARCHAR2, and RAW data types. To use extended data types, set the `MAX_STRING_SIZE` parameter to EXTENDED. For more information, see [Extended Data Types](#) in the Oracle documentation.

If you don't want to use extended data types, keep the `MAX_STRING_SIZE` parameter set to STANDARD (the default). When this parameter is set to STANDARD, the size limits are 4,000 bytes for the VARCHAR2 and NVARCHAR2 data types, and 2,000 bytes for the RAW data type.

You can enable extended data types on a new or existing DB instance. For new DB instances, DB instance creation time is typically longer when you enable extended data types. For existing DB instances, the DB instance is unavailable during the conversion process.

The following are considerations for a DB instance with extended data types enabled:

- When you enable extended data types for a DB instance, you can't change the DB instance back to use the standard size for data types. After a DB instance is converted to use extended data types, if you set the `MAX_STRING_SIZE` parameter back to STANDARD it results in the `incompatible-parameters` status.
- When you restore a DB instance that uses extended data types, you must specify a parameter group with the `MAX_STRING_SIZE` parameter set to EXTENDED. During restore, if you specify the default

parameter group or any other parameter group with `MAX_STRING_SIZE` set to `STANDARD` it results in the `incompatible-parameters` status.

- We recommend that you don't enable extended data types for Oracle DB instances running on the `t2.micro` DB instance class.

When the DB instance status is `incompatible-parameters` because of the `MAX_STRING_SIZE` setting, the DB instance remains unavailable until you set the `MAX_STRING_SIZE` parameter to `EXTENDED` and reboot the DB instance.

Enabling Extended Data Types for a New DB Instance

To enable extended data types for a new DB instance

1. Set the `MAX_STRING_SIZE` parameter to `EXTENDED` in a parameter group.

To set the parameter, you can either create a new parameter group or modify an existing parameter group.

For more information, see [Working with DB Parameter Groups \(p. 169\)](#).

2. Create a new Amazon RDS Oracle DB instance, and associate the parameter group with `MAX_STRING_SIZE` set to `EXTENDED` with the DB instance.

For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).

Enabling Extended Data Types for an Existing DB Instance

When you modify a DB instance to enable extended data types, the data in the database is converted to use the extended sizes. The DB instance is unavailable during the conversion. The amount of time it takes to convert the data depends on the DB instance class used by the DB instance and the size of the database.

Note

After you enable extended data types, you can't perform a point-in-time restore to a time during the conversion. You can restore to the time immediately before the conversion or after the conversion.

To enable extended data types for an existing DB instance

1. Take a snapshot of the database.

If there are invalid objects in the database, Amazon RDS tries to recompile them. The conversion to extended data types can fail if Amazon RDS can't recompile an invalid object. The snapshot enables you to restore the database if there is a problem with the conversion. Always check for invalid objects before conversion and fix or drop those invalid objects. For production databases, we recommend testing the conversion process on a copy of your DB instance first.

For more information, see [Creating a DB Snapshot \(p. 216\)](#).

2. Set the `MAX_STRING_SIZE` parameter to `EXTENDED` in a parameter group.

To set the parameter, you can either create a new parameter group or modify an existing parameter group.

For more information, see [Working with DB Parameter Groups \(p. 169\)](#).

3. Modify the DB instance to associate it with the parameter group with `MAX_STRING_SIZE` set to `EXTENDED`.

For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

4. Reboot the DB instance for the parameter change to take effect.

For more information, see [Rebooting a DB Instance \(p. 129\)](#).

Creating a DB Instance Running the Oracle Database Engine

The basic building block of Amazon RDS is the DB instance. This is the environment in which you run your Oracle databases.

Important

You must complete the tasks in the [Setting Up for Amazon RDS \(p. 5\)](#) section before you can create or connect to a DB instance.

For an example that walks you through the process of creating and connecting to a sample DB instance, see [Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance \(p. 36\)](#).

AWS Management Console

To launch an Oracle DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the AWS Management Console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
4. Choose **Create database** to open the **Select engine** page.

The Oracle editions that are available vary by AWS Region.

Select engine

Engine options

Amazon Aurora

Amazon
Aurora

MySQL



MariaDB



PostgreSQL



Oracle

ORACLE®

Microsoft SQL Server



Oracle

Edition

Oracle Enterprise Edition

Efficient, reliable, and secure database management system that delivers comprehensive high-end capabilities for mission-critical applications and demanding database workloads.

Oracle Standard Edition

Affordable and full-featured database management system supporting up to 32 vCPUs.

Oracle Standard Edition One

Affordable and full-featured database management system supporting up to 16 vCPUs.

Oracle Standard Edition Two

Affordable and full-featured database management system supporting up to 16 vCPUs. Oracle Database Standard Edition Two is a replacement for Standard Edition and Standard Edition One.

Only enable options eligible for RDS Free Usage Tier [info](#)

Cancel

Next

5. In the **Select engine** window, choose the **Select** button for the Oracle DB engine you want to use and then choose **Next**.
6. The next step asks if you are planning to use the DB instance you are creating for production. If you are, choose **Production**. When you choose **Production**, the following are preselected in a later step:
 - **Multi-AZ deployment** failover option
 - **Provisioned IOPS** storage option
 - **Enable deletion protection** option
7. Choose **Next** to continue. The **Specify DB details** page appears.

On the **Specify DB details** page, specify your DB instance information. For information about each setting, see [Settings for Oracle DB Instances \(p. 746\)](#).

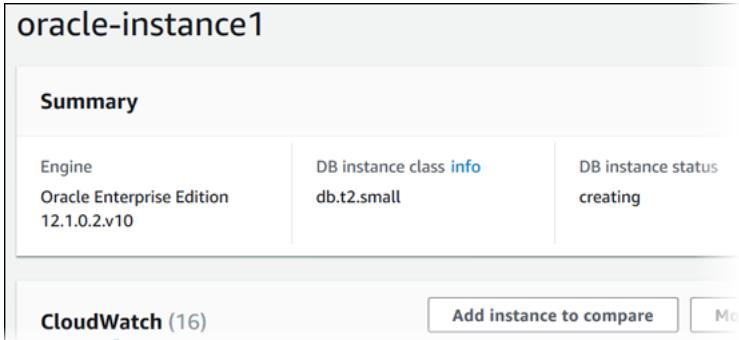
The screenshot shows the 'Specify DB details' page. At the top, there's a link to 'Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator'. Below that, the 'DB engine' is set to 'Oracle Database Enterprise Edition'. Under 'License model', the dropdown shows 'bring-your-own-license'. For 'DB engine version', it's set to 'Oracle 12.1.0.2.v14'. The 'DB instance class' is 'db.r4.xlarge'. In the 'Multi-AZ deployment' section, the radio button for 'Create replica in different zone' is selected, with a note explaining it creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups. The 'Storage type' is set to 'Provisioned IOPS (SSD)'.

8. Choose **Next** to continue. The **Configure advanced settings** page appears.

On the **Configure advanced settings** page, provide additional information that RDS needs to launch the DB instance. For information about each setting, see [Settings for Oracle DB Instances \(p. 746\)](#).

9. Choose **Create database**.
10. On the final page, choose **View DB instance details**.

On the RDS console, the details for the new DB instance appear. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and storage allocated, it could take several minutes for the new instance to be available.



CLI

To create an Oracle DB instance by using the AWS CLI, call the [create-db-instance](#) command with the parameters below. For information about each setting, see [Settings for Oracle DB Instances \(p. 746\)](#).

- `--db-instance-identifier`
- `--db-instance-class`
- `--db-security-groups`
- `--db-subnet-group`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--allocated-storage`
- `--backup-retention-period`

Example

The following command will launch the example DB instance.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
--engine oracle-se1 \
--db-instance-identifier mydbinstance \
--allocated-storage 20 \
--db-instance-class db.m1.small \
--db-security-groups mydbsecuritygroup \
--db-subnet-group mydbsubnetgroup \
--master-username masterawsuser \
--master-user-password masteruserpassword \
--backup-retention-period 3
```

For Windows:

```
aws rds create-db-instance ^
--engine oracle-se1 ^
--db-instance-identifier mydbinstance ^
--allocated-storage 20 ^
--db-instance-class db.m1.small ^
--db-security-groups mydbsecuritygroup ^
--db-subnet-group mydbsubnetgroup ^
--master-username masterawsuser ^
--master-user-password masteruserpassword ^
```

```
--backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE mydbinstance db.m1.small oracle-se1 20 sa creating 3 **** n
11.2.0.4.v1
SECGROUP default active
PARAMGRP default.oracle-se1-11.2 in-sync
```

API

To create an Oracle DB instance by using the Amazon RDS API, call the [CreateDBInstance](#) action with the parameters below. For information about each setting, see [Settings for Oracle DB Instances \(p. 746\)](#).

- `AllocatedStorage`
- `BackupRetentionPeriod`
- `DBInstanceState`
- `DBInstanceIdentifier`
- `DBSecurityGroups`
- `DBSubnetGroup`
- `Engine`
- `MasterUsername`
- `MasterUserPassword`

Example

```
https://rds.amazonaws.com/
?Action=CreateDBInstance
&AllocatedStorage=250
&BackupRetentionPeriod=3
&DBInstanceState=db.m1.large
&DBInstanceIdentifier=mydbinstance
&DBSecurityGroups.member.1=mysecuritygroup
&DBSubnetGroup=mydbsubnetgroup
&Engine=oracle-se1
&MasterUserPassword=masteruserpassword
&MasterUsername=masterawsuser
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-1/rds/aws4_request
&X-Amz-Date=20140305T185838Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=b441901545441d3c7a48f63b5b1522c5b2b37c137500c93c45e209d4b3a064a3
```

Settings for Oracle DB Instances

The following table contains details about settings that you choose when you create an Oracle DB instance.

Setting	Setting Description
Allocated storage	The amount of storage to allocate your DB instance (in gigabytes). In some cases, allocating a higher amount of

Setting	Setting Description
	<p>storage for your DB instance than the size of your database can improve I/O performance.</p> <p>For more information, see DB instance storage (p. 103).</p>
Auto minor version upgrade	<p>Amazon RDS does not support automatic minor version upgrades for DB instances running Oracle. You must modify your DB instance manually to perform a minor version upgrade.</p> <p>Some options, such as Oracle Locator, Oracle Multimedia, and Oracle Spatial, require that you enable automatic minor version upgrades. Upgrades for DB instances that use these options are installed during your scheduled maintenance window, and an outage occurs during the upgrade. You can't disable automatic minor version upgrades at the same time as you modify the option group to remove such an option.</p>
Availability zone	<p>The availability zone for your DB instance. Use the default of No Preference unless you need to specify a particular Availability Zone.</p> <p>For more information, see Regions and Availability Zones (p. 101).</p>
Backup retention period	<p>The number of days that you want automatic backups of your DB instance to be retained. For any non-trivial instance, you should set this value to 1 or greater.</p> <p>For more information, see Working With Backups (p. 208).</p>
Backup window	<p>The time period during which Amazon RDS automatically takes a backup of your DB instance. Unless you have a specific time that you want to have your database backup, use the default of No Preference.</p> <p>For more information, see Working With Backups (p. 208).</p>
Character set name	<p>The character set for your DB instance. The default value of AL32UTF8 is for the Unicode 5.0 UTF-8 Universal character set. You cannot change the character set after the DB instance is created.</p> <p>For more information, see Oracle Character Sets Supported in Amazon RDS (p. 784).</p>
Copy tags to snapshots	<p>Select this option to copy any DB instance tags to a DB snapshot when you create a snapshot.</p> <p>For more information, see Tagging Amazon RDS Resources (p. 138).</p>

Setting	Setting Description
Database name	The name for the database on your DB instance. The name must begin with a letter and contain up to 8 alpha-numeric characters. You can't specify the string NULL, or any other reserved word, for the database name. If you do not provide a name, Amazon RDS does not create a database on the DB instance you are creating.
Database port	The port that you want to access the DB instance through. Oracle installations default to port 1521.
DB engine version	The version of Oracle that you want to use.
DB instance class	The DB instance class that you want to use. For more information, see DB Instance Class (p. 82) and DB Instance Class Support for Oracle (p. 720) .
DB instance identifier	The name for your DB instance. The name must be unique for your account and AWS Region. You can add some intelligence to the name, such as including the AWS Region and DB engine you chose, for example <code>oracle-instance1</code> .
DB parameter group	A parameter group for your DB instance. You can choose the default parameter group or you can create a custom parameter group. For more information, see Working with DB Parameter Groups (p. 169) and Modifying Oracle sqlnet.ora Parameters (p. 767) .
Deletion protection	Enable deletion protection to prevent your DB instance from being deleted. If you create a production DB instance with the AWS Management Console, deletion protection is enabled by default. For more information, see Deleting a DB Instance (p. 135) .
Encryption	Enable Encryption to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 389) .
Enhanced monitoring	Enable enhanced monitoring to gather metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 256) .
License model	The license model that you want to use. Choose license-included to use the general license agreement for Oracle. Choose bring-your-own-license to use your existing Oracle license. For more information, see Oracle Licensing (p. 719) .

Setting	Setting Description
Maintenance window	The 30 minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose No Preference . For more information, see The Amazon RDS Maintenance Window (p. 120) .
Master username	The name that you use as the master user name to log on to your DB instance with all database privileges. This user account is used to log into the DB instance and is granted DBA privileges. For more information, see Oracle Security (p. 723) .
Master password	The password for your master user account. The password must contain from 8 to 30 printable ASCII characters (excluding /, ", and @).
Multi-AZ deployment	Create replica in different zone to create a standby replica of your DB instance in another availability zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose No . For more information, see Regions and Availability Zones (p. 101) .
Option group	An option group for your DB instance. You can choose the default option group or you can create a custom option group. For more information, see Working with Option Groups (p. 156) .
Public accessibility	Yes to give your DB instance a public IP address. This means that it is accessible outside the VPC (the DB instance also needs to be in a public subnet in the VPC). Choose No if you want the DB instance to only be accessible from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 422) .
Storage type	The storage type for your DB instance. For more information, see Amazon RDS Storage Types (p. 103) .
Subnet group	This setting depends on the platform you are on. If you are a new customer to AWS, choose default , which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC.

Setting	Setting Description
Virtual Private Cloud (VPC)	<p>This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC. If you are creating a DB instance on the previous E2-Classic platform, choose Not in VPC.</p> <p>For more information, see Amazon Virtual Private Cloud (VP Cs) and Amazon RDS (p. 412).</p>
VPC security groups	<p>If you are a new customer to AWS, choose Create new VPC security group. Otherwise, choose Select existing VPC security groups, and select security groups you previously created.</p> <p>When you choose Create new VPC security group in the RDS console, a new security group is created with an inbound rule that allows access to the DB instance from the IP address detected in your browser.</p> <p>For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 399).</p>

Related Topics

- [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#)
- [Connecting to a DB Instance Running the Oracle Database Engine \(p. 751\)](#)
- [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#)
- [Deleting a DB Instance \(p. 135\)](#)

Connecting to a DB Instance Running the Oracle Database Engine

After Amazon RDS provisions your Oracle DB instance, you can use any standard SQL client application to connect to the DB instance. In this topic, you connect to a DB instance that is running the Oracle database engine by using Oracle SQL Developer or SQL*Plus.

For an example that walks you through the process of creating and connecting to a sample DB instance, see [Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance \(p. 36\)](#).

Finding the Endpoint of Your DB Instance

Each Amazon RDS DB instance has an endpoint, and each endpoint has the DNS name and port number for the DB instance. To connect to your DB instance using a SQL client application, you need the DNS name and port number for your DB instance.

You can find the endpoint for a DB instance using the Amazon RDS console or the AWS CLI.

AWS Management Console

To find the endpoint using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the upper-right corner of the console, choose the AWS Region of your DB instance.
3. Find the DNS name and port number for your DB instance.
 - a. Choose **Databases** to display a list of your DB instances.
 - b. Choose the Oracle DB instance name to display the instance details.
 - c. On the **Connectivity** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.

oracle-instance1

Summary

DB Name
oracle-instance1

Role
Instance

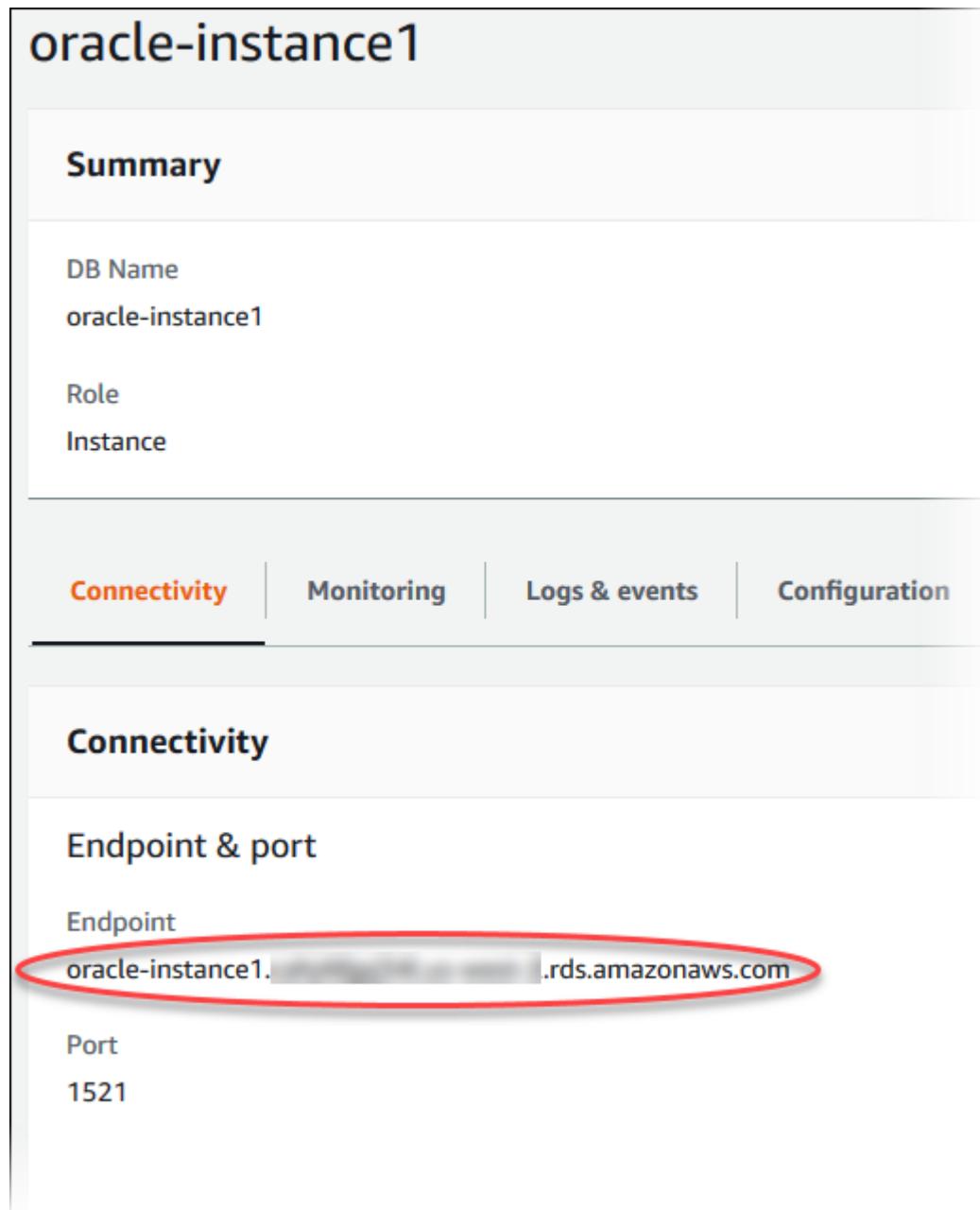
Connectivity Monitoring Logs & events Configuration

Connectivity

Endpoint & port

Endpoint
oracle-instance1.XXXXXXXXXX.rds.amazonaws.com

Port
1521



CLI

To find the endpoint of an Oracle DB instance by using the AWS CLI, call the [describe-db-instances](#) command.

Example To find the endpoint using the AWS CLI

```
aws rds describe-db-instances
```

Search for `Endpoint` in the output to find the DNS name and port number for your DB instance. The `Address` line in the output contains the DNS name. The following is an example of the JSON endpoint output.

```
"Endpoint": {  
    "HostedZoneId": "Z1PVIF0B656C1W",  
    "Port": 3306,  
    "Address": "myinstance.123456789012.us-west-2.rds.amazonaws.com"  
},
```

Note

The output might contain information for multiple DB instances.

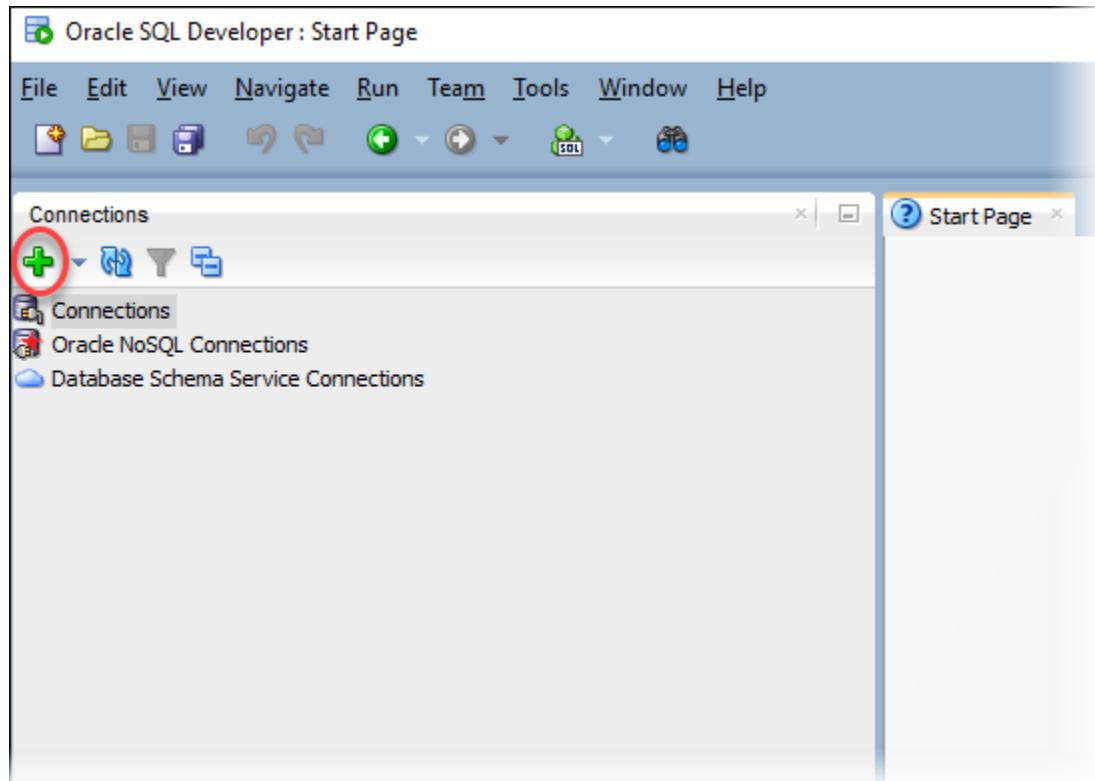
Connecting to Your DB Instance Using Oracle SQL Developer

In this procedure, you connect to your DB instance by using Oracle SQL Developer. To download a standalone version of this utility, see the [Oracle SQL Developer Downloads page](#).

To connect to your DB instance, you need its DNS name and port number. For information about finding the DNS name and port number for a DB instance, see [Finding the Endpoint of Your DB Instance \(p. 751\)](#).

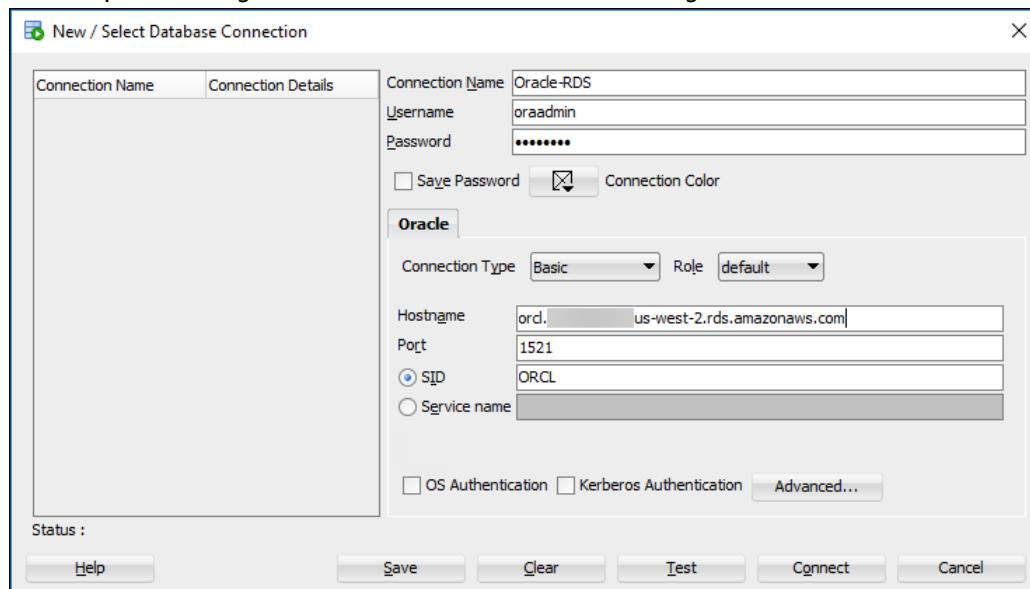
To connect to a DB instance using SQL Developer

1. Start Oracle SQL Developer.
2. On the **Connections** tab, choose the **add (+)** icon.



3. In the **New/Select Database Connection** dialog box, provide the information for your DB instance:
 - For **Connection Name**, enter a name that describes the connection, such as Oracle-RDS.
 - For **Username**, enter the name of the database administrator for the DB instance.
 - For **Password**, enter the password for the database administrator.
 - For **Hostname**, enter the DNS name of the DB instance.
 - For **Port**, enter the port number.
 - For **SID**, enter the Oracle database SID.

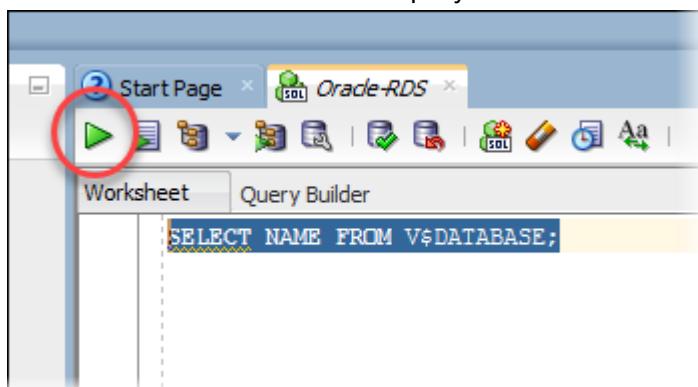
The completed dialog box should look similar to the following.



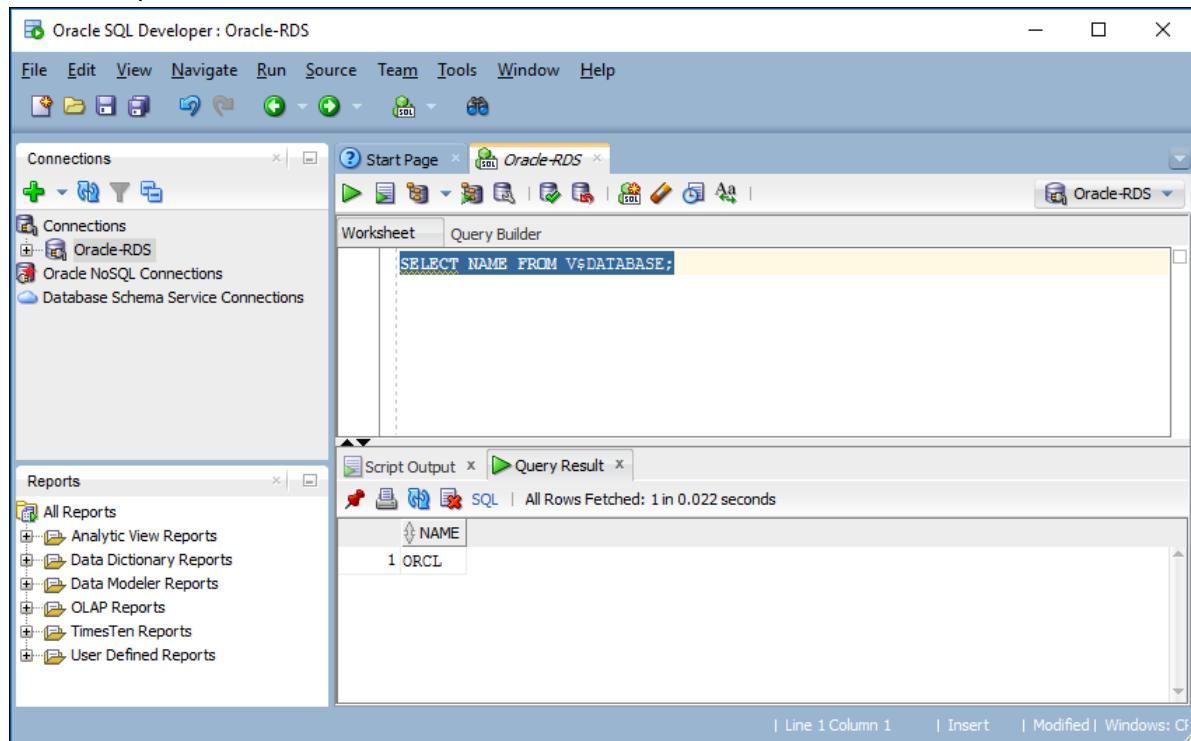
4. Click **Connect**.
5. You can now start creating your own databases and running queries against your DB instance and databases as usual. To run a test query against your DB instance, do the following:
 - a. In the **Worksheet** tab for your connection, enter the following SQL query.

```
SELECT NAME FROM V$DATABASE;
```

- b. Choose the **execute** icon to run the query.



SQL Developer returns the database name.



Connecting to Your DB Instance Using SQL*Plus

You can use a utility like SQL*Plus to connect to an Amazon RDS DB instance running Oracle. To download a standalone version of SQL*Plus, see [SQL*Plus User's Guide and Reference](#).

To connect to your DB instance, you need its DNS name and port number. For information about finding the DNS name and port number for a DB instance, see [Finding the Endpoint of Your DB Instance \(p. 751\)](#).

Example To connect to an Oracle DB instance using SQL*Plus

In the following examples, substitute the user name of your DB instance administrator. Also, substitute the DNS name for your DB instance, and then include the port number and the Oracle SID. The SID value is the name of the DB instance's database that you specified when you created the DB instance, and not the name of the DB instance.

For Linux, OS X, or Unix:

```
sqlplus 'user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))(CONNECT_DATA=(SID=database_name)))'
```

For Windows:

```
sqlplus user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))(CONNECT_DATA=(SID=database_name)))
```

You should see output similar to the following.

```
SQL*Plus: Release 12.1.0.2.0 Production on Mon Aug 21 09:42:20 2017
```

After you enter the password for the user, the SQL prompt appears.

```
SQL>
```

Note

The shorter format connection string (Easy connect or EZCONNECT), such as `sqlplus USER/PASSWORD@LONGER-THAN-63-CHARS-RDS-ENDPOINT-HERE:1521/DATABASE_IDENTIFIER`, might encounter a maximum character limit and should not be used to connect.

Security Group Considerations

For you to connect to your DB instance, it must be associated with a security group that contains the IP addresses and network configuration that you use to access the DB instance. You might have associated your DB instance with an appropriate security group when you created it. If you assigned a default, non-configured security group when you created the DB instance, the DB instance firewall prevents connections.

If you need to create a new security group to enable access, the type of security group that you create depends on which Amazon EC2 platform your DB instance is on. To determine your platform, see [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#). In general, if your DB instance is on the *EC2-Classic* platform, you create a DB security group; if your DB instance is on the *VPC* platform, you create a VPC security group. For information about creating a new security group, see [Controlling Access with Security Groups \(p. 394\)](#).

After you create the new security group, you modify your DB instance to associate it with the security group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

You can enhance security by using SSL to encrypt connections to your DB instance. For more information, see [Oracle Secure Sockets Layer \(p. 817\)](#).

Dedicated and Shared Server Processes

Server processes handle user connections to an Oracle DB instance. By default, the Oracle DB instance uses dedicated server processes. With dedicated server processes, each server process services only one user process. You can optionally configure shared server processes. With shared server processes, each server process can service multiple user processes.

You might consider using shared server processes when a high number of user sessions are using too much memory on the server. You might also consider shared server processes when sessions connect and disconnect very often, resulting in performance issues. There are also disadvantages to using shared server processes. For example, they can strain CPU resources, and they are more complicated to configure and administer.

For more information about dedicated and shared server processes, see [About Dedicated and Shared Server Processes](#) in the Oracle documentation. For more information about configuring shared server

processes on an Amazon RDS Oracle DB instance, see [How do I configure Amazon RDS for Oracle Database to work with shared servers?](#) in the Knowledge Center.

Troubleshooting the Connection to Your Oracle DB Instance

The following are issues you might encounter when you try to connect to your Oracle DB instance.

Issue	Troubleshooting Suggestions
Unable to connect to your DB instance.	For a newly created DB instance, the DB instance has a status of creating until it is ready to use. When the state changes to available , you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new DB instance is available.
Unable to connect to your DB instance.	If you can't send or receive communications over the port that you specified when you created the DB instance, you can't connect to the DB instance. Check with your network administrator to verify that the port you specified for your DB instance allows inbound and outbound communication.
Unable to connect to your DB instance.	<p>The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the security group for the DB instance might not match. The problem is most likely the egress or ingress rules on your firewall. For more information about security groups, see Controlling Access with Security Groups (p. 394).</p> <p>To walk through the process of setting up rules for your security group, see Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 427).</p>
Connect failed because target host or object does not exist – Oracle, Error: ORA-12545	<p>Make sure that you specified the server name and port number correctly. For Server name, enter the DNS name from the console.</p> <p>For information about finding the DNS name and port number for a DB instance, see Finding the Endpoint of Your DB Instance (p. 751).</p>
Invalid username/password; logon denied – Oracle, Error: ORA-01017	You were able to reach the DB instance, but the connection was refused. This is usually caused by providing an incorrect user name or password. Verify the user name and password, and then retry.

Modifying a DB Instance Running the Oracle Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. In this topic, you learn how to modify an Amazon RDS Oracle DB instance, and about the settings for Oracle instances. We recommend that you test any changes on a test instance before modifying a production instance, so that you fully understand the impact of each change. This practice is especially important when upgrading database versions.

After you modify your DB instance settings, you can apply the changes immediately, or apply them during the next maintenance window for the DB instance. Some modifications cause an interruption by restarting the DB instance.

In addition to modifying Oracle instances as described directly following, you can also change settings for sqlnet.ora parameters for an Oracle DB instance as described in [Modifying Oracle sqlnet.ora Parameters \(p. 767\)](#), at the end of this topic.

AWS Management Console

To modify an Oracle DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want. For information about each setting, see [Settings for Oracle DB Instances \(p. 759\)](#).
5. When all the changes are as you want them, choose **Continue** and check the summary of modifications.
6. To apply the changes immediately, choose **Apply immediately**. Choosing this option can cause an outage in some cases. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify an Oracle DB instance by using the AWS CLI, call the `modify-db-instance` command. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see [Settings for Oracle DB Instances \(p. 759\)](#).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days). The code enables automatic minor version upgrades by using `--auto-minor-version-upgrade`. To disable automatic minor version upgrades, use `--no-auto-minor-version-upgrade`. The changes are applied during the next maintenance window by using `--no-apply-immediately`. Use `--apply-immediately` to apply the changes immediately. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--backup-retention-period 7 \
--auto-minor-version-upgrade \
--no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--backup-retention-period 7 ^
--auto-minor-version-upgrade ^
--no-apply-immediately
```

API

To modify an Oracle DB instance by using the Amazon RDS API, call the [ModifyDBInstance](#) action. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see [Settings for Oracle DB Instances \(p. 759\)](#).

Example

The following code modifies *mydbinstance* by setting the backup retention period to 1 week (7 days) and enabling automatic minor version upgrades. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
?Action=ModifyDBInstance
&ApplyImmediately=false
&AutoMinorVersionUpgrade=true
&BackupRetentionPeriod=7
&DBInstanceIdentifier=mydbinstance
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-10-31
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
&X-Amz-Date=20131016T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for Oracle DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated storage	<p>The storage, in gigabytes, that you want to allocate for your DB instance. You can only increase the allocated storage, you can't reduce the allocated storage.</p> <p>You can't modify allocated storage if the DB instance status is <i>storage-</i></p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	No downtime. Performance may be degraded during the change.

Setting	Setting Description	When the Change Occurs	Downtime Notes
	<p>optimization or if the allocated storage for the DB instance has been modified in the last six hours.</p> <p>The maximum storage allowed depends on the storage type. For more information, see DB instance storage (p. 103).</p>		
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.	–	–
Backup retention period	<p>The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0.</p> <p>For more information, see Working With Backups (p. 208).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false and you change the setting from a non-zero value to another non-zero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.</p>	An outage occurs if you change from 0 to a non-zero value, or from a non-zero value to 0.
Backup window	<p>The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours.</p> <p>For more information, see Working With Backups (p. 208).</p>	The change is applied asynchronously, as soon as possible.	–
Certificate authority	The certificate that you want to use.	–	–
Copy tags to snapshots	<p>If you have any DB instance tags, this option copies them when you create a DB snapshot.</p> <p>For more information, see Tagging Amazon RDS Resources (p. 138).</p>	The change occurs immediately. This setting ignores the Apply immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Database port	<p>The port that you want to use to access the database.</p> <p>The port value must not match any of the port values specified for options in the option group for the DB instance.</p>	The change occurs immediately. This setting ignores the Apply immediately setting.	The DB instance is rebooted immediately.
DB engine version	<p>The version of the Oracle database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications.</p> <p>We do not recommend upgrading micro DB instances because they have limited CPU resources and the upgrade process may take hours to complete. An alternative to upgrading micro DB instances with small storage (10-20 GiB) is to copy your data using Data Pump, where we also recommend testing before migrating your production instances.</p> <p>For more information, see Upgrading the Oracle DB Engine (p. 770).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change.
DB instance class	<p>The DB instance class that you want to use.</p> <p>For more information, see DB Instance Class (p. 82) and DB Instance Class Support for Oracle (p. 720).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change.
DB instance identifier	<p>The DB instance identifier. This value is stored as a lowercase string.</p> <p>For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 126).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change. The DB instance is rebooted.

Setting	Setting Description	When the Change Occurs	Downtime Notes
DB parameter group	The parameter group that you want associated with the DB instance. For more information, see Working with DB Parameter Groups (p. 169) and Modifying Oracle sqlnet.ora Parameters (p. 767) .	The parameter group change occurs immediately.	An outage doesn't occur during this change. When you change the parameter group, changes to some parameters are applied to the DB instance immediately without a reboot. Changes to other parameters are applied only after the DB instance is rebooted. For more information, see Rebooting a DB Instance (p. 129) .
Deletion protection	Enable deletion protection to prevent your DB instance from being deleted. For more information, see Deleting a DB Instance (p. 135) .	–	–
Enhanced Monitoring	Enable enhanced monitoring to enable gathering metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 256) .	–	–
License model	license-included to use the general license agreement for Oracle. bring-your-own-license to use your existing Oracle license. For more information, see Oracle Licensing (p. 719) .	If Apply immediately is set to true, the change occurs immediately. If Apply immediately is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
Log exports	Select the types of Oracle database log files to publish to Amazon CloudWatch Logs. For more information, see Oracle Database Log Files (p. 328) .	The change occurs immediately. This setting ignores the Apply immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Maintenance Window	<p>The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours.</p> <p>If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied.</p> <p>For more information, see The Amazon RDS Maintenance Window (p. 120).</p>	The change occurs immediately. This setting ignores the Apply immediately setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.
Multi-AZ deployment	<p>Yes to deploy your DB instance in multiple Availability Zones; otherwise, No.</p> <p>For more information, see Regions and Availability Zones (p. 101).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	–
New master password	The password for your master user. The password must contain from 8 to 30 alphanumeric characters.	The change is applied asynchronously, as soon as possible. This setting ignores the Apply immediately setting.	–
Option group	<p>The option group that you want associated with the DB instance.</p> <p>For more information, see Working with Option Groups (p. 156).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	<p>When you add the APEX options to an existing DB instance, a brief outage occurs while your DB instance is automatically restarted.</p> <p>When you add the OEM option to an existing DB instance, the change can cause a brief (sub-second) period during which new connections are rejected. Existing connections are not interrupted.</p>

Setting	Setting Description	When the Change Occurs	Downtime Notes
Public accessibility	<p>Yes to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. No to make the DB instance accessible only from inside the VPC.</p> <p>For more information, see Hiding a DB Instance in a VPC from the Internet (p. 422).</p>	The change occurs immediately. This setting ignores the Apply immediately setting.	–
Security group	<p>The security group you want associated with the DB instance.</p> <p>For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 399).</p>	The change is applied asynchronously, as soon as possible. This setting ignores the Apply immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Storage type	<p>The storage type that you want to use.</p> <p>For more information, see Amazon RDS Storage Types (p. 103).</p>	<p>If Apply immediately is set to true, the change occurs immediately.</p> <p>If Apply immediately is set to false, the change occurs during the next maintenance window.</p>	<p>The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place.</p> <ul style="list-style-type: none"> • From General Purpose (SSD) to Magnetic. • From General Purpose (SSD) to Provisioned IOPS (SSD), if the DB instance is single-AZ. There is no outage for a multi-AZ DB instance. • From Magnetic to General Purpose (SSD). • From Magnetic to Provisioned IOPS (SSD). • From Provisioned IOPS (SSD) to Magnetic. • From Provisioned IOPS (SSD) to General Purpose (SSD), if the DB instance is single-AZ. There is no outage for a multi-AZ DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Subnet group	<p>The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC.</p> <p>For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 426).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change. The DB instance is rebooted.

Modifying Oracle sqlnet.ora Parameters

The sqlnet.ora file includes parameters that configure Oracle Net features on Oracle database servers and clients. Using the parameters in the sqlnet.ora file, you can modify properties for connections in and out of the database.

For more information about why you might set sqlnet.ora parameters, see [Configuring Profile Parameters](#) in the Oracle documentation.

Setting sqlnet.ora Parameters

Amazon RDS Oracle parameter groups include a subset of sqlnet.ora parameters. You set them in the same way that you set other Oracle parameters. The `sqlnetora.` prefix identifies which parameters are sqlnet.ora parameters. For example, in an Oracle parameter group in Amazon RDS, the `default_sdu_size` sqlnet.ora parameter is `sqlnetora.default_sdu_size`.

For information about managing parameter groups and setting parameter values, see [Working with DB Parameter Groups \(p. 169\)](#).

Supported sqlnet.ora Parameters

Amazon RDS supports the following sqlnet.ora parameters. Changes to dynamic sqlnet.ora parameters take effect immediately.

Parameter	Valid Values	Static/ Dynamic	Description
<code>sqlnetora.default_sdu_size</code>	Oracle 11g: 512 to 65535 Oracle 12c: – 512 to 2097152	Dynamic	The session data unit (SDU) size, in bytes. The SDU is the amount of data that is put in a buffer and sent across the network at one time.
<code>sqlnetora.diag_ora_file_enabled</code>	ON or OFF	Dynamic	A value that enables or disables Automatic Diagnostic Repository (ADR) tracing. ON specifies that ADR file tracing is used. OFF specifies that non-ADR file tracing is used.
<code>sqlnetora.recv_timeout</code>	268435456	Dynamic	The buffer space limit for receive operations of sessions, supported by the TCP/IP, TCP/IP with SSL, and SDP protocols.
<code>sqlnetora.send_timeout</code>	268435456	Dynamic	The buffer space limit for send operations of sessions, supported by the TCP/IP, TCP/IP with SSL, and SDP protocols.
<code>sqlnetora.sqlnet.expire_time</code>	Dynamic		Time interval, in minutes, to send a check to verify that client-server connections are active.
<code>sqlnetora.sqlnet.timeout_connect</code>	7200	Dynamic	Time, in seconds, for a client to connect with the database server and provide the necessary authentication information.
<code>sqlnetora.sqlnet.timeout_bound_connect</code>	7200	Dynamic	Time, in seconds, for a client to establish an Oracle Net connection to the DB instance.

Parameter	Valid Values	Static/ Dynamic	Description
sqlnet.ora.sqlnet.reco_time	Dynamic 7200	Dynamic	Time, in seconds, for a database server to wait for client data after establishing a connection.
sqlnet.ora.sqlnet.send_time	Dynamic 7200	Dynamic	Time, in seconds, for a database server to complete a send operation to clients after establishing a connection.
sqlnet.ora.tcp_connect_time	Dynamic 7200	Dynamic	Time, in seconds, for a client to establish a TCP connection to the database server.
sqlnet.ora.trace_level_server	Dynamic 16, OFF, USER, ADMIN, SUPPORT	Dynamic	For non-ADR tracing, turns server tracing on at a specified level or turns it off.

The default value for each supported sqlnet.ora parameter is the Oracle default for the release. For information about default values for Oracle 12c, see [Parameters for the sqlnet.ora File](#) in the 12c Oracle documentation. For information about default values for Oracle 11g, see [Parameters for the sqlnet.ora File](#) in the 11g Oracle documentation.

Viewing sqlnet.ora Parameters

You can view sqlnet.ora parameters and their settings using the AWS Management Console, the AWS CLI, or a SQL client.

Viewing sqlnet.ora Parameters Using the Console

For information about viewing parameters in a parameter group, see [Working with DB Parameter Groups \(p. 169\)](#).

In Oracle parameter groups, the `sqlnet.ora.` prefix identifies which parameters are sqlnet.ora parameters.

Viewing sqlnet.ora Parameters Using the AWS CLI

To view the sqlnet.ora parameters that were configured in an Oracle parameter group, use the AWS CLI `describe-db-parameters` command.

To view all of the sqlnet.ora parameters for an Oracle DB instance, call the AWS CLI `download-db-log-file-portion` command. Specify the DB instance identifier, the log file name, and the type of output.

Example

The following code lists all of the sqlnet.ora parameters for `mydbinstance`.

For Linux, OS X, or Unix:

```
aws rds download-db-log-file-portion \
--db-instance-identifier mydbinstance \
--log-file-name trace/sqlnet-parameters \
--output text
```

For Windows:

```
aws rds download-db-log-file-portion ^
--db-instance-identifier mydbinstance ^
--log-file-name trace/sqlnet-parameters ^
--output text
```

Viewing sqlnet.ora Parameters Using a SQL Client

After you connect to the Oracle DB instance in a SQL client, the following query lists the sqlnet.ora parameters.

```
SELECT * FROM TABLE
  (rdsadmin.rds_file_util.read_text_file(
    p_directory => 'BDUMP',
    p_filename  => 'sqlnet-parameters'));
```

For information about connecting to an Oracle DB instance in a SQL client, see [Connecting to a DB Instance Running the Oracle Database Engine \(p. 751\)](#).

Upgrading the Oracle DB Engine

When Amazon RDS supports a new version of Oracle, you can upgrade your DB instances to the new version. Amazon RDS supports the following upgrades to an Oracle DB instance:

- **Major Version Upgrades** – from 11g to 12c.

In general, a major engine version upgrade can introduce changes that are not compatible with existing applications. In contrast, a minor version upgrade includes only changes that are backward-compatible with existing applications.

You must modify the DB instance manually to perform a major version upgrade. Minor version upgrades occur automatically if you enable auto minor version upgrades on your DB instance. In all other cases, you must modify the DB instance manually to perform a minor version upgrade.

An outage occurs while the upgrade takes place. The time for the outage varies based on your engine version and the size of your DB instance.

For information about what Oracle versions are available on Amazon RDS, see [Oracle Database Engine Release Notes \(p. 921\)](#).

Topics

- [Overview of Upgrading \(p. 770\)](#)
- [Major Version Upgrades \(p. 771\)](#)
- [Oracle Minor Version Upgrades \(p. 771\)](#)
- [Oracle SE2 Upgrade Paths \(p. 772\)](#)
- [Option and Parameter Group Considerations \(p. 772\)](#)
- [Testing an Upgrade \(p. 773\)](#)
- [Upgrading an Oracle DB Instance \(p. 773\)](#)

Overview of Upgrading

Amazon RDS takes two DB snapshots during the upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken after the upgrade completes.

Note

Amazon RDS only takes DB snapshots if you have set the backup retention period for your DB instance to a number greater than 0. To change your backup retention period, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

After an upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the DB snapshot that was taken before the upgrade to create a new DB instance.

If your DB instance is in a Multi-AZ deployment, both the primary and standby replicas are upgraded. If no operating system updates are required, the primary and standby DB instances are upgraded at the same time, and you experience an outage until the upgrade is complete.

If your DB instance is in a Multi-AZ deployment, and operating system updates are required, the operating system updates are applied when you request the database upgrade. In this case, the operating system is updated on the standby DB instance, and the standby DB instance is upgraded. After that upgrade completes, the primary DB instance fails over to the standby DB instance, and the

operating system is updated on the new standby DB instance (the former primary DB instance), and that database is upgraded.

Major Version Upgrades

Amazon RDS supports the following major version upgrades:

- Oracle DB instances running Oracle version 12.1.0.2 to Oracle version 12.2.0.1
- Oracle DB instances running Oracle version 11.2.0.4 to Oracle version 12.2.0.1
- Oracle DB instances running Oracle version 11.2.0.4 to Oracle version 12.1.0.2.v5 and higher

To perform a major version upgrade, modify the DB instance manually. Major version upgrades don't occur automatically.

In some cases, your current Oracle DB instance might be running on a DB instance class that isn't supported for the version to which you are upgrading. In such a case, you must migrate the DB instance to a supported DB instance class before you upgrade. For more information about the supported DB instance classes for each version and edition of Amazon RDS Oracle, see [DB Instance Class \(p. 82\)](#).

Before you perform a major version upgrade, Oracle recommends that you gather optimizer statistics on the DB instance that you are upgrading. Gathering optimizer statistics can reduce DB instance downtime during the upgrade. To gather optimizer statistics, connect to the DB instance as the master user, and run the `DBMS_STATS.GATHER_DICTIONARY_STATS` procedure, as in the following example.

```
EXEC DBMS_STATS.GATHER_DICTIONARY_STATS;
```

For more information, see [Gathering Optimizer Statistics to Decrease Oracle Database Downtime](#) in the Oracle documentation.

Note

Major version upgrades aren't supported for deprecated Oracle versions, such as Oracle version 11.2.0.3 and 11.2.0.2.

Major version downgrades aren't supported.

A major version upgrade from 11g to 12c must upgrade to an Oracle Patch Set Update (PSU) that was released in the same month or later.

For example, a major version upgrade from Oracle version 11.2.0.4.v14 to Oracle version 12.1.0.2.v11 is supported. However, a major version upgrade from Oracle version 11.2.0.4.v14 to Oracle version 12.1.0.2.v9 isn't supported. This is because Oracle version 11.2.0.4.v14 was released in October 2017, and Oracle version 12.1.0.2.v9 was released in July 2017. For information about the release date for each Oracle PSU, see [Oracle Database Engine Release Notes \(p. 921\)](#).

Oracle Minor Version Upgrades

A minor version upgrade applies an Oracle PSU in a major version.

The following minor version upgrades aren't supported.

Current Version	Upgrade Not Supported
12.1.0.2.v6	12.1.0.2.v7
12.1.0.2.v5	12.1.0.2.v7
12.1.0.2.v5	12.1.0.2.v6

Note

Minor version downgrades aren't supported.

Oracle SE2 Upgrade Paths

The following table shows supported upgrade paths to Standard Edition Two (SE2). For more information about the License Included and Bring Your Own License (BYOL) models, see [Oracle Licensing \(p. 719\)](#).

Your Existing Configuration	Supported SE2 Configuration
12.2.0.1 SE2, BYOL	12.2.0.1 SE2, BYOL or License Included
12.1.0.2 SE2, BYOL	12.2.0.1 SE2, BYOL or License Included 12.1.0.2 SE2, BYOL or License Included
11.2.0.4 SE1, BYOL or License Included	12.2.0.1 SE2, BYOL or License Included
11.2.0.4 SE, BYOL	12.1.0.2 SE2, BYOL or License Included

To upgrade from your existing configuration to a supported SE2 configuration, use a supported upgrade path. For more information, see [Major Version Upgrades \(p. 771\)](#).

Option and Parameter Group Considerations

Option Group Considerations

If your DB instance uses a custom option group, in some cases Amazon RDS can't automatically assign your DB instance a new option group. For example, this occurs when you upgrade to a new major version. In those cases, you must specify a new option group when you upgrade. We recommend that you create a new option group, and add the same options to it as in your existing custom option group.

For more information, see [Creating an Option Group \(p. 157\)](#) or [Making a Copy of an Option Group \(p. 159\)](#).

If your DB instance uses a custom option group that contains the APEX option, in some cases you can reduce the time it takes to upgrade your DB instance by upgrading your version of APEX at the same time as your DB instance. For more information, see [Upgrading the APEX Version \(p. 794\)](#).

Parameter Group Considerations

If your DB instance uses a custom parameter group, in some cases Amazon RDS can't automatically assign your DB instance a new parameter group. For example, this occurs when you upgrade to a new major version. In those cases, you must specify a new parameter group when you upgrade. We recommend that you create a new parameter group, and configure the parameters as in your existing custom parameter group.

For more information, see [Creating a DB Parameter Group \(p. 170\)](#) or [Copying a DB Parameter Group \(p. 174\)](#).

Testing an Upgrade

Before you perform a major version upgrade on your DB instance, you should thoroughly test your database and all applications that access the database for compatibility with the new version. We recommend that you use the following procedure.

To test a major version upgrade

1. Review the Oracle upgrade documentation for the new version of the database engine to see if there are compatibility issues that might affect your database or applications. For more information, see [Database Upgrade Guide](#) in the Oracle documentation.
2. If your DB instance uses a custom option group, create a new option group compatible with the new version you are upgrading to. For more information, see [Option Group Considerations \(p. 772\)](#).
3. If your DB instance uses a custom parameter group, create a new parameter group compatible with the new version you are upgrading to. For more information, see [Parameter Group Considerations \(p. 772\)](#).
4. Create a DB snapshot of the DB instance to be upgraded. For more information, see [Creating a DB Snapshot \(p. 216\)](#).
5. Restore the DB snapshot to create a new test DB instance. For more information, see [Restoring from a DB Snapshot \(p. 218\)](#).
6. Modify this new test DB instance to upgrade it to the new version, by using one of the following methods:
 - [Upgrading the Engine Version of a DB Instance Using the Console \(p. 123\)](#)
 - [Upgrading the Engine Version of a DB Instance Using the AWS CLI \(p. 124\)](#)
 - [Upgrading the Engine Version of a DB Instance Using the RDS API \(p. 124\)](#)
7. Perform testing:
 - Run as many of your quality assurance tests against the upgraded DB instance as needed to ensure that your database and application work correctly with the new version.
 - Implement any new tests needed to evaluate the impact of any compatibility issues that you identified in step 1.
 - Test all stored procedures, functions, and triggers.
 - Direct test versions of your applications to the upgraded DB instance. Verify that the applications work correctly with the new version.
 - Evaluate the storage used by the upgraded instance to determine if the upgrade requires additional storage. You might need to choose a larger instance class to support the new version in production. For more information, see [DB Instance Class \(p. 82\)](#).
8. If all tests pass, then perform the upgrade on your production DB instance. We recommend that you don't allow write operations to the DB instance until you confirm that everything is working correctly.

Upgrading an Oracle DB Instance

For information about manually or automatically upgrading an Oracle DB instance, see [Upgrading a DB Instance Engine Version \(p. 123\)](#).

Upgrading an Oracle DB Snapshot

If you have existing manual DB snapshots, you might want to upgrade a snapshot to a later version of the Oracle database engine.

When Oracle stops providing patches for a version, and therefore Amazon RDS deprecates the version, you can upgrade your snapshots that correspond to the deprecated version. For more information, see [Oracle Engine Version Management \(p. 735\)](#).

The following snapshot upgrades are currently supported.

Current Snapshot Version	Supported Snapshot Upgrade
12.1.0.1	12.1.0.2.v8
11.2.0.3	11.2.0.4.v11
11.2.0.2	11.2.0.4.v12

Amazon RDS supports upgrading snapshots in all AWS Regions.

AWS Management Console

To upgrade an Oracle DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Snapshots**, and then select the DB snapshot that you want to upgrade.
3. Choose **Actions**, and then choose **Modify Snapshot**. The **Modify DB Snapshot** page appears.
4. For **DB engine version**, choose the version to upgrade the snapshot to.
5. (Optional) For **Option group**, choose the option group for the upgraded DB snapshot. The same option group considerations apply when upgrading a DB snapshot as when upgrading a DB instance. For more information, see [Option Group Considerations \(p. 772\)](#).
6. Choose **Modify Snapshot** to save your changes.

Alternatively, choose **Cancel** to cancel your changes.

CLI

To upgrade an Oracle DB snapshot by using the AWS CLI, call the `modify-db-snapshot` command with the following parameters:

- `--db-snapshot-identifier` – The name of the DB snapshot.
- `--engine-version` – The version to upgrade the snapshot to.

You might also need to include the following parameter. The same option group considerations apply when upgrading a DB snapshot as when upgrading a DB instance. For more information, see [Option Group Considerations \(p. 772\)](#).

- `--option-group-name` – The option group for the upgraded DB snapshot.

Example

The following example upgrades a DB snapshot.

For Linux, OS X, or Unix:

```
aws rds modify-db-snapshot \
    --db-snapshot-identifier <mydbsnapshot> \
    --engine-version <11.2.0.4.v12> \
    --option-group-name <default:oracle-se1-11-2>
```

For Windows:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier <mydbsnapshot> ^
    --engine-version <11.2.0.4.v12> ^
    --option-group-name <default:oracle-se1-11-2>
```

API

To upgrade an Oracle DB snapshot by using the Amazon RDS API, call the [ModifyDBSnapshot](#) action with the following parameters:

- `DBSnapshotIdentifier` – The name of the DB snapshot.
- `EngineVersion` – The version to upgrade the snapshot to.

You might also need to include the following parameter. The same option group considerations apply when upgrading a DB snapshot as when upgrading a DB instance. For more information, see [Option Group Considerations \(p. 772\)](#).

- `OptionGroupName` – The option group for the upgraded DB snapshot.

Example

The following example upgrades a DB snapshot.

```
https://rds.amazonaws.com/
    ?Action=ModifyDBSnapshot
    &DBSnapshotIdentifier=mydbsnapshot
    &EngineVersion=11.2.0.4.v12
    &OptionGroupName=default:oracle-se1-11-2
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757fffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- [Oracle Database Engine Release Notes \(p. 921\)](#)
- [Upgrading the Oracle DB Engine \(p. 770\)](#)
- [Applying Updates for a DB Instance \(p. 118\)](#)

Importing Data into Oracle on Amazon RDS

How you import data into an Amazon RDS DB instance depends on the amount of data you have and the number and variety of database objects in your database. For example, you can use Oracle SQL Developer to import a simple, 20 MB database. You can use Oracle Data Pump to import complex databases, or databases that are several hundred megabytes or several terabytes in size.

You can also use AWS Database Migration Service (AWS DMS) to import data into an Amazon RDS DB instance. AWS DMS can migrate databases without downtime and, for many database engines, continue ongoing replication until you are ready to switch over to the target database. You can migrate to Oracle from either the same database engine or a different database engine using AWS DMS. If you are migrating from a different database engine, you can use the AWS Schema Conversion Tool to migrate schema objects that are not migrated by AWS DMS. For more information about AWS DMS, see [What is AWS Database Migration Service](#).

Before you use any of these migration techniques, we recommend the best practice of taking a backup of your database. After you import the data, you can back up your Amazon RDS DB instances by creating snapshots. Later, you can restore the database from the snapshots. For more information, see [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#).

Oracle SQL Developer

For small databases, you can use Oracle SQL Developer, a graphical Java tool distributed without cost by Oracle. You can install this tool on your desktop computer (Windows, Linux, or Mac) or on one of your servers. Oracle SQL Developer provides options for migrating data between two Oracle databases, or for migrating data from other databases, such as MySQL, to Oracle. Oracle SQL Developer is best suited for migrating small databases. We recommend that you read the Oracle SQL Developer product documentation before you begin migrating your data.

After you install SQL Developer, you can use it to connect to your source and target databases. Use the **Database Copy** command on the Tools menu to copy your data to your Amazon RDS instance.

To download Oracle SQL Developer, go to <http://www.oracle.com/technetwork/developer-tools/sql-developer>.

Oracle also has documentation on how to migrate from other databases, including MySQL and SQL Server. For more information, see <http://www.oracle.com/technetwork/database/migration> in the Oracle documentation.

Oracle Data Pump

Oracle Data Pump is a long-term replacement for the Oracle Export/Import utilities and is the preferred way to move large amounts of data from an Oracle installation to an Amazon RDS DB instance. You can use Oracle Data Pump for several scenarios:

- Import data from an Oracle database (either on-premises or Amazon EC2 instance) to an Amazon RDS Oracle DB instance
- Import data from an Amazon RDS Oracle DB instance to an Oracle database (either on-premises or Amazon EC2 instance)
- Import data between Amazon RDS Oracle DB instances (for example, to migrate data from EC2-Classic to VPC)

To download Oracle Data Pump utilities, go to <http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html>.

For compatibility considerations when migrating between versions of Oracle Database, see [the Oracle documentation](#).

The following process uses Oracle Data Pump and the `DBMS_FILE_TRANSFER` package. The process connects to a source Oracle instance (which can be an on-premises or Amazon EC2 instance, or an Amazon RDS Oracle DB instance) and exports data using the `DBMS_DATAPUMP` package. It then uses the `DBMS_FILE_TRANSFER.PUT_FILE` method to copy the dump file from the Oracle instance to the `DATA_PUMP_DIR` directory on the target Amazon RDS Oracle DB instance that is connected using a database link. The final step imports the data from the copied dump file into the Amazon RDS Oracle DB instance using the `DBMS_DATAPUMP` package.

The process has the following requirements:

- You must have execute privileges on the `DBMS_FILE_TRANSFER` and `DBMS_DATAPUMP` packages.
- You must have write privileges to the `DATA_PUMP_DIR` directory on the source DB instance.
- You must ensure that you have enough storage space to store the dump file on the source instance and the target DB instance.

Note

This process imports a dump file into the `DATA_PUMP_DIR` directory, a preconfigured directory on all Oracle DB instances. This directory is located on the same storage volume as your data files. When you import the dump file, the existing Oracle data files will use more space, so you should make sure that your DB instance can accommodate that additional use of space as well. The imported dump file is not automatically deleted or purged from the `DATA_PUMP_DIR` directory. Use `UTL_FILE.FREMOVE` to remove the imported dump file.

The import process using Oracle Data Pump and the `DBMS_FILE_TRANSFER` package has the following steps:

- Step 1: Grant privileges to user on the Amazon RDS target instance
- Step 2: Grant privileges to user on source database
- Step 3: Use `DBMS_DATAPUMP` to create a dump file
- Step 4: Create a database link to the target DB instance
- Step 5: Use `DBMS_FILE_TRANSFER` to copy the exported dump file to the target DB instance
- Step 6: Use `DBMS_DATAPUMP` to import the data file on the target DB instance
- Step 7: Clean up

Step 1: Grant privileges to user on the Amazon RDS target instance

1. Use SQL Plus or Oracle SQL Developer to connect to the Amazon RDS target Oracle DB instance into which the data will be imported. Connect as the Amazon RDS master user. For information about connecting to the DB instance, see [Connecting to a DB Instance Running the Oracle Database Engine \(p. 751\)](#).
2. Create the required tablespaces before you import the data. For more information, see [Creating and Sizing Tablespaces \(p. 855\)](#).
3. If the user account into which the data will be imported does not exist, create the user account and grant the necessary permissions and roles. If you will import data into multiple user schemas, create each user account and grant the necessary privileges and roles to it.

For example, the following commands create a new user and grant the necessary permissions and roles to import the data into the user's schema:

```
create user schema_1 identified by <password>;
grant create session, resource to schema_1;
alter user schema_1 quota 100M on users;
```

This example grants the new user the CREATE SESSION privilege and the RESOURCE role. Additional privileges and roles might be required depending on the database objects you will import.

Note

Replace *schema_1* with the name of your schema in this step and in the following steps.

Step 2: Grant privileges to user on source database

Use SQL Plus or Oracle SQL Developer to connect to the Oracle instance that contains the data to be imported. If necessary, create a user account and grant the necessary permissions.

Note

If the source database is an Amazon RDS instance, you can skip this step. You will use your Amazon RDS master user account to perform the export.

The following commands create a new user and grant the necessary permissions:

```
create user export_user identified by <password>;
grant create session, create table, create database link to export_user;
alter user export_user quota 100M on users;
grant read, write on directory data_pump_dir to export_user;
grant select_catalog_role to export_user;
grant execute on dbms_datapump to export_user;
grant execute on dbms_file_transfer to export_user;
```

Step 3: Use DBMS_DATAPUMP to create a dump file

Use SQL Plus or Oracle SQL Developer to connect to the source Oracle instance with an administrative user or with the user you created in Step 2. If the source database is an Amazon RDS Oracle DB instance, connect with the Amazon RDS master user. Next, use the Oracle Data Pump utility to create a dump file.

The following script creates a dump file named *sample.dmp* in the DATA_PUMP_DIR directory.

```
DECLARE
hdnl NUMBER;
BEGIN
hdnl := DBMS_DATAPUMP.OPEN( operation => 'EXPORT', job_mode => 'SCHEMA', job_name=>null);
DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'sample.dmp', directory =>
'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_dump_file);
DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'exp.log', directory =>
'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_log_file);
DBMS_DATAPUMP.METADATA_FILTER(hdnl, 'SCHEMA_EXPR', 'IN (''SCHEMA_1'')');
DBMS_DATAPUMP.START_JOB(hdnl);
END;
/
```

Note

Data Pump jobs are started asynchronously. For information about monitoring a Data Pump job, see [Monitoring Job Status](#) in the Oracle documentation.

Step 4: Create a database link to the target DB instance

Create a database link between your source instance and your target DB instance. Note that your local Oracle instance must have network connectivity to the DB instance in order to create a database link and to transfer your export dump file.

Perform this step connected with the same user account as the previous step.

If you are creating a database link between two DB instances inside the same VPC or peered VPCs, the two DB instances should have a valid route between them. The security group of each DB instance must allow ingress to and egress from the other DB instance. The security group inbound and outbound rules can refer to security groups from the same VPC or a peered VPC. For more information, see [Adjusting Database Links for Use with DB Instances in a VPC \(p. 859\)](#).

The following command creates a database link named *to_rds* that connects to the Amazon RDS master user at the target DB instance:

```
create database link to_rds connect to <master_user_account> identified by <password>
using '(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<dns or ip address of remote db>)
(PORT=<listener port>))(CONNECT_DATA=(SID=<remote SID>)))';
```

Step 5: Use DBMS_FILE_TRANSFER to copy the exported dump file to the target DB instance

Use DBMS_FILE_TRANSFER to copy the dump file from the source database instance to the target DB instance. The following script copies a dump file named sample.dmp from the source instance to a target database link named *to_rds* (created in the previous step):

```
BEGIN
DBMS_FILE_TRANSFER.PUT_FILE(
source_directory_object      => 'DATA_PUMP_DIR',
source_file_name              => 'sample.dmp',
destination_directory_object => 'DATA_PUMP_DIR',
destination_file_name        => 'sample_copied.dmp',
destination_database          => 'to_rds'
);
END;
/
```

Step 6: Use DBMS_DATAPUMP to import the data file on the target DB instance

Use Oracle Data Pump to import the schema in the DB instance. Note that additional options such as METADATA_REMAP might be required.

Connect to the DB instance with the Amazon RDS master user account to perform the import.

```
DECLARE
hdnl NUMBER;
BEGIN
hdnl := DBMS_DATAPUMP.OPEN( operation => 'IMPORT', job_mode => 'SCHEMA', job_name=>null);
DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'sample_copied.dmp', directory =>
'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_dump_file);
DBMS_DATAPUMP.METADATA_FILTER(hdnl,'SCHEMA_EXPR','IN (''SCHEMA_1''));
DBMS_DATAPUMP.START_JOB(hdnl);
END;
```

/

You can verify the data import by viewing the user's tables on the DB instance. For example, the following query returns the number of tables for `schema_1`:

```
select count(*) from dba_tables where owner='SCHEMA_1';
```

Step 7: Clean up

After the data has been imported, you can delete the files you no longer want to keep. You can list the files in the DATA_PUMP_DIR using the following command:

```
select * from table(RDSADMIN.RDS_FILE_UTIL.LISTDIR('DATA_PUMP_DIR')) order by mtime;
```

The following command can be used to delete files in the DATA_PUMP_DIR that you no longer require:

```
exec utl_file.fremove('DATA_PUMP_DIR', '<file name>');
```

For example, the following command deletes the file named "sample_copied.dmp":

```
exec utl_file.fremove('DATA_PUMP_DIR', 'sample_copied.dmp');
```

Oracle Export/Import Utilities

The Oracle Export/Import utilities are best suited for migrations where the data size is small and data types such as binary float and double are not required. The import process creates the schema objects so you do not need to run a script to create them beforehand, making this process well suited for databases with small tables. The following example demonstrates how these utilities can be used to export and import specific tables.

To download Oracle export and import utilities, go to <http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html>.

Export the tables from the source database using the command below. Substitute username/password as appropriate.

```
exp cust_dba@ORCL FILE=exp_file.dmp TABLES=(tab1,tab2,tab3) LOG=exp_file.log
```

The export process creates a binary dump file that contains both the schema and data for the specified tables. Now this schema and data can be imported into a target database using the command:

```
imp cust_dba@targetdb FROMUSER=cust_schema TOUSER=cust_schema \
TABLES=(tab1,tab2,tab3) FILE=exp_file.dmp LOG=imp_file.log
```

There are other variations of the Export and Import commands that might be better suited to your needs. See Oracle's documentation for full details.

Oracle SQL*Loader

Oracle SQL*Loader is well suited for large databases that have a limited number of objects in them. Since the process involved in exporting from a source database and loading to a target database is very specific to the schema, the following example creates the sample schema objects, exports from a source, and then loads it into a target database.

To download Oracle SQL*Loader, go to <http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html>.

1. Create a sample source table using the command below.

```
create table customer_0 tablespace users as select rownum id, o.* from all_objects o, all_objects x where rownum <= 1000000;
```

2. On the target Amazon RDS instance, create a destination table that is used to load the data.

```
create table customer_1 tablespace users as select 0 as id, owner, object_name, created from all_objects where 1=2;
```

3. The data is exported from the source database to a flat file with delimiters. This example uses SQL*Plus for this purpose. For your data, you will likely need to generate a script that does the export for all the objects in the database.

```
alter session set nls_date_format = 'YYYY/MM/DD HH24:MI:SS'; set linesize 800
HEADING OFF FEEDBACK OFF array 5000 pagesize 0 spool customer_0.out SET
MARKUP HTML PREFORMAT ON SET COLSEP ',' SELECT id, owner, object_name,
created FROM customer_0; spool off
```

4. You need to create a control file to describe the data. Again, depending on your data, you will need to build a script that does this step.

```
cat << EOF > sqldr_1.ctl
load data
infile customer_0.out
into table customer_1
APPEND
fields terminated by "," optionally enclosed by ''
(
id          POSITION(01:10)      INTEGER EXTERNAL,
owner       POSITION(12:41)      CHAR,
object_name POSITION(43:72)      CHAR,
created     POSITION(74:92)      date "YYYY/MM/DD HH24:MI:SS"
)
```

If needed, copy the files generated by the preceding code to a staging area, such as an Amazon EC2 instance.

5. Finally, import the data using SQL*Loader with the appropriate username and password for the target database.

```
sqlldr cust_dba@targetdb control=sqldr_1.ctl BINDSIZE=10485760 READSIZE=10485760
ROWS=1000
```

Oracle Materialized Views

You can also make use of Oracle materialized view replication to migrate large datasets efficiently. Replication allows you to keep the target tables in sync with the source on an ongoing basis, so the actual cutover to Amazon RDS can be done later, if needed. The replication is set up using a database link from the Amazon RDS instance to the source database.

One requirement for materialized views is to allow access from the target database to the source database. In the following example, access rules were enabled on the source database to allow the Amazon RDS target database to connect to the source over SQLNet.

1. Create a user account on both source and Amazon RDS target instances that can authenticate with the same password.

```
create user dblink_user identified by <password>
default tablespace users
temporary tablespace temp; grant create session to dblink_user; grant select
any table to dblink_user; grant select any dictionary to dblink_user;
```

2. Create a database link from the Amazon RDS target instance to the source instance using the newly created dblink_user.

```
create database link remote_site
connect to dblink_user identified by <password>
using '(description=(address=(protocol=tcp) (host=<myhost>) (port=<listener port>))
(connect_data=(sid=<sourcedb sid>)))';
```

3. Test the link:

```
select * from v$instance@remote_site;
```

4. Create a sample table with primary key and materialized view log on the source instance.

```
create table customer_0 tablespace users as select rownum id, o.* from
all_objects o, all_objects x where rownum <= 1000000; alter table customer_0
add constraint pk_customer_0 primary key (id) using index; create
materialized view log on customer_0;
```

5. On the target Amazon RDS instance, create a materialized view.

```
CREATE MATERIALIZED VIEW customer_0      BUILD IMMEDIATE      REFRESH FAST      AS
SELECT * FROM cust_dba.customer_0@remote_site;
```

Oracle Character Sets Supported in Amazon RDS

The following table lists the Oracle database character sets that are supported in Amazon RDS. You can use a value from this table with the `--character-set-name` parameter of the AWS CLI [create-db-instance](#) command or with the `CharacterSetName` parameter of the Amazon RDS API [CreateDBInstance](#) action.

Setting the `NLS_LANG` environment parameter in your client's environment is the simplest way to specify locale behavior for Oracle. This parameter sets the language and territory used by the client application and the database server. It also indicates the client's character set, which corresponds to the character set for data entered or displayed by a client application. Amazon RDS lets you set the character set when you create a DB instance. For more information on the `NLS_LANG` and character sets, see [What is a Character set or Code Page? in the Oracle documentation](#).

Value	Description
AL32UTF8	Unicode 5.0 UTF-8 Universal character set (default)
AR8ISO8859P6	ISO 8859-6 Latin/Arabic
AR8MSWIN1256	Microsoft Windows Code Page 1256 8-bit Latin/Arabic
BLT8ISO8859P13	ISO 8859-13 Baltic
BLT8MSWIN1257	Microsoft Windows Code Page 1257 8-bit Baltic
CL8ISO8859P5	ISO 8859-5 Latin/Cyrillic
CL8MSWIN1251	Microsoft Windows Code Page 1251 8-bit Latin/Cyrillic
EE8ISO8859P2	ISO 8859-2 East European
EL8ISO8859P7	ISO 8859-7 Latin/Greek
EE8MSWIN1250	Microsoft Windows Code Page 1250 8-bit East European
EL8MSWIN1253	Microsoft Windows Code Page 1253 8-bit Latin/Greek
IW8ISO8859P8	ISO 8859-8 Latin/Hebrew
IW8MSWIN1255	Microsoft Windows Code Page 1255 8-bit Latin/Hebrew
JA16EUC	EUC 24-bit Japanese
JA16EUCTILDE	Same as JA16EUC except for mapping of wave dash and tilde to and from Unicode
JA16SJIS	Shift-JIS 16-bit Japanese
JA16SJISTILDE	Same as JA16SJIS except for mapping of wave dash and tilde to and from Unicode
KO16MSWIN949	Microsoft Windows Code Page 949 Korean

Value	Description
NE8ISO8859P10	ISO 8859-10 North European
NEE8ISO8859P4	ISO 8859-4 North and Northeast European
TH8TISASCII	Thai Industrial Standard 620-2533-ASCII 8-bit
TR8MSWIN1254	Microsoft Windows Code Page 1254 8-bit Turkish
US7ASCII	ASCII 7-bit American
UTF8	Unicode 3.0 UTF-8 Universal character set, CESU-8 compliant
VN8MSWIN1258	Microsoft Windows Code Page 1258 8-bit Vietnamese
WE8ISO8859P1	Western European 8-bit ISO 8859 Part 1
WE8ISO8859P15	ISO 8859-15 West European
WE8ISO8859P9	ISO 8859-9 West European and Turkish
WE8MSWIN1252	Microsoft Windows Code Page 1252 8-bit West European
ZHS16GBK	GBK 16-bit Simplified Chinese
ZHT16HKSCS	Microsoft Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001. Character set conversion is based on Unicode 3.0.
ZHT16MSWIN950	Microsoft Windows Code Page 950 Traditional Chinese
ZHT32EUC	EUC 32-bit Traditional Chinese

You can also set the following National Language Support (NLS) initialization parameters at the instance level for an Oracle DB instance in Amazon RDS:

- NLS_DATE_FORMAT
- NLS_LENGTH_SEMANTICS
- NLS_NCHAR_CONV_EXCP
- NLS_TIME_FORMAT
- NLS_TIME_TZ_FORMAT
- NLS_TIMESTAMP_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT

For information about modifying instance parameters, see [Working with DB Parameter Groups \(p. 169\)](#).

You can set other NLS initialization parameters in your SQL client. For example, the following statement sets the NLS_LANGUAGE initialization parameter to GERMAN in a SQL client that is connected to an Oracle DB instance:

```
ALTER SESSION SET NLS_LANGUAGE=GERMAN;
```

For information about connecting to an Oracle DB instance with a SQL client, see [Connecting to a DB Instance Running the Oracle Database Engine \(p. 751\)](#).

Options for Oracle DB Instances

Following, you can find a description of options, or additional features, that are available for Amazon RDS instances running the Oracle DB engine. To enable these options, you add them to an option group, and then associate the option group with your DB instance. For more information, see [Working with Option Groups \(p. 156\)](#).

Some options require additional memory to run on your DB instance. For example, Oracle Enterprise Manager Database Control uses about 300 MB of RAM. If you enable this option for a small DB instance, you might encounter performance problems due to memory constraints. You can adjust the Oracle parameters so that the database requires less RAM. Alternatively, you can scale up to a larger DB instance.

Amazon RDS supports the following options for Oracle DB instances.

Option	Option ID
Oracle Application Express (p. 788)	APEX
	APEX-DEV
Oracle Enterprise Manager (p. 795)	OEM
	OEM_AGENT
Oracle Java Virtual Machine (p. 804)	JVM
Oracle Label Security (p. 807)	OLS
Oracle Locator (p. 810)	LOCATOR
Oracle Multimedia (p. 813)	MULTIMEDIA
Oracle Native Network Encryption (p. 815)	NATIVE_NETWORK_ENCRYPTION
Oracle Secure Sockets Layer (p. 817)	SSL
Oracle Spatial (p. 823)	SPATIAL
Oracle SQLT (p. 825)	SQLT
Oracle Statspack (p. 830)	STATSPACK
Oracle Time Zone (p. 833)	Timezone
Oracle Transparent Data Encryption (p. 836)	TDE
Oracle UTL_MAIL (p. 838)	UTL_MAIL
Oracle XML DB (p. 840)	XMLDB

Oracle Application Express

Amazon RDS supports Oracle Application Express (APEX) through the use of the `APEX` and `APEX-DEV` options. Oracle APEX can be deployed as a run-time environment or as a full development environment for web-based applications. Using Oracle APEX, developers can build applications entirely within the web browser. For more information, see [Oracle Application Express](#) in the Oracle documentation.

Oracle APEX consists of two main components:

- A *repository* that stores the metadata for APEX applications and components. The repository consists of tables, indexes, and other objects that are installed in your Amazon RDS DB instance.
- A *listener* that manages HTTP communications with Oracle APEX clients. The listener accepts incoming connections from web browsers, forwards them to the Amazon RDS DB instance for processing, and then sends results from the repository back to the browsers. The APEX Listener was renamed Oracle Rest Data Services (ORDS) in Oracle 12c.

When you add the Amazon RDS APEX options to your DB instance, Amazon RDS installs the Oracle APEX repository only. You must install the Oracle APEX Listener on a separate host, such as an Amazon EC2 instance, an on-premises server at your company, or your desktop computer.

The APEX option uses storage on the DB instance class for your DB instance.

Following are the supported versions and approximate storage requirements for Oracle APEX for Oracle 12c version 12.2 on Amazon RDS:

- Oracle APEX version 5.1.4.v1 – 220 MiB

Following are the supported versions and approximate storage requirements for Oracle APEX for Oracle 12c version 12.1 on Amazon RDS:

- Oracle APEX version 5.1.4.v1 – 220 MiB
- Oracle APEX version 5.1.2.v1 – 150 MiB
- Oracle APEX version 5.0.4.v1 – 140 MiB
- Oracle APEX version 4.2.6.v1 – 160 MiB

Following are the supported versions and approximate storage requirements for Oracle APEX for Oracle 11g on Amazon RDS:

- Oracle APEX version 5.1.4.v1 – 220 MiB
- Oracle APEX version 5.1.2.v1 – 150 MiB
- Oracle APEX version 5.0.4.v1 – 140 MiB
- Oracle APEX version 4.2.6.v1 – 160 MiB
- Oracle APEX version 4.1.1.v1 – 130 MiB

Note

Oracle APEX 5 for Oracle 11g isn't supported when the DB instance class used by the DB instance has only one vCPU. For information about DB instance classes, see [DB Instance Class \(p. 82\)](#).

Prerequisites for Oracle APEX and APEX Listener

The following are prerequisites for using Oracle APEX and APEX Listener:

- You must have SQL*Plus to perform administrative tasks on your DB instance.
- You must have the following software installed on the host computer that acts as the Oracle APEX Listener:
 - The Java Runtime Environment (JRE).
 - Oracle Net Services, to enable the Oracle APEX Listener to connect to your Amazon RDS instance.

Adding the Amazon RDS APEX Options

The general process for adding the Amazon RDS APEX options to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the options to the option group.
3. Associate the option group with the DB instance.

When you add the Amazon RDS APEX options, a brief outage occurs while your DB instance is automatically restarted.

To add the APEX options to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - a. For **Engine**, choose the Oracle edition that you want to use. The APEX options are supported on all editions.
 - b. For **Major engine version**, choose **11.2** or **12.1**.

For more information, see [Creating an Option Group \(p. 157\)](#).

2. Add the options to the option group. If you want to deploy only the Oracle APEX run-time environment, add only the **APEX** option. If you want to deploy the full development environment, add both the **APEX** and **APEX-DEV** options.
 - For Oracle 12c, add the **APEX** and **APEX-DEV** options.
 - For Oracle 11g, first add the **XMLDB** option as a prerequisite, then add the **APEX** and **APEX-DEV** options.

For **Version**, choose the version of **APEX** that you want to use. If you don't choose a version, version 4.1.1.v1 is the default for 11g, and version 4.2.6.v1 is the default for 12c.

Important

If you add the APEX options to an existing option group that is already attached to one or more DB instances, a brief outage occurs while all the DB instances are automatically restarted.

For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).

3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. When you add the APEX options to an existing DB instance, a brief outage occurs while your DB instance is automatically restarted. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Unlocking the Public User Account

After the Amazon RDS APEX options are installed, you must change the password for the APEX public user account, and then unlock the account. You can do this by using the Oracle SQL*Plus command line utility. Connect to your DB instance as the master user, and issue the following commands. Replace `new_password` with a password of your choice.

```
alter user APEX_PUBLIC_USER identified by new_password;
alter user APEX_PUBLIC_USER account unlock;
```

Configuring RESTful Services for Oracle APEX

To configure RESTful services in APEX (not needed for APEX 4.1.1.V1), use SQL*Plus to connect to your DB instance as the master user, and then run the `rdsadmin.rdsadmin_run_apex_rest_config` stored procedure. When you run the stored procedure, you provide passwords for the following users:

- `APEX_LISTENER`
- `APEX_REST_PUBLIC_USER`

The stored procedure runs the `apex_rest_config.sql` script, which creates new database accounts for these users.

Note

Configuration isn't required for Oracle APEX version 4.1.1.v1. For this Oracle APEX version only, you don't need to run the stored procedure.

The following command runs the stored procedure.

```
exec rdsadmin.rdsadmin_run_apex_rest_config('apex_listener_password',
'apex_rest_public_user_password');
```

Installing and Configuring the APEX Listener

You are now ready to install and configure a listener for use with Oracle APEX. You can use one of these products for this purpose:

- For APEX version 5.0 and later, use Oracle Rest Data Services (ORDS)
- For APEX version 4.1.1, use Oracle APEX Listener version 1.1.4
- Oracle HTTP Server and `mod_plsql`

Note

Amazon RDS doesn't support the Oracle XML DB HTTP server with the embedded PL/SQL gateway; you can't use this as an APEX Listener. In general, Oracle recommends against using the embedded PL/SQL gateway for applications that run on the internet.

You must install the APEX Listener on a separate host such as an Amazon EC2 instance, an on-premises server at your company, or your desktop computer.

The following procedure shows you how to install and configure the APEX Listener. We assume that the name of your host is `myapexhost.example.com`, and that your host is running Linux.

To install and configure the APEX Listener

1. Log in to `myapexhost.example.com` as root.

2. Create a nonprivileged OS user to own the APEX Listener installation. The following command creates a new user named *apexuser*.

```
useradd -d /home/apexuser apexuser
```

The following command assigns a password to the new user.

```
passwd apexuser;
```

3. Log in to `myapexhost.example.com` as `apexuser`, and download the APEX installation file from Oracle to your `/home/apexuser` directory:
 - <http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html>
 - [Oracle Application Express Prior Release Archives](#)
4. Unzip the file in the `/home/apexuser` directory.

```
unzip apex_<version>.zip
```

After you unzip the file, there is an `apex` directory in the `/home/apexuser` directory.

5. While you are still logged into `myapexhost.example.com` as `apexuser`, download the APEX Listener file from Oracle to your `/home/apexuser` directory:
 - <http://www.oracle.com/technetwork/developer-tools/apex-listener/downloads/index.html>
6. Create a new directory and open the APEX Listener file:

Listener Type	Instructions
ORDS	Run the following code: <pre>mkdir /home/apexuser/ORDS cd /home/apexuser/ORDS unzip/ords.<version>.zip</pre>
APEX Listener	Run the following code: <pre>mkdir /home/apexuser/apexlistener cd /home/apexuser/apexlistener unzip/apex_listener.<version>.zip</pre>

7. While you are still in the directory from the previous step, run the listener program.

Listener Type	Instructions
ORDS	Run the following code: <pre>java -jar ords.war setup</pre> <p>The program prompts you for the following information. The default values are in brackets.</p> <ul style="list-style-type: none"> • The location to store configuration data

Listener Type	Instructions
	<p>Type /home/apexuser/ORDS.</p> <ul style="list-style-type: none"> • The name of the database server [localhost] <p>Choose the default or type the correct value.</p> <ul style="list-style-type: none"> • The database listen port [1521] <p>Choose the default or type the correct value.</p> <ul style="list-style-type: none"> • Database service name or database SID [1] <p>Choose 1 to specify the database service name, or choose 2 to specify the database SID.</p> <ul style="list-style-type: none"> • Database SID [xe] <p>Choose the default or type the correct value.</p> <ul style="list-style-type: none"> • Verify/install Oracle REST Data Services schema or skip this step [1] <p>Choose 2.</p> <ul style="list-style-type: none"> • Use PL/SQL Gateway or skip this step [1] <p>Choose the default.</p> <ul style="list-style-type: none"> • PL/SQL Gateway database user name [APEX_PUBLIC_USER] <p>Choose the default.</p> <ul style="list-style-type: none"> • Database password for APEX_PUBLIC_USER <p>Type the password.</p> <ul style="list-style-type: none"> • Specify passwords for Application Express RESTful Services database users (APEX_LISTENER, APEX_REST_PUBLIC_USER) or skip this step [1] <p>Choose 2 for APEX 4.1.1.V1; choose 1 for all other APEX versions.</p> <ul style="list-style-type: none"> • [Not needed for APEX 4.1.1.v1] Database password for APEX_LISTENER <p>Type the password (if required).</p> <ul style="list-style-type: none"> • [Not needed for APEX 4.1.1.v1] Database password for APEX_REST_PUBLIC_USER <p>Type the password (if required).</p>

Listener Type	Instructions
APEX Listener	<p>Run the following code:</p> <pre>java -Dapex.home=./apex -Dapex.images=/home/apexuser/apex/images -Dapex.erase -jar ./apex.war</pre> <p>The program prompts you for the following:</p> <ul style="list-style-type: none"> • The APEX Listener Administrator user name. The default is <i>adminlistener</i>. • A password for the APEX Listener Administrator. • The APEX Listener Manager user name. The default is <i>managerlistener</i>. • A password for the APEX Listener Administrator. <p>The program prints a URL that you need in order to complete the configuration, as follows:</p> <pre>INFO: Please complete configuration at: http://localhost:8080/apex/listenerConfigure Database is not yet configured</pre> <p>Leave the APEX Listener running. It needs to continue running for you to use Oracle Application Express. When you have finished this configuration procedure, you can run the listener in the background.</p> <p>From your web browser, go to the URL provided by the APEX Listener program. The Oracle Application Express Listener administration window appears. Type the following information:</p> <ul style="list-style-type: none"> • Username – <i>APEX_PUBLIC_USER</i> • Password – the password for <i>APEX_PUBLIC_USER</i>. This password is the one that you specified earlier when you configured the APEX repository. For more information, see Unlocking the Public User Account (p. 790). • Connection Type – Basic • Hostname – the endpoint of your Amazon RDS DB instance, such as <i>mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com</i>. • Port – 1521 • SID – the name of the database on your Amazon RDS DB instance, such as <i>mydb</i>. <p>Choose Apply. The APEX administration window appears.</p>

8. You must set a password for the APEX admin user. To do this, use SQL*Plus to connect to your DB instance as the master user, and then issue the following commands:

```
EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
grant APEX_ADMINISTRATOR_ROLE to master;
@/home/apexuser/apex/apxchpwd.sql
```

Replace *master* with your master user name. When the *apxchpwd.sql* script prompts you, type a new admin password.

9. For ORDS, start the APEX Listener. Run the following code:

```
java -jar ords.war
```

The first time you start the APEX Listener, you are prompted to provide the location of the APEX Static resources. This images folder is located in the /apex/images directory in the installation directory for APEX.

10. Return to the APEX administration window in your browser and choose **Administration**. Next, choose **Application Express Internal Administration**. When you are prompted for credentials, type the following information:

- **User name** – admin
- **Password** – the password you set using the apxchpwd.sql script

Choose **Login**, and then set a new password for the admin user.

The APEX Listener is now ready for use.

Upgrading the APEX Version

Important

Back up your DB instance before you upgrade APEX. For more information, see [Creating a DB Snapshot \(p. 216\)](#) and [Testing an Upgrade \(p. 773\)](#).

To upgrade APEX with your DB instance, do the following:

- Create a new option group for the upgraded version of your DB instance.
- Add the upgraded versions of APEX and APEX-DEV to the new option group. Be sure to include any other options that your DB instance uses. For more information, see [Option Group Considerations \(p. 772\)](#).
- When you upgrade your DB instance, specify the new option group for your upgraded DB instance.

After you upgrade your version of APEX, the APEX schema for the previous version might still exist in your database. If you don't need it anymore, you can drop the old APEX schema from your database after you upgrade.

If you upgrade the APEX version and RESTful services were not configured in the previous APEX version, we recommend that you configure RESTful services. For more information, see [Configuring RESTful Services for Oracle APEX \(p. 790\)](#).

If you are planning to do a major version upgrade of your DB instance, and you are using an APEX version that is not compatible with your target database version, you can upgrade your version of APEX before you upgrade your DB instance. Upgrading APEX first can reduce the amount of time it takes to upgrade your DB instance.

Note

After upgrading APEX, install and configure a listener for use with the upgraded version. For instructions, see [Installing and Configuring the APEX Listener \(p. 790\)](#).

Removing the APEX Option

You can remove the Amazon RDS APEX options from a DB instance. To remove the APEX options from a DB instance, do one of the following:

- To remove the APEX options from multiple DB instances, remove the APEX options from the option group they belong to. This change affects all DB instances that use the option group. When you

remove the APEX options from an option group that is attached to multiple DB instances, a brief outage occurs while all the DB instances are restarted.

For more information, see [Removing an Option from an Option Group \(p. 167\)](#).

- To remove the APEX options from a single DB instance, modify the DB instance and specify a different option group that doesn't include the APEX options. You can specify the default (empty) option group, or a different custom option group. When you remove the APEX options, a brief outage occurs while your DB instance is automatically restarted.

For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

When you remove the APEX options from a DB instance, the APEX schema is removed from your database.

Oracle Enterprise Manager

Amazon RDS supports Oracle Enterprise Manager (OEM). OEM is the Oracle product line for integrated management of enterprise information technology.

Amazon RDS supports OEM through the following options.

Option	Option ID	Support For
OEM Database (p. 796)	OEM	OEM Database Express 12c
		OEM 11g Database Control
OEM Management Agent (p. 799)	OEM_AGENT	OEM Cloud Control for 13c
		OEM Cloud Control for 12c

Note

You can use OEM Database or OEM Management Agent, but not both.

Oracle Enterprise Manager Database Express

Amazon RDS supports Oracle Enterprise Manager (OEM) Database Express through the use of the OEM option. Amazon RDS supports the following versions of OEM database:

- Oracle Enterprise Manager Database Express 12c
- Oracle Enterprise Manager 11g Database Control

OEM Database Express and Database Control are similar tools that have a web-based interface for Oracle database administration. For more information about these tools, see [Accessing Enterprise Manager Database Express 12c](#) and [Accessing Enterprise Manager 11g Database Control](#) in the Oracle documentation.

The following are some limitations to using OEM Database:

- OEM Database is not supported on the following DB instance classes: db.t2.micro, db.t2.small, db.m1.small.

For more information about DB instance classes, see [DB Instance Class Support for Oracle \(p. 720\)](#).

- OEM 11g Database Control is not compatible with the following time zones: America/Argentina/Buenos_Aires, America/Matamoros, America/Monterrey, America/Toronto, Asia/Ashgabat, Asia/Dhaka, Asia/Kathmandu, Asia/Kolkata, Asia/Ulaanbaatar, Atlantic/Cape_Verde, Australia/Eucla, Pacific/Kiritimati.

For more information about time zone support, see [Oracle Time Zone \(p. 833\)](#).

OEM Database Option Settings

Amazon RDS supports the following settings for the OEM option.

Option Setting	Valid Values	Description
Port	An integer value	The port on the DB instance that listens for OEM Database. The default for OEM Database Express 12c is 5500. The default for OEM 11g Database Control is 1158.
Security Groups	—	A security group that has access to Port .

Adding the OEM Database Option

The general process for adding the OEM option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the OEM option, you don't need to restart your DB instance. As soon as the option group is active, the OEM Database is active.

To add the OEM option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - a. For **Engine** choose the oracle edition for your DB instance.
 - b. For **Major engine version** choose **11.2**, **12.1**, or **12.2** for your DB instance.

For more information, see [Creating an Option Group \(p. 157\)](#).
2. Add the **OEM** option to the option group, and configure the option settings. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#). For more information about each setting, see [OEM Database Option Settings \(p. 796\)](#).
3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Using OEM Database

After you enable the OEM option, you can begin using the OEM Database tool from your web browser.

You can access either OEM Database Control or OEM Database Express from your web browser. For example, if the endpoint for your Amazon RDS DB instance is `mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com`, and your OEM port is 1158, then the URL to access the OEM Database Control the following.

`https://mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com:1158/em`

When you access either tool from you web browser, a login window appears that prompts you for a user name and password. Type the master user name and master password for your DB instance. You are now ready to manage your Oracle databases.

Modifying OEM Database Settings

After you enable OEM Database, you can modify the Security Groups setting for the option.

You can't modify the OEM port number after you have associated the option group with a DB instance. To change the OEM port number for a DB instance, do the following:

1. Create a new option group.
2. Add the OEM option with the new port number to the new option group.
3. Remove the existing option group from the DB instance.
4. Add the new option group to the DB instance.

For more information about how to modify option settings, see [Modifying an Option Setting \(p. 164\)](#). For more information about each setting, see [OEM Database Option Settings \(p. 796\)](#).

Removing the OEM Database Option

You can remove the OEM option from a DB instance. After you remove the OEM option, you don't need to restart your DB instance.

To remove the OEM option from a DB instance, do one of the following:

- Remove the OEM option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#)
- Modify the DB instance and specify a different option group that doesn't include the OEM option. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Related Topics

- [Working with Option Groups \(p. 156\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)

Oracle Management Agent for Enterprise Manager Cloud Control

Amazon RDS supports Oracle Enterprise Manager (OEM) Management Agent through the use of the OEM_AGENT option. Amazon RDS supports Management Agent for the following versions of OEM:

- Oracle Enterprise Manager Cloud Control for 13c
- Oracle Enterprise Manager Cloud Control for 12c

Management Agent is a software component that monitors targets running on hosts and communicates that information to the middle-tier Oracle Management Service (OMS). For more information, see [Overview of Oracle Enterprise Manager Cloud Control 12c](#) and [Overview of Oracle Enterprise Manager Cloud Control 13c](#) in the Oracle documentation.

The following are some limitations to using Management Agent:

- Administrative tasks such as job execution and database patching, that require host credentials, are not supported.
- Host metrics and the process list are not guaranteed to reflect the actual system state.
- Autodiscovery is not supported. You must manually add database targets.
- OMS module availability depends on your database edition. For example, the database performance diagnosis and tuning module is only available for Oracle Database Enterprise Edition.
- Management Agent consumes additional memory and computing resources. If you experience performance problems after enabling the OEM_AGENT option, we recommend that you scale up to a larger DB instance class. For more information, see [DB Instance Class \(p. 82\)](#) and [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Prerequisites for Management Agent

The following are prerequisites for using Management Agent:

- An Amazon RDS DB instance running Oracle version 12.2.0.1, 12.1.0.2, or 11.2.0.4.
- At least 3.3 GiB of storage space for OEM 13c2.
- At least 3 GiB of storage space for OEM 13c1.
- At least 2 GiB of storage space for OEM 12c.
- An Oracle Management Service (OMS), configured to connect to your Amazon RDS DB instance.
 - For OMS 13c2 with Oracle patch 25163555 applied, use OEM Agent 13.2.0.0.v2 or later.
 - Use OMSPatcher to apply the patch.
 - For unpatched OMS 13c2, use OEM Agent 13.2.0.0.v1.
 - In most cases, you need to configure your VPC to allow connections from OMS to your DB instance. If you are not familiar with Amazon Virtual Private Cloud (Amazon VPC), we recommend that you complete the steps in [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#) before continuing.

Additional configuration is required to allow your OMS host and your Amazon RDS DB instance to communicate. You must also do the following:

- To connect from the Management Agent to your OMS, if your OMS is behind a firewall, you must add the IP addresses of your DB instances to your OMS.

- To connect from your OMS to the Management Agent, if your OMS has a publicly resolvable host name, you must add the OMS address to a security group. Your security group must have inbound rules that allow access to the DB instance port and the Management Agent port. For an example of creating a security and adding inbound rules, see [Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance \(p. 427\)](#).
- To connect from your OMS to the Management Agent, if your OMS doesn't have a publicly resolvable host name, use one of the following:
 - If your OMS is hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance in a private VPC, you can set up VPC peering to connect from OMS to Management Agent. For more information, see [A DB Instance in a VPC Accessed by an EC2 Instance in a Different VPC \(p. 415\)](#).
 - If your OMS is hosted on-premises, you can set up a VPN connection to allow access from OMS to Management Agent. For more information, see [A DB Instance in a VPC Accessed by a Client Application Through the Internet \(p. 417\)](#) or [VPN Connections](#).

Management Agent Option Settings

Amazon RDS supports the following settings for the Management Agent option. When adding the `OEM_AGENT` option, all of the settings are required.

Note

All of the settings are required.

Option Setting	Valid Values	Description
Version (AGENT_VERSION)	13.2.0.0.v2 13.2.0.0.v1 13.1.0.0.v1 12.1.0.5.v1 12.1.0.4.v1	The version of the Management Agent software. The AWS CLI option name is <code>OptionVersion</code> . Note In the AWS GovCloud (US-West) region, only versions 13.2.0.0.v1 and 13.2.0.0.v2 are available.
Port (AGENT_PORT)	An integer value	The port on the DB instance that listens for the OMS host. The default is 3872. Your OMS host must belong to a security group that has access to this port. The AWS CLI option name is <code>Port</code> .
Security Groups	Existing security groups	A security group that has access to Port . Your OMS host must belong to this security group. The AWS CLI option name is <code>VpcSecurityGroupMemberships</code> or <code>DBSecurityGroupMemberships</code> .
OMS_HOST	A string value, for example <code>my.example.oms</code>	The publicly accessible host name or IP address of the OMS.

Option Setting	Valid Values	Description
		The AWS CLI option name is OEM_HOST.
OMS_PORT	An integer value	<p>The HTTPS upload port on the OMS Host that listens for the Management Agent.</p> <p>To determine the HTTPS upload port, connect to the OMS host, and run the following command (which requires the SYSMAN password):</p> <pre>emctl status oms -details</pre> <p>The AWS CLI option name is OMS_PORT.</p>
AGENT_REGISTRATION_PASSWORD	String value	<p>The password that the Management Agent uses to authenticate itself with the OMS. We recommend that you create a persistent password in your OMS before enabling the OEM_AGENT option. With a persistent password you can share a single Management Agent option group among multiple Amazon RDS databases.</p> <p>The AWS CLI option name is AGENT_REGISTRATION_PASSWORD.</p>

Adding the Management Agent Option

The general process for adding the Management Agent option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the Management Agent option, you don't need to restart your DB instance. As soon as the option group is active, the OEM Agent is active.

AWS Management Console

To add the Management Agent option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - a. For **Engine** choose the oracle edition for your DB instance.
 - b. For **Major engine version** choose **11.2**, **12.1**, or **12.2** for your DB instance.

For more information, see [Creating an Option Group \(p. 157\)](#).

2. Add the **OEM_AGENT** option to the option group, and configure the option settings. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#). For more information about each setting, see [Management Agent Option Settings \(p. 800\)](#).
3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

AWS CLI

The following example uses the AWS CLI `add-option-to-option-group` command to add the **OEM_AGENT** option to an option group called `myoptiongroup`.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options
    OptionName=OEM_AGENT,OptionVersion=13.1.0.0.v1,Port=3872,VpcSecurityGroupMemberships=sg-1234567890,Opt
    {Name=OMS_PORT,Value=4903},{Name=AGENT_REGISTRATION_PASSWORD,Value=password}] \
    --apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
    --options
    OptionName=OEM_AGENT,OptionVersion=13.1.0.0.v1,Port=3872,VpcSecurityGroupMemberships=sg-1234567890,Opt
    {Name=OMS_PORT,Value=4903},{Name=AGENT_REGISTRATION_PASSWORD,Value=password}] ^
    --apply-immediately
```

Using the Management Agent

After you enable the Management Agent option, use the following procedure to begin using it.

To use the Management Agent

1. Unlock and reset the DBSNMP account credential, by running the following code on your target database on your DB instance, and using your master user account.

```
ALTER USER dbsnmp IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

2. Add your targets to the OMS console manually:
 - a. In your OMS console, choose **Setup, Add Target, Add Targets Manually**.
 - b. Choose **Add Targets Declaratively by Specifying Target Monitoring Properties**.
 - c. For **Target Type**, choose **Database Instance**.

- d. For **Monitoring Agent**, choose the agent with the same identifier as your Amazon RDS DB instance identifier.
- e. Choose **Add Manually**.
- f. Enter the endpoint for the Amazon RDS DB instance, or select it from the host name list. Ensure that the specified host name matches the endpoint of the Amazon RDS DB instance.

For information about finding the endpoint for your Amazon RDS DB instance, see [Finding the Endpoint of Your DB Instance \(p. 751\)](#).
- g. Specify the following database properties:
 - For **Target name**, type a name.
 - For **Database system name**, type a name.
 - For **Monitor username**, type `dbsnmp`.
 - For **Monitor password**, type the password from Step 1.
 - For **Role**, type `normal`.
 - For **Oracle home path**, type `/oracle`.
 - For **Listener Machine name**, the agent identifier already appears.
 - For **Port**, type the database port. The RDS default port is 1521.
 - For **Database name**, type the name of your database.
- h. Choose **Test Connection**.
- i. Choose **Next**. The target database appears in your list of monitored resources.

Modifying Management Agent Settings

After you enable the Management Agent, you can modify settings for the option. For more information about how to modify option settings, see [Modifying an Option Setting \(p. 164\)](#). For more information about each setting, see [Management Agent Option Settings \(p. 800\)](#).

Removing the Management Agent Option

You can remove the OEM Agent from a DB instance. After you remove the OEM Agent, you don't need to restart your DB instance.

To remove the OEM Agent from a DB instance, do one of the following:

- Remove the OEM Agent option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#)
- Modify the DB instance and specify a different option group that doesn't include the OEM Agent option. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Related Topics

- [Working with Option Groups \(p. 156\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)

Oracle Java Virtual Machine

Amazon RDS supports Oracle Java Virtual Machine (JVM) through the use of the `JVM` option. Oracle Java provides a SQL schema and functions that facilitate Oracle Java features in an Oracle database. For more information, see [Introduction to Java in Oracle Database](#) in the Oracle documentation.

You can use Oracle JVM with the following Oracle Database versions:

- Oracle 12c, 12.2.0.1, all versions
- Oracle 12c, 12.1.0.2.v13 or later
- Oracle 11g, 11.2.0.4.v17 or later

Prerequisites for Oracle JVM

The following are prerequisites for using Oracle Java:

- Your DB instance must be inside a virtual private cloud (VPC). For more information, see [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#).
- Your DB instance must be of a large enough class. Oracle Java isn't supported for the db.m1.small, db.t2.micro, or db.t2.small DB instance classes. For more information, see [DB Instance Class \(p. 82\)](#).
- Your DB instance must have the **Auto minor version upgrade** option enabled. This option enables your DB instance to receive minor DB engine version upgrades automatically when they become available. Amazon RDS uses this option to update your DB instance to the latest Oracle Patch Set Update (PSU). In particular, it does so in cases where there are security vulnerabilities with a Common Vulnerability Scoring System (CVSS) score of 9.0 or greater or other announced security vulnerabilities. For more information, see [Settings for Oracle DB Instances \(p. 759\)](#).
- If your DB instance is running on major version 11.2, you must install the `XMLDB` option. For more information, see [Oracle XML DB \(p. 840\)](#).

Best Practices for Oracle JVM

The following are best practices for using Oracle Java:

- For maximum security, use the `JVM` option with Secure Sockets Layer (SSL). For more information, see [Oracle Secure Sockets Layer \(p. 817\)](#).
- Configure your DB instance to restrict network access. For more information, see [Scenarios for Accessing a DB Instance in a VPC \(p. 414\)](#) and [Working with an Amazon RDS DB Instance in a VPC \(p. 420\)](#).

Adding the Oracle JVM Option

The following is the general process for adding the `JVM` option to a DB instance:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

There is a brief outage while the `JVM` option is added. After you add the option, you don't need to restart your DB instance. As soon as the option group is active, Oracle Java is available.

To add the JVM option to a DB instance

1. Determine the option group that you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - For **Engine**, choose the DB engine used by the DB instance (**oracle-ee**, **oracle-se**, **oracle-se1**, or **oracle-se2**).
 - For **Major engine version**, choose **11.2** or **12.1** for your DB instance.

For more information, see [Creating an Option Group \(p. 157\)](#).

2. Add the **JVM** option to the option group. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).
3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).
4. Grant the required permissions to users.

The Amazon RDS master user has the permissions to use the **JVM** option by default. If other users require these permissions, connect to the DB instance as the master user in a SQL client and grant the permissions to the users.

The following example grants the permissions to use the **JVM** option to the `test_proc` user.

```
create user test_proc identified by password;
CALL dbms_java.grant_permission('TEST_PROC',
  'oracle.aurora.security.JServerPermission', 'LoadClassInPackage.*', ''');
```

After the user is granted the permissions, the following query should return output.

```
select * from dba_java_policy where grantee='TEST_PROC';
```

Note

The Oracle user name is case-sensitive, and it usually has all uppercase characters.

Removing the Oracle JVM Option

You can remove the **JVM** option from a DB instance. There is a brief outage while the option is removed. After you remove the **JVM** option, you don't need to restart your DB instance.

Warning

Removing the **JVM** option can result in data loss if the DB instance is using data types that were enabled as part of the option. Back up your data before proceeding. For more information, see [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#).

To remove the **JVM** option from a DB instance, do one of the following:

- Remove the `JVM` option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#).
- Modify the DB instance and specify a different option group that doesn't include the `JVM` option. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Oracle Label Security

Amazon RDS supports Oracle Label Security for Oracle Enterprise Edition, version 12c, through the use of the OLS option.

Most database security controls access at the object level. Oracle Label Security provides fine-grained control of access to individual table rows. For example, you can use Label Security to enforce regulatory compliance with a policy-based administration model. You can use Label Security policies to control access to sensitive data, and restrict access to only users with the appropriate clearance level. For more information, see [Introduction to Oracle Label Security](#) in the Oracle documentation.

Important

For Oracle 12c version 12.2 on Amazon RDS, Oracle Label Security is a permanent and persistent option. You can't remove Oracle Label Security from an Oracle version 12.2 DB instance.

Prerequisites for Oracle Label Security

The following are prerequisites for using Oracle Label Security:

- Your DB instance must use the Bring Your Own License model. For more information, see [Oracle Licensing \(p. 719\)](#).
- You must have a valid license for Oracle Enterprise Edition with Software Update License and Support.
- Your Oracle license must include the Label Security option.

Adding the Oracle Label Security Option

The general process for adding the Oracle Label Security option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the Label Security option, as soon as the option group is active, Label Security is active.

To add the Label Security option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - a. For **Engine**, choose **oracle-ee**.
 - b. For **Major engine version**, choose **12.1 or 12.2**.

For more information, see [Creating an Option Group \(p. 157\)](#).

2. Add the **OLS** option to the option group. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).

Important

If you add Label Security to an existing option group that is already attached to one or more DB instances, all the DB instances are restarted.

3. Apply the option group to a new or existing DB instance:

- For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).

- For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. When you add the Label Security option to an existing DB instance, a brief outage occurs while your DB instance is automatically restarted. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Using Oracle Label Security

To use Oracle Label Security, you create policies that control access to specific rows in your tables. For more information, see [Creating an Oracle Label Security Policy](#) in the Oracle documentation.

When you work with Label Security, you perform all actions as the LBAC_DBA role. The master user for your DB instance is granted the LBAC_DBA role. You can grant the LBAC_DBA role to other users so that they can administer Label Security policies.

For Amazon RDS for Oracle 12.2 DB instances, you must grant access to the OLS_ENFORCEMENT package to any new users who require access to Oracle Label Security. To grant access to the OLS_ENFORCEMENT package, connect to the DB instance as the master user and run the following SQL statement:

```
GRANT ALL ON LBACSYS.OLS_ENFORCEMENT TO username;
```

You can configure Label Security through the Oracle Enterprise Manager (OEM) Cloud Control. Amazon RDS supports the OEM Cloud Control through the Management Agent option. For more information, see [Oracle Management Agent for Enterprise Manager Cloud Control \(p. 799\)](#).

Removing the Oracle Label Security Option

You can remove Oracle Label Security from a DB instance.

To remove Label Security from a DB instance, do one of the following:

- To remove Label Security from multiple DB instances, remove the Label Security option from the option group they belong to. This change affects all DB instances that use the option group. When you remove Label Security from an option group that is attached to multiple DB instances, all the DB instances are restarted. For more information, see [Removing an Option from an Option Group \(p. 167\)](#).
- To remove Label Security from a single DB instance, modify the DB instance and specify a different option group that doesn't include the Label Security option. You can specify the default (empty) option group, or a different custom option group. When you remove the Label Security option, a brief outage occurs while your DB instance is automatically restarted. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Troubleshooting

The following are issues you might encounter when you use Oracle Label Security.

Issue	Troubleshooting Suggestions
When you try to create a policy, you see an error message similar to the following: <code>insufficient authorization for the SYSDBA package.</code>	A known issue with Oracle's Label Security feature prevents users with usernames of 16 or 24 characters from running Label Security commands. You can create a new user with a different number of characters, grant LBAC_DBA to the new

Issue	Troubleshooting Suggestions
	user, log in as the new user, and run the OLS commands as the new user. For additional information, please contact Oracle support.

Related Topics

- [Working with Option Groups \(p. 156\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)

Oracle Locator

Amazon RDS supports Oracle Locator through the use of the `LOCATOR` option. Oracle Locator provides capabilities that are typically required to support internet and wireless service-based applications and partner-based GIS solutions. Oracle Locator is a limited subset of Oracle Spatial. For more information, see [Oracle Locator](#) in the Oracle documentation.

Important

If you use Oracle Locator, Amazon RDS automatically updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a Common Vulnerability Scoring System (CVSS) score of 9+ or other announced security vulnerabilities.

Amazon RDS supports Oracle Locator for the following editions and versions of Oracle:

- Oracle Standard Edition (SE2) or Enterprise Edition, version 12.2.0.1, all versions
- Oracle Standard Edition (SE2) or Enterprise Edition, version 12.1.0.2.v13 or later
- Oracle Standard Edition (SE, SE1) or Enterprise Edition, version 11.2.0.4.v17 or later

Prerequisites for Oracle Locator

The following are prerequisites for using Oracle Locator:

- Your DB instance must be inside a virtual private cloud (VPC). For more information, see [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#).
- Your DB instance must be of sufficient class. Oracle Locator is not supported for the db.m1.small, db.t2.micro, or db.t2.small DB instance classes. For more information, see [DB Instance Class Support for Oracle \(p. 720\)](#).
- Your DB instance must have Auto Minor Version Upgrade enabled. Amazon RDS updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a CVSS score of 9+ or other announced security vulnerabilities. For more information, see [Settings for Oracle DB Instances \(p. 759\)](#).
- If your DB instance is running on major version 11.2, you must install the `XMLDB` option. For more information, see [Oracle XML DB \(p. 840\)](#).

Best Practices for Oracle Locator

The following are best practices for using Oracle Locator:

- For maximum security, use the `LOCATOR` option with Secure Sockets Layer (SSL). For more information, see [Oracle Secure Sockets Layer \(p. 817\)](#).
- Configure your DB instance to restrict access to your DB instance. For more information, see [Scenarios for Accessing a DB Instance in a VPC \(p. 414\)](#) and [Working with an Amazon RDS DB Instance in a VPC \(p. 420\)](#).

Adding the Oracle Locator Option

The following is the general process for adding the `LOCATOR` option to a DB instance:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

There is a brief outage while the `LOCATOR` option is added. After you add the option, you don't need to restart your DB instance. As soon as the option group is active, Oracle Locator is available.

To add the `LOCATOR` option to a DB instance

1. Determine the option group that you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - a. For **Engine**, choose the oracle edition for your DB instance.
 - b. For **Major engine version**, choose **11.2** or **12.1** for your DB instance.

For more information, see [Creating an Option Group \(p. 157\)](#).
2. Add the `LOCATOR` option to the option group. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).
3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Using Oracle Locator

After you enable the Oracle Locator option, you can begin using it. You should only use Oracle Locator features. Don't use any Oracle Spatial features unless you have a license for Oracle Spatial.

For a list of features that are supported for Oracle Locator, see [Features Included with Locator](#) in the Oracle documentation.

For a list of features that are not supported for Oracle Locator, see [Features Not Included with Locator](#) in the Oracle documentation.

Removing the Oracle Locator Option

You can remove the `LOCATOR` option from a DB instance. There is a brief outage while the option is removed. After you remove the `LOCATOR` option, you don't need to restart your DB instance.

Warning

Removing the `LOCATOR` option can result in data loss if the DB instance is using data types that were enabled as part of the option. Back up your data before proceeding. For more information, see [Backing Up and Restoring Amazon RDS Instances \(p. 207\)](#).

To remove the `LOCATOR` option from a DB instance, do one of the following:

- Remove the `LOCATOR` option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#).
- Modify the DB instance and specify a different option group that doesn't include the `LOCATOR` option. This change affects a single DB instance. You can specify the default (empty) option group or a different custom option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Related Topics

- [Oracle Spatial \(p. 823\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)
- [Working with Option Groups \(p. 156\)](#)

Oracle Multimedia

Amazon RDS supports Oracle Multimedia through the use of the `MULTIMEDIA` option. You can use Oracle Multimedia to store, manage, and retrieve images, audio, video, and other heterogeneous media data. For more information, see [Oracle Multimedia](#) in the Oracle documentation.

Important

If you use Oracle Multimedia, Amazon RDS automatically updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a Common Vulnerability Scoring System (CVSS) score of 9+ or other announced security vulnerabilities.

Amazon RDS supports Oracle Multimedia for the following editions and versions of Oracle:

- Oracle Enterprise Edition, version 12.2.0.1, all versions
- Oracle Enterprise Edition, version 12.1.0.2.v13 or later
- Oracle Enterprise Edition, version 11.2.0.4.v17 or later

Prerequisites for Oracle Multimedia

The following are prerequisites for using Oracle Multimedia:

- Your DB instance must be inside a virtual private cloud (VPC). For more information, see [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#).
- Your DB instance must be of sufficient class. Oracle Multimedia is not supported for the db.m1.small, db.t2.micro, or db.t2.small DB instance classes. For more information, see [DB Instance Class Support for Oracle \(p. 720\)](#).
- Your DB instance must have Auto Minor Version Upgrade enabled. Amazon RDS updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a CVSS score of 9+ or other announced security vulnerabilities. For more information, see [Settings for Oracle DB Instances \(p. 759\)](#).
- If your DB instance is running on major version 11.2, you must install the `XMLDB` option. For more information, see [Oracle XML DB \(p. 840\)](#).

Best Practices for Oracle Multimedia

The following are best practices for using Oracle Multimedia:

- For maximum security, use the `MULTIMEDIA` option with Secure Sockets Layer (SSL). For more information, see [Oracle Secure Sockets Layer \(p. 817\)](#).
- Configure your DB instance to restrict access to your DB instance. For more information, see [Scenarios for Accessing a DB Instance in a VPC \(p. 414\)](#) and [Working with an Amazon RDS DB Instance in a VPC \(p. 420\)](#).

Adding the Oracle Multimedia Option

The following is the general process for adding the `MULTIMEDIA` option to a DB instance:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

There is a brief outage while the `MULTIMEDIA` option is added. After you add the option, you don't need to restart your DB instance. As soon as the option group is active, Oracle Multimedia is available.

To add the **MULTIMEDIA** option to a DB instance

1. Determine the option group that you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - a. For **Engine**, choose **oracle-ee**.
 - b. For **Major engine version**, choose **11.2** or **12.1** for your DB instance.

For more information, see [Creating an Option Group \(p. 157\)](#).
2. Add the **MULTIMEDIA** option to the option group. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).
3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Removing the Oracle Multimedia Option

You can remove the **MULTIMEDIA** option from a DB instance. There is a brief outage while the option is removed. After you remove the **MULTIMEDIA** option, you don't need to restart your DB instance.

Warning

Removing the **MULTIMEDIA** option can result in data loss if the DB instance is using data types that were enabled as part of the option. Back up your data before proceeding. For more information, see [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#).

To remove the **MULTIMEDIA** option from a DB instance, do one of the following:

- Remove the **MULTIMEDIA** option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#).
- Modify the DB instance and specify a different option group that doesn't include the **MULTIMEDIA** option. This change affects a single DB instance. You can specify the default (empty) option group or a different custom option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Related Topics

- [Working with Option Groups \(p. 156\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)

Oracle Native Network Encryption

Amazon RDS supports Oracle native network encryption (NNE). With native network encryption, you can encrypt data as it moves to and from a DB instance. Amazon RDS supports NNE for all editions of Oracle.

A detailed discussion of Oracle native network encryption is beyond the scope of this guide, but you should understand the strengths and weaknesses of each algorithm and key before you decide on a solution for your deployment. For information about the algorithms and keys that are available through Oracle native network encryption, see [Configuring Network Data Encryption](#) in the Oracle documentation. For more information about AWS security, see the [AWS Security Center](#).

Note

You can use Native Network Encryption or Secure Sockets Layer, but not both. For more information, see [Oracle Secure Sockets Layer \(p. 817\)](#).

NNE Option Settings

Amazon RDS supports the following settings for the NNE option.

Option Setting	Valid Values	Default Value	Description
<code>SQLNET.ENCRYPTION_SERVER</code>	Rejected, Requested, Required	Requested	The encryption behavior when a client, or a server acting as a client, connects to the DB instance. Requested indicates that the DB instance does not require traffic from the client to be encrypted.
<code>SQLNET.CRYPTO_CHECKSUM_SERVER</code>	Rejected, Requested, Required	Requested	The data integrity behavior when a client, or a server acting as a client, connects to the DB instance. Requested indicates that the DB instance does not require the client to perform a checksum.
<code>SQLNET.ENCRYPTION_TYPES_SERVER</code>	RC4_256, AES192, AES256, AES128, 3DES168, RC4_128, 3DES112, RC4_56, DES, RC4_40, DES40	RC4_256, AES192, AES256, AES128, 3DES168, RC4_128, 3DES112, RC4_56, DES, RC4_40, DES40	A list of encryption algorithms used by the DB instance. The DB instance will use each algorithm, in order, to attempt to decrypt the client input until an algorithm succeeds or until the end of the list is reached. Amazon RDS uses the following default list from Oracle. You can change the order or limit the algorithms that the DB instance will accept. <ol style="list-style-type: none"> 1. RC4_256: RSA RC4 (256-bit key size) 2. AES256: AES (256-bit key size) 3. AES192: AES (192-bit key size) 4. 3DES168: 3-key Triple-DES (112-bit effective key size)

Option Setting	Valid Values	Default Value	Description
			5. RC4_128: RSA RC4 (128-bit key size) 6. AES128: AES (128-bit key size) 7. 3DES112: 2-key Triple-DES (80-bit effective key size) 8. RC4_56: RSA RC4 (56-bit key size) 9. DES: Standard DES (56-bit key size) 10RC4_40: RSA RC4 (40-bit key size) 11DES40: DES40 (40-bit key size)
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	SHA512, SHA384, SHA512, SHA1, MD5	MD5	The checksum algorithm.

Adding the NNE Option

The general process for adding the NNE option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the NNE option, as soon as the option group is active, NNE is active.

To add the NNE option to a DB instance

1. For **Engine**, choose the Oracle edition that you want to use. NNE is supported on all editions.
2. For **Major engine version**, choose **11.2**, **12.1**, or **12.2**.

For more information, see [Creating an Option Group \(p. 157\)](#).

3. Add the **NNE** option to the option group. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).

Note

After you add the NNE option, you don't need to restart your DB instances. As soon as the option group is active, NNE is active.

4. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. After you add the NNE option, you don't need to restart your DB instance. As soon as the option group is active, NNE is active. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Using NNE

With Oracle native network encryption, you can also specify network encryption on the client side. On the client (the computer used to connect to the DB instance), you can use the sqlnet.ora file to specify the following client settings: SQLNET.CRYPTO_CHECKSUM_CLIENT, SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT, SQLNET.ENCRYPTION_CLIENT, and SQLNET.ENCRYPTION_TYPES_CLIENT. For information, see [Configuring Network Data Encryption and Integrity for Oracle Servers and Clients](#) in the Oracle documentation.

Sometimes, the DB instance will reject a connection request from an application, for example, if there is a mismatch between the encryption algorithms on the client and on the server.

To test Oracle native network encryption , add the following lines to the sqlnet.ora file on the client:

```
DIAG_ADR_ENABLED=off
TRACE_DIRECTORY_CLIENT=/tmp
TRACE_FILE_CLIENT=nettrace
TRACE_LEVEL_CLIENT=16
```

These lines generate a trace file on the client called /tmp/nettrace* when the connection is attempted. The trace file contains information on the connection. For more information about connection-related issues when you are using Oracle Native Network Encryption, see [About Negotiating Encryption and Integrity](#) in the Oracle documentation.

Modifying NNE Settings

After you enable NNE, you can modify settings for the option. For more information about how to modify option settings, see [Modifying an Option Setting \(p. 164\)](#). For more information about each setting, see [NNE Option Settings \(p. 815\)](#).

Removing the NNE Option

You can remove NNE from a DB instance.

To remove NNE from a DB instance, do one of the following:

- To remove NNE from multiple DB instances, remove the NNE option from the option group they belong to. This change affects all DB instances that use the option group. After you remove the NNE option, you don't need to restart your DB instances. For more information, see [Removing an Option from an Option Group \(p. 167\)](#).
- To remove NNE from a single DB instance, modify the DB instance and specify a different option group that doesn't include the NNE option. You can specify the default (empty) option group, or a different custom option group. After you remove the NNE option, you don't need to restart your DB instance. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Related Topics

- [Working with Option Groups \(p. 156\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)

Oracle Secure Sockets Layer

You enable Secure Sockets Layer (SSL) encryption for an Oracle DB instance by adding the Oracle SSL option to the option group associated with an Oracle DB instance. You specify the port you want to communicate over using SSL. You must configure SQL*Plus as shown in this following section.

You enable SSL encryption for an Oracle DB instance by adding the Oracle SSL option to the option group associated with the DB instance. Amazon RDS uses a second port, as required by Oracle, for SSL connections. This approach allows both clear text and SSL-encrypted communication to occur at the same time between a DB instance and SQL*Plus. For example, you can use the port with clear text communication to communicate with other resources inside a VPC while using the port with SSL-encrypted communication to communicate with resources outside the VPC.

Note

You can use Secure Sockets Layer or Native Network Encryption, but not both. For more information, see [Oracle Native Network Encryption \(p. 815\)](#).

You can use SSL encryption with the following Oracle database versions and editions:

- 12.2.0.1: All versions, all editions including Standard Edition Two
- 12.1.0.2: All versions, all editions including Standard Edition Two
- 11.2.0.4: All versions, Enterprise Edition
- 11.2.0.4: Version 6 and later, Standard Edition, Standard Edition One, Enterprise Edition

Note

You cannot use both SSL and Oracle native network encryption (NNE) on the same instance. If you use SSL encryption, you must disable any other connection encryption.

TLS Versions for the Oracle SSL Option

Amazon RDS for Oracle supports Transport Layer Security (TLS) versions 1.0 and 1.2. To use the Oracle SSL option, you must use the `SQLNET.SSL_VERSION` option setting. Following are the allowed values for this option setting:

- "1.0" – Clients can connect to the DB instance using TLS 1.0 only.
- "1.2" – Clients can connect to the DB instance using TLS 1.2 only.
- "1.2 or 1.0" – Clients can connect to the DB instance using either TLS 1.2 or 1.0.

To use the Oracle SSL option, the `SQLNET.SSL_VERSION` option setting is also required:

- For existing Oracle SSL options, `SQLNET.SSL_VERSION` is set to "1.0" automatically. You can change the setting if necessary.
- When you add a new Oracle SSL option, you must set `SQLNET.SSL_VERSION` explicitly to a valid value.

The following table shows the TLS option settings that are supported for different Oracle engine versions and editions.

Oracle Engine Version	<code>SQLNET.SSL_VERSION="1.0"</code>	<code>SQLNET.SSL_VERSION="1.2 or 1.0"</code>	<code>SQLNET.SSL_VERSION="1.2"</code>
12.2.0.1 (All editions)	Supported	Supported	Supported
12.1.0.2 (All editions)	Supported	Supported	Supported
11.2.0.4 (Oracle EE)	Supported	Supported for 11.2.0.4.v8 and higher	Supported for 11.2.0.4.v8 and higher
11.2.0.4 (Oracle SE1)	Supported	Not supported	Not supported
11.2.0.4 (Oracle SE)	Supported	Not supported	Not supported

Configuring SQL*Plus to Use SSL with an Oracle DB Instance

You must configure SQL*Plus before connecting to an Oracle DB instance that uses the Oracle SSL option.

Note

To allow access to the DB instance from the appropriate clients, ensure that your security groups are configured correctly. For more information, see [Controlling Access with Security Groups \(p. 394\)](#). Also, these instructions are for SQL*Plus and other clients that directly use an Oracle home. For JDBC connections, see [Setting Up an SSL Connection Over JDBC \(p. 820\)](#).

To configure SQL*Plus to use SSL to connect to an Oracle DB instance

1. Set the ORACLE_HOME environment variable to the location of your Oracle home directory.

The path to your Oracle home directory depends on your installation. The following example sets the ORACLE_HOME environment variable.

```
prompt>export ORACLE_HOME=/home/user/app/user/product/12.1.0/dbhome_1
```

For information about setting Oracle environment variables, see [SQL*Plus Environment Variables](#) in the Oracle documentation, and also see the Oracle installation guide for your operating system.

2. Append \$ORACLE_HOME/lib to the LD_LIBRARY_PATH environment variable.

The following is an example that sets the LD_LIBRARY_PATH environment variable.

```
prompt>export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
```

3. Create a directory for the Oracle wallet at \$ORACLE_HOME/ssl_wallet.

The following is an example that creates the Oracle wallet directory.

```
prompt>mkdir $ORACLE_HOME/ssl_wallet
```

4. Download the RDS CA certificates file from <https://s3.amazonaws.com/rds-downloads/rds-ca-2015-root.pem> and then put the file in the ssl_wallet directory.

The RDS CA certificates file for AWS GovCloud (US-West) is available at <https://s3-us-gov-west-1.amazonaws.com/rds-downloads/rds-ca-2012-us-gov-west-1.pem>.

5. In the \$ORACLE_HOME/network/admin directory, modify or create the tnsnames.ora file and include the following entry.

```
<net_service_name>= (DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCPS)
    (HOST = <endpoint>) (PORT = <ssl port number>)))(CONNECT_DATA = (SID = <database
    name>))
    (SECURITY = (SSL_SERVER_CERT_DN =
    "C=US,ST=Washington,L=Seattle,O=Amazon.com,OU=RDS,CN=<endpoint>")))
```

6. In the same directory, modify or create the sqlnet.ora file and include the following parameters.

```
WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY = $ORACLE_HOME/
    ssl_wallet)))
```

```
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_VERSION = 1.0
SSL_CIPHER_SUITES = (SSL_RSA_WITH_AES_256_CBC_SHA)
SSL_SERVER_DN_MATCH = ON
```

- Run the following commands to create the Oracle wallet.

```
prompt>orapki wallet create -wallet $ORACLE_HOME/ssl_wallet -auto_login_only
prompt>orapki wallet add -wallet $ORACLE_HOME/ssl_wallet -trusted_cert -cert
$ORACLE_HOME/ssl_wallet/rds-ca-2015-root.pem -auto_login_only
```

Connecting to an Oracle DB Instance Using SSL

After you configure SQL*Plus to use SSL as described previously, you can connect to the Oracle DB instance with the SSL option. For example, you can connect using SQL*Plus and a `<net_service_name>` in a tnsnames.ora file.

```
sqlplus <mydbuser>@<net_service_name>
```

You can also connect to the DB instance using SQL*Plus without using a tnsnames.ora file by using the following command.

```
sqlplus '<mydbuser>@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCPS)(HOST = <endpoint>) (PORT
= <ssl port number>))(CONNECT_DATA = (SID = <database name>)))'
```

You can also connect to the Oracle DB instance without using SSL. For example, the following command connects to the DB instance through the clear text port without SSL encryption.

```
sqlplus '<mydbuser>@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <endpoint>) (PORT
= <port number>))(CONNECT_DATA = (SID = <database name>)))'
```

If you want to close Transmission Control Protocol (TCP) port access, create a security group with no IP address ingresses and add it to the instance. This addition closes connections over the TCP port, while still allowing connections over the SSL port that are specified from IP addresses within the range permitted by the SSL option security group.

Setting Up an SSL Connection Over JDBC

To use an SSL connection over JDBC, you must create a keystore, trust the Amazon RDS root CA certificate, and use the code snippet specified following.

To create the keystore in JKS format, use the following command. For more information about creating the keystore, see the [Oracle documentation](#).

```
keytool -keystore clientkeystore -genkey -alias client
```

Next, take the following steps to trust the Amazon RDS root CA certificate.

To trust the Amazon RDS root CA certificate

1. Download the Amazon RDS root CA certificate from <https://s3.amazonaws.com/rds-downloads/rds-ca-2015-root.pem>.
2. Convert the certificate to .der format using the following command.

```
openssl x509 -outform der -in rds-ca-2015-root.pem -out rds-ca-2015-root.der
```

3. Import the certificate into the keystore using the following command.

```
keytool -import -alias rds-root -keystore clientkeystore -file rds-ca-2015-root.der
```

The following code example shows how to set up the SSL connection using JDBC.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class OracleSslConnectionTest {
    private static final String DB_SERVER_NAME = "<dns-name-provided-by-amazon-rds>";
    private static final Integer SSL_PORT = "<ssl-option-port-configured-in-option-group>";
    private static final String DB_SID = "<oracle-sid>";
    private static final String DB_USER = "<user name>";
    private static final String DB_PASSWORD = "<password>";
    // This key store has only the prod root ca: https://s3.amazonaws.com/rds-downloads/
    rds-ca-2015-root.pem
    private static final String KEY_STORE_FILE_PATH = "<file-path-to-keystore>";
    private static final String KEY_STORE_PASS = "<keystore-password>";

    public static void main(String[] args) throws SQLException {
        final Properties properties = new Properties();
        final String connectionString = String.format(
            "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=%d)) "
        (CONNECT_DATA=(SID=%s)))",
            DB_SERVER_NAME, SSL_PORT, DB_SID);
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);
        properties.put("oracle.jdbc.J2EE13Compliant", "true");
        properties.put("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        properties.put("javax.net.ssl.trustStoreType", "JKS");
        properties.put("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);
        final Connection connection = DriverManager.getConnection(connectionString,
        properties);
        // If no exception, that means handshake has passed, and an SSL connection can be
        opened
    }
}
```

Enforcing a DN Match with an SSL Connection

You can use the Oracle parameter `SSL_SERVER_DN_MATCH` to enforce that the distinguished name (DN) for the database server matches its service name. If you enforce the match verifications, then SSL ensures that the certificate is from the server. If you don't enforce the match verification, then SSL performs the check but allows the connection, regardless if there is a match. If you do not enforce the match, you allow the server to potentially fake its identify.

To enforce DN matching, add the DN match property and use the connection string specified below.

Add the property to the client connection to enforce DN matching.

```
properties.put("oracle.net.ssl_server_dn_match", "TRUE");
```

Use the following connection string to enforce DN matching when using SSL.

```
final String connectionString = String.format(
    "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=%d))" +
    "(CONNECT_DATA=(SID=%s))" +
    "(SECURITY = (SSL_SERVER_CERT_DN =
    \"C=US,ST=Washington,L=Seattle,O=Amazon.com,OU=RDS,CN=%s\")))",
    DB_SERVER_NAME, SSL_PORT, DB_SID, DB_SERVER_NAME);
```

Oracle Spatial

Amazon RDS supports Oracle Spatial through the use of the `SPATIAL` option. Oracle Spatial provides a SQL schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial data in an Oracle database. For more information, see [Spatial Concepts](#) in the Oracle documentation.

Important

If you use Oracle Spatial, Amazon RDS automatically updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a Common Vulnerability Scoring System (CVSS) score of 9+ or other announced security vulnerabilities.

Amazon RDS supports Oracle Spatial for the following editions and versions of Oracle:

- Oracle Enterprise Edition, version 12.2.0.1, all versions
- Oracle Enterprise Edition, version 12.1.0.2.v13 or later
- Oracle Enterprise Edition, version 11.2.0.4.v17 or later

Prerequisites for Oracle Spatial

The following are prerequisites for using Oracle Spatial:

- Your DB instance must be inside a virtual private cloud (VPC). For more information, see [Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform \(p. 412\)](#).
- Your DB instance must be of sufficient class. Oracle Spatial is not supported for the db.m1.small, db.t2.micro, or db.t2.small DB instance classes. For more information, see [DB Instance Class Support for Oracle \(p. 720\)](#).
- Your DB instance must have Auto Minor Version Upgrade enabled. Amazon RDS updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a CVSS score of 9+ or other announced security vulnerabilities. For more information, see [Settings for Oracle DB Instances \(p. 759\)](#).
- If your DB instance is running on major version 11.2, you must install the `XMLDB` option. For more information, see [Oracle XML DB \(p. 840\)](#).
- An Oracle Spatial license from Oracle. For more information, see [Oracle Spatial and Graph](#) in the Oracle documentation.

Best Practices for Oracle Spatial

The following are best practices for using Oracle Spatial:

- For maximum security, use the `SPATIAL` option with Secure Sockets Layer (SSL). For more information, see [Oracle Secure Sockets Layer \(p. 817\)](#).
- Configure your DB instance to restrict access to your DB instance. For more information, see [Scenarios for Accessing a DB Instance in a VPC \(p. 414\)](#) and [Working with an Amazon RDS DB Instance in a VPC \(p. 420\)](#).

Adding the Oracle Spatial Option

The following is the general process for adding the `SPATIAL` option to a DB instance:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

There is a brief outage while the `SPATIAL` option is added. After you add the option, you don't need to restart your DB instance. As soon as the option group is active, Oracle Spatial is available.

To add the `SPATIAL` option to a DB instance

1. Determine the option group that you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - a. For **Engine**, choose `oracle-ee`.
 - b. For **Major engine version**, choose **11.2** or **12.1** for your DB instance.

For more information, see [Creating an Option Group \(p. 157\)](#).
2. Add the `SPATIAL` option to the option group. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).
3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Removing the Oracle Spatial Option

You can remove the `SPATIAL` option from a DB instance. There is a brief outage while the option is removed. After you remove the `SPATIAL` option, you don't need to restart your DB instance.

Warning

Removing the `SPATIAL` option can result in data loss if the DB instance is using data types that were enabled as part of the option. Back up your data before proceeding. For more information, see [Backing Up and Restoring Amazon RDS DB Instances \(p. 207\)](#).

To remove the `SPATIAL` option from a DB instance, do one of the following:

- Remove the `SPATIAL` option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#).
- Modify the DB instance and specify a different option group that doesn't include the `SPATIAL` option. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Related Topics

- [Oracle Locator \(p. 810\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)
- [Working with Option Groups \(p. 156\)](#)

Oracle SQLT

Amazon RDS supports Oracle SQLTXPLAIN (SQLT) through the use of the SQLT option.

The Oracle EXPLAIN PLAN statement can determine the execution plan of a SQL statement. It can verify whether the Oracle optimizer chooses a certain execution plan, such as a nested loops join. It also helps you understand the optimizer's decisions, such as why it chose a nested loops join over a hash join. So EXPLAIN PLAN helps you understand the statement's performance.

SQLT is an Oracle utility that produces a report. The report includes object statistics, object metadata, optimizer-related initialization parameters, and other information that a database administrator can use to tune a SQL statement for optimal performance. SQLT produces an HTML report with hyperlinks to all of the sections in the report.

Unlike Automatic Workload Repository or Statspack reports, SQLT works on individual SQL statements. SQLT is a collection of SQL, PL/SQL, and SQL*Plus files that collect, store, and display performance data.

Amazon RDS for Oracle currently supports the following versions of SQLT:

- 12.1.160429
- 12.2.180331

To download SQLT and access instructions for using it:

- Log in to your My Oracle Support account, and open the following documents:
 - To download SQLT: [Document 215187.1](#)
 - For SQLT usage instructions: [Document 1614107.1](#)
 - For frequently asked questions about SQLT: [Document 1454160.1](#)
 - For information about reading SQLT output: [Document 1456176.1](#)
 - For interpreting the Main report: [Document 1922234.1](#)

You can use SQLT with any edition of the following Oracle Database versions:

- Oracle 12c, 12.2.0.1
- Oracle 12c, 12.1.0.2
- Oracle 11g, 11.2.0.4

Amazon RDS does not support the following SQLT methods:

- XPORE
- XHUME

Prerequisites for SQLT

The following are prerequisites for using SQLT:

- You must remove users and roles that are required by SQLT, if they exist.

The SQLT option creates the following users and roles on a DB instance:

- SQLTXPLAIN user
- SQLTXADMIN user
- SQLT_USER_ROLE role

If your DB instance has any of these users or roles, log in to the DB instance using a SQL client, and drop them using the following statements:

```
DROP USER SQLTXPLAIN CASCADE;
DROP USER SQLTXADMIN CASCADE;
DROP ROLE SQLT_USER_ROLE CASCADE;
```

- You must remove tablespaces that are required by SQLT, if they exist.

The SQLT option creates the following tablespaces on a DB instance:

- RDS_SQLT_TS
- RDS_TEMP_SQLT_TS

If your DB instance has these tablespaces, log in to the DB instance using a SQL client, and drop them.

SQLT Option Settings

SQLT can work with licensed features that are provided by the Oracle Tuning Pack and the Oracle Diagnostics Pack. The Oracle Tuning Pack includes the SQL Tuning Advisor, and the Oracle Diagnostics Pack includes the Automatic Workload Repository. The SQLT settings enable or disable access to these features from SQLT.

Amazon RDS supports the following settings for the SQLT option.

Option Setting	Valid Values	Default Value	Description
LICENSE_PACK	T, D, N	T	<p>The Oracle Management Packs that you want to access with SQLT. Enter one of the following values:</p> <ul style="list-style-type: none"> • T indicates that you have a license for the Oracle Tuning Pack and the Oracle Diagnostics Pack, and you want to access the SQL Tuning Advisor and Automatic Workload Repository from SQLT. • D indicates that you have a license for the Oracle Diagnostics Pack, and you want to access the Automatic Workload Repository from SQLT. • N indicates that you don't have a license for the Oracle Tuning Pack and the Oracle Diagnostics Pack, or that you have a license for one or both of them, but you don't want SQLT to access them. <p>Note Amazon RDS does not provide licenses for these Oracle Management Packs. If you indicate that you want to use a pack that is not included in your DB instance, you can use SQLT with the DB instance. However, SQLT can't access the pack, and the SQLT report doesn't include the data for the pack. For example, if you specify T, but the DB instance doesn't include the Oracle Tuning Pack, SQLT works on the DB instance, but the</p>

Option Setting	Valid Values	Default Value	Description
			report it generates doesn't contain data related to the Oracle Tuning Pack.
VERSION	2016-04-29, 2016-04-29, 2018-03-31	v1	The version of SQLT that you want to install.

Adding the SQLT Option

The following is the general process for adding the SQLT option to a DB instance:

1. Create a new option group, or copy or modify an existing option group.
2. Add the SQLT option to the option group.
3. Associate the option group with the DB instance.

After you add the SQLT option, as soon as the option group is active, SQLT is active.

To add the SQLT option to a DB instance

1. Determine the option group that you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - a. For **Engine**, choose the Oracle edition that you want to use. The SQLT option is supported on all editions.
 - b. For **Major engine version**, choose **11.2**, **12.1**, or **12.2**.

For more information, see [Creating an Option Group \(p. 157\)](#).

2. Add the **SQLT** option to the option group. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).
3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).
4. (Optional) Verify the SQLT installation on each DB instance with the SQLT option.
 - a. Use a SQL client to connect to the DB instance as the master user.

For information about connecting to an Oracle DB instance using a SQL client, see [Connecting to a DB Instance Running the Oracle Database Engine \(p. 751\)](#).

 - b. Run the following query:

```
SELECT sqltxplain.sqlt$a.get_param('tool_version') sqlt_version FROM DUAL;
```

The query returns the current version of the SQLT option on Amazon RDS. 12.1.160429 is an example of a version of SQLT that is available on Amazon RDS.

5. Change the passwords of the users that are created by the SQLT option.
 - a. Use a SQL client to connect to the DB instance as the master user.
 - b. Run the following SQL statement to change the password for the `SQLTXADMIN` user:

```
ALTER USER SQLTXADMIN IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

- c. Run the following SQL statement to change the password for the `SQLTXPLAIN` user:

```
ALTER USER SQLTXPLAIN IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

Note

Upgrading SQLT requires uninstalling an older version of SQLT and then installing the new version. So, all SQLT metadata can be lost when you upgrade SQLT. A major version upgrade of a database also uninstalls and re-installs SQLT. An example of a major version upgrade is an upgrade from Oracle 11g to Oracle 12c.

Using SQLT

SQLT works with the Oracle SQL*Plus utility.

To use SQLT

1. Download the SQLT .zip file from [Document 215187.1](#) on the My Oracle Support site.

Note

You can't download SQLT 12.1.160429 from the My Oracle Support site. Oracle has deprecated this older version.

2. Unzip the SQLT .zip file.
3. From a command prompt, change to the `sqlt/run` directory on your file system.
4. From the command prompt, open SQL*Plus, and connect to the DB instance as the master user.

For information about connecting to a DB instance using SQL*Plus, see [Connecting to a DB Instance Running the Oracle Database Engine \(p. 751\)](#).

5. Get the SQL ID of a SQL statement:

```
SELECT SQL_ID FROM V$SQL WHERE SQL_TEXT='sql_statement';
```

Your output is similar to the following:

```
SQL_ID
-----
chvsmttqjzjkn
```

6. Analyze a SQL statement with SQLT:

```
START sqltxtract.sql sql_id sqltxplain_user_password
```

For example, for the SQL ID chvsmttqjzjkn, enter the following:

```
START sqltxtract.sql chvsmttqjzjkn sqltxplain_user_password
```

SQLT generates the HTML report and related resources as a .zip file in the directory from which the SQLT command was run.

7. (Optional) To enable application users to diagnose SQL statements with SQLT, grant SQLT_USER_ROLE to each application user with the following statement:

```
GRANT ROLE SQLT_USER_ROLE TO application_user_name;
```

Note

Oracle does not recommend running SQLT with the SYS user or with users that have the DBA role. It is a best practice to run SQLT diagnostics using the application user's account, by granting SQLT_USER_ROLE to the application user.

Upgrading the SQLT Option

With Amazon RDS for Oracle, you can upgrade the SQLT option from version 12.1.160429 to version 12.2.180331. To upgrade the SQLT option, complete steps 1–3 in [Using SQLT \(p. 828\)](#) for the new version of SQLT. Also, if you granted privileges for the previous version of SQLT in step 7 of that section, grant the privileges again for the new SQLT version.

Upgrading the SQLT option results in the loss of the older SQLT version's metadata. The older SQLT version's schema and related objects are dropped, and the newer version of SQLT is installed. For more information about the changes in SQLT version 12.2.180331, see [Document 1614201.1](#) on the My Oracle Support site.

Note

Version downgrades are not supported.

Modifying SQLT Settings

After you enable SQLT, you can modify the LICENSE_PACK and VERSION settings for the option.

For more information about how to modify option settings, see [Modifying an Option Setting \(p. 164\)](#). For more information about each setting, see [SQLT Option Settings \(p. 826\)](#).

Removing the SQLT Option

You can remove SQLT from a DB instance.

To remove SQLT from a DB instance, do one of the following:

- To remove SQLT from multiple DB instances, remove the SQLT option from the option group to which the DB instances belong. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#).

- To remove SQLT from a single DB instance, modify the DB instance and specify a different option group that doesn't include the SQLT option. You can specify the default (empty) option group or a different custom option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Related Topics

- [Working with Option Groups \(p. 156\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)

Oracle Statspack

The Oracle Statspack option installs and enables the Oracle Statspack performance statistics feature. Oracle Statspack is a collection of SQL, PL/SQL, and SQL*Plus scripts that collect, store, and display performance data. For information about using Oracle Statspack, see [Oracle Statspack](#) in the Oracle documentation.

Note

Oracle Statspack is no longer supported by Oracle and has been replaced by the more advanced Automatic Workload Repository (AWR). AWR is available only for Oracle Enterprise Edition customers who have purchased the Diagnostics Pack. Oracle Statspack can be used with any Oracle DB engine on Amazon RDS.

The following steps show you how to work with Oracle Statspack on Amazon RDS:

1. If you have an existing DB instance that has the PERFSTAT account already created and you want to use Oracle Statspack with it, you must drop the PERFSTAT account before adding the Statspack option to the option group associated with your DB instance. If you attempt to add the Statspack option to an option group associated with a DB instance that already has the PERFSTAT account created, you get an error and the RDS event RDS-Event-0058 is generated.

If you have already installed Statspack, and the PERFSTAT account is associated with Statspack, then skip this step, and do not drop the PERFSTAT user.

You can drop the PERFSTAT account by running the following command:

```
DROP USER perfstat CASCADE;
```

2. Add the Statspack option to an option group and then associate that option group with your DB instance. Amazon RDS installs the Statspack scripts on the DB instance and then sets up the PERFSTAT user account, the account you use to run the Statspack scripts. If you have installed Statspack, skip this step.
3. After Amazon RDS has installed Statspack on your DB instance, you must log in to the DB instance using your master user name and master password. You must then reset the PERFSTAT password from the randomly generated value Amazon RDS created when Statspack was installed. After you have reset the PERFSTAT password, you can log in using the PERFSTAT user account and run the Statspack scripts.

Use the following command to reset the password:

```
ALTER USER perfstat IDENTIFIED BY <new_password> ACCOUNT UNLOCK;
```

4. After you have logged on using the PERFSTAT account, you can either manually create a Statspack snapshot or create a job that will take a Statspack snapshot after a given time interval. For example, the following job creates a Statspack snapshot every hour:

```
variable jn number;
execute dbms_job.submit(:jn, 'statspack.snap;', sysdate,'trunc(SYSDATE+1/24,''HH24''));
commit;
```

- Once you have created at least two Statspack snapshots, you can view them using the following query:

```
select snap_id, snap_time from stats$snapshot order by 1;
```

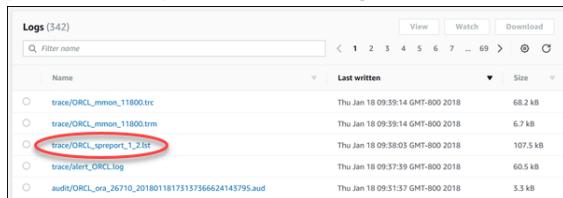
- To create a Statspack report, you choose two snapshots to analyze and run the following Amazon RDS command:

```
exec RDSADMIN.RDS_RUN_SPREPORT(<begin snap>,<end snap>);
```

For example, the following Amazon RDS command would create a report based on the interval between Statspack snapshots 1 and 2:

```
exec RDSADMIN.RDS_RUN_SPREPORT(1,2);
```

The file name of the Statspack report that is generated includes the number of the two Statspack snapshots used. For example, a report file created using Statspack snapshots 1 and 2 would be named ORCL_spreport_1_2.lst. You can download the Statspack report by selecting the report in the **Log** section of the DB instance details on the RDS console and clicking **Download** or you can use the trace file procedures explained in [Working with Oracle Trace Files \(p. 328\)](#).



Name	Last written	Size
trace/ORCL_mmon_11800.trc	Thu Jan 18 09:59:14 GMT-800 2018	68.2 kB
trace/ORCL_mmon_11800.trm	Thu Jan 18 09:59:14 GMT-800 2018	6.7 kB
trace/ORCL_spreport_1_2.lst	Thu Jan 18 09:58:03 GMT-800 2018	107.5 kB
trace/alert_ORCL.log	Thu Jan 18 09:37:59 GMT-800 2018	60.5 kB
audit/ORCL ora_26710_20180118173157366624145795.aud	Thu Jan 18 09:51:57 GMT-800 2018	3.5 kB

If an error occurs when producing the report, an error file is created using the same naming conventions but with an extension of .err. For example, if an error occurred while creating a report using Statspack snapshots 1 and 7, the report file would be named ORCL_spreport_1_7.err. You can download the error report by selecting the report in the Log section of the RDS console and clicking **Download** or use the trace file procedures explained in [Working with Oracle Trace Files \(p. 328\)](#).

Oracle Statspack does some basic checking before running the report, so you could also see error messages displayed at the command prompt. For example, if you attempt to generate a report based on an invalid range, such as the beginning Statspack snapshot value is larger than the ending Statspack snapshot value, the error message is displayed at the command prompt and no error file is created.

```
exec RDSADMIN.RDS_RUN_SPREPORT(2,1);
*
ERROR at line 1:
ORA-20000: Invalid snapshot IDs. Find valid ones in perfstat.stats$snapshot.
```

If you use an invalid number for one of the Statspack snapshots, the error message will also be displayed at the command prompt. For example, if you have 20 Statspack snapshots but request that a report be run using Statspack snapshots 1 and 50, the command prompt will display an error.

```
exec RDSADMIN.RDS_RUN_SPREPORT(1,50);
*
ERROR at line 1:
```

ORA-20000: Could not find both snapshot IDs

For more information about how to use Oracle Statspack, including information on adjusting the amount of data captured by adjusting the snapshot level, go to the Oracle [Statspack documentation page](#).

To remove Oracle Statspack files, use the following command:

```
execute statspack.purge(<begin snap>, <end snap>);
```

Oracle Time Zone

You can use the time zone option to change the system time zone used by your Oracle DB instance. For example, you might change the time zone of a DB instance to be compatible with an on-premises environment, or a legacy application. The time zone option changes the time zone at the host level. Changing the time zone impacts all date columns and values, including SYSDATE and SYSTIMESTAMP.

The time zone option differs from the `rdsadmin_util.alter_db_time_zone` command. The `alter_db_time_zone` command changes the time zone only for certain data types. The time zone option changes the time zone for all date columns and values. For more information about `alter_db_time_zone`, see [Setting the Database Time Zone \(p. 857\)](#).

Prerequisites for Time Zone

The time zone option is a permanent and persistent option. You can't remove the option from an option group after you add it. You can't remove the option group from a DB instance after you add it. You can't modify the time zone setting of the option to a different time zone.

We strongly urge you to take a DB snapshot of your DB instance before adding the time zone option to a DB instance. By using a snapshot you can recover the DB instance if you set the time zone option incorrectly. For more information, see [Creating a DB Snapshot \(p. 216\)](#).

We strongly urge you to test the time zone option on a test DB instance before you add it to a production DB instance. Adding the time zone option can cause problems with tables that use system date to add dates or times. You should analyze your data and applications to determine the impact of changing the time zone.

Time Zone Option Settings

Amazon RDS supports the following settings for the time zone option.

Option Setting	Valid Values	Description
TIME_ZONE	One of the available time zones. For the full list, see Available Time Zones (p. 835) .	The new time zone for your DB instance.

Adding the Time Zone Option

The general process for adding the time zone option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

When you add the time zone option, a brief outage occurs while your DB instance is automatically restarted.

AWS Management Console

To add the time zone option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

- a. For **Engine** choose the oracle edition for your DB instance.
- b. For **Major engine version** choose **11.2**, **12.1**, or **12.2** for your DB instance.

For more information, see [Creating an Option Group \(p. 157\)](#).

2. Add the **Timezone** option to the option group, and configure the option settings.

Important

If you add the time zone option to an existing option group that is already attached to one or more DB instances, a brief outage occurs while all the DB instances are automatically restarted.

For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#). For more information about each setting, see [Time Zone Option Settings \(p. 833\)](#).

3. Apply the option group to a new or existing DB instance:

- For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
- For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. When you add the time zone option to an existing DB instance, a brief outage occurs while your DB instance is automatically restarted. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

CLI

The following example uses the AWS CLI `add-option-to-option-group` command to add the `Timezone` option and the `TIME_ZONE` option setting to an option group called `myoptiongroup`. The time zone is set to `Africa/Cairo`.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
--option-group-name "myoptiongroup" \
--options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=Africa/Cairo}]" \
--apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group ^
--option-group-name "myoptiongroup" ^
--options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=Africa/Cairo}]" ^
--apply-immediately
```

Modifying Time Zone Settings

The time zone option is a permanent and persistent option. You can't remove the option from an option group after you add it. You can't remove the option group from a DB instance after you add it. You can't modify the time zone setting of the option to a different time zone. If you set the time zone incorrectly, restore a snapshot of your DB instance from before you added the time zone option.

Removing the Time Zone Option

The time zone option is a permanent and persistent option. You can't remove the option from an option group after you add it. You can't remove the option group from a DB instance after you add it. To remove

the time zone option, restore a snapshot of your DB instance from before you added the time zone option.

Available Time Zones

The following values can be used for the time zone option.

Zone	Time Zone
Africa	Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Lagos, Africa/Luanda, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek
America	America/Araguaina, America/Argentina/Buenos_Aires, America/Asuncion, America/Bogota, America/Caracas, America/Chicago, America/Chihuahua, America/Cuiaba, America/Denver, America/Detroit, America/Fortaleza, America/Godthab, America/Guatemala, America/Halifax, America/Lima, America/Los_Angeles, America/Manaus, America/Matamoros, America/Mexico_City, America/Monterrey, America/Montevideo, America/New_York, America/Phoenix, America/Santiago, America/Sao_Paulo, America/Tijuana, America/Toronto
Asia	Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Hong_Kong, Asia/Irkutsk, Asia/Jakarta, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Kolkata, Asia/Krasnoyarsk, Asia/Magadan, Asia/Manila, Asia/Muscat, Asia/Novosibirsk, Asia/Rangoon, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran, Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan
Atlantic	Atlantic/Azores, Atlantic/Cape_Verde
Australia	Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Eucla, Australia/Hobart, Australia/Lord_Howe, Australia/Perth, Australia/Sydney
Brazil	Brazil/DeNoronha, Brazil/East
Canada	Canada/Newfoundland, Canada/Saskatchewan
Etc	Etc/GMT-3
Europe	Europe/Amsterdam, Europe/Athens, Europe/Berlin, Europe/Dublin, Europe/Helsinki, Europe/Kaliningrad, Europe/London, Europe/Madrid, Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Rome, Europe/Sarajevo
Pacific	Pacific/Apia, Pacific/Auckland, Pacific/Chatham, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Kiritimati, Pacific/Marquesas, Pacific/Samoa, Pacific/Tongatapu, Pacific/Wake
US	US/Alaska, US/Central, US/East-Indiana, US/Eastern, US/Pacific
UTC	UTC

Related Topics

- [Working with Option Groups \(p. 156\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)

Oracle Transparent Data Encryption

Amazon RDS supports Oracle Transparent Data Encryption (TDE), a feature of the Oracle Advanced Security option available in Oracle Enterprise Edition. This feature automatically encrypts data before it is written to storage and automatically decrypts data when the data is read from storage.

Oracle Transparent Data Encryption is used in scenarios where you need to encrypt sensitive data in case data files and backups are obtained by a third party or when you need to address security-related regulatory compliance issues.

Note

You can use the TDE option or AWS CloudHSM Classic, but not both. For more information, see [Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys \(p. 886\)](#).

The TDE option is a permanent option that cannot be removed from an option group, and that option group cannot be removed from a DB instance once it is associated with a DB instance. You cannot disable TDE from a DB instance once that instance is associated with an option group with the Oracle TDE option.

A detailed explanation about Oracle Transparent Data Encryption is beyond the scope of this guide. For information about using Oracle Transparent Data Encryption, see [Securing Stored Data Using Transparent Data Encryption](#). For more information about Oracle Advanced Security, see [Oracle Advanced Security](#) in the Oracle documentation. For more information on AWS security, see the [AWS Security Center](#).

TDE Encryption Modes

Oracle Transparent Data Encryption supports two encryption modes: TDE tablespace encryption and TDE column encryption. TDE tablespace encryption is used to encrypt entire application tables. TDE column encryption is used to encrypt individual data elements that contain sensitive data. You can also apply a hybrid encryption solution that uses both TDE tablespace and column encryption.

Note

Amazon RDS manages the Oracle Wallet and TDE master key for the DB instance. You do not need to set the encryption key using the command `ALTER SYSTEM set encryption key`.

For information about TDE best practices, see [Oracle Advanced Security Transparent Data Encryption Best Practices](#).

Once the option is enabled, you can check the status of the Oracle Wallet by using the following command:

```
SELECT * FROM v$encryption_wallet;
```

To create an encrypted tablespace, use the following command:

```
CREATE TABLESPACE encrypt_ts ENCRYPTION DEFAULT STORAGE (ENCRYPT);
```

To specify the encryption algorithm, use the following command:

```
CREATE TABLESPACE encrypt_ts ENCRYPTION USING 'AES256' DEFAULT STORAGE (ENCRYPT);
```

Note that the previous commands for encrypting a tablespace are the same as the commands you would use with an Oracle installation not on Amazon RDS, and the `ALTER TABLE` syntax to encrypt a column is also the same as the commands you would use for an Oracle installation not on Amazon RDS.

You should determine if your DB instance is associated with an option group that has the **TDE** option. To view the option group that a DB instance is associated with, you can use the RDS console, the [describe-db-instance](#) AWS CLI command, or the API action [DescribeDBInstances](#).

To comply with several security standards, Amazon RDS is working to implement automatic periodic master key rotation.

Adding the TDE Option

The process for using Oracle Transparent Data Encryption (TDE) with Amazon RDS is as follows:

1. If the DB instance is not associated with an option group that has the **TDE** option enabled, you must either create an option group and add the **TDE** option or modify the associated option group to add the **TDE** option. For information about creating or modifying an option group, see [Working with Option Groups \(p. 156\)](#). For information about adding an option to an option group, see [Adding an Option to an Option Group \(p. 160\)](#).
2. Associate the DB instance with the option group with the **TDE** option. For information about associating a DB instance with an option group, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Removing the TDE Option

If you no longer want to use the TDE option with a DB instance, you must decrypt all your data on the DB instance, copy the data to a new DB instance that is not associated with an option group with TDE enabled, and then delete the original instance. You can rename the new instance to be the same name as the previous DB instance if you prefer.

Using TDE with Data Pump

You can use Oracle Data Pump to import or export encrypted dump files. Amazon RDS supports the password encryption mode (ENCRYPTION_MODE=PASSWORD) for Oracle Data Pump. Amazon RDS does not support transparent encryption mode (ENCRYPTION_MODE=TRANSPARENT) for Oracle Data Pump. For more information about using Oracle Data Pump with Amazon RDS, see [Oracle Data Pump \(p. 777\)](#).

Related Topics

- [Working with Option Groups \(p. 156\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)

Oracle UTL_MAIL

Amazon RDS supports Oracle UTL_MAIL through the use of the UTL_MAIL option and SMTP servers. You can send email directly from your database by using the UTL_MAIL package. Amazon RDS supports UTL_MAIL for the following versions of Oracle:

- Oracle version 12.2.0.1, all versions
- Oracle version 12.1.0.2.v5 and later
- Oracle version 11.2.0.4.v9 and later

The following are some limitations to using UTL_MAIL:

- UTL_MAIL does not support Transport Layer Security (TLS) and therefore emails are not encrypted.
- UTL_MAIL does not support authentication with SMTP servers.
- You can only send a single attachment in an email.
- You can't send attachments larger than 32 K.
- You can only use ASCII and Extended Binary Coded Decimal Interchange Code (EBCDIC) character encodings.
- SMTP port (25) is throttled based on the elastic network interface owner's policies.

When you enable UTL_MAIL, only the master user for your DB instance is granted the execute privilege. If necessary, the master user can grant the execute privilege to other users so that they can use UTL_MAIL.

Important

We recommend that you enable Oracle's built-in auditing feature to track the use of UTL_MAIL procedures.

Prerequisites for Oracle UTL_MAIL

The following are prerequisites for using Oracle UTL_MAIL:

- One or more SMTP servers, and the corresponding IP addresses or public or private Domain Name Server (DNS) names. For more information about private DNS names resolved through a custom DNS server, see [Setting Up a Custom DNS Server \(p. 853\)](#).
- For Oracle versions prior to 12c, your DB instance must also use the XML DB option. For more information, see [Oracle XML DB \(p. 840\)](#).

Adding the Oracle UTL_MAIL Option

The general process for adding the Oracle UTL_MAIL option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the UTL_MAIL option, as soon as the option group is active, UTL_MAIL is active.

To add the UTL_MAIL option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

- a. For **Engine**, choose the edition of Oracle you want to use.
- b. For **Major engine version**, choose **11.2 or 12.1**.

For more information, see [Creating an Option Group \(p. 157\)](#).

2. Add the **UTL_MAIL** option to the option group. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).
3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).
 - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Using Oracle UTL_MAIL

After you enable the **UTL_MAIL** option, you must configure the SMTP server before you can begin using it.

You configure the SMTP server by setting the **SMTP_OUT_SERVER** parameter to a valid IP address or public DNS name. For the **SMTP_OUT_SERVER** parameter, you can specify a comma-separated list of the addresses of multiple servers. If the first server is unavailable, **UTL_MAIL** tries the next server, and so on.

You can set the default **SMTP_OUT_SERVER** for a DB instance by using a [DB parameter group](#). You can set the **SMTP_OUT_SERVER** parameter for a session by running the following code on your database on your DB instance.

```
ALTER SESSION SET smtp_out_server = mailserver.domain.com:25;
```

After the **UTL_MAIL** option is enabled, and your **SMTP_OUT_SERVER** is configured, you can send mail by using the **SEND** procedure. For more information, see [UTL_MAIL](#) in the Oracle documentation.

Removing the Oracle UTL_MAIL Option

You can remove Oracle **UTL_MAIL** from a DB instance.

To remove **UTL_MAIL** from a DB instance, do one of the following:

- To remove **UTL_MAIL** from multiple DB instances, remove the **UTL_MAIL** option from the option group they belong to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#).
- To remove **UTL_MAIL** from a single DB instance, modify the DB instance and specify a different option group that doesn't include the **UTL_MAIL** option. You can specify the default (empty) option group, or a different custom option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Troubleshooting

The following are issues you might encounter when you use **UTL_MAIL** with Amazon RDS.

- Throttling. SMTP port (25) is throttled based on the elastic network interface owner's policies. If you can successfully send email by using **UTL_MAIL**, and you see the error ORA-29278: **SMTP transient**

error: 421 Service not available, you are possibly being throttled. If you experience throttling with email delivery, we recommend that you implement a backoff algorithm. For more information about backoff algorithms, see [Error Retries and Exponential Backoff in AWS](#) and [How to handle a "Throttling – Maximum sending rate exceeded" error](#).

You can request that this throttle be removed. For more information, see [How do I remove the throttle on port 25 from my EC2 instance?](#)

Related Topics

- [Working with Option Groups \(p. 156\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)

Oracle XML DB

Oracle XML DB adds native XML support to your DB instance. With XML DB, you can store and retrieve structured or unstructured XML, in addition to relational data.

XML DB is pre-installed on Oracle version 12c and later. Amazon RDS supports Oracle XML DB for version 11g through the use of the XMLDB option. After you apply the XMLDB option to your DB instance, you have full access to the Oracle XML DB repository; no post-installation tasks are required.

Note

The Amazon RDS XMLDB option does not provide support for the Oracle XML DB Protocol Server.

Adding the Oracle XML DB Option

The general process for adding the Oracle XML DB option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the XML DB option, as soon as the option group is active, XML DB is active.

To add the XML DB option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
 - a. For **Engine**, choose the edition of Oracle you want to use.
 - b. For **Major engine version**, choose **11.2**.

For more information, see [Creating an Option Group \(p. 157\)](#).

2. Add the **XMLDB** option to the option group. For more information about adding options, see [Adding an Option to an Option Group \(p. 160\)](#).
3. Apply the option group to a new or existing DB instance:
 - For a new DB instance, you apply the option group when you launch the instance. For more information, see [Creating a DB Instance Running the Oracle Database Engine \(p. 742\)](#).

- For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Removing the Oracle XML DB Option

You can remove the XML DB option from a DB instance running version 11g.

To remove the XML DB option from a DB instance running version 11g, do one of the following:

- To remove the XMLDB option from multiple DB instances, remove the XMLDB option from the option group they belong to. This change affects all DB instances that use the option group. For more information, see [Removing an Option from an Option Group \(p. 167\)](#).
- To remove the XMLDB option from a single DB instance, modify the DB instance and specify a different option group that doesn't include the XMLDB option. You can specify the default (empty) option group, or a different custom option group. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Common DBA Tasks for Oracle DB Instances

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the Oracle database engine. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

The following are common DBA tasks for DB instances running Oracle:

- [System Tasks \(p. 845\)](#)

Disconnecting a Session (p. 845)	Amazon RDS method: <code>disconnect</code> Oracle method: <code>alter system disconnect session</code>
Killing a Session (p. 845)	Amazon RDS method: <code>kill</code> Oracle method: <code>alter system kill session</code>
Enabling and Disabling Restricted Sessions (p. 846)	Amazon RDS method: <code>restricted_session</code> Oracle method: <code>alter system enable restricted session</code>
Flushing the Shared Pool (p. 847)	Amazon RDS method: <code>flush_shared_pool</code> Oracle method: <code>alter system flush shared_pool</code>
Flushing the Buffer Cache (p. 847)	Amazon RDS method: <code>flush_buffer_cache</code> Oracle method: <code>alter system flush buffer_cache</code>
Granting SELECT or EXECUTE Privileges to SYS Objects (p. 847)	Amazon RDS method: <code>grant_sys_object</code> Oracle method: <code>grant</code>
Revoking SELECT or EXECUTE Privileges on SYS Objects (p. 849)	Amazon RDS method: <code>revoke_sys_object</code> Oracle method: <code>revoke</code>
Granting Privileges to Non-Master Users (p. 849)	Amazon RDS method: <code>grant</code> Oracle method: <code>grant</code>
Modifying DBMS_SCHEDULER Jobs (p. 850)	Amazon RDS method: <code>dbms_scheduler.set_attribute</code>

	Oracle method: <code>dbms_scheduler.set_attribute</code>
Creating Custom Functions to Verify Passwords (p. 850)	Amazon RDS method: <code>create_verify_function</code> Amazon RDS method: <code>create_passthrough_verify_fcn</code>
Setting Up a Custom DNS Server (p. 853)	—

- [Database Tasks \(p. 854\)](#)

Changing the Global Name of a Database (p. 855)	Amazon RDS method: <code>rename_global_name</code> Oracle method: <code>alter database rename</code>
Creating and Sizing Tablespaces (p. 855)	Amazon RDS method: <code>create tablespace</code> Oracle method: <code>alter database</code>
Setting the Default Tablespace (p. 856)	Amazon RDS method: <code>alter_default_tablespace</code> Oracle method: <code>alter database default tablespace</code>
Setting the Default Temporary Tablespace (p. 856)	Amazon RDS method: <code>alter_default_temp_tablespace</code> Oracle method: <code>alter database default temporary tablespace</code>
Checkpointing the Database (p. 856)	Amazon RDS method: <code>checkpoint</code> Oracle method: <code>alter system checkpoint</code>
Setting Distributed Recovery (p. 856)	Amazon RDS method: <code>enable_distr_recovery</code> Oracle method: <code>alter system enable distributed recovery</code>
Setting the Database Time Zone (p. 857)	Amazon RDS method: <code>alter_db_time_zone</code> Oracle method: <code>alter database set time_zone</code>

Working with Automatic Workload Repository (AWR) (p. 858)	—
Adjusting Database Links for Use with DB Instances in a VPC (p. 859)	—

- [Log Tasks \(p. 866\)](#)

Setting Force Logging (p. 866)	Amazon RDS method: <code>force_logging</code> Oracle method: <code>alter database force logging</code>
Setting Supplemental Logging (p. 867)	Amazon RDS method: <code>alter_supplemental_logging</code> Oracle method: <code>alter database add supplemental log</code>
Switching Online Log Files (p. 868)	Amazon RDS method: <code>switch_logfile</code> Oracle method: <code>alter system switch logfile</code>
Adding Online Redo Logs (p. 868)	Amazon RDS method: <code>add_logfile</code>
Dropping Online Redo Logs (p. 868)	Amazon RDS method: <code>drop_logfile</code>
Resizing Online Redo Logs (p. 869)	—
Retaining Archived Redo Logs (p. 871)	Amazon RDS method: <code>set_configuration</code>
Accessing Transaction Logs (p. 872)	Amazon RDS method: <code>create_archivelog_dir</code> Amazon RDS method: <code>create_onlinelog_dir</code>

- [Miscellaneous Tasks \(p. 873\)](#)

Creating New Directories in the Main Data Storage Space (p. 873)	Amazon RDS method: <code>create_directory</code> Oracle method: <code>create directory</code>
Listing Files in a DB Instance Directory (p. 874)	Amazon RDS method: <code>listdir</code> Oracle method: —

[Reading Files in a DB Instance Directory \(p. 874\)](#)

Amazon RDS method:
`read_text_file`

Oracle method: —

Common DBA System Tasks for Oracle DB Instances

This section describes how you can perform common DBA tasks related to the system on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

Disconnecting a Session

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.disconnect` to disconnect the current session by ending the dedicated server process. The `disconnect` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>sid</code>	number	—	required	The session identifier.
<code>serial</code>	number	—	required	The serial number of the session.
<code>method</code>	varchar	'IMMEDIATE'	optional	Valid values are ' <code>IMMEDIATE</code> ' or ' <code>POST_TRANSACTION</code> '.

The following example disconnects a session:

```
begin
    rdsadmin.rdsadmin_util.disconnect(
        sid    => sid,
        serial => serial_number);
end;
/
```

To get the session identifier and the session serial number, query the `V$SESSION` view. The following example gets all sessions for the user `AWSUSER`:

```
select SID, SERIAL#, STATUS from V$SESSION where USERNAME = 'AWSUSER';
```

The database must be open to use this method. For more information about disconnecting a session, see [ALTER SYSTEM](#) in the Oracle documentation.

Killing a Session

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.kill` to kill a session. The `kill` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>sid</code>	number	—	required	The session identifier.

Parameter Name	Data Type	Default	Required	Description
serial	number	—	required	The serial number of the session.
method	varchar	null	optional	Valid values are ' <code>IMMEDIATE</code> ' or ' <code>PROCESS</code> '.

The following example kills a session:

```
begin
    rdsadmin.rdsadmin_util.kill(
        sid    => sid,
        serial => serial_number);
end;
/
```

To get the session identifier and the session serial number, query the `V$SESSION` view. The following example gets all sessions for the user `AWSUSER`:

```
select SID, SERIAL#, STATUS from V$SESSION where USERNAME = 'AWSUSER';
```

You can specify either `IMMEDIATE` or `PROCESS` as a value for the `method` parameter. Specifying `PROCESS` as the enables you to kill the processes associated with a session. You should only do this if killing the session using `IMMEDIATE` as the `method` value was unsuccessful.

Enabling and Disabling Restricted Sessions

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.restricted_session` to enable and disable restricted sessions. The `restricted_session` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_enable</code>	boolean	true	optional	Set to <code>true</code> to enable restricted sessions, <code>false</code> to disable restricted sessions.

The following example shows how to enable and disable restricted sessions.

```
/* Verify that the database is currently unrestricted. */

select LOGINS from V$INSTANCE;

LOGINS
-----
ALLOWED

/* Enable restricted sessions */

exec rdsadmin.rdsadmin_util.restricted_session(p_enable => true);

/* Verify that the database is now restricted. */
```

```

select LOGINS from V$INSTANCE;

LOGINS
-----
RESTRICTED

/* Disable restricted sessions */

exec rdsadmin.rdsadmin_util.restricted_session(p_enable => false);

/* Verify that the database is now unrestricted again. */

select LOGINS from V$INSTANCE;

LOGINS
-----
ALLOWED

```

Flushing the Shared Pool

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.flush_shared_pool` to flush the shared pool. The `flush_shared_pool` procedure has no parameters.

The following example flushes the shared pool.

```
exec rdsadmin.rdsadmin_util.flush_shared_pool;
```

Flushing the Buffer Cache

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.flush_buffer_cache` to flush the buffer cache. The `flush_buffer_cache` procedure has no parameters.

The following example flushes the buffer cache.

```
exec rdsadmin.rdsadmin_util.flush_buffer_cache;
```

Granting SELECT or EXECUTE Privileges to SYS Objects

Usually you transfer privileges by using roles, which can contain many objects. You can grant privileges to a single object by using the Amazon RDS procedure `rdsadmin.rdsadmin_util.grant_sys_object`. The procedure only grants privileges that the master account already has via a role or direct grant.

The `grant_sys_object` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_obj_name</code>	varchar2	—	required	The name of the object to grant privileges for. The object can be a directory, function, package, procedure, sequence, table, or view. Object names must be spelled exactly as they

Parameter Name	Data Type	Default	Required	Description
				appear in DBA_OBJECTS. Most system objects are defined in upper case, so we recommend you try that first.
p_grantee	varchar2	—	required	The name of the object to grant privileges to. The object can be a schema or a role.
p_privilege	varchar2	null	required	—
p_grant_option	boolean	false	optional	Set to true to use the with grant option. The p_grant_option parameter is supported for Oracle versions 11.2.0.4.v8 and later, and 12.1.0.2.v4 and later.

The following example grants select privileges on an object named V_\$SESSION to a user named USER1:

```
begin
    rdsadmin.rdsadmin_util.grant_sys_object(
        p_obj_name  => 'V_$SESSION',
        p_grantee   => 'USER1',
        p_privilege => 'SELECT');
end;
/
```

The following example grants select privileges on an object named V_\$SESSION to a user named USER1 with the grant option:

```
begin
    rdsadmin.rdsadmin_util.grant_sys_object(
        p_obj_name      => 'V_$SESSION',
        p_grantee       => 'USER1',
        p_privilege     => 'SELECT',
        p_grant_option => true);
end;
/
```

To be able to grant privileges on an object, your account must have those privileges granted to it directly with the grant option, or via a role granted using with admin option. In the most common case, you may want to grant SELECT on a DBA view that has been granted to the SELECT_CATALOG_ROLE role. If that role isn't already directly granted to your user using with admin option, then you won't be able to transfer the privilege. If you have the DBA privilege, then you can grant the role directly to another user.

The following example grants the SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE to USER1. Since the with admin option is used, USER1 can now grant access to SYS objects that have been granted to SELECT_CATALOG_ROLE.

```
grant SELECT_CATALOG_ROLE to USER1 with admin option;
grant EXECUTE_CATALOG_ROLE to USER1 with admin option;
```

Objects already granted to `PUBLIC` do not need to be re-granted. If you use the `grant_sys_object` procedure to re-grant access, the procedure call succeeds.

Revoking SELECT or EXECUTE Privileges on SYS Objects

You can revoke privileges on a single object by using the Amazon RDS procedure `rdsadmin.rdsadmin_util.revoke_sys_object`. The procedure only revokes privileges that the master account already has via a role or direct grant.

The `revoke_sys_object` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_obj_name</code>	varchar2	—	required	The name of the object to revoke privileges for. The object can be a directory, function, package, procedure, sequence, table, or view. Object names must be spelled exactly as they appear in <code>DBA_OBJECTS</code> . Most system objects are defined in upper case, so we recommend you try that first.
<code>p_revokee</code>	varchar2	—	required	The name of the object to revoke privileges for. The object can be a schema or a role.
<code>p_privilege</code>	varchar2	null	required	—

The following example revokes select privileges on an object named `V_$SESSION` from a user named `USER1`:

```
begin
    rdsadmin.rdsadmin_util.revoke_sys_object(
        p_obj_name  => 'V_$SESSION',
        p_revokee   => 'USER1',
        p_privilege => 'SELECT');
end;
/
```

Granting Privileges to Non-Master Users

You can grant select privileges for many objects in the `SYS` schema by using the `SELECT_CATALOG_ROLE` role. The `SELECT_CATALOG_ROLE` role gives users `SELECT` privileges on data dictionary views. The following example grants the role `SELECT_CATALOG_ROLE` to a user named `user1`.

```
grant SELECT_CATALOG_ROLE to user1;
```

You can grant execute privileges for many objects in the `SYS` schema by using the `EXECUTE_CATALOG_ROLE` role. The `EXECUTE_CATALOG_ROLE` role gives users `EXECUTE` privileges

for packages and procedures in the data dictionary. The following example grants the role EXECUTE_CATALOG_ROLE to a user named *user1*:

```
grant EXECUTE_CATALOG_ROLE to user1;
```

The following example gets the permissions that the roles SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE allow:

```
select *
  from ROLE_TAB_PRIVS
 where ROLE in ('SELECT_CATALOG_ROLE', 'EXECUTE_CATALOG_ROLE')
order by ROLE, TABLE_NAME asc;
```

The following example creates a non-master user named *user1*, grants the CREATE SESSION privilege, and grants the SELECT privilege on a database named *sh.sales*:

```
create user user1 identified by password;
grant CREATE SESSION to user1;
grant SELECT on sh.sales TO user1;
```

Modifying DBMS_SCHEDULER Jobs

You can use the Oracle procedure dbms_scheduler.set_attribute to modify DBMS_SCHEDULER jobs. For more information, see [DBMS_SCHEDULER](#) and [SET_ATTRIBUTE Procedure](#) in the Oracle documentation.

When working with Amazon RDS DB instances, prepend the schema name **SYS** to the object name. The following example sets the resource plan attribute for the Monday window object.

```
begin
    dbms_scheduler.set_attribute(
        name      => 'SYS.MONDAY_WINDOW',
        attribute => 'RESOURCE_PLAN',
        value     => 'resource_plan_1');
end;
/
```

Creating Custom Functions to Verify Passwords

You can create a custom password verification function in two ways. If you want to use standard verification logic, and to store your function in the **SYS** schema, use the `create_verify_function` procedure. If you want to use custom verification logic, or you don't want to store your function in the **SYS** schema, use the `create_passthrough_verify_fcn` procedure.

The `create_verify_function` Procedure

The `create_verify_function` procedure is supported for Oracle version 11.2.0.4.v9 and later, and 12.1.0.2.v5 and later.

You can create a custom function to verify passwords by using the Amazon RDS procedure `rdsadmin.rdsadmin_password_verify.create_verify_function`. The `create_verify_function` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_verify_function_name</code>	varchar2	—	required	The name for your custom function. This function is

Parameter Name	Data Type	Default	Required	Description
				created for you in the SYS schema. You assign this function to user profiles.
p_min_length	number	8	optional	The minimum number of characters required.
p_max_length	number	256	optional	The maximum number of characters allowed.
p_min_letters	number	1	optional	The minimum number of letters required.
p_min_uppercase	number	0	optional	The minimum number of uppercase letters required.
p_min_lowercase	number	0	optional	The minimum number of lowercase letters required.
p_min_digits	number	1	optional	The minimum number of digits required.
p_min_special	number	0	optional	The minimum number of special characters required.
p_min_different_chars	number	3	optional	The minimum number of distinct characters required.
p_disallow_username	boolean	true	optional	Set to true to disallow the username in the password.
p_disallow_reverse	boolean	true	optional	Set to true to disallow the reverse of the username in the password.
p_disallow_db_name	boolean	true	optional	Set to true to disallow the database or server name in the password.
p_disallow_simple_string	boolean	true	optional	Set to true to disallow simple strings as the password.
p_disallow_whitespace	boolean	false	optional	Set to true to disallow white space characters in the password.
p_disallow_at_sign	boolean	false	optional	Set to true to disallow the @ character in the password.

You can create multiple password verification functions.

There are restrictions on the name of your custom function. Your custom function can't have the same name as an existing system object, the name can be no more than 30 characters long, and the name must include one of the following strings: PASSWORD, VERIFY, COMPLEXITY, ENFORCE, or STRENGTH.

The following example creates a function named `CUSTOM_PASSWORD_FUNCTION`. The function requires that a password has at least 12 characters, 2 uppercase characters, 1 digit, and 1 special character, and that the password disallows the @ character.

```
begin
    rdsadmin.rdsadmin_password_verify.create_verify_function(
        p_verify_function_name => 'CUSTOM_PASSWORD_FUNCTION',
        p_min_length          => 12,
        p_min_uppercase       => 2,
        p_min_digits          => 1,
        p_min_special         => 1,
        p_disallow_at_sign   => true);
end;
/
```

To see the text of your verification function, query `DBA_SOURCE`. The following example gets the text of a custom password function named `CUSTOM_PASSWORD_FUNCTION`.

```
col text format a150

select TEXT
  from DBA_SOURCE
 where OWNER = 'SYS' and NAME = 'CUSTOM_PASSWORD_FUNCTION'
order by LINE;
```

To associate your verification function with a user profile, use `alter profile`. The following example associates a verification function with the `DEFAULT` user profile.

```
alter profile DEFAULT limit PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

To see what user profiles are associated with what verification functions, query `DBA_PROFILES`. The following example gets the profiles that are associated with the custom verification function named `CUSTOM_PASSWORD_FUNCTION`.

```
select *
  from DBA_PROFILES
 where RESOURCE = 'PASSWORD' and LIMIT = 'CUSTOM_PASSWORD_FUNCTION';

PROFILE          RESOURCE_NAME          RESOURCE  LIMIT
-----          -----
-----          -----
DEFAULT          PASSWORD_VERIFY_FUNCTION          PASSWORD
```

The following example gets all profiles and the password verification functions that they are associated with.

```
select *
  from DBA_PROFILES
 where RESOURCE_NAME = 'PASSWORD_VERIFY_FUNCTION';

PROFILE          RESOURCE_NAME          RESOURCE  LIMIT
-----          -----
-----          -----
DEFAULT          PASSWORD_VERIFY_FUNCTION          PASSWORD
CUSTOM_PASSWORD_FUNCTION
RDSADMIN          PASSWORD_VERIFY_FUNCTION          PASSWORD  NULL
```

The `create_passthrough_verify_fcn` Procedure

The `create_passthrough_verify_fcn` procedure is supported for Oracle version 11.2.0.4.v11 and later, and 12.1.0.2.v7 and later.

You can create a custom function to verify passwords by using the Amazon RDS procedure `rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn`. The `create_passthrough_verify_fcn` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_verify_function_name</code>	varchar2	—	required	The name for your custom verification function. This is a wrapper function that is created for you in the SYS schema, and it doesn't contain any verification logic. You assign this function to user profiles.
<code>p_target_owner</code>	varchar2	—	required	The schema owner for your custom verification function.
<code>p_target_function_name</code>	varchar2	—	required	The name of your existing custom function that contains the verification logic. Your custom function must return a boolean. Your function should return <code>true</code> if the password is valid and <code>false</code> if the password is invalid.

The following example creates a password verification function that uses the logic from the function named `PASSWORD_LOGIC_EXTRA_STRONG`.

```
begin
    rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn(
        p_verify_function_name => 'CUSTOM_PASSWORD_FUNCTION',
        p_target_owner         => 'TEST_USER',
        p_target_function_name => 'PASSWORD_LOGIC_EXTRA_STRONG');
end;
/
```

To associate the verification function with a user profile, use `alter profile`. The following example associates the verification function with the `DEFAULT` user profile.

```
alter profile DEFAULT limit PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

Setting Up a Custom DNS Server

Amazon RDS supports outbound network access on your DB instances running Oracle. For more information about outbound network access, including prerequisites, see [Using `utl_http`, `utl_tcp`, and `utl_smtp` with an Oracle DB Instance \(p. 738\)](#).

Amazon RDS Oracle allows Domain Name Service (DNS) resolution from a custom DNS server owned by the customer. You can resolve only fully qualified domain names from your Amazon RDS DB instance through your custom DNS server.

After you set up your custom DNS name server, it takes up to 30 minutes to propagate the changes to your DB instance. After the changes are propagated to your DB instance, all outbound network traffic requiring a DNS lookup queries your DNS server over port 53.

To set up a custom DNS server for your Oracle Amazon RDS DB instance, do the following:

- From the DHCP options set attached to your VPC, set the `domain-name-servers` option to the IP address of your DNS name server. For more information, see [DHCP Options Sets](#).

Note

The `domain-name-servers` option accepts up to four values, but your Amazon RDS DB instance uses only the first value.

- Ensure that your DNS server can resolve all lookup queries, including public DNS names, Amazon EC2 private DNS names, and customer-specific DNS names. If the outbound network traffic contains any DNS lookups that your DNS server can't handle, your DNS server must have appropriate upstream DNS providers configured.
- Configure your DNS server to produce User Datagram Protocol (UDP) responses of 512 bytes or less.
- Configure your DNS server to produce Transmission Control Protocol (TCP) responses of 1024 bytes or less.
- Configure your DNS server to allow inbound traffic from your Amazon RDS DB instances over port 53. If your DNS server is in an Amazon VPC, the VPC must have a security group that contains inbound rules that allow UDP and TCP traffic on port 53. If your DNS server is not in an Amazon VPC, it must have appropriate firewall whitelisting to allow UDP and TCP inbound traffic on port 53.

For more information, see [Security Groups for Your VPC](#) and [Adding and Removing Rules](#).

- Configure the VPC of your Amazon RDS DB instance to allow outbound traffic over port 53. Your VPC must have a security group that contains outbound rules that allow UDP and TCP traffic on port 53.

For more information, see [Security Groups for Your VPC](#) and [Adding and Removing Rules](#).

- The routing path between the Amazon RDS DB instance and the DNS server has to be configured correctly to allow DNS traffic.
 - If the Amazon RDS DB instance and the DNS server are not in the same VPC, a peering connection has to be setup between them. For more information, see [What is VPC Peering?](#)

Related Topics

- [Common DBA Database Tasks for Oracle DB Instances \(p. 854\)](#)
- [Common DBA Log Tasks for Oracle DB Instances \(p. 866\)](#)
- [Common DBA Miscellaneous Tasks for Oracle DB Instances \(p. 873\)](#)

Common DBA Database Tasks for Oracle DB Instances

This section describes how you can perform common DBA tasks related to databases on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

Changing the Global Name of a Database

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.rename_global_name` to change the global name of a database. The `rename_global_name` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_new_global_name</code>	varchar2	—	required	The new global name for the database.

The database must be open for the name change to occur. For more information about changing the global name of a database, see [ALTER DATABASE](#) in the Oracle documentation.

The following example changes the global name of a database to `new_global_name`.

```
exec rdsadmin.rdsadmin_util.rename_global_name(p_new_global_name => 'new_global_name');
```

Creating and Sizing Tablespaces

Amazon RDS only supports Oracle Managed Files (OMF) for data files, log files and control files. When you create data files and log files, you can't specify the physical file names.

By default, tablespaces are created with auto-extend enabled, and no maximum size. Because of these default settings, tablespaces can grow to consume all allocated storage. We recommend that you specify an appropriate maximum size on permanent and temporary tablespaces, and that you carefully monitor space usage.

The following example creates a tablespace named `users2` with a starting size of 1 gigabyte and a maximum size of 10 gigabytes:

```
create tablespace users2 datafile size 1G autoextend on maxsize 10G;
```

The following example creates temporary tablespace named `temp01`:

```
create temporary tablespace temp01;
```

The Oracle `ALTER DATABASE` system privilege is not available on Amazon RDS. We recommend that you don't use smallfile tablespaces, because you can only perform some operations, such as resizing existing datafiles, by using the `ALTER DATABASE` statement.

You can resize a bigfile tablespace by using `ALTER TABLESPACE`. You can specify the size in kilobytes (K), megabytes (M), gigabytes (G), or terabytes (T).

The following example resizes a bigfile tablespace named `users2` to 200 MB:

```
alter tablespace users2 resize 200M;
```

The following example adds an additional datafile to a smallfile tablespace named `users2`:

```
alter tablespace users2 add datafile size 100000M autoextend on next 250m maxsize UNLIMITED;
```

Setting the Default Tablespace

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.alter_default_tablespace` to set the default tablespace. The `alter_default_tablespace` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>tablespace_name</code>	<code>varchar</code>	—	required	The name of the default tablespace.

The following example sets the default tablespace to `users2`:

```
exec rdsadmin.rdsadmin_util.alter_default_tablespace(tablespace_name => 'users2');
```

Setting the Default Temporary Tablespace

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.alter_default_temp_tablespace` to set the default temporary tablespace. The `alter_default_temp_tablespace` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>tablespace_name</code>	<code>varchar</code>	—	required	The name of the default temporary tablespace.

The following example sets the default temporary tablespace to `temp01`:

```
exec rdsadmin.rdsadmin_util.alter_default_temp_tablespace(tablespace_name => 'temp01');
```

Checkpointing the Database

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.checkpoint` to checkpoint the database. The `checkpoint` procedure has no parameters.

The following example checkpoints the database:

```
exec rdsadmin.rdsadmin_util.checkpoint;
```

Setting Distributed Recovery

You can use the Amazon RDS procedures `rdsadmin.rdsadmin_util.enable_distr_recovery` and `disable_distr_recovery` to set distributed recovery. The procedures have no parameters.

The following example enables distributed recovery:

```
exec rdsadmin.rdsadmin_util.enable_distr_recovery;
```

The following example disables distributed recovery:

```
exec rdsadmin.rdsadmin_util.disable_distr_recovery;
```

Setting the Database Time Zone

There are two different ways that you can set the time zone of your Amazon RDS Oracle database:

- You can use the `Timezone` option.

The `Timezone` option changes the time zone at the host level and impacts all date columns and values such as `SYSDATE`. For more information about the `Timezone` option, see [Oracle Time Zone \(p. 833\)](#).

- You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.alter_db_time_zone`.

The `alter_db_time_zone` procedure changes the time zone for only certain data types, and doesn't change `SYSDATE`. There are additional restrictions on setting the time zone listed in the [Oracle documentation](#).

The `alter_db_time_zone` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_new_tz</code>	<code>varchar2</code>	—	required	The new time zone as a named region or an absolute offset from Coordinated Universal Time (UTC). Valid offsets range from -12:00 to +14:00.

The following example changes the time zone to UTC plus 3 hours:

```
exec rdsadmin.rdsadmin_util.alter_db_time_zone(p_new_tz => '+3:00');
```

The following example changes the time zone to the time zone of the Africa/Algiers region:

```
exec rdsadmin.rdsadmin_util.alter_db_time_zone(p_new_tz => 'Africa/Algiers');
```

After you alter the time zone by using the `alter_db_time_zone` procedure, you must reboot the DB instance for the change to take effect. For more information, see [Rebooting a DB Instance \(p. 129\)](#).

Working with Oracle External Tables

Oracle external tables are tables with data that is not in the database. Instead, the data is in external files that the database can access. By using external tables, you can access data without loading it into the database. For more information about external tables, see [Managing External Tables](#) in the Oracle documentation.

With Amazon RDS, you can store external table files in directory objects. You can create a directory object, or you can use one that is predefined in the Oracle database, such as the `DATA_PUMP_DIR` directory. For information about creating directory objects, see [Creating New Directories in the Main Data Storage Space \(p. 873\)](#). You can query the `ALL_DIRECTORIES` view to list the directory objects for your Amazon RDS Oracle DB instance.

Note

Directory objects point to the main data storage space (Amazon EBS volume) used by your instance. The space used—along with data files, redo logs, audit, trace, and other files—counts against allocated storage.

You can move an external data file from one Oracle database to another by using the [DBMS_FILE_TRANSFER](#) package or the [UTL_FILE](#) package. The external data file is moved from a directory on the source database to the specified directory on the destination database. For information about using DBMS_FILE_TRANSFER, see [Oracle Data Pump \(p. 777\)](#).

After you move the external data file, you can create an external table with it. The following example creates an external table that uses the `emp_xt_file1.txt` file in the `USER_DIR1` directory:

```
CREATE TABLE emp_xt (
    emp_id      NUMBER,
    first_name  VARCHAR2(50),
    last_name   VARCHAR2(50),
    user_name   VARCHAR2(20)
)
ORGANIZATION EXTERNAL (
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY USER_DIR1
    ACCESS PARAMETERS (
        RECORDS DELIMITED BY NEWLINE
        FIELDS TERMINATED BY ','
        MISSING FIELD VALUES ARE NULL
        (emp_id,first_name,last_name,user_name)
    )
    LOCATION ('emp_xt_file1.txt')
)
PARALLEL
REJECT LIMIT UNLIMITED;
```

Suppose that you want to move data that is in an Amazon RDS Oracle DB instance into an external data file. In this case, you can populate the external data file by creating an external table and selecting the data from the table in the database. For example, the following SQL statement creates the `orders_xt` external table by querying the `orders` table in the database.

```
CREATE TABLE orders_xt
ORGANIZATION EXTERNAL
(
    TYPE ORACLE_DATAPUMP
    DEFAULT DIRECTORY DATA_PUMP_DIR
    LOCATION ('orders_xt.dmp')
)
AS SELECT * FROM orders;
```

In this example, the data is populated in the `orders_xt.dmp` file in the `DATA_PUMP_DIR` directory.

Working with Automatic Workload Repository (AWR)

If you use Oracle Database Enterprise Edition and want to use Automatic Workload Repository (AWR), you can enable AWR by changing the `CONTROL_MANAGEMENT_PACK_ACCESS` parameter.

Oracle AWR includes several report generation scripts, such as `awrrpt.sql`, that are installed on the host server. You do not have direct access to the host, but you can copy the scripts from another installation of Oracle Database.

Adjusting Database Links for Use with DB Instances in a VPC

To use Oracle database links with Amazon RDS DB instances inside the same VPC or peered VPCs, the two DB instances should have a valid route between them. Verify the valid route between the DB instances by using your VPC routing tables and network access control list (ACL).

The security group of each DB instance must allow ingress to and egress from the other DB instance. The inbound and outbound rules can refer to security groups from the same VPC or a peered VPC. For more information, see [Updating Your Security Groups to Reference Peered VPC Security Groups](#).

If you have configured a custom DNS server using the DHCP Option Sets in your VPC, your custom DNS server must be able to resolve the name of the database link target. For more information, see [Setting Up a Custom DNS Server \(p. 853\)](#).

For more information about using database links with Oracle Data Pump, see [Oracle Data Pump \(p. 777\)](#).

Setting the Default Edition for a DB Instance

You can redefine database objects in a private environment called an edition. You can use edition-based redefinition to upgrade an application's database objects with minimal downtime.

You can set the default edition of an Amazon RDS Oracle DB instance using the Amazon RDS procedure `rdsadmin.rdsadmin_util.alter_default_edition`.

The following example sets the default edition for the Amazon RDS Oracle DB instance to `RELEASE_V1`.

```
exec rdsadmin.rdsadmin_util.alter_default_edition('RELEASE_V1');
```

The following example sets the default edition for the Amazon RDS Oracle DB instance back to the Oracle default.

```
exec rdsadmin.rdsadmin_util.alter_default_edition('ORA$BASE');
```

For more information about Oracle edition-based redefinition, see [About Editions and Edition-Based Redefinition](#) in the Oracle documentation.

Validating DB Instance Files

You can use the Amazon RDS package `rdsadmin.rdsadmin_rman_util` to validate Amazon RDS Oracle DB instance files, such as data files, server parameter files (SPFILEs), and control files.

Note

The `rdsadmin.rdsadmin_rman_util` package provides capabilities that are available with Oracle Recovery Manager (RMAN) validation. While Amazon RDS does not use RMAN for backups, you can use the package to execute RMAN validation commands against the database, control file, SPFILE, tablespaces, or data files. For more information about RMAN validation, see [Validating Database Files and Backups](#) and `VALIDATE` in the Oracle documentation.

Validating a DB Instance

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_database` to validate all of the relevant files used by an Amazon RDS Oracle DB instance.

Parameter Name	Data Type	Valid Values	Default	Required	Description
<code>p_validation_type</code>	varchar2	'PHYSICAL','PHYSICAL+LOGICAL'	'Optional'		The level of corruption detection.

Parameter Name	Data Type	Valid Values	Default	Required	Description
					<p>Specify 'PHYSICAL' to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer.</p> <p>Specify 'PHYSICAL+LOGICAL' to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.</p>
p_parallel	number	A valid integer between 1 and 254 for Oracle Database Enterprise Edition (EE) 1 for other Oracle Database editions	1	Optional	Number of channels.
p_section_size_mb	number	A valid integer	NULL	Optional	<p>The section size in megabytes (MB).</p> <p>Validates in parallel by dividing each file into the specified section size.</p> <p>When NULL, the parameter is ignored.</p>

Parameter Name	Data Type	Valid Values	Default	Required	Description
p_rman_to_dbms_output	boolean	TRUE, FALSE	FALSE	Optional	<p>When TRUE, the RMAN output is sent to the DBMS_OUTPUT package in addition to a file in the BDUMP directory. When using SQL*Plus, execute SET SERVEROUTPUT ON to see the output.</p> <p>When FALSE, the RMAN output is only sent to a file in the BDUMP directory.</p>

The following example validates the DB instance using the default values for the parameters:

```
exec rdsadmin.rdsadmin_rman_util.validate_database;
```

The following example validates the DB instance using the specified values for the parameters:

```
BEGIN
    rdsadmin.rdsadmin_rman_util.validate_database(
        p_validation_type      => 'PHYSICAL+LOGICAL',
        p_parallel             => 4,
        p_section_size_mb     => 10,
        p_rman_to_dbms_output => FALSE);
END;
/
```

When the p_rman_to_dbms_output parameter is set to FALSE, the RMAN output is written to a file in the BDUMP directory.

To view the files in the BDUMP directory, run the following SELECT statement:

```
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP')) order by mtime;
```

To view the contents of a file in the BDUMP directory, run the following SELECT statement:

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP', 'rds-rman-validate-nnn.txt'));
```

Replace the file name with the name of the file you want to view.

Validating a Tablespace

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_tablespace` to validate the files associated with a tablespace.

Parameter Name	Data Type	Valid Values	Default	Required	Description
<code>p_tablespace_name</code>	varchar2	A valid tablespace name	—	Required	The name of the tablespace.
<code>p_validation_type</code>	varchar2	'PHYSICAL' 'PHYSICAL+LOGICAL'	'PHYSICAL'	Optional	<p>The level of corruption detection.</p> <p>Specify 'PHYSICAL' to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer.</p> <p>Specify 'PHYSICAL+LOGICAL' to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.</p>
<code>p_parallel</code>	number	A valid integer between 1 and 254 for Oracle Database Enterprise Edition (EE) 1 for other Oracle Database editions	1	Optional	Number of channels.
<code>p_section_size_mb</code>	number	A valid integer	NULL	Optional	<p>The section size in megabytes (MB).</p> <p>Validates in parallel by dividing each file into the specified section size.</p> <p>When NULL, the parameter is ignored.</p>

Parameter Name	Data Type	Valid Values	Default	Required	Description
p_rman_to_dbms_output	boolean	TRUE, FALSE	FALSE	Optional	<p>When TRUE, the RMAN output is sent to the DBMS_OUTPUT package in addition to a file in the BDUMP directory. When using SQL*Plus, execute SET SERVEROUTPUT ON to see the output.</p> <p>When FALSE, the RMAN output is only sent to a file in the BDUMP directory.</p>

Validating a Control File

You can use the Amazon RDS procedure

`rdsadmin.rdsadmin_rman_util.validate_current_controlfile` to validate only the control file used by an Amazon RDS Oracle DB instance.

Parameter Name	Data Type	Valid Values	Default	Required	Description
p_validation_type	varchar2	'PHYSICAL' 'PHYSICAL+LOGICAL'	'PHYSICAL'	Optional	<p>The level of corruption detection.</p> <p>Specify 'PHYSICAL' to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer.</p> <p>Specify 'PHYSICAL+LOGICAL' to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.</p>
p_rman_to_dbms_output	boolean	TRUE, FALSE	FALSE	Optional	When TRUE, the RMAN output is sent to the DBMS_OUTPUT package in addition to a file in the BDUMP directory. When using SQL*Plus, execute SET SERVEROUTPUT ON to see the output.

Parameter Name	Data Type	Valid Values	Default	Required	Description
					When FALSE, the RMAN output is only sent to a file in the BDUMP directory.

Validating an SPFILE

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_spfile` to validate only the server parameter file (SPFILE) used by an Amazon RDS Oracle DB instance.

Parameter Name	Data Type	Valid Values	Default	Required	Description
<code>p_validation_type</code>	varchar2	'PHYSICAL' 'PHYSICAL+LOGICAL'	'PHYSICAL'	Optional	<p>The level of corruption detection.</p> <p>Specify 'PHYSICAL' to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer.</p> <p>Specify 'PHYSICAL+LOGICAL' to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.</p>
<code>p_rman_to_dbms_output</code>	boolean	TRUE, FALSE	FALSE	Optional	<p>When TRUE, the RMAN output is sent to the DBMS_OUTPUT package in addition to a file in the BDUMP directory. When using SQL*Plus, execute SET SERVEROUTPUT ON to see the output.</p> <p>When FALSE, the RMAN output is only sent to a file in the BDUMP directory.</p>

Validating a Data File

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_datafile` to validate a data file.

Parameter Name	Data Type	Valid Values	Default	Required	Description
p_datafile	varchar2	A valid datafile ID number or a valid datafile name including complete path	—	Required	The datafile ID number (from v\$datafile.file#) or the full datafile name including the path (from v\$datafile.name).
p_from_block	number	A valid integer	NULL	Optional	The number of the block where the validation starts within the data file. When NULL, 1 is used.
p_to_block	number	A valid integer	NULL	Optional	The number of the block where the validation ends within the data file. When NULL, the max block in the data file is used.
p_validation_type	varchar2	'PHYSICAL' 'PHYSICAL+LOGICAL'	Optional	Optional	<p>The level of corruption detection.</p> <p>Specify 'PHYSICAL' to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer.</p> <p>Specify 'PHYSICAL+LOGICAL' to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.</p>
p_parallel	number	A valid integer between 1 and 254 for Oracle Database Enterprise Edition (EE) 1 for other Oracle	1	Optional	Number of channels.

Parameter Name	Data Type	Valid Values	Default	Required	Description
		Database editions			
p_section_size_mb	number	A valid integer	NULL	Optional	<p>The section size in megabytes (MB).</p> <p>Validates in parallel by dividing each file into the specified section size.</p> <p>When NULL, the parameter is ignored.</p>
p_rman_to_dbms_output	boolean	TRUE, FALSE	FALSE	Optional	<p>When TRUE, the RMAN output is sent to the DBMS_OUTPUT package in addition to a file in the BDUMP directory. When using SQL*Plus, execute SET SERVEROUTPUT ON to see the output.</p> <p>When FALSE, the RMAN output is only sent to a file in the BDUMP directory.</p>

Related Topics

- [Common DBA System Tasks for Oracle DB Instances \(p. 845\)](#)
- [Common DBA Log Tasks for Oracle DB Instances \(p. 866\)](#)
- [Common DBA Miscellaneous Tasks for Oracle DB Instances \(p. 873\)](#)

Common DBA Log Tasks for Oracle DB Instances

This section describes how you can perform common DBA tasks related to logging on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

For more information, see [Oracle Database Log Files \(p. 328\)](#).

Setting Force Logging

In force logging mode, Oracle logs all changes to the database except changes in temporary tablespaces and temporary segments (NOLOGGING clauses are ignored). For more information, see [Specifying FORCE LOGGING Mode](#) in the Oracle documentation.

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.force_logging` to set force logging. The `force_logging` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
p_enable	boolean	true	optional	Set to true to put the database in force logging mode, false to remove the database from force logging mode.

The following example puts the database in force logging mode.

```
exec rdsadmin.rdsadmin_util.force_logging(p_enable => true);
```

Setting Supplemental Logging

Supplemental logging ensures that LogMiner and products that use LogMiner technology have sufficient information to support chained rows and storage arrangements such as cluster tables. For more information, see [Supplemental Logging](#) in the Oracle documentation.

Oracle Database doesn't enable supplemental logging by default. You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.alter_supplemental_logging` to enable and disable supplemental logging. For more information about how Amazon RDS manages the retention of archived redo logs for Oracle DB instances, see [Retaining Archived Redo Logs \(p. 871\)](#).

The `alter_supplemental_logging` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
p_action	varchar2	—	required	'ADD' to add supplemental logging, 'DROP' to drop supplemental logging.
p_type	varchar2	null	optional	The type of supplemental logging. Valid values are 'ALL', 'FOREIGN KEY', 'PRIMARY KEY', or 'UNIQUE'.

The following example enables supplemental logging:

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD');
end;
/
```

The following example enables supplemental logging for all fixed-length maximum size columns:

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD',
        p_type    => 'ALL');
end;
/
```

The following example enables supplemental logging for primary key columns:

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD',
        p_type    => 'PRIMARY KEY');
end;
/
```

Switching Online Log Files

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.switch_logfile` to switch log files. The `switch_logfile` procedure has no parameters.

The following example switches log files.

```
exec rdsadmin.rdsadmin_util.switch_logfile;
```

Adding Online Redo Logs

An Amazon RDS DB instance running Oracle starts with four online redo logs, 128 MB each. You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.add_logfile` to add additional redo logs.

For any version of Oracle, the `add_logfile` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
bytes	positive	null	optional	The size of the log file in bytes.

The `add_logfile` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
p_size	varchar2	—	required	The size of the log file. You can specify the size in kilobytes (K), megabytes (M), or gigabytes (G).

The following command adds a 100 MB log file:

```
exec rdsadmin.rdsadmin_util.add_logfile(p_size => '100M');
```

Dropping Online Redo Logs

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.drop_logfile` to drop redo logs. The `drop_logfile` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
grp	positive	—	required	The group number of the log.

The following example drops the log with group number 3:

```
exec rdsadmin.rdsadmin_util.drop_logfile(grp => 3);
```

You can only drop logs that have a status of unused or inactive. The following example gets the statuses of the logs:

```
select GROUP#, STATUS from V$LOG;

GROUP#      STATUS
----- -----
1           CURRENT
2           INACTIVE
3           INACTIVE
4           UNUSED
```

Resizing Online Redo Logs

An Amazon RDS DB instance running Oracle starts with four online redo logs, 128 MB each. The following example shows how you can use Amazon RDS procedures to resize your logs from 128 MB each to 512 MB each.

```
/* Query V$LOG to see the logs.          */
/* You start with 4 logs of 128 MB each. */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#      BYTES      STATUS
----- -----
1           134217728  INACTIVE
2           134217728  CURRENT
3           134217728  INACTIVE
4           134217728  INACTIVE

/* Add four new logs that are each 512 MB */

exec rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);
exec rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);
exec rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);
exec rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);

/* Query V$LOG to see the logs. */
/* Now there are 8 logs.          */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#      BYTES      STATUS
----- -----
1           134217728  INACTIVE
2           134217728  CURRENT
3           134217728  INACTIVE
4           134217728  INACTIVE
5           536870912  UNUSED
6           536870912  UNUSED
7           536870912  UNUSED
8           536870912  UNUSED

/* Drop each inactive log using the group number. */
```

```
exec rdsadmin.rdsadmin_util.drop_logfile(grp => 1);
exec rdsadmin.rdsadmin_util.drop_logfile(grp => 3);
exec rdsadmin.rdsadmin_util.drop_logfile(grp => 4);

/* Query V$LOG to see the logs. */
/* Now there are 5 logs. */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#      BYTES      STATUS
-----
2          134217728  CURRENT
5          536870912  UNUSED
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED

/* Switch logs so that group 2 is no longer current. */

exec rdsadmin.rdsadmin_util.switch_logfile;

/* Query V$LOG to see the logs. */
/* Now one of the new logs is current. */

SQL>select GROUP#, BYTES, STATUS from V$LOG;

GROUP#      BYTES      STATUS
-----
2          134217728  ACTIVE
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED

/* Issue a checkpoint to clear log 2. */

exec rdsadmin.rdsadmin_util.checkpoint;

/* Query V$LOG to see the logs. */
/* Now the final original log is inactive. */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#      BYTES      STATUS
-----
2          134217728  INACTIVE
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED

# Drop the final inactive log.

exec rdsadmin.rdsadmin_util.drop_logfile(grp => 2);

/* Query V$LOG to see the logs. */
/* Now there are four 512 MB logs. */
```

```
select GROUP#, BYTES, STATUS from V$LOG;

GROUP#    BYTES      STATUS
-----  -----
5          536870912 CURRENT
6          536870912 UNUSED
7          536870912 UNUSED
8          536870912 UNUSED
```

Retaining Archived Redo Logs

You can retain archived redo logs locally on your DB instance for use with products like Oracle LogMiner (DBMS_LOGMNR). After you have retained the redo logs, you can use LogMiner to analyze the logs. For more information, see [Using LogMiner to Analyze Redo Log Files](#) in the Oracle documentation.

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.set_configuration` to retain archived redo logs. The `set_configuration` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>name</code>	varchar	—	required	The name of the configuration to update.
<code>value</code>	varchar	—	required	The value for the configuration.

The following example retains 24 hours of redo logs:

```
begin
    rdsadmin.rdsadmin_util.set_configuration(
        name  => 'archivelog retention hours',
        value => '24');
end;
/
commit;
```

Note

The commit is required for the change to take effect.

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.show_configuration` to view how long archived redo logs are retained for your DB instance.

The following example shows the log retention time:

```
set serveroutput on
exec rdsadmin.rdsadmin_util.show_configuration;
```

The output shows the current setting for `archivelog retention hours`. The following output shows that archived redo logs are retained for 48 hours:

```
NAME:archivelog retention hours
VALUE:48
DESCRIPTION:ArchiveLog expiration specifies the duration in hours before archive/redo log files are automatically deleted.
```

Because the archived redo logs are retained on your DB instance, ensure that your DB instance has enough allocated storage for the retained logs. To determine how much space your DB instance has used in the last X hours, you can run the following query, replacing X with the number of hours:

```
select sum(BLOCKS * BLOCK_SIZE) bytes
  from V$ARCHIVED_LOG
 where FIRST_TIME >= SYSDATE-(X/24) and DEST_ID=1;
```

Archived redo logs are only generated if the backup retention period of your DB instance is greater than zero. By default the backup retention period is greater than zero, so unless you explicitly set yours to zero, archived redo logs are generated for your DB instance. To modify the backup retention period for your DB instance, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

After the archived redo logs are removed from your DB instance, you can't download them again to your DB instance. Amazon RDS retains the archived redo logs outside of your DB instance to support restoring your DB instance to a point in time. Amazon RDS retains the archived redo logs outside of your DB instance based on the backup retention period configured for your DB instance. To modify the backup retention period for your DB instance, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Note

If you are using JDBC on Linux to download archived redo logs, and you experience long latency times and connection resets, it could be caused by the default random number generator setting on your Java client. We recommend setting your JDBC drivers to use a non-blocking random number generator.

Accessing Transaction Logs

Accessing transaction logs is supported for Oracle version 11.2.0.4.v11 and later, and 12.1.0.2.v7 and later.

You might want to access your online and archived redo log files for mining with external tools such as GoldenGate, Attunity, Informatica, and others. If you want to access your online and archived redo log files, you must first create directory objects that provide read-only access to the physical file paths.

The following code creates directories that provide read-only access to your online and archived redo log files:

Important

This code also revokes the `DROP ANY DIRECTORY` privilege.

```
exec rdsadmin.rdsadmin_master_util.create_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

After you create directory objects for your online and archived redo log files, you can read the files by using PL/SQL. For more information about reading files from directory objects, see [Listing Files in a DB Instance Directory \(p. 874\)](#) and [Reading Files in a DB Instance Directory \(p. 874\)](#).

The following code drops the directories for your online and archived redo log files:

```
exec rdsadmin.rdsadmin_master_util.drop_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.drop_onlinelog_dir;
```

The following code grants and revokes the `DROP ANY DIRECTORY` privilege:

```
exec rdsadmin.rdsadmin_master_util.revoke_drop_any_directory;
```

```
exec rdsadmin.rdsadmin_master_util.grant_drop_any_directory;
```

Related Topics

- [Common DBA System Tasks for Oracle DB Instances \(p. 845\)](#)
- [Common DBA Database Tasks for Oracle DB Instances \(p. 854\)](#)
- [Common DBA Miscellaneous Tasks for Oracle DB Instances \(p. 873\)](#)

Common DBA Miscellaneous Tasks for Oracle DB Instances

This section describes how you can perform miscellaneous DBA tasks on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

Creating New Directories in the Main Data Storage Space

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.create_directory` to create directories. You can create up to 10,000 directories, all located in your main data storage space.

The `create_directory` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_directory_name</code>	<code>varchar2</code>	—	required	The name of the new directory.

The following example creates a new directory named `product_descriptions`:

```
exec rdsadmin.rdsadmin_util.create_directory(p_directory_name => 'product_descriptions');
```

You can list the directories by querying `DBA_DIRECTORIES`. The system chooses the actual host pathname automatically. The following example gets the directory path for the directory named `product_descriptions`:

```
select DIRECTORY_PATH
  from DBA_DIRECTORIES
 where DIRECTORY_NAME='product_descriptions';

DIRECTORY_PATH
-----
/rdsdbdata/userdirs/01
```

The master user name for the DB instance has read and write privileges in the new directory, and can grant access to other users. Execute privileges are not available for directories on a DB instance. Directories are created in your main data storage space and will consume space and I/O bandwidth.

You can drop a directory that you created by using the Oracle `drop_directory` command. Dropping a directory doesn't remove its contents. Because the `create_directory()` method can reuse pathnames, files in dropped directories can appear in a newly created directory. Before you drop a directory, you should use `UTL_FILE.FREMOVE` to remove files from the directory.

Listing Files in a DB Instance Directory

You can use the Amazon RDS procedure `rdsadmin.rds_file_util.listdir` to list the files in a directory. The `listdir` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_directory</code>	varchar2	—	required	The name of the directory to list.

The following example lists the files in the directory named `product_descriptions`:

```
select * from table
  (rdsadmin.rds_file_util.listdir(p_directory => 'product_descriptions'));
```

Reading Files in a DB Instance Directory

You can use the Amazon RDS procedure `rdsadmin.rds_file_util.read_text_file` to read a text file. The `read_text_file` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
<code>p_directory</code>	varchar2	—	required	The name of the directory that contains the file.
<code>p_filename</code>	varchar2	—	required	The name of the file to read.

The following example reads the file `rice.txt` from the directory `product_descriptions`:

```
select * from table
  (rdsadmin.rds_file_util.read_text_file(
    p_directory => 'product_descriptions',
    p_filename  => 'rice.txt'));
```

Related Topics

- [Common DBA System Tasks for Oracle DB Instances \(p. 845\)](#)
- [Common DBA Database Tasks for Oracle DB Instances \(p. 854\)](#)
- [Common DBA Log Tasks for Oracle DB Instances \(p. 866\)](#)

Related Topics

- [Oracle Database Log Files \(p. 328\)](#)
- [Options for Oracle DB Instances \(p. 787\)](#)
- [Tools and Third-Party Software for Oracle DB Instances \(p. 875\)](#)

Tools and Third-Party Software for Oracle DB Instances

This section provides information about tools and third-party software for Oracle DB instances on Amazon RDS.

Topics

- [Setting Up Amazon RDS to Host Tools and Third-Party Software for Oracle \(p. 875\)](#)
- [Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys \(p. 886\)](#)
- [Using Oracle GoldenGate with Amazon RDS \(p. 902\)](#)
- [Using the Oracle Repository Creation Utility on Amazon RDS for Oracle \(p. 913\)](#)
- [Installing a Siebel Database on Oracle on Amazon RDS \(p. 918\)](#)

Setting Up Amazon RDS to Host Tools and Third-Party Software for Oracle

You can use Amazon RDS to host an Oracle DB instance that supports software and components such as the following:

- Siebel Customer Relationship Management (CRM)
- Oracle Fusion Middleware Metadata — installed by the Repository Creation Utility (RCU)

The following procedures help you create an Oracle DB instance on Amazon RDS that you can use to host additional software and components for Oracle.

Creating an Amazon VPC for Use with an Oracle Database

In the following procedure, you create an Amazon VPC, a private subnet, and a security group. Because your Amazon RDS DB instance needs to be available only to your middle-tier components, and not to the public Internet, your Amazon RDS DB instance is hosted in a private subnet, providing greater security.

To create an Amazon VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the top-right corner of the AWS Management Console, choose the AWS Region for your VPC. This example uses the US West (Oregon) region.
3. In the upper-left corner, choose **VPC Dashboard**, and then choose **Start VPC Wizard**.
4. On the page **Step 1: Select a VPC Configuration**, choose **VPC with Public and Private Subnets**, and then choose **Select**.
5. On the page **Step 2: VPC with Public and Private Subnets**, shown following, set these values:

Option	Value
IPv4 CIDR block	10.0.0.0/16 For more information about selecting CIDR blocks for your VPC, see VPC Sizing .

Option	Value
IPv6 CIDR block	No IPv6 CIDR Block
VPC name	The name for your VPC, for example vpc-1 .
Public subnet's IPv4 CIDR	10.0.0.0/24 For more information about subnet sizing, see Subnet Sizing .
Availability Zone	An Availability Zone for your AWS Region.
Public subnet name	The name for your public subnet, for example subnet-public-1 .
Private subnet's IPv4 CIDR	10.0.1.0/24 For more information about subnet sizing, see Subnet Sizing .
Availability Zone	An Availability Zone for your AWS Region.
Private subnet name	The name for your private subnet, for example subnet-private-1 .
Instance type	An instance type for your NAT instance, for example t2.small . Note If you don't see Instance type in the console, choose Use a NAT instance instead .
Key pair name	No key pair
Service endpoints	None
Enable DNS hostnames	Yes
Hardware tenancy	Default

Step 2: VPC with Public and Private Subnets

IPv4 CIDR block: *	<input type="text" value="10.0.0.0/16"/> (65531 IP addresses available)	
IPv6 CIDR block:	<input checked="" type="radio"/> No IPv6 CIDR Block <input type="radio"/> Amazon provided IPv6 CIDR block	
VPC name:	<input type="text" value="vpc-1"/>	
Public subnet's IPv4 CIDR: *	<input type="text" value="10.0.0.0/24"/> (251 IP addresses available)	
Availability Zone: *	<input type="text" value="us-west-2a"/>	
Public subnet name:	<input type="text" value="subnet-public-1"/>	
Private subnet's IPv4 CIDR: *	<input type="text" value="10.0.1.0/24"/> (251 IP addresses available)	
Availability Zone: *	<input type="text" value="us-west-2a"/>	
Private subnet name:	<input type="text" value="subnet-private-1"/>	
You can add more subnets after AWS creates the VPC.		
Specify the details of your NAT instance (Instance rates apply). Use a NAT gateway instead		
Instance type: *	<input type="text" value="t2.small"/>	
Key pair name:	<input type="text" value="No key pair"/>	
Service endpoints		
Add Endpoint		
Enable DNS hostnames: *	<input checked="" type="radio"/> Yes <input type="radio"/> No	
Hardware tenancy: *	<input type="text" value="Default"/>	
Cancel and Exit	Back	Create VPC

- Choose **Create VPC**.

An Amazon RDS DB instance in a VPC requires at least two private subnets or at least two public subnets, to support Multi-AZ deployment. For more information about working with multiple Availability Zones, see [Regions and Availability Zones \(p. 101\)](#). Because your database is private, add a second private subnet to your VPC.

To create an additional subnet

- Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
- In the top-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
- In the upper-left corner, choose **VPC Dashboard**, choose **Subnets**, and then choose **Create Subnet**.

4. On the **Create Subnet** page, set the following values.

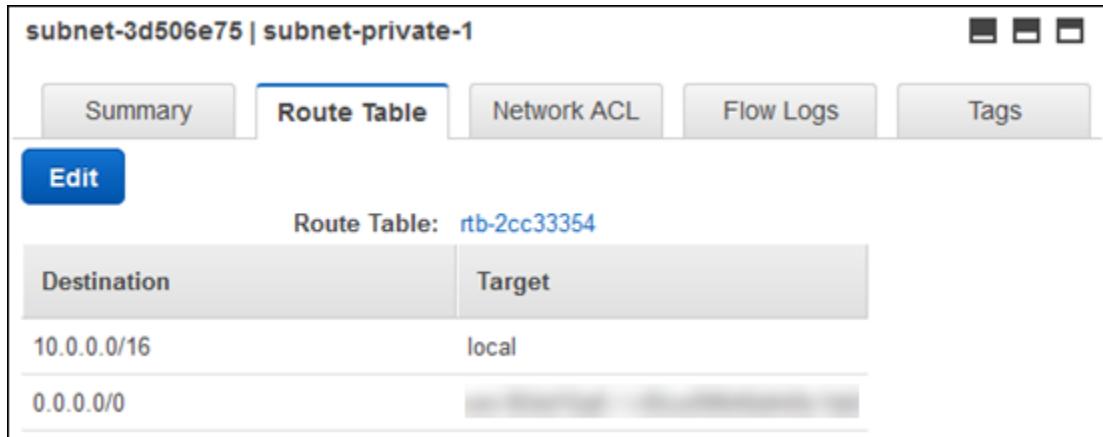
Option	Value
Name tag	The name for your second private subnet, for example subnet-private-2 .
VPC	Your VPC, for example vpc-1 .
Availability Zone	An Availability Zone for your AWS Region. Note Choose an Availability Zone different from the one that you chose for the first private subnet.
CIDR block	10.0.2.0/24

5. Choose **Yes, Create**.

Both private subnets must use the same route table. In the following procedure, you check to make sure the route tables match, and if not you edit one of them.

To ensure the subnets use the same route table.

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the top-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
3. In the upper-left corner, choose **VPC Dashboard**, choose **Subnets**, and then choose your first private subnet, for example **subnet-private-1**.
4. At the bottom of the console, choose the **Route Table** tab, shown following.



5. Make a note of the route table, for example **rtb-0d9fc668**.
6. In the list of subnets, choose the second private subnet, for example **subnet-private-2**.
7. At the bottom of the console, choose the **Route Table** tab.
8. If the route table for the second subnet is not the same as the route table for the first subnet, edit it to match:
 - a. Choose **Edit**.
 - b. For **Change to**, choose the route table that matches your first subnet.
 - c. Choose **Save**.

A security group acts as a virtual firewall for your DB instance to control inbound and outbound traffic. In the following procedure, you create a security group for your DB instance. For more information about security groups, see [Security Groups for Your VPC](#).

To create a VPC security group for a Private Amazon RDS DB Instance

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the top-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
3. In the upper-left corner, choose **VPC Dashboard**, choose **Security Groups**, and then choose **Create Security Group**.
4. On the page **Create Security Group**, set the following values.

Option	Value
Name tag	The name for your security group, for example sgdb-1 .
Group name	The name for your security group, for example sgdb-1 .
Description	A description for your security group.
VPC	Your VPC, for example vpc-1 .

5. Choose **Yes, Create**.

In the following procedure, you add rules to your security group to control inbound traffic to your DB instance. For more information about inbound rules, see [Security Group Rules](#).

To add inbound rules to the security group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the top-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
3. In the upper-left corner, choose **VPC Dashboard**, choose **Security Groups**, and then choose your security group, for example **sgdb-1**.
4. At the bottom of the console, choose the **Inbound Rules** tab, and then choose **Edit**.
5. Set these values, as shown following.

Option	Value
Type	Oracle (1521)
Protocol	TCP (6)
Port Range	1521
Source	The identifier of your security group. When you choose the box, you see the name of your security group, for example sgdb-1 .



6. Choose **Save**.

Creating an Oracle DB Instance

You can use Amazon RDS to host an Oracle DB instance. In the following procedure, you create the Oracle DB instance.

To launch an Oracle DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top-right corner of the AWS Management Console, choose the AWS Region for your DB instance. Choose the same AWS Region as your VPC.
3. Choose **Databases** and then choose **Create database**.
4. On the page **Select engine**, choose **Oracle**, and then choose **Oracle Database Enterprise Edition**.

Select engine

Engine options

- Amazon Aurora
Amazon Aurora
- MySQL

- MariaDB

- PostgreSQL

- Oracle
ORACLE®
- Microsoft SQL Server


Oracle

Edition

Oracle Enterprise Edition
Efficient, reliable, and secure database management system that delivers comprehensive high-end capabilities for mission-critical applications and demanding database workloads.

Oracle Standard Edition
Affordable and full-featured database management system supporting up to 32 vCPUs.

Oracle Standard Edition One
Affordable and full-featured database management system supporting up to 16 vCPUs.

Oracle Standard Edition Two
Affordable and full-featured database management system supporting up to 16 vCPUs.
Oracle Database Standard Edition Two is a replacement for Standard Edition and Standard Edition One.

Only enable options eligible for RDS Free Usage Tier [info](#)

[Cancel](#) [Next](#)

5. Choose **Next**.
6. On the page **Choose use case**, choose **Production**, and then choose **Next**.

Note

For a DB instance for development and testing, you can choose **Dev/Test**.

7. On the page **Specify DB details**, shown following, set the following values.

Option	Value
License model	bring-your-own-license

Option	Value
DB engine version	The Oracle version you want to use. Use the latest Oracle 12c version.
DB instance class	The DB instance class you want to use. For more information, see DB Instance Class (p. 82) .
Multi-AZ deployment	<p>Create replica in different zone. Multi-AZ deployment creates a standby replica of your DB instance in another Availability Zone for failover support. Multi-AZ is recommended for production workloads. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 101).</p> <p>Note For development and testing, you can choose No.</p>
Storage type	<p>Provisioned IOPS (SSD). Provisioned IOPS (input/output operations per second) is recommended for production workloads. For more information about storage, see DB instance storage (p. 103).</p> <p>Note For development and testing, you can choose General Purpose (SSD).</p>
Allocated storage	The storage to allocate for your database. Allocate at least 20 GiB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon RDS Storage Types (p. 103) and Guidelines for Creating Oracle Database Tablespaces .
Provisioned IOPS	<p>The amount of provisioned IOPS to be initially allocated for the DB instance. This value must be a multiple between 3 and 10 of the storage amount for the DB instance. This value must also be an integer multiple of 1,000.</p> <p>Note For development and testing, you do not need Provisioned IOPS.</p>
DB instance identifier	The name for DB instance, for example oracle-instance .
Master username	The master username for the DB instance, for example oracle_mu .
Master password and Confirm password	A password that contains from 8 to 30 printable ASCII characters (excluding /, ", and @) for your master user password. Retype the password in the Confirm Password box.

Specify DB details

Instance specifications
Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#).

DB engine: Oracle Database Enterprise Edition

License model [info](#): bring-your-own-license

DB engine version [info](#): Oracle 12.1.0.2.v10

DB instance class [info](#): db.m4.xlarge — 4 vCPU, 16 GiB RAM

Multi-AZ deployment [info](#):

Create replica in different zone
Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

No

Storage type [info](#): Provisioned IOPS (SSD)

Allocated storage: 100 GB
(Minimum: 100 GB, Maximum: 16384 GB)

Provisioned IOPS [info](#): 1000

Settings

DB instance identifier [info](#)
Specify a name that is unique for all DB instances owned by your AWS account in the current region.

oracle-instance

DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance".

Master username [info](#)
Specify an alphanumeric string that defines the login ID for the master user.

oracle_mu

Master Username must start with a letter.

Master password [info](#): ······

Confirm password [info](#): ······

Master Password must be at least eight characters long, as in "mypassword".

[Cancel](#) [Previous](#) **Next**

8. Choose **Next**.
9. On the page **Configure advanced settings**, shown following, set the following values.

Option	Value
Virtual Private Cloud (VPC)	Your VPC, for example vpc-1 .
Subnet group	Create new DB Subnet Group
Public accessibility	No
Availability zone	No Preference
VPC security groups	Choose Select existing VPC security groups , and choose your VPC security group, for example sgdb-1 .
Database name	The name for your database, for example db1 .
Database port	1521
DB parameter group	The default parameter group.
Option group	The default option group.
Copy tags to snapshots	This option, when chosen, specifies to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .
Character set name	A character set for your DB instance. The default value of AL32UTF8 is for the Unicode 5.0 UTF-8 Universal character set. You can't change the character set after the DB instance is created.
Enable encryption	Enable Encryption or Disable Encryption . A value of Enable Encryption enables encryption at rest for this DB instance, and you can choose a master key. For more information, see Encrypting Amazon RDS Resources (p. 389) .
Backup retention period	The number of days you want to retain automatic backups of your database. For most DB instances, you should set this value to 1 or greater.
Backup window	Unless you have a specific time that you want to have your database backup, use the default of No Preference .
Enhanced monitoring	Enable enhanced monitoring to gather metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 256) .
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.
Maintenance window	Choose Select window and choose the 30-minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter to you, choose No Preference .

10. On the final page of the wizard, choose **View DB instance details**.

On the RDS console, the details for the new DB instance appear. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and storage allocated, it could take several minutes for the new instance to be available.

Additional Amazon RDS Interfaces

In the preceding procedures, we use the AWS Management Console to perform tasks. Amazon Web Services also provides the AWS Command Line Interface (AWS CLI), and an application programming interface (API). You can use the AWS CLI or the API to automate many of the tasks for managing Amazon RDS, including tasks to manage an Oracle DB instance with Amazon RDS.

For more information, see [AWS Command Line Interface Reference for Amazon RDS](#) and [Amazon RDS API Reference](#).

Related Topics

- [Setting Up for Amazon RDS \(p. 5\)](#)
- [Using the Oracle Repository Creation Utility on Amazon RDS for Oracle \(p. 913\)](#)
- [Installing a Siebel Database on Oracle on Amazon RDS \(p. 918\)](#)
- [Scenarios for Accessing a DB Instance in a VPC \(p. 414\)](#)
- [Connecting to a DB Instance Running the Oracle Database Engine \(p. 751\)](#)

Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys

You can use AWS CloudHSM Classic with an Amazon RDS DB instance running Oracle Enterprise Edition to store keys when you use Oracle Transparent Data Encryption (TDE). AWS CloudHSM Classic is a service that provides a hardware appliance called a hardware security module (HSM) that performs secure key storage and cryptographic operations. You enable an Amazon RDS DB instance to use AWS CloudHSM Classic by setting up an HSM appliance, setting the proper permissions for cross-service access, and then setting up Amazon RDS and the DB instance that will use AWS CloudHSM Classic.

Important

Review the following availability and pricing information before you setup AWS CloudHSM Classic:

- Amazon RDS supports AWS CloudHSM Classic for Oracle DB instances in the following regions: US East (N. Virginia), US West (Oregon), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), EU (Frankfurt), EU (Ireland).
- AWS CloudHSM Classic pricing:

AWS CloudHSM Classic pricing information is available on the [AWS CloudHSM Classic pricing page](#).

- AWS CloudHSM Classic upfront fee refund (API and CLI Tools):

You are charged an upfront fee for each new AWS CloudHSM Classic instance that you create by using the [CreateHsm](#) API operation or the [create-hsm](#) AWS CLI command. If you accidentally provision an HSM instance that you don't need, first delete the HSM instance by using the [DeleteHsm](#) API operation or the [delete-hsm](#) AWS CLI command. You can then request a refund of the upfront fee at the [AWS Support Center](#), by creating a new case and choosing **Account and Billing Support**.

The number of Oracle databases you can support on a single AWS CloudHSM Classic partition will depend on the rotation schedule you choose for your data. You should rotate your keys as often as your data needs require. The [PCI-DSS documentation](#) and the [National Institute of Standards and Technology \(NIST\)](#) provide guidance on appropriate key rotation frequency. You can maintain approximately 10,000 symmetric master keys per AWS CloudHSM Classic device. Note that after key rotation the old master key remains on the partition and is still counted against the per-partition maximum.

AWS CloudHSM Classic works with Amazon Virtual Private Cloud (Amazon VPC). An appliance is provisioned inside your VPC with a private IP address that you specify, providing simple and private network connectivity to your Amazon RDS DB instance. Your HSM appliances are dedicated exclusively to you and are isolated from other AWS customers. For more information, see [Amazon Virtual Private Cloud \(VPCs\) and Amazon RDS \(p. 412\)](#) and [Creating a DB Instance in a VPC \(p. 422\)](#).

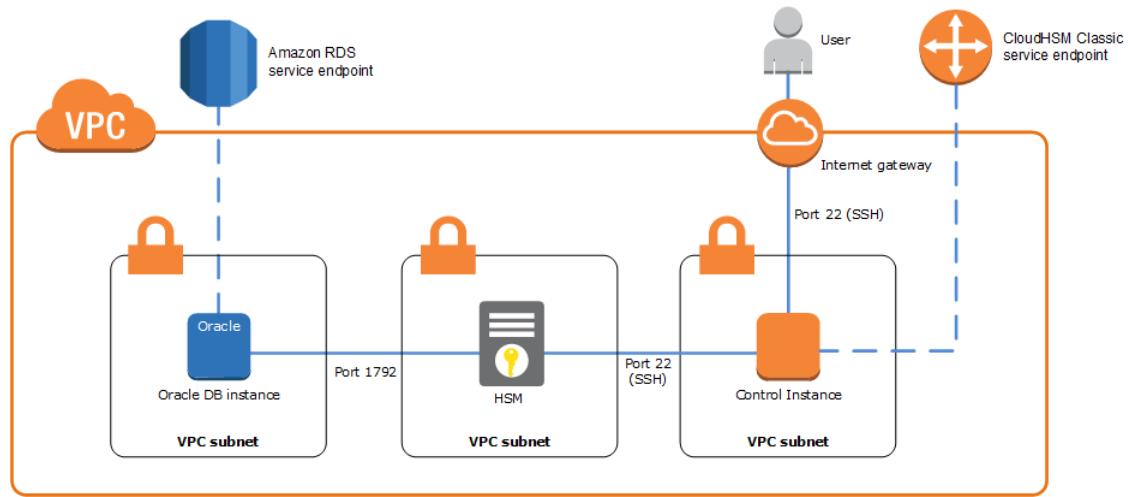
To use AWS CloudHSM Classic with an Amazon RDS Oracle DB instance, you must complete the following tasks, which are explained in detail in the following sections:

- [Setting Up AWS CloudHSM Classic to Work with Amazon RDS \(p. 888\)](#)
- [Setting Up Amazon RDS to Work with AWS CloudHSM Classic \(p. 891\)](#)

When you complete the entire setup, you should have the following AWS components.

- An AWS CloudHSM Classic control instance that will communicate with the HSM appliance using port 22, and the AWS CloudHSM Classic endpoint. The AWS CloudHSM Classic control instance is an Amazon EC2 instance that is in the same VPC as the HSMs and is used to manage the HSMs.

- An Amazon RDS Oracle DB instance that will communicate with the Amazon RDS service endpoint, as well as the HSM appliance using port 1792.



Topics

- [Setting Up AWS CloudHSM Classic to Work with Amazon RDS \(p. 888\)](#)
- [Setting Up Amazon RDS to Work with AWS CloudHSM Classic \(p. 891\)](#)
- [Verifying the HSM Connection, the Oracle Keys in the HSM, and the TDE Key \(p. 899\)](#)
- [Restoring Encrypted DB Instances \(p. 900\)](#)
- [Managing a Multi-AZ Failover \(p. 901\)](#)

Setting Up AWS CloudHSM Classic to Work with Amazon RDS

To use AWS CloudHSM Classic with an Oracle DB instance using TDE, you must first complete the tasks required to setup AWS CloudHSM Classic. The tasks are explained in detail in the following sections.

Amazon RDS supports AWS CloudHSM Classic for Oracle DB instances in the following regions: US East (N. Virginia), US West (Oregon), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), EU (Frankfurt), EU (Ireland).

Completing the AWS CloudHSM Classic Prerequisites

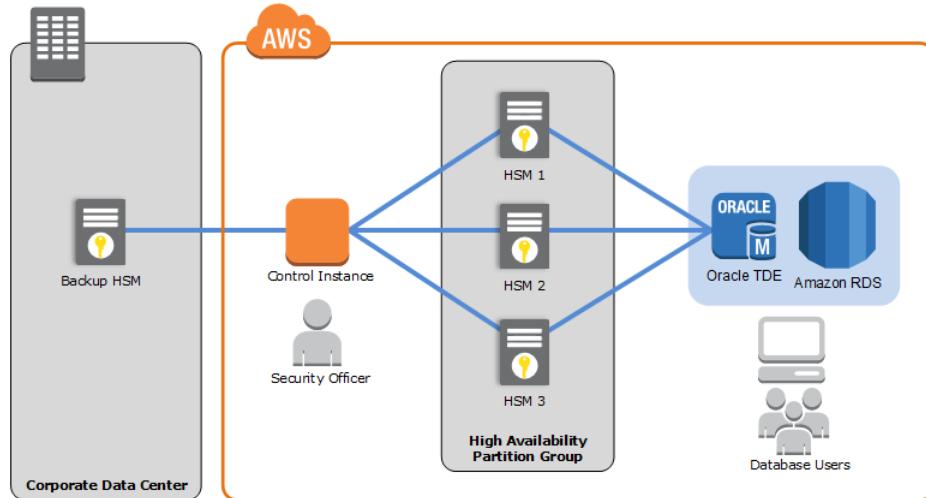
Follow the procedure in the [Setting Up AWS CloudHSM](#) section in the *AWS CloudHSM Classic User Guide* to setup an AWS CloudHSM Classic environment.

Installing the AWS CloudHSM Classic Command Line Interface Tools

Follow the instructions in the [Setting Up the AWS CloudHSM CLI Tools](#) section in the *AWS CloudHSM Classic User Guide* to install the AWS CloudHSM Classic command line interface tools on your AWS CloudHSM Classic control instance.

Configuring Your HSMs

The recommended configuration for using AWS CloudHSM Classic with Amazon RDS is to use three AWS CloudHSM Classic appliances configured into a high-availability (HA) partition group. A minimum of three HSMs are suggested for HA purposes. Even if two of your HSMs are unavailable, your keys will still be available to Amazon RDS.



Important

Initializing an HSM sets the password for the HSM security officer account (also known as the HSM administrator). Record the security officer password on your [Password Worksheet \(p. 890\)](#) and do not lose it. We recommend that you print out a copy of the [Password Worksheet \(p. 890\)](#), use it to record your AWS CloudHSM Classic passwords, and store it in a secure place. We also recommended that you store at least one copy of this worksheet in secure off-site storage. AWS does not have the ability to recover your key material from an HSM for which you do not have the proper HSM security officer credentials.

To provision and initialize your HSMs using the AWS CloudHSM Classic CLI tools, perform the following steps from your control instance:

1. Following the instructions in [Creating Your HSMs with the CLI](#), provision the number of HSMs you need for your configuration. When you provision your HSMs, make note of the ARN of each HSM because you will need these to initialize your HSMs and create your high-availability partition group.

2. Following the instructions in [Initializing Your HSMs](#), initialize each of your HSMs.

Creating Your High-Availability Partition Group

After your HSMs are initialized, create an HA partition group with the initialized HSMs. Creating an HA partition group is a three-step process. You create the HA partition group, add your HSMs to the HA partition group, and register the clients for use with the HA partition group.

To create and initialize an HA partition group

1. Following the instructions in the [Create the HA Partition Group](#) section in the *AWS CloudHSM Classic User Guide*, create your HA partition group. Save the HA partition group ARN returned from the `create-hapg` command for later use.

Save the partition password on your [Password Worksheet \(p. 890\)](#).
2. Following the instructions in [Registering a Client with a High-Availability Partition Group](#), create, register, and assign the clients to use with your HA partition group.

Repeat this process to add additional partitions if necessary. One partition can support multiple Oracle databases.

Password Worksheet

Use the following worksheet to compile information for your AWS CloudHSM Classic appliances. Print this page and use it to record your AWS CloudHSM Classic passwords, and store it in a secure place. We also recommend that you store at least one copy of this worksheet in secure off-site storage.

Security Officer Password

This password was set when you initialized the AWS CloudHSM Classic appliance.

Manager Password (Optional)

This password was optionally set with the `user password manager` command on the AWS CloudHSM Classic appliance.

Partition Passwords

Setting Up Amazon RDS to Work with AWS CloudHSM Classic

To use AWS CloudHSM Classic with an Oracle DB instance using Oracle TDE, you must do the following tasks:

- Ensure that the security group associated with the Oracle DB instance allows access to the HSM port 1792.
- Create a DB subnet group that uses the same subnets as those in the VPC used by your HSMs, and then assign that DB subnet group to your Oracle DB instance.
- Set up the Amazon RDS CLI.
- Add IAM permissions for Amazon RDS to use when accessing AWS CloudHSM Classic.
- Add the **TDE_HSM** option to the option group associated with your Oracle DB instance using the Amazon RDS CLI.
- Add two new DB instance parameters to the Oracle DB instance that will use AWS CloudHSM Classic. The `tde-credential-arn` parameter is the Amazon Resource Number (ARN) of the high-availability (HA) partition group returned from the `create-hapg` command. The `tde-credential-password` is the partition password you used when you initialized the HA partition group.

The Amazon RDS CLI documentation can be found at [What Is the AWS Command Line Interface?](#) and the section [Getting Set Up with the AWS Command Line Interface](#). General instructions on using the AWS CLI can be found at [Using the AWS Command Line Interface](#).

The following sections show you how to set up the Amazon RDS CLI, add the required permissions for RDS to access your HSMs, create an option group with the **TDE_HSM** option, and how to create or modify a DB instance that will use the **TDE_HSM** option.

Security Group

To allow the RDS instance to communicate with the HSM, the security group ENI assigned to the HSM appliance must authorize ingress connectivity on TCP port 1792 from the DB instance. Additionally, the Network ACL associated with the HSM's ENI must permit ingress TCP port 1792 from the RDS instance, and egress connections from the HSM to the Dynamic Port range on the RDS instance. For more information about the Dynamic TCP Port range, please see the [Amazon VPC documentation](#).

If you used the AWS CloudFormation template to create your AWS CloudHSM Classic environment, modify the security group that allows SSH and NTLS from the public subnet. If you didn't use the AWS CloudFormation template, modify the security group associated with the ENI assigned to the HSM appliance.

DB Subnet Group

The DB subnet group that you assign to your Oracle DB instance must have the same subnets as those in the VPC used by the AWS CloudHSM Classic. For information about how to create a DB subnet group, see [Creating a DB Subnet Group](#), or you can use the AWS CLI `create-db-subnet-group` command to create the DB subnet group.

Setting Up the Amazon RDS CLI

The Amazon RDS CLI can be installed on a computer running the Linux or Windows operating system and that has Java version 1.6 or higher installed.

The following steps install and configure the Amazon RDS CLI:

1. Download the Amazon RDS CLI from [here](#). Unzip the file.

2. Set the following environment variables:

```
AWS_RDS_HOME - <The directory where the deployment files were copied to>  
JAVA_HOME - <Java Installation home directory>
```

You can check that the environment variables are set correctly by running the following command for Linux or Windows should list describe-db-instances and other AWS CLI commands.

For Linux, OS X, or Unix:

```
ls ${AWS_RDS_HOME}/bin
```

For Windows:

```
dir %AWS_RDS_HOME%\bin
```

3. Add \${AWS_RDS_HOME}/bin (Linux) or %AWS_RDS_HOME%\bin (Windows) to your path
4. Add the RDS service URL information for your AWS region to your shell configuration. For example:

```
export RDS_URL=https://rds.us-east-1.amazonaws.com  
export SERVICE_SIG_NAME=rds
```

5. If you are on a Linux system, set execute permissions on all files in the bin directory using the following command:

```
chmod +x ${AWS_RDS_HOME}/bin/*
```

6. Provide the Amazon RDS CLI with your AWS user credentials. There are two ways you can provide credentials: AWS keys, or using X.509 certificates.

If you are using AWS keys, do the following:

- a. Edit the credential file included in the zip file, \${AWS_RDS_HOME}/credential-file-path.template, to add your AWS credentials. If you are on a Linux system, limit permissions to the owner of the credential file:

```
$ chmod 600 <credential file>
```

- b. Alternatively, you can provide the following option with every command:

```
aws rds <AWSCLIcommand> --aws-credential-file <credential file>
```

- c. Or you can explicitly specify credentials on the command line: --I ACCESS_KEY --S SECRET_KEY

If you are using X.509 certifications, do the following:

- a. Save your certificate and private keys to files: e.g. my-cert.pem and my-pk.pem.
- b. Set the following environment variables:

```
EC2_CERT=<path_to_my_cert>  
EC2_PRIVATE_KEY=<path_to_my_private_key>
```

- c. Or you can specify the files directly on command-line for every command:

For Linux, OS X, or Unix:

```
aws rds <AWSCLIcommand> \  
--ec2-cert-file-path <path_to_my_cert> \  
--
```

```
--ec2-private-key-file-path <path_to_my_private_key>
```

For Windows:

```
aws rds <AWSCLIcommand> ^
--ec2-cert-file-path <path_to_my_cert> ^
--ec2-private-key-file-path <path_to_my_private_key>
```

You can test that you have set up the AWS CLI correctly by running the following commands. The first command should output the usage page for all Amazon RDS commands. The second command should output information on all DB instances for the account you are using.

```
aws rds --help
aws rds describe-db-instances --headers
```

Adding IAM Permissions for Amazon RDS to Access the AWS CloudHSM Classic

You can use a single AWS account to work with Amazon RDS and AWS CloudHSM Classic or you can use two separate accounts, one for Amazon RDS and one for AWS CloudHSM Classic. This section provides information on both processes.

Topics

- [Adding IAM Permissions for a Single Account for Amazon RDS to Access the AWS CloudHSM Classic API \(p. 893\)](#)
- [Using Separate AWS CloudHSM Classic and Amazon RDS Accounts for Amazon RDS to Access AWS CloudHSM Classic \(p. 894\)](#)

Adding IAM Permissions for a Single Account for Amazon RDS to Access the AWS CloudHSM Classic API

To create an IAM role that Amazon RDS uses to access the AWS CloudHSM Classic API, use the following procedure. Amazon RDS checks for the presence of this IAM role when you create or modify a DB instance that uses AWS CloudHSM Classic.

To create an IAM role for Amazon RDS to access the AWS CloudHSM Classic API

1. Open the [IAM console](#).
 2. In the left navigation pane, click **Roles**.
 3. Click **Create role**.
 4. Click **AWS Service** and choose **RDS**.
 5. Under **Select your use case**, choose **RDS**.
 6. Choose **Next: Permissions**.
 7. On the **Attached permissions policy** page, choose **Next: Review**.
 8. In **Role name** on the **Review** page, type **RDSCloudHsmAuthorization**. Currently, you must use this name.
- In **Role description**, you can also type a description for the role.
9. Review the information and then click **Create role**.

Using Separate AWS CloudHSM Classic and Amazon RDS Accounts for Amazon RDS to Access AWS CloudHSM Classic

If you want to separately manage your AWS CloudHSM Classic and Amazon RDS resources, you can use the two services with separate accounts. To use two different accounts, you must set up each account as described in the following section.

To use two accounts, you must have the following:

- An account that is enabled for the AWS CloudHSM Classic service and that is the owner of your hardware security module (HSM) devices. Generally, this account is your AWS CloudHSM Classic account, with a customer ID of HSM_ACCOUNT_ID.
- An account for Amazon RDS that you can use to create and manage a DB instance that uses Oracle TDE. Generally, this account is your DB account, with a customer ID DB_ACCOUNT_ID.

To add DB account permission to access AWS CloudHSM Classic resources under the AWS CloudHSM Classic account

1. Create the IAM policy.

- Open the [IAM console](#).
- Log in using your DB account.
- In the navigation pane, choose **Policies**.
- Choose **Create policy**.
- Choose **JSON tab**.
- Copy the following policy information and paste it into the policy text field:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "sts:AssumeRole"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

- Choose **Review policy**.
- In **Name**, type **AssumeRole**. You can also add an optional **Description** value.
- Choose **Create policy**.
- Create the role.
 - In the navigation pane of the [IAM console](#), choose **Roles**.
 - Choose **Create role**.
 - Under **AWS service**, choose **RDS**.
 - Under **Select your use case**, choose **RDS**.
 - Choose **Next: Permissions**.
 - Choose **Next: Review**.
 - In **Role name** on the **Review** page, type **RDSCloudHsmAssumeAuthorization**. Currently, you must use this name.

In **Role description**, you can also type a description for the role.

- h. Choose **Create Role**.
3. Attach the policy to the role.
 - a. In the navigation pane of the [IAM console](#), choose **Roles**.
 - b. In the **Search** field, enter **RDSCloudHsmAssumeAuthorization**, and click the role when it appears in the list.
 - c. On the **Permissions** tab, detach the following default roles from the policy:
 - **AmazonRDSDirectoryServiceAccess**
 - **RDSCloudHsmAuthorizationRole**
- To detach a role, click the X associated with the role on the right, and then click **Detach**.
- d. On the **Permissions** tab, choose **Attach policy**.
- e. On the **Attach policy** page, enter **AssumeRole** in the **Search** field.
- f. When it appears in the list, select the **AssumeRole** policy.
- g. Choose **Attach policy**.

To revise the AWS CloudHSM Classic account to trust permission to access AWS CloudHSM Classic resources under the AWS CloudHSM Classic account

1. Open the [IAM console](#).
2. Log in using your AWS CloudHSM Classic account.
3. In the left navigation pane, choose **Roles**.
4. In the **Search** field, enter **RDSCloudHsmAuthorization**, and click the role when it appears in the list. This role is the one created for a single account CloudHSM-RDS.
5. Choose the **Trust relationships** tab.
6. Choose **Edit trust relationship**.
7. Add your DB account as a trusted account. The policy document should look like the following, with your DB account replacing the **<DB_ACCOUNT_ID>** placeholder:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "rds.amazonaws.com",  
                "AWS": [ "arn:aws:iam:<DB_ACCOUNT_ID>:role/RDSCloudHsmAssumeAuthorization"  
                ]  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

8. Choose **Update Trust Policy**.

Creating an Amazon VPC Using the DB Account That Can Connect to Your HSM

HSM appliances are provisioned into an HSM-specific Amazon VPC. By default, only hosts inside the HSM VPC can see the HSM devices. Thus, all DB instances need to be created inside the HSM VPC or in a VPC that can be linked to the HSM VPC using VPC peering.

To use AWS CloudHSM Classic with an Amazon RDS DB instance in a different VPC (which you create under your DB account, as described in [Creating a DB Instance in a VPC \(p. 422\)](#)), you set up VPC peering from the VPC containing the DB instance to the HSM-specific VPC that contains your HSM appliances.

To set up VPC peering between the two VPCs

1. Use an existing VPC created under your DB account, or create a new VPC using your DB account. The VPC should not have any CIDR ranges that overlap with the CIDR ranges of the HSM-specific VPC.
2. Perform VPC peering between the DB VPC and the HSM VPC. For instructions, go to [VPC Peering in the Amazon Virtual Private Cloud User Guide](#).
3. Ensure that the VPC routing table is correctly associated with the VPC subnet and the VPC security group on the HSM network interface.

Note that you must configure both VPCs' routing tables so that network traffic goes to the correct VPC (from the DB VPC to the HSM VPC, and from the HSM VPC to the DB VPC). The two VPCs don't need to share the same security group, though the security groups must not prevent network traffic between the two VPCs.

Creating an Option Group with the TDE_HSM Option

The **TDE_HSM** option can be added to an existing option group just like other Oracle options, or you can create a new option group and add the **TDE_HSM** option. The following Amazon RDS CLI example creates an option group for Oracle Enterprise Edition 11.2 named *tdehsm-option-group*.

For Linux, OS X, or Unix:

```
aws rds create-option-group \
--option-group-name tdehsm-option-group \
--option-group-description "Option Group with TDE_HSM" \
--engine-name oracle-ee \
--major-engine-version 11.2
```

For Windows:

```
aws rds create-option-group ^
--option-group-name tdehsm-option-group ^
--option-group-description "Option Group with TDE_HSM" ^
--engine-name oracle-ee ^
--major-engine-version 11.2
```

The output of the command should appear similar to the following example:

```
OPTIONGROUP  tdehsm-option-group  oracle-ee  11.2  Option Group with TDE_HSM  n
```

Once the option group has been created, you can use the following command to add the **TDE_HSM** option to the option group.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
--option-group-name tdehsm-option-group \
```

```
--option-name OptionName=TDE_HSM
```

For Windows:

```
aws rds add-option-to-option-group ^
--option-group-name tdehsm-option-group ^
--option-name OptionName=TDE_HSM
```

The output of the command should appear similar to the following example:

```
OPTION TDE_HSM y n Oracle Advanced Security - TDE with HSM
```

Adding the AWS CloudHSM Classic Parameters to an Oracle DB Instance

An Oracle Enterprise Edition DB instance that uses AWS CloudHSM Classic must have two new parameters added to the DB instance. The `tde-credential-arn` and `tde-credential-password` parameters are new parameters you must include when creating a new DB instance or when modifying an existing DB instance to use AWS CloudHSM Classic.

Creating a New Oracle DB Instance with Additional Parameters for AWS CloudHSM Classic

When creating a new DB instance to use with AWS CloudHSM Classic, there are several requirements:

- You must include the option group that contains the `TDE_HSM` option
- You must provide values for the `tde-credential-arn` and `tde-credential-password` parameters. The `tde-credential-arn` parameter value is the Amazon Resource Number (ARN) of the HA partition group returned from the `create-hapg` command. You can also retrieve the ARNs of all of your high-availability partition groups with the `list-hapgs` command.

The `tde-credential-password` is the partition password you used when you initialized the HA partition group.

- The IAM Role that provides cross-service access must be created.
- You must create an Oracle Enterprise Edition DB instance.

The following command creates a new Oracle Enterprise Edition DB instance called `HsmInstance-test01` that includes the two parameters that provide AWS CloudHSM Classic access and uses an option group called `tdehsm-option-group`.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
--db-instance-identifier HsmInstance-test01 \
--db-instance-class <instance class> \
--engine oracle-ee \
--tde-credential-arn <ha partition group ARN> \
--tde-credential-password <partition password> \
--db-name <Oracle DB instance name> \
--db-subnet-group-name <subnet group name> \
--connection-timeout <connection timeout value> \
--master-user-password <master user password> \
--master-username <master user name> \
--allocated-storage <storage value> \
--option-group-name <TDE option group>
```

For Windows:

```
aws rds create-db-instance ^
```

```
--db-instance-identifier HsmInstance-test01 ^
--db-instance-class <instance class> ^
--engine oracle-ee ^
--tde-credential-arn <ha partition group ARN> ^
--tde-credential-password <partition password> ^
--db-name <Oracle DB instance name> ^
--db-subnet-group-name <subnet group name> ^
--connection-timeout <connection timeout value> ^
--master-user-password <master user password> ^
--master-username <master user name> ^
--allocated-storage <storage value> ^
--option-group-name <TDE option group>
```

The output of the command should appear similar to the following example:

```
DBINSTANCE hsminstance-test01 db.m1.medium oracle-ee 40 fooooo creating
1 **** n 11.2.0.4.v7 bring-your-own-license AL52UTF8 n
    VPCSECGROUP sg-922xvc2fd active
SUBNETGROUP dev-test test group Complete vpc-3facfe54
    SUBNET subnet-1fd6a337 us-east-1e Active
    SUBNET subnet-28aeff43 us-east-1c Active
    SUBNET subnet-5daeff36 us-east-1b Active
    SUBNET subnet-2caeef47 us-east-1d Active
PARAMGRP default.oracle-ee-11.2 in-sync
OPTIONGROUP tdehsm-option-group pending-apply
```

Modifying an Existing DB Instance to Add Parameters for AWS CloudHSM Classic

The following command modifies an existing Oracle Enterprise Edition DB instance and adds the `tde-credential-arn` and `tde-credential-password` parameters. Note that you must also include in the command the option group that contains the **TDE_HSM** option.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier hsm03 \
--tde-credential-arn <ha partition group ARN> \
--tde-credential-password <partition password> \
--option-group <tde hsm option group> \
--apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier hsm03 ^
--tde-credential-arn <ha partition group ARN> ^
--tde-credential-password <partition password> ^
--option-group <tde hsm option group> ^
--apply-immediately
```

The output of the command should appear similar to the following example:

```
DBINSTANCE hsm03 2014-04-03T18:48:53.106Z db.m1.medium oracle-ee 40 fooooo available
hsm03.c1iibpgwvdf0.us-east-1.rds.amazonaws.com 1521 us-east-1e 1
n 11.2.0.4.v7 bring-your-own-license AL32UTF8 n
    VPCSECGROUP sg-922dc2fd active
SUBNETGROUP dev-test test group Complete vpc-3faffe54
    SUBNET subnet-1fd6a337 us-east-1e Active
    SUBNET subnet-28aeff43 us-east-1c Active
```

```
SUBNET subnet-5daeff36 us-east-1b Active
SUBNET subnet-2caeef47 us-east-1d Active
PARAMGRP default.oracle-ee-11.2 in-sync
OPTIONGROUP tdehsm-option-group pending-apply
OPTIONGROUP default:oracle-ee-11-2 pending-removal
```

Verifying the HSM Connection, the Oracle Keys in the HSM, and the TDE Key

Once you have completed all the set up steps, you can verify the HSM is working properly for TDE key storage. Connect to the Oracle DB instance using a SQL utility such as *sqlplus* on a client computer or from the Amazon EC2 control instance if it has *sqlplus* installed. For more information on connecting to an Oracle DB instance, see [Connecting to a DB Instance Running the Oracle Database Engine](#).

Note

Before you continue, you must verify that the option group that you created for your Oracle instance returns a status of *in-sync*. You can verify this passing the DB instance identifier to the *describe-db-instances* command.

Verifying the HSM Connection

You can verify the connection between an Oracle DB instance and the HSM. Connect to the Oracle DB instance and use the following command:

```
select * from v$encryption_wallet;
```

If the HSM connection is working, the command should return a status of *OPEN*. The output of the command is similar to the following example:

```
WRL_TYPE
-----
WRL_PARAMETER
-----
STATUS
-----
HSM
OPEN

1 row selected.
```

Verifying the Oracle Keys in the HSM

Once Amazon RDS starts and Oracle is running, Oracle creates two master keys on the HSM. Do the following steps to confirm the existence of the master keys in the HSM. You can run these commands from the prompt on the Amazon EC2 control instance or from the Amazon RDS Oracle DB instance.

1. Use SSH to connect to the HSM appliance. The following command

```
$ ssh manager@10.0.203.58
```

2. Log in to the HSM as the HSM manager

```
$ hsm login
```

3. Once you have successfully logged in, the Luna Shell prompt appears ([hostname]lunash:>). Display the contents of the HSM partition that corresponds to the Oracle DB instance using TDE. Look for two symmetric key objects that begin with "ORACLE.TDE.HSM."

```
lunash:>part showContents -par <hapg_label> -password <partition_password>
```

The following output is an example of the information returned from the command:

```
Partition Name: hapg_label
Partition SN: 154749011
Storage (Bytes): Total=102701, Used=348, Free=102353
Number objects: 2

Object Label: ORACLE.TDE.HSM.MK.0699468E1DC88E4F27BF426176B94D4907
Object Type: Symmetric Key

Object Label: ORACLE.TSE.HSM.MK.0784B1918AB6C19483189B2296FAE261C70203
Object Type: Symmetric Key

Command Result : 0 (Success)
```

Verifying the TDE Key

The final step to verifying that the TDE key is correctly stored in the HSM is to create an encrypted tablespace. The following commands creates an encrypted tablespace and shows that it is encrypted.

```
SQL> create tablespace encrypted_ts
  datafile size 50M encryption using 'AES128'
  default storage (encrypt)
/
SQL> select tablespace_Name, encrypted from dba_tablespaces where encrypted='YES'
```

The following sample output shows that the tablespace was encrypted:

TABLESPACE_NAME	ENC
ENCRYPTED_TS	YES

Restoring Encrypted DB Instances

To restore an encrypted Oracle DB instance, you can use your existing AWS CloudHSM Classic HA partition group or create a new HA partition group and copy the contents from the original partition group to the new partition group. Please update the SafeNet client on your HSM control instance if you would like to use your existing HA partition group. Then use the `restore-db-instance-from-db-snapshot` command to restore the DB instance.

To restore the instance, perform the following procedure:

1. On your AWS CloudHSM Classic control instance, create a new HA partition group as shown in [Creating Your High-Availability Partition Group \(p. 889\)](#). When you create the new HA partition group, you must specify the same partition password as the original HA partition group. Make a note of the ARN of the new HA partition group, which you will need in the next two steps.
2. On your AWS CloudHSM Classic control instance, clone the contents of the existing HA partition group to the new HA partition group with the `clone-hapg` command.

For Linux, OS X, or Unix:

```
cloudhsm clone-hapg --conf_file ~/cloudhsm.conf \
--src-hapg-arn <src_arn> \
```

```
--dest-hapg-arn <dest_arn> \
--client-arn <client_arn> \
--partition-password <partition_password>
```

For Windows:

```
cloudhsm clone-hapg --conf_file ~/cloudhsm.conf ^
--src-hapg-arn <src_arn> ^
--dest-hapg-arn <dest_arn> ^
--client-arn <client_arn> ^
--partition-password <partition_password>
```

The parameters are as follows:

<src_arn>

The identifier of the existing HA partition group.

<dest_arn>

The identifier of the new HA partition group created in the previous step.

<client_arn>

The identifier of the HSM client.

<partition_password>

The password for the member partitions. Both HA partition groups must have the same partition password.

3. To restore the DB instance, use the AWS CLI [restore-db-instance-from-db-snapshot](#) command. For the parameter `tde-credential-arn`, specify the ARN of the new HA partition group in. For the parameter `tde-credential-password`, specify the partition password for the HA partition group.

Managing a Multi-AZ Failover

You do not need to set up a AWS CloudHSM Classic HA partition group for your standby DB instance if you are using a Multi-AZ deployment. In fact, the details of a failover are handled automatically for you. During a failover, the standby instance becomes the new primary instance and the HSM continues to work with the new primary instance.

Using Oracle GoldenGate with Amazon RDS

Oracle GoldenGate (GoldenGate) is used to collect, replicate, and manage transactional data between databases. It is a log-based change data capture (CDC) and replication software package used with Oracle databases for online transaction processing (OLTP) systems. GoldenGate creates trail files that contain the most recent changed data from the source database and then pushes these files to the target database. You can use GoldenGate with Amazon RDS for Active-Active database replication, zero-downtime migration and upgrades, disaster recovery, data protection, and in-region and cross-region replication.

The following are important points to know when working with GoldenGate on Amazon RDS:

- You are responsible for setting up and managing GoldenGate for use with Amazon RDS.
- You are responsible for managing GoldenGate licensing (bring-your-own-license) for use with Amazon RDS in all AWS regions. For more information, see [Oracle Licensing \(p. 719\)](#).
- Amazon RDS supports GoldenGate for Oracle Database Standard Edition Two (SE2), Standard Edition One (SE1), Standard Edition (SE), and Enterprise Edition (EE).
- Amazon RDS supports GoldenGate for database version 11.2.0.4, 12.1.0.2, or 12.2.0.1.
- Amazon RDS supports GoldenGate version 11.2.1 and later, including 12.1, 12.2, and 12.3.
- Amazon RDS supports migration and replication across Oracle databases using GoldenGate. We do not support nor prevent customers from migrating or replicating across heterogeneous databases.
- You can use GoldenGate on Amazon RDS Oracle DB instances that use Oracle Transparent Data Encryption (TDE). To maintain the integrity of replicated data, you should configure encryption on the GoldenGate hub using EBS encrypted volumes or trail file encryption. You should also configure encryption for data sent between the GoldenGate hub and the source and target database instances. Amazon RDS Oracle DB instances support encryption with [Oracle Secure Sockets Layer \(p. 817\)](#) or [Oracle Native Network Encryption \(p. 815\)](#).
- GoldenGate DDL is supported with GoldenGate version 12.1 and later when using Integrated capture mode.

Overview

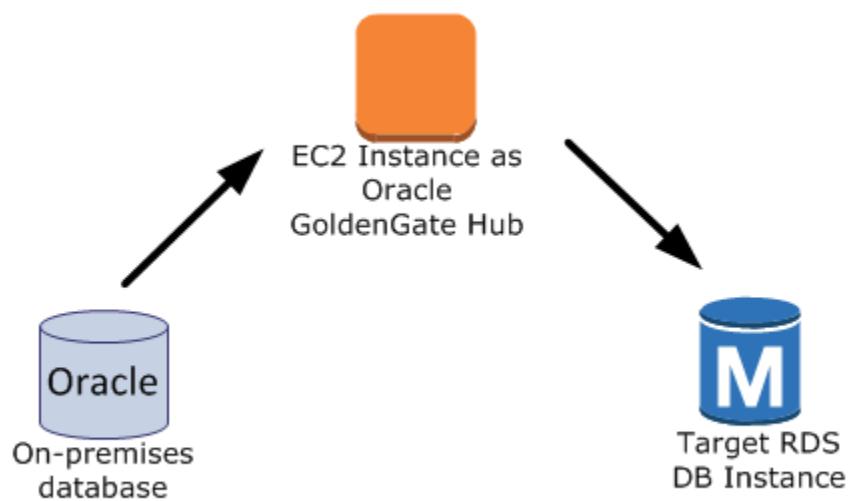
The GoldenGate architecture for use with Amazon RDS consists of three decoupled modules. The source database can be either an on-premises Oracle database, an Oracle database on an EC2 instance, or an Oracle database on an Amazon RDS DB instance. Next, the GoldenGate hub, which moves transaction information from the source database to the target database, can be either an EC2 instance with Oracle Database 11.2.0.4 and with GoldenGate 11.2.1 installed, or an on-premises Oracle installation. You can have more than one EC2 hub, and we recommend that you use two hubs if you are using GoldenGate for cross-region replication. Finally, the target database can be either on an Amazon RDS DB instance, on an EC2 instance, or on an on-premises location.

GoldenGate on Amazon RDS supports the following common scenarios:

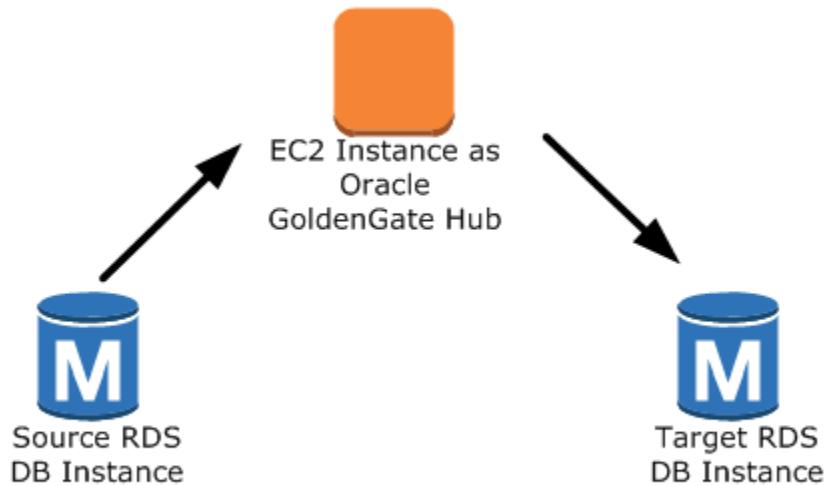
Scenario 1: An on-premises Oracle source database and on-premises GoldenGate hub, that provides data to a target Amazon RDS DB instance.



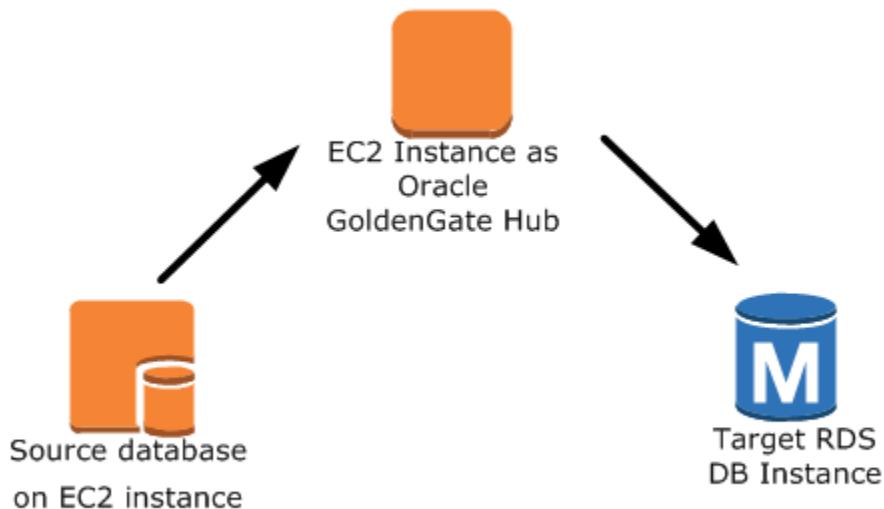
Scenario 2: An on-premises Oracle database that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance.



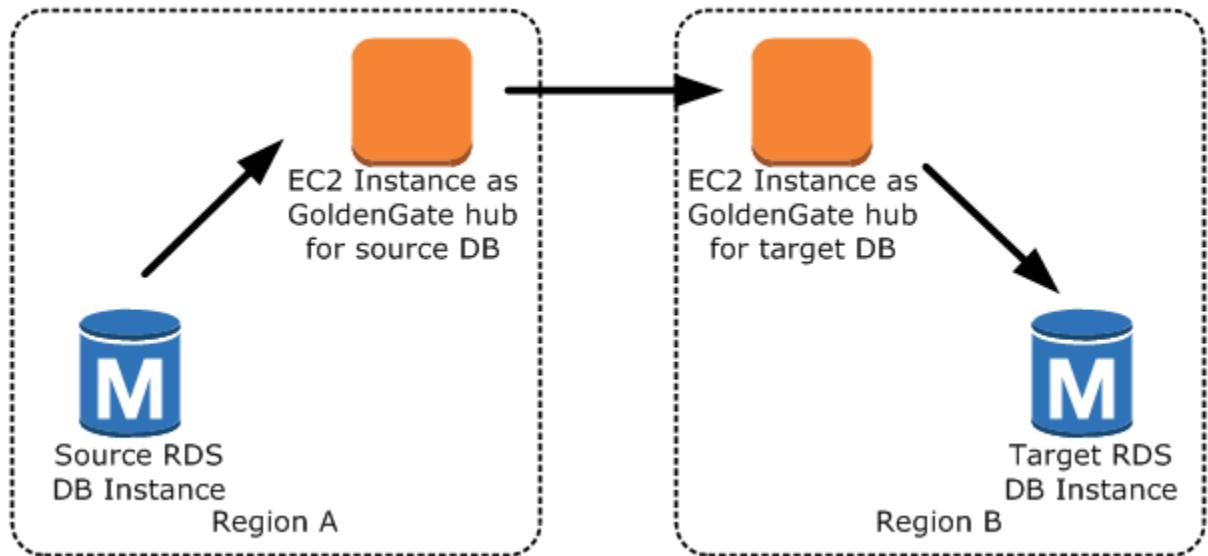
Scenario 3: An Oracle database on an Amazon RDS DB instance that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance.



Scenario 4: An Oracle database on an Amazon EC2 instance that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance.



Scenario 5: An Oracle database on an Amazon RDS DB instance connected to an Amazon EC2 instance hub in the same region, connected to an Amazon EC2 instance hub in a different region that provides data to the target Amazon RDS DB instance in the same region as the second EC2 instance hub.



Note

Any issues that impact running GoldenGate on an on-premises environment will also impact running GoldenGate on AWS. We strongly recommend that you monitor the GoldenGate hub to ensure that EXTRACT and REPLICAT are resumed if a failover occurs. Since the GoldenGate hub is run on an Amazon EC2 instance, Amazon RDS does not manage the GoldenGate hub and cannot ensure that it is running.

You can use GoldenGate using Amazon RDS to upgrade to major versions of Oracle. For example, you can use GoldenGate using Amazon RDS to upgrade from an Oracle version 8 on-premises database to an Oracle database running version 11.2.0.4 on an Amazon RDS DB instance.

To set up GoldenGate using Amazon RDS, you configure the hub on the EC2 instance, and then configure the source and target databases. The following steps show how to set up GoldenGate for use with Amazon RDS. Each step is explained in detail in the following sections:

- [Setting Up a GoldenGate Hub on EC2 \(p. 905\)](#)
- [Setting Up a Source Database for Use with GoldenGate on Amazon RDS \(p. 906\)](#)
- [Setting Up a Target Database for Use with GoldenGate on Amazon RDS \(p. 909\)](#)
- [Working with the EXTRACT and REPLICAT Utilities of GoldenGate \(p. 910\)](#)

Setting Up a GoldenGate Hub on EC2

There are several steps to creating a GoldenGate hub on an Amazon EC2 instance. First, you create an EC2 instance with a full installation of Oracle DBMS 11g version 11.2.0.4. The EC2 instance must also have GoldenGate 11.2.1 software installed, and you must have Oracle patch 13328193 installed. For more information about installing GoldenGate, see the [Oracle documentation](#).

Since the EC2 instance that is serving as the GoldenGate hub stores and processes the transaction information from the source database into trail files, you must have enough allocated storage to store the trail files. You must also ensure that the EC2 instance has enough processing power to manage the amount of data being processed and enough memory to store the transaction information before it is written to the trail file.

The following tasks set up a GoldenGate hub on an Amazon EC2 instance; each task is explained in detail in this section. The tasks include:

- Add an alias to the tnsname.ora file
- Create the GoldenGate subdirectories
- Update the GLOBALS parameter file
- Configure the mgr.prm file and start the *manager*

Add the following entry to the tnsname.ora file to create an alias. For more information on the tnsname.ora file, see the [Oracle documentation](#).

```
$ cat /example/config/tnsnames.ora
TEST=
(DESCRIPTION=
  (ENABLE=BROKEN)
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP)(HOST=goldengate-test.abcdef12345.us-west-2.rds.amazonaws.com)
     (PORT=8200))
  )
  (CONNECT_DATA=
    (SID=ORCL)
  )
)
```

Next, create subdirectories in the GoldenGate directory using the EC2 command line shell and *ggsci*, the GoldenGate command interpreter. The subdirectories are created under the gg directory and include directories for parameter, report, and checkpoint files.

```
prompt$ cd /gg
prompt$ ./ggsci
GGSCI> CREATE SUBDIRS
```

Create a GLOBALS parameter file using the EC2 command line shell. Parameters that affect all GoldenGate processes are defined in the GLOBALS parameter file. The following example creates the necessary file:

```
$ cd $GGHOME
$ vi GLOBALS
CheckpointTable oggadm1.oggchkpt
```

The last step in setting up and configuring the GoldenGate hub is to configure the *manager*. Add the following lines to the mgr.prm file, then start the *manager* using *ggsci*:

```
PORT 8199
PurgeOldExtracts ./dirdat/*, UseCheckpoints, MINKEEPDAYS 5
```

```
GGSCI> start mgr
```

Once you have completed these steps, the GoldenGate hub is ready for use. Next, you set up the source and target databases.

Setting Up a Source Database for Use with GoldenGate on Amazon RDS

When your source database is running version 11.2.0.4 or later, there are three tasks you need to accomplish to set up a source database for use with GoldenGate:

- Set the compatible parameter to 11.2.0.4 or later.

- Set the `ENABLE_GOLDENGATE_REPLICATION` parameter to *True*. This parameter turns on supplemental logging for the source database. If your source database is on an Amazon RDS DB instance, you must have a parameter group assigned to the DB instance with the `ENABLE_GOLDENGATE_REPLICATION` parameter set to *true*. For more information about the `ENABLE_GOLDENGATE_REPLICATION` parameter, see the [Oracle documentation](#).
- Set the retention period for archived redo logs for the GoldenGate source database.
- Create a GoldenGate user account on the source database.
- Grant the necessary privileges to the GoldenGate user.

The source database must have the `compatible` parameter set to 11.2.0.4 or later. If you are using an Oracle database on an Amazon RDS DB instance as the source database, you must have a parameter group with the `compatible` parameter set to 11.2.0.4 or later associated with the DB instance. If you change the `compatible` parameter in a parameter group associated with the DB instance, the change requires an instance reboot. You can use the following Amazon RDS CLI commands to create a new parameter group and set the `compatible` parameter. Note that you must associate the new parameter group with the source DB instance:

For Linux, OS X, or Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name example-goldengate \
    --description "Parameters to allow GoldenGate" \
    --db-parameter-group-family oracle-ee-11.2

aws rds modify-db-parameter-group \
    --db-parameter-group-name example-goldengate \
    --parameters "ParameterName=compatible, ParameterValue=11.2.0.4, ApplyMethod=pending-reboot"

aws rds modify-db-instance \
    --db-instance-identifier example-test \
    --db-parameter-group-name example-goldengate \
    --apply-immediately

aws rds reboot-db-instance \
    --db-instance-identifier example-test
```

For Windows:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name example-goldengate ^
    --description "Parameters to allow GoldenGate" ^
    --db-parameter-group-family oracle-ee-11.2

aws rds modify-db-parameter-group ^
    --db-parameter-group-name example-goldengate ^
    --parameters "ParameterName=compatible, ParameterValue=11.2.0.4, ApplyMethod=pending-reboot"

aws rds modify-db-instance ^
    --db-instance-identifier example-test ^
    --db-parameter-group-name example-goldengate ^
    --apply-immediately

aws rds reboot-db-instance ^
    --db-instance-identifier example-test
```

Always retain the parameter group with the `compatible` parameter. If you restore an instance from a DB snapshot, you must modify the restored instance to use the parameter group that has a matching or

greater compatible parameter value. This should be done as soon as possible after the restore action and will require a reboot of the instance.

The `ENABLE_GOLDENGATE_REPLICATION` parameter, when set to *True*, turns on supplemental logging for the source database and configures the required GoldenGate permissions. If your source database is on an Amazon RDS DB instance, you must have a parameter group assigned to the DB instance with the `ENABLE_GOLDENGATE_REPLICATION` parameter set to *true*. For more information about the `ENABLE_GOLDENGATE_REPLICATION` parameter, see the [Oracle documentation](#).

The source database must also retain archived redo logs. For example, the following command sets the retention period for archived redo logs to 24 hours:

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours', 24);
```

The duration for log retention is specified in hours. The duration should exceed any potential downtime of the source instance or any potential communication/networking issues to the source instance, so that GoldenGate can recover logs from the source instance as needed. The absolute minimum value required is one (1) hour of logs retained.

A log retention setting that is too small will result in the following message:

```
ERROR OGG-02028 Failed to attach to logmining server OGG$<extract_name> error 26927 -  
ORA-26927: altering an outbound server with a remote capture is not allowed.
```

Because these logs are retained on your DB instance, you need to ensure that you have enough storage available on your instance to accommodate the log files. To see how much space you have used in the last "X" hours, use the following query, replacing "X" with the number of hours.

```
select sum(blocks * block_size) bytes from v$archived_log  
where next_time>=sysdate-X/24 and dest_id=1;
```

GoldenGate runs as a database user and must have the appropriate database privileges to access the redo and archive logs for the source database, so you must create a GoldenGate user account on the source database. For more information about the permissions for a GoldenGate user account, see the sections 4, section 4.4, and table 4.1 in the [Oracle documentation](#).

The following statements create a user account named `oggadm1`:

```
CREATE tablespace administrator;  
CREATE USER oggadm1 IDENTIFIED BY "XXXXXX"  
  default tablespace ADMINISTRATOR temporary tablespace TEMP;
```

Finally, grant the necessary privileges to the GoldenGate user account. The following statements grant privileges to a user named `oggadm1`:

```
grant create session, alter session to oggadm1;  
grant resource to oggadm1;  
grant select any dictionary to oggadm1;  
grant flashback any table to oggadm1;  
grant select any table to oggadm1;  
grant select_catalog_role to <RDS instance master username> with admin option;  
exec RDSADMIN.RDSADMIN_UTIL.GRANT_SYS_OBJECT ('DBA_CLUSTERS', 'OGGADM1');  
grant execute on dbms_flashback to oggadm1;  
grant select on SYS.v_$database to oggadm1;  
grant alter any table to oggadm1;
```

```
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE (grantee=>'OGGADM1',
    privilege_type=>'capture',
    grant_select_privileges=>true,
    do_grants=>TRUE);
```

Setting Up a Target Database for Use with GoldenGate on Amazon RDS

The following tasks set up a target DB instance for use with GoldenGate:

- Set the compatible parameter to 11.2.0.4 or later
- Set the ENABLE_GOLDENGATE_REPLICATION parameter to *True*. If your target database is on an Amazon RDS DB instance, you must have a parameter group assigned to the DB instance with the ENABLE_GOLDENGATE_REPLICATION parameter set to *true*. For more information about the ENABLE_GOLDENGATE_REPLICATION parameter, see the [Oracle documentation](#).
- Create and manage a GoldenGate user account on the target database
- Grant the necessary privileges to the GoldenGate user

GoldenGate runs as a database user and must have the appropriate database privileges, so you must create a GoldenGate user account on the target database. The following statements create a user named *oggadm1*:

```
create tablespace administrator;
create tablespace administrator_idx;
CREATE USER oggadm1 IDENTIFIED BY "XXXXXX"
    default tablespace ADMINISTRATOR
    temporary tablespace TEMP;
alter user oggadm1 quota unlimited on ADMINISTRATOR;
alter user oggadm1 quota unlimited on ADMINISTRATOR_IDX;
```

Finally, grant the necessary privileges to the GoldenGate user account. The following statements grant privileges to a user named *oggadm1*:

```
grant create session      to oggadm1;
grant alter session       to oggadm1;
grant CREATE CLUSTER     to oggadm1;
grant CREATE INDEXTYPE   to oggadm1;
grant CREATE OPERATOR    to oggadm1;
grant CREATE PROCEDURE   to oggadm1;
grant CREATE SEQUENCE    to oggadm1;
grant CREATE TABLE        to oggadm1;
grant CREATE TRIGGER     to oggadm1;
grant CREATE TYPE         to oggadm1;
grant select any dictionary to oggadm1;
grant create any table   to oggadm1;
grant alter any table    to oggadm1;
grant lock any table     to oggadm1;
grant select any table   to oggadm1;
grant insert any table   to oggadm1;
grant update any table   to oggadm1;
grant delete any table   to oggadm1;

EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE
    (grantee=>'OGGADM1',privilege_type=>'apply',
    grant_select_privileges=>true, do_grants=>TRUE);
```

Working with the EXTRACT and REPLICAT Utilities of GoldenGate

The GoldenGate utilities `EXTRACT` and `REPLICAT` work together to keep the source and target databases in sync via incremental transaction replication using trail files. All changes that occur on the source database are automatically detected by `EXTRACT`, then formatted and transferred to trail files on the GoldenGate on-premises or EC2-instance hub. After initial load is completed, the data is read from these files and replicated to the target database by the `REPLICAT` utility.

Running GoldenGate's EXTRACT Utility

The `EXTRACT` utility retrieves, converts, and outputs data from the source database to trail files. `EXTRACT` queues transaction details to memory or to temporary disk storage. When the transaction is committed to the source database, `EXTRACT` flushes all of the transaction details to a trail file for routing to the GoldenGate on-premises or EC2-instance hub and then to the target database.

The following tasks enable and start the `EXTRACT` utility:

- Configure the `EXTRACT` parameter file on the GoldenGate hub (on-premises or EC2 instance). The following listing shows an example `EXTRACT` parameter file.

```
EXTRACT EABC
SETENV (ORACLE_SID=ORCL)
SETENV (NLSLANG=AL32UTF8)

USERID oggadm1@TEST, PASSWORD XXXXXX
EXTTRAIL /path/to/goldengate/dirdat/ab

IGNOREREPLICATES
GETAPPLOPS
TRANLOGOPTIONS EXCLUDEUSER OGGADM1

TABLE EXAMPLE.TABLE;
```

- On the GoldenGate hub, launch the GoldenGate command line interface (`ggsci`). Log into the source database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Add a checkpoint table for the database:

```
add checkpointtable
```

- Add transdata to turn on supplemental logging for the database table:

```
add transdata <user>.<table>
```

Alternatively, you can add transdata to turn on supplemental logging for all tables in the database:

```
add transdata <user>.*
```

- Using the `ggsci` command line, enable the `EXTRACT` utility using the following commands:

```
add extract <extract name> tranlog, INTEGRATED tranlog, begin now
add exttrail <path-to-trail-from-the param-file>
    extract <extractname-from-paramfile>,
    MEGABYTES Xm
```

- Register the EXTRACT utility with the database so that the archive logs are not deleted. This allows you to recover old, uncommitted transactions if necessary. To register the EXTRACT utility with the database, use the following command:

```
register EXTRACT <extract process name>, DATABASE
```

- To start the EXTRACT utility, use the following command:

```
start <extract process name>
```

Running GoldenGate's REPLICAT Utility

The REPLICAT utility is used to "push" transaction information in the trail files to the target database.

The following tasks enable and start the REPLICAT utility:

- Configure the REPLICAT parameter file on the GoldenGate hub (on-premises or EC2 instance). The following listing shows an example REPLICAT parameter file.

```
REPLICAT RABC
SETENV (ORACLE_SID=ORCL)
SETENV (NLSLANG=AL32UTF8)

USERID oggadm1@TARGET, password XXXXXX

ASSUMETARGETDEFS
MAP EXAMPLE.TABLE, TARGET EXAMPLE.TABLE;
```

- Launch the GoldenGate command line interface (ggsci). Log into the target database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Using the ggsci command line, add a checkpoint table. Note that the user indicated should be the GoldenGate user account, not the target table schema owner. The following example creates a checkpoint table named *gg_checkpoint*.

```
add checkpointtable <user>.gg_checkpoint
```

- To enable the REPLICAT utility, use the following command:

```
add replicat <replicat name> EXTTRAIL <extract trail file> CHECKPOINTTABLE
<user>.gg_checkpoint
```

- To start the REPLICAT utility, use the following command:

```
start <replicat name>
```

Troubleshooting Issues When Using GoldenGate with Amazon RDS

This section explains the most common issues when using GoldenGate with Amazon RDS.

Topics

- [Log Retention \(p. 912\)](#)
- [GoldenGate appears to be properly configured but replication is not working \(p. 912\)](#)

Log Retention

You must have log retention enabled. If you do not, or if the retention value is too small, you will see the following message:

```
2014-03-06 06:17:27  ERROR  OGG-00446  error 2 (No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log
for sequence 1306Not able to establish initial position for begin time 2014-03-06
06:16:55.
```

GoldenGate appears to be properly configured but replication is not working

For pre-existing tables, GoldenGate needs to be told which SCN it should work from. Take the following steps to fix this issue:

- Launch the GoldenGate command line interface (ggsci). Log into the source database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Using the ggsci command line, set up the start SCN for the EXTRACT process. The following example sets the SCN to 223274 for the extract:

```
ALTER EXTRACT <extract process name> SCN 223274
start <extract process name>
```

- Log into the target database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Using the ggsci command line, set up the start SCN for the REPLICAT process. The following example sets the SCN to 223274 for the REPLICAT:

```
start <replicat process name> atcsn 223274
```

Using the Oracle Repository Creation Utility on Amazon RDS for Oracle

You can use Amazon RDS to host an Oracle DB instance that holds the schemas to support your Fusion Middleware components. Before you can use Fusion Middleware components, you must create and populate schemas for them in your database. You create and populate the schemas by using the Oracle Repository Creation Utility (RCU).

You can store the schemas for any Fusion Middleware components in your Amazon RDS DB instance. The following is a list of schemas that have been verified to install correctly:

- Analytics (ACTIVITIES)
- Audit Services (IAU)
- Audit Services Append (IAU_APPEND)
- Audit Services Viewer (IAU_VIEWER)
- Discussions (DISCUSSIONS)
- Metadata Services (MDS)
- Oracle Business Intelligence (BIPLATFORM)
- Oracle Platform Security Services (OPSS)
- Portal and Services (WEBCENTER)
- Portlet Producers (PORTLET)
- Service Table (STB)
- SOA Infrastructure (SOAINFRA)
- User Messaging Service (UCSUMS)
- WebLogic Services (WLS)

Licensing and Versions

Amazon RDS supports Oracle Repository Creation Utility (RCU) version 12c only. You can use the RCU in the following configurations:

- RCU 12c with Oracle database 12.2.0.1
- RCU 12c with Oracle database 12.1.0.2.v4 or later
- RCU 12c with Oracle database 11.2.0.4.v8 or later

Before you can use RCU, you need a license for Oracle Fusion Middleware. You also need to follow the Oracle licensing guidelines for the Oracle database that hosts the repository. For more information, see [Oracle Fusion Middleware Licensing Information User Manual](#) in the Oracle documentation.

Fusion MiddleWare supports repositories on Oracle Database Enterprise Edition and Standard Editions (SE, SE One, or SE Two). Oracle recommends Enterprise Edition for production installations that require partitioning and installations that require online index rebuild.

Before you create your Oracle DB instance, confirm the Oracle database version that you need to support the components that you want to deploy. You can use the Certification Matrix to find the requirements for the Fusion Middleware components and versions you want to deploy. For more information, see [Oracle Fusion Middleware Supported System Configurations](#) in the Oracle documentation.

Amazon RDS supports Oracle database version upgrades as needed. For more information, see [Upgrading a DB Instance Engine Version \(p. 123\)](#).

Before You Begin

Before you begin, you need an Amazon VPC. Because your Amazon RDS DB instance needs to be available only to your Fusion Middleware components, and not to the public Internet, your Amazon RDS DB instance is hosted in a private subnet, providing greater security. For information about how to create an Amazon VPC for use with an Oracle DB instance, see [Creating an Amazon VPC for Use with an Oracle Database \(p. 875\)](#).

Before you begin, you also need an Oracle DB instance. For information about how to create an Oracle DB instance for use with Fusion Middleware metadata, see [Creating an Oracle DB Instance \(p. 880\)](#).

Recommendations

The following are some recommendations for working with your DB instance in this scenario:

- We recommend that you use Multi-AZ for production workloads. For more information about working with multiple Availability Zones, see [Regions and Availability Zones \(p. 101\)](#).
- For additional security, Oracle recommends that you use Transparent Data Encryption (TDE) to encrypt your data at rest. If you have an Enterprise Edition license that includes the Advanced Security Option, you can enable encryption at rest by using the TDE option. For more information, see [Oracle Transparent Data Encryption \(p. 836\)](#).

Amazon RDS also provides an encryption at rest option for all database editions. For more information, see [Encrypting Amazon RDS Resources \(p. 389\)](#).

- Configure your VPC Security Groups to allow communication between your application servers and your Amazon RDS DB instance. The application servers that host the Fusion Middleware components can be on Amazon EC2 or on-premises.

Using the Oracle Repository Creation Utility

You use the Oracle Repository Creation Utility (RCU) to create and populate the schemas to support your Fusion Middleware components.

Running RCU Using the Command Line in One Step

If you don't need to edit any of your schemas before populating them, you can run RCU in a single step. Otherwise, see the following section for running RCU in multiple steps.

You can run the RCU in silent mode by using the command-line parameter `-silent`. When you run RCU in silent mode, you can avoid typing passwords on the command line by creating a text file containing the passwords. Create a text file with the password for `dbUser` on the first line, and the password for each component on subsequent lines. You specify the name of the password file as the last parameter to the RCU command.

Example

The following example creates and populates schemas for the SOA Infrastructure component (and its dependencies) in a single step.

For Linux, OS X, or Unix:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
```

```
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-f < /tmp/passwordfile.txt
```

For more information, see [Running Repository Creation Utility from the Command Line](#) in the Oracle documentation.

Running RCU Using the Command Line in Multiple Steps

If you need to manually edit your schema scripts, you can run the RCU in multiple steps:

1. Run RCU in **Prepare Scripts for System Load** mode by using the `-generateScript` command-line parameter to create the scripts for your schemas.
2. Manually edit and run the generated script `script_systemLoad.sql`.
3. Run RCU again in **Perform Product Load** mode by using the `-dataLoad` command-line parameter to populate the schemas.
4. Run the generated clean-up script `script_postDataLoad.sql`.

You can run the RCU in silent mode by using the command-line parameter `-silent`. When you run RCU in silent mode, you can avoid typing passwords on the command line by creating a text file containing the passwords. Create a text file with the password for `dbUser` on the first line, and the password for each component on subsequent lines. You specify the name of the password file as the last parameter to the RCU command.

Example

The following example creates schema scripts for the SOA Infrastructure component (and its dependencies).

For Linux, OS X, or Unix:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-generateScript \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
[-encryptTablespace true] \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
```

```
-component WLS \
-component SOAINFRA \
-scriptLocation /tmp/rcuscripts \
-f < /tmp/passwordfile.txt
```

Now you can edit the generated script, connect to your Oracle DB instance, and run the script. The generated script is named `script_systemLoad.sql`. For information about connecting to your Oracle DB instance, see [Connecting to Your Sample Oracle DB Instance \(p. 39\)](#).

The following example populates the schemas for the SOA Infrastructure component (and its dependencies).

For Linux, OS X, or Unix:

```
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-dataLoad \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-f < /tmp/passwordfile.txt
```

To finish, you connect to your Oracle DB instance, and run the clean-up script. The script is named `script_postDataLoad.sql`.

For more information, see [Running Repository Creation Utility from the Command Line](#) in the Oracle documentation.

Running RCU in Interactive Mode

To use the RCU graphical user interface, you can run RCU in interactive mode. To run RCU in interactive mode, include the `-interactive` parameter and omit the `-silent` parameter. For more information, see [Understanding Repository Creation Utility Screens](#) in the Oracle documentation.

Example

The following example starts RCU in interactive mode and pre-populates the connection information.

For Linux, OS X, or Unix:

```
export ORACLE_HOME=u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-interactive \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal
```

Known Issues

The following are some known issues for working with RCU, with some troubleshooting suggestions:

- Oracle Managed Files (OMF) — Amazon RDS uses OMF data files to simplify storage management. You can customize tablespace attributes, such as size and extent management. However, specifying a data file name when you run RCU causes tablespace code to fail with ORA-20900. The RCU can be used with OMF in the following ways:
 - In RCU 12.2.1.0 and later, use the `-honorOMF` command-line parameter.
 - In RCU 12.1.0.3 and later, use multiple steps and edit the generated script. For more information, see [Running RCU Using the Command Line in Multiple Steps \(p. 915\)](#).
- SYSDBA — Because Amazon RDS is a managed service, you don't have full SYSDBA access to your Oracle DB instance. However, RCU 12c supports users with lower privileges. In most cases, the master user privilege is sufficient to create repositories. In some cases, the RCU might fail with ORA-01031 when attempting to grant SYS object privileges. You can retry and run the `RDSADMIN_UTIL.GRANT_SYS_OBJECT()` stored procedure, or contact AWS Support.
- Dropping Enterprise Scheduler Service — When you use the RCU to drop an Enterprise Scheduler Service repository, the RCU might fail with `Error: Component drop check failed`.

Related Topics

- [Oracle Licensing \(p. 719\)](#)

Installing a Siebel Database on Oracle on Amazon RDS

You can use Amazon RDS to host a Siebel Database on an Oracle DB instance. The Siebel Database is part of the Siebel Customer Relationship Management (CRM) application architecture. For an illustration, see [Generic Architecture of Siebel Business Application](#).

Use the following topic to help set up a Siebel Database on an Oracle DB instance on Amazon RDS. You can also find out how to use Amazon Web Services to support the other components required by the Siebel CRM application architecture.

Note

To install a Siebel Database on Oracle on Amazon RDS, you need to use the master user account. You don't need SYSDBA privilege; master user privilege is sufficient. For more information, see [Master User Account Privileges \(p. 407\)](#).

Licensing and Versions

To install a Siebel Database on Amazon RDS, you must use your own Oracle Database license, and your own Siebel license. You must have the appropriate Oracle Database license (with Software Update License and Support) for the DB instance class and Oracle Database edition. For more information, see [Oracle Licensing \(p. 719\)](#).

Oracle Database Enterprise Edition is the only edition certified by Siebel for this scenario. Amazon RDS supports Siebel CRM version 15.0 or 16.0. Use Oracle 12c, version 12.1.0.2.0. For the procedures following, we use Siebel CRM version 15.0 and Oracle 12.1.0.2 or 12.2.0.1. For more information, see [Oracle 12c with Amazon RDS \(p. 724\)](#).

Amazon RDS supports database version upgrades. For more information, see [Upgrading a DB Instance Engine Version \(p. 123\)](#).

Before You Begin

Before you begin, you need an Amazon VPC. Because your Amazon RDS DB instance needs to be available only to your Siebel Enterprise Server, and not to the public Internet, your Amazon RDS DB instance is hosted in a private subnet, providing greater security. For information about how to create an Amazon VPC for use with Siebel CRM, see [Creating an Amazon VPC for Use with an Oracle Database \(p. 875\)](#).

Before you begin, you also need an Oracle DB instance. For information about how to create an Oracle DB instance for use with Siebel CRM, see [Creating an Oracle DB Instance \(p. 880\)](#).

Installing and Configuring a Siebel Database

After you create your Oracle DB instance, you can install your Siebel Database. You install the database by creating table owner and administrator accounts, installing stored procedures and functions, and then running the Siebel Database Configuration Wizard. For more information, see [Installing the Siebel Database on the RDBMS](#).

To run the Siebel Database Configuration Wizard, you need to use the master user account. You don't need SYSDBA privilege; master user privilege is sufficient. For more information, see [Master User Account Privileges \(p. 407\)](#).

Using Other Amazon RDS Features with a Siebel Database

After you create your Oracle DB instance, you can use additional Amazon RDS features to help you customize your Siebel Database.

Collecting Statistics with the Oracle Statspack Option

You can add features to your DB instance through the use of options in DB option groups. When you created your Oracle DB instance, you used the default DB option group. If you want to add features to your database, you can create a new option group for your DB instance.

If you want to collect performance statistics on your Siebel Database, you can add the Oracle Statspack feature. For more information, see [Oracle Statspack \(p. 830\)](#).

Some option changes are applied immediately, and some option changes are applied during the next maintenance window for the DB instance. For more information, see [Working with Option Groups \(p. 156\)](#). After you create a customized option group, modify your DB instance to attach it. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

Performance Tuning with Parameters

You manage your DB engine configuration through the use of parameters in a DB parameter group. When you created your Oracle DB instance, you used the default DB parameter group. If you want to customize your database configuration, you can create a new parameter group for your DB instance.

When you change a parameter, depending on the type of the parameter, the changes are applied either immediately or after you manually reboot the DB instance. For more information, see [Working with DB Parameter Groups \(p. 169\)](#). After you create a customized parameter group, modify your DB instance to attach it. For more information, see [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#).

To optimize your Oracle DB instance for Siebel CRM, you can customize certain parameters. The following table shows some recommended parameter settings. For more information about performance tuning Siebel CRM, see [Siebel CRM Performance Tuning Guide](#).

Parameter Name	Default Value	Guidance for Optimal Siebel CRM Performance
<code>_always_semi_join</code>	<code>CHOOSE</code>	OFF
<code>_b_tree_bitmap_p</code>	<code>TRUE</code>	FALSE
<code>_like_with_bind_</code>	<code>EQUALITY</code>	TRUE
<code>_no_or_expansion</code>	<code>FALSE</code>	FALSE
<code>_optimizer_join_</code>	<code>TRUE</code>	<code>sanity_check</code> TRUE
<code>_optimizer_max_p</code>	<code>2000</code>	100
<code>_optimizer_sortmerge_</code>	<code>join_enabled</code>	<code>FALSE</code>
<code>_partition_view_</code>	<code>ENABLED</code>	FALSE
<code>open_cursors</code>	300	At least 2000 .

Creating Snapshots

After you create your Siebel Database, you can copy the database by using the snapshot features of Amazon RDS. For more information, see [Creating a DB Snapshot \(p. 216\)](#) and [Restoring from a DB Snapshot \(p. 218\)](#).

Support for Other Siebel CRM Components

In addition to your Siebel Database, you can also use Amazon Web Services to support the other components of your Siebel CRM application architecture. You can find more information about the support provided by Amazon AWS for additional Siebel CRM components in the following table.

Siebel CRM Component	Amazon AWS Support
Siebel Enterprise (with one or more Siebel Servers)	You can host your Siebel Servers on Amazon Elastic Compute Cloud (Amazon EC2) instances. You can use Amazon EC2 to launch as many or as few virtual servers as you need. Using Amazon EC2, you can scale up or down easily to handle changes in requirements. For more information, see What Is Amazon EC2? You can put your servers in the same VPC with your DB instance and use the VPC security group to access the database. For more information, see Working with an Amazon RDS DB Instance in a VPC (p. 420) .
Web Servers (with Siebel Web Server Extensions)	You can install multiple Web Servers on multiple EC2 instances. You can then use Elastic Load Balancing to distribute incoming traffic among the instances. For more information, see What Is Elastic Load Balancing?
Siebel Gateway Name Server	You can host your Siebel Gateway Name Server on an EC2 instance. You can then put your server in the same VPC with the DB instance and use the VPC security group to access the database. For more information, see Working with an Amazon RDS DB Instance in a VPC (p. 420) .

Related Topics

- [Connecting to a DB Instance Running the Oracle Database Engine \(p. 751\)](#)

Oracle Database Engine Release Notes

Updates to your Amazon RDS for Oracle DB instances keep them current. If you apply updates, you can be confident that your DB instance is running a stable, common version of the database software that has been regression-tested by both Oracle and Amazon. We don't support applying one-off patches to individual DB instances.

Oracle Version 12.2.0.1

For Amazon RDS for Oracle version 12.2.0.1, Amazon RDS incorporates bug fixes from Oracle by using Release Updates (RUs) and Release Updates Revisions (RURs).

The following RUs and RURs are applied to the Amazon RDS for Oracle version 12.2.0.1:

- [12.2.0.1.ru-2018-10.rur-2018-10.r1 \(p. 922\)](#)

Oracle Versions 12.1.0.2 and 11.2.0.4

For Amazon RDS for Oracle versions 12.1.0.2 and 11.2.0.4, Amazon RDS incorporates bug fixes from Oracle via their quarterly Database Patch Set Updates (PSUs). You can be confident that your DB instance is running a stable, common version of the database software that has been regression-tested by both Oracle and Amazon. We don't support applying one-off patches to individual DB instances.

To find what Oracle Patch Set Updates (PSUs) are applied to Amazon RDS for Oracle versions 12.1.0.2 and 11.2.0.4, see the following table.

PSU	Version 12.1.0.2	Version 11.2.0.4
2018 October	12.1.0.2.v14 (p. 925)	11.2.0.4.v18 (p. 944)
2018 July	12.1.0.2.v13 (p. 927)	11.2.0.4.v17 (p. 946)
2018 April	12.1.0.2.v12 (p. 929)	11.2.0.4.v16 (p. 947)
2018 January	12.1.0.2.v11 (p. 931)	11.2.0.4.v15 (p. 949)
2017 October	12.1.0.2.v10 (p. 932)	11.2.0.4.v14 (p. 950)
2017 July	12.1.0.2.v9 (p. 934)	11.2.0.4.v13 (p. 951)
2017 April	12.1.0.2.v8 (p. 935)	11.2.0.4.v12 (p. 953)
2017 January	12.1.0.2.v7 (p. 937)	11.2.0.4.v11 (p. 954)
2016 October	12.1.0.2.v6 (p. 938)	11.2.0.4.v10 (p. 955)
2016 July	12.1.0.2.v5 (p. 939)	11.2.0.4.v9 (p. 956)
2016 April	12.1.0.2.v4 (p. 940)	11.2.0.4.v8 (p. 957)
2016 January	12.1.0.2.v3 (p. 941)	11.2.0.4.v7 (p. 959)
2015 October	12.1.0.2.v2 (p. 942)	11.2.0.4.v6 (p. 960) 11.2.0.4.v5 (p. 960)
2015 April	12.1.0.2.v1 (p. 943)	11.2.0.4.v4 (p. 961)

PSU	Version 12.1.0.2	Version 11.2.0.4
2014 October	—	11.2.0.4.v3 (p. 962)
2014 July	—	11.2.0.4.v2 (p. 963) (Deprecated)
2014 January	—	11.2.0.4.v1 (p. 963)

Topics

- [Database Engine: 12.2.0.1 \(p. 922\)](#)
- [Database Engine: 12.1.0.2 \(p. 924\)](#)
- [Database Engine: 11.2.0.4 \(p. 944\)](#)

Database Engine: 12.2.0.1

For Oracle 12c version 12.2.0.1, Oracle changed the way it releases Oracle Database updates. Instead of Patch Set Updates (PSUs), Oracle supplies Release Updates (RUs) and Release Updates Revisions (RURs). RUs contain optimizer changes, feature additions, and security fixes. RURs only contain security fixes for the two preceding quarterly patch cycles. With this new system, you have more control over the features that you install with each update.

The naming conventions have also changed for Oracle 12c version 12.2.0.1 versions. In previous versions, Amazon RDS for Oracle used the PSU naming convention of *oracle-version.vpatch-version*. The *patch-version* corresponded with an Oracle PSU. For example, in Oracle for Amazon RDS version 12.1.0.2.v13, the v13 part of the version number corresponds with an Oracle PSU.

Oracle 12c version 12.2.0.1 naming conventions account for both RU and RUR updates. For example, the first Amazon RDS for Oracle version available is 12.2.0.1.ru-2018-10.rur-2018-10.r1. In this example, 12.2 is the major version, and 0.1 is the minor version. The revision version has the following parts:

- ru-2018-10 – the October RU
- rur-2018-10 – the October RUR for the October RU
- r1 – Internal Amazon RDS revision, which lets Amazon RDS differentiate between emergency patches of pre-existing RU/RURs

For more information about the new Oracle Database versioning system, see the posts [Differences between PSU / BP and RU / RUR](#) at the Upgrade your Database – NOW! blog and [RU and RUR patches for Oracle 12.2](#) at the Oracle–Help blog.

Version 12.2.0.1.ru-2018-10.rur-2018-10.r1

Version 12.2.0.1.ru-2018-10.rur-2018-10.r1 adds support for the following:

- October 2018 Release Update: 12.2.0.1.181016 (28662603)

Oracle Release Update 12.2.0.1.181016, Released October 2018

Bugs fixed: 28390273, 28571483, 28483184, 8480838, 13554903, 14221306, 14690846, 15931756, 16002385, 16438495, 16727454, 16942578, 17027695, 17533661, 17947871, 18308268, 18521691, 18594510, 18774543, 19072655, 19211433, 19285025, 19327292, 19526548, 19614243, 19647894, 19649997, 19721304, 20003668, 20087519, 20118035, 20120236, 20324049, 20436508, 20532077, 20591151, 20620169, 20736227, 20756305, 20866970, 20976443, 21143725, 21147908, 21159907,

21178363, 21186167, 21216226, 21320338, 21433452 21479706, 21520266, 21547051, 21744603, 21882528, 21981529, 21985256 22007324, 22070853, 22072543, 22087683, 22104866, 22179537, 22347493 22364044, 22367053, 22379010, 22446455, 22495673, 22503283, 22503297 22504793, 22530986, 22564336, 22568728, 22581771, 22594071, 22599050 22628825, 22645009, 22654475, 22700845, 22729345, 22826067, 22843979 22845846, 22864303, 22898198, 22950945, 22970869, 22981722, 23019710 23026585, 23035249, 23055900, 23061453, 23065002, 23066146, 23080557 23105538, 23110523, 23125560, 23126545, 23127945, 23151677, 23179662 23184263, 23197730, 23234232, 23249829, 23271203, 23300142, 23310101 23312077, 23481673, 23491861, 23499160, 23521523, 23527363, 23533647 23548817, 23567857, 23572982, 23581777, 23588722, 23599216, 23600861 23602213, 23645516, 23665623, 23709062, 23715460, 23730961, 23733981 23735292, 23741944, 23746128, 23749454, 24010030, 24289874, 24294174 24303148, 24307571, 24308349, 24326444, 24326846, 24332831, 24334708 24336249, 24337882, 24341675, 24343905, 24345420, 24346821, 24348685 24350620, 24368004, 24371491, 24373756, 24374976, 24376875, 24376878 24385983, 24401351, 24403922, 24415926, 24421668, 24423416, 24425056 24425998, 24435982, 24437162, 24443539, 24457597, 24461826, 24467122 24468470, 24470606, 24473736, 24485034, 24485161, 24485174, 24486059 24486237, 24509056, 24534401, 24554533, 24555417, 24556967, 24560906 24563422, 24570598, 24573817, 24578718, 24578797, 24589081, 24589590 24593740, 24595699, 24600330, 24609592, 24609996, 24616637, 24617969 24623975, 24624166, 24642495, 24654629, 24655717, 24664211, 24668398 24674197, 24674955, 24676172, 24677696, 24680959, 24689376, 24692973 24693290, 24699619, 24710696, 24713381, 24714096, 24717183, 24717859 24718260, 24719799, 24735430, 24737064, 24737403, 24737581, 24744383 24744686, 24757934, 24759556, 24760407, 24766309, 24786669, 24792678 24793511, 24796092, 24797119, 24800423, 24801152, 24802934, 24811725 24812047, 24827228, 24827654, 24831514, 24835919, 24843188, 24844549 24845157, 24848746, 24848923, 24850622, 24907917, 24908321, 24911709 24912588, 24922704, 24923080, 24923215, 24923338, 24923790, 24929210 24938784, 24940060, 24942749, 24953434, 24957555, 24960044, 24966594 24966788, 24968162, 24976007, 24978100, 25027852, 25029022, 25029423 25034396, 25036474, 25044977, 25045228, 25050160, 25051628, 25057811 25058080, 25062592, 25063971, 25065563, 25072986, 25078611, 25086233 25087436, 25093872, 25098160, 25099339, 25099497, 25099758, 25100063 25100579, 25103996, 25107662, 25110233, 25120284, 25120742, 25121089 25123585, 25124363, 25129925, 25140197, 25145163, 25145215, 25150925 25159176, 25162645, 25164293, 25166187, 25171084, 25175723, 25176408 25178032, 25178101, 25178179, 25179774, 25182817, 25184555, 25186079 25191872, 25192044, 25192729, 25199585, 25201454, 25202355, 25203656 25206864, 25207410, 25209912, 25210268, 25210499, 25211628, 25223839 25224242, 25225795, 25226665, 25227381, 25230945, 25237577, 25240590 25241448, 25241625, 25244807, 25248384, 25251648, 25257085, 25259611 25262869, 25263960, 25265499, 25283790, 25287072, 25296876, 25299227 25299807, 25300427, 25305405, 25307368, 25309116, 25313154, 25313411 25316758, 25317989, 25320555, 25323525, 25328518, 25329664, 25335249 25335360, 25335790, 25337332, 25337640, 25348956, 25353983, 25357142 25362958, 25382812, 25383204, 25384462, 25386748, 25388896, 25392535 25395696, 25397936, 25405813, 25410017, 25410180, 25410802, 25410877 25411036, 25417050, 25417056, 25417958, 25425451, 25425760, 25427662 25429959, 25430120, 25433696, 25435038, 25437699, 25440818, 25444961 25451531, 25455795, 25457409, 25459958, 25462714, 25463844, 25472112 25476149, 25478885, 25479164, 25489342, 25489367, 25489607, 25492379 25498930, 25498994, 25516250, 25524955, 25528838, 25530080, 25530814 25535668, 25536819, 25537470, 25539063, 25540738, 25546580, 25546608 25547901, 25551676, 25553616, 25554787, 25555252, 25557886, 25558986 25560487, 25561296, 25569149, 25570929, 25575348, 25575628, 25579458 25579761, 25594901, 25597525, 25598473, 25600342, 25600421, 25602488 25603923, 25606091, 25607726, 25612095, 25614866, 25616268, 25616359 25616417, 25616645, 25631933, 25633101, 25634317, 25634348, 25635149 25638456, 25639019, 25643818, 25643931, 25646373, 25647325, 25648731 25653109, 25654459, 25654936, 25655390, 25655966, 25659655, 25660847 25661819, 25662088, 25662101, 25662524, 25669791, 25670786, 25672640 25674386, 25680221, 25685152, 25686739, 25687460, 25691904, 25694206 25695903, 25700654, 25710420, 25715167, 25717371, 25722055, 25722608 25722720, 25728085, 25729507, 25734963, 25736747, 25739065, 25754606 25757748, 25760195, 25762221, 25764020, 25766822, 25768681, 25772669 25774077, 25775213, 25780343, 25784002, 25785331, 25785441, 25788879 25789041, 25789277, 25789579, 25790353, 25797092, 25797124, 25797305 25800464, 25803545, 25807997, 25810704, 25813931, 25818707, 25822410 25823754, 25825910, 25826740, 25830492, 25832935, 25834581, 25838361 25852885, 25856821, 25858672, 25861398,

25865785, 25870579, 25871177 25871639, 25871753, 25872127, 25872389, 25874050, 25874678, 25882264 25885148, 25888073, 25890056, 25890673, 25894239, 25895224, 25897615 25904273, 25904490, 25906117, 25911724, 25914276, 25919622, 25932524 25932728, 25933494, 25941836, 25943271, 25945130, 25947799, 25953857 25954022, 25954054, 25957038, 25963024, 25964954, 25967544, 25967985 25970731, 25973152, 25975723, 25977302, 25980605, 25980770, 25981498 25982666, 25990907, 25995938, 26006257, 26007010, 26019148, 26024732 26025681, 26029780, 26032573, 26036748, 26037215, 26038086, 26039623 26040483, 26045732, 26078437, 26078493, 26080410, 26083298, 26088426 26088836, 26090767, 26091640, 26091786, 26095327, 26095405, 26096382 26108080, 26110632, 26111842, 26121990, 26137367, 26138085, 26149904 26153977, 26169341, 26169345, 26170715, 26176002, 26187943, 26189861 26198757, 26198926, 26201113, 26223039, 26237431, 26237773, 26242031 26243698, 26244115, 26245237, 26249718, 26256131, 26259265, 26261327 26263328, 26263721, 26269790, 26271001, 26277439, 26285933, 26308650 26309047, 26318627, 26323308, 26324769, 26327624, 26330994, 26331743 26333141, 26338953, 26351334, 26353617, 26358670, 26362821, 26366517 26367012, 26374791, 26375250, 26380097, 26385189, 26388538, 26396790 26399626, 26399691, 26406387, 26412540, 26418088, 26420561, 26421667 26422277, 26426526, 26430737, 26434999, 26435073, 26436168, 26438612 26440749, 26442308, 26444601, 26444887, 26446098, 26452606, 26475419 26476244, 26478970, 26479173, 26486365, 26492866, 26493289, 26498354 26513709, 26521043, 26522439, 26523432, 26526726, 26536320, 26537307 26542135, 26544823, 26545688, 26546070, 26546664, 26546754, 26548363 26556014, 26569225, 26575788, 26582460, 26584641, 26597140, 26599395 26608137, 26609942, 26615291, 26615690, 26623652, 26626879, 26629381 26633355, 26633558, 26635897, 26637273, 26637824, 26639167, 26641610 26650226, 26658759, 26659182, 26680105, 26712331, 26714910, 26717528 26727397, 26729494, 26729611, 26740700, 26744595, 26751106, 26751171 26758193, 26764561, 26765212, 26775602, 26784509, 26794786, 26797591 26802503, 26820076, 26822620, 26828994, 26840654, 26844870, 26849779 26875822, 26883456, 26896659, 26898563, 26907327, 26908788, 26909100 26909504, 26911000, 26939314, 26944190, 26963310, 26966916, 26967713 26969321, 26970717, 26981902, 26983259, 26986173, 26992964, 27006664 27009164, 27013146, 27028251, 27034890, 27038986, 27039712, 27044297 27052607, 27060167, 27060859, 27073314, 27079140, 27087426, 27090765 27093423, 27110878, 27117822, 27119621, 27124624, 27125872, 27133662 27135647, 27135993, 27138325, 27142373, 27153641, 27161071, 27162405 27163928, 27165231, 27169796, 27170305, 27181537, 27199245, 27207110 27213224, 27229389, 27244337, 27248917, 27250547, 27251690, 27255377 27256000, 27259307, 27262945, 27274536, 27276231, 27285244, 27292213 27293599, 27302711, 27302730, 27304410, 27305039, 27314206, 27314390 27321179, 27329612, 27333106, 27334316, 27338912, 27338946, 27339115 27345231, 27346709, 27348081, 27349393, 27351628, 27359178, 27367194 27370965, 27375542, 27394703, 27395416, 27396624, 27396813, 27400598 27405645, 27416997, 27433870, 27434193, 27439835, 27441326, 27442041 27466597, 27493674, 27501373, 27501413, 27502420, 27504770, 27505229 27508985, 27510959, 27534509, 27544973, 27548131, 27555481, 27558861 27560602, 27567477, 27595973, 27607563, 27611612, 27613080, 27620808 27687880, 27688036, 27688692, 27691920, 27691939, 27698953, 27700466 27709046, 27726780, 27740424, 27748954, 27757888, 27799032, 27835925 27847259, 27882176, 27898015, 27940876, 27945870, 27951817, 27959048 27994325, 27997875, 27998003, 28000269, 28033429, 28040776, 28074713 28090453, 28099662, 28140658, 28171079, 28174827, 28184554, 28188330 28218832, 28226179, 28290434, 28305001, 28320399, 28354603, 28437315 28454242, 28508557, 28522441

Database Engine: 12.1.0.2

The following versions are available for database engine 12.1.0.2:

- [Version 12.1.0.2.v14 \(p. 925\)](#)
- [Version 12.1.0.2.v13 \(p. 927\)](#)
- [Version 12.1.0.2.v12 \(p. 929\)](#)
- [Version 12.1.0.2.v11 \(p. 931\)](#)
- [Version 12.1.0.2.v10 \(p. 932\)](#)
- [Version 12.1.0.2.v9 \(p. 934\)](#)

- [Version 12.1.0.2.v8 \(p. 935\)](#)
- [Version 12.1.0.2.v7 \(p. 937\)](#)
- [Version 12.1.0.2.v6 \(p. 938\)](#)
- [Version 12.1.0.2.v5 \(p. 939\)](#)
- [Version 12.1.0.2.v4 \(p. 940\)](#)
- [Version 12.1.0.2.v3 \(p. 941\)](#)
- [Version 12.1.0.2.v2 \(p. 942\)](#)
- [Version 12.1.0.2.v1 \(p. 943\)](#)

Version 12.1.0.2.v14

Version 12.1.0.2.v14 adds support for the following:

- Patch 28259833: Oracle Database Patch Set Update 12.1.0.2.181016
- Patch 28440711: Oracle JVM Patch Set Update 12.1.0.2.181016
- Patch 28125601: DSTv32 for RDBMS (TZDATA2018E)
- Patch 28127287: DSTv32 for OJVM (TZDATA2018E)
- Patch 17969866: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 20394750: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 21171382: DBMS_STATS Patch
- Patch 28697469: JSON Database Patch
- Patch 20033733: KGL heap size patch

Oracle patch 28259833, released October 2018

Bugs fixed: 19309466, 19902195, 18250893, 25437699, 19383839, 16756406, 18456643, 26546664, 22364044, 18845653, 19915271, 20172151, 18417036, 23713236, 24796092, 23140259, 19243521, 19658708, 18272672, 21153266, 19174430, 22243719, 20688221, 20493163, 21387964, 13542050, 22734547, 21623164, 19012119, 19932634, 19869255, 22232606, 18681056, 23324000, 25427662, 22068305, 24589081, 19439759, 19303936, 22916353, 24835538, 22353346, 21106027, 26444887, 23088803, 22529728, 26256131, 19134173, 20447445, 21188584, 19390567, 26513709, 25780343, 19769480, 21097043, 21225209, 26245237, 20677396, 19284031, 19450314, 19016730, 20919320, 22075064, 22551446, 22721409, 18440095, 22496904, 16439813, 18354830, 20596234, 22022760, 20936905, 23197103, 21514877, 26111842, 18990023, 22492533, 20173897, 24624166, 17210525, 21260431, 20181030, 25056052, 19370504, 21868720, 23068169, 19124589, 19402853, 19888853, 24341675, 17722075, 20882568, 25653109, 23026585, 18604692, 20717081, 25546608, 27370965, 19081128, 22173980, 23514710, 19178851, 20951038, 22168163, 25161298, 20569094, 24308635, 19791377, 19050649, 20920911, 19189525, 19469538, 27052607, 20598042, 22458049, 18988834, 23302839, 25307368, 17409174, 22729345, 22842151, 19238590, 16941434, 20387265, 24397438, 20673810, 23108128, 20356733, 22380919, 18436647, 23065323, 20825533, 19124336, 22294260, 24790914, 20284155, 25539063, 17365043, 25914276, 20952966, 22961508, 19176223, 21300341, 23237313, 18288842, 22353199, 22083366, 25670786, 21419850, 26898563, 19577410, 23294548, 24737064, 19931709, 25423453, 25547060, 23533807, 27726780, 24600330, 25600421, 18122373, 20043616, 23124895, 18856999, 21450666, 18893947, 20076781, 26633558, 26029780, 21196809, 21354456, 23725036, 20464614, 19562381, 24808595, 27375542, 19189317, 25669791, 18307021, 21917884, 19708632, 27213224, 25633101, 20711718, 18973548, 25982666, 19718981, 22826718, 25655390, 23567857, 21773465, 20250147, 19197175, 26263721, 19597439, 21387128, 22007324, 19180770, 19879746, 21785691, 20424183, 24285405, 26544823, 20322560, 22228324, 23172924, 22520320, 21575362, 25058080, 22365117, 22645009, 25165496, 18774543, 20124446, 21429602, 26153977, 19371175, 21863727, 18940497, 19074147, 22923409, 25489342, 21380789, 19154375, 19044962, 19532017, 19662635, 22374754, 20560611, 25654936, 21492036, 18705806, 19578247, 22024071, 22238921, 22809871, 21184223, 23089357, 19404068, 18921743, 19065677, 19018447,

19018206, 18308268, 19777862 22223463, 19304354, 22519146, 27199245, 20890311, 22977256, 21142837 20869721, 24555417, 22179537, 21756699, 20217801, 18819908, 22760595 25483815, 23007241, 19593445, 21080143, 27351628, 20031873, 18618122 24737581, 26784509, 24739928, 18966843, 19077215, 20704450, 19068970 20543011, 19023822, 24713381, 20432873, 21756677, 20328248, 18674047 18849537, 25459958, 20315311, 22897344, 27534509, 25178179, 19308965 18948177, 19468991, 20868862, 21780146, 20466628, 21756661, 20397490 23315153, 19706965, 20302006, 24831514, 23240358, 22178855, 19032777 20862087, 19329654, 18974476, 20603378, 20859910, 19307662, 21847223 20281121, 19075256, 19076343, 18866977, 22808310, 25635149, 20844426 20904530, 20441797, 21442094, 25079710, 24674955, 18840932, 18740837 20294666, 25602488, 21517440, 22062517, 27337759, 19174942, 20671094 21889720, 18411216, 20117253, 24386767, 20641666, 25264559, 22092979 21625179, 20879709, 23003979, 20165574, 19272708, 19547370, 22624709 23084507, 20228093, 21281532, 19805359, 19461270, 19434529, 18799063 20378086, 17008068, 21246723, 20831538, 20424899, 20361671, 18674024 19689979, 24411921, 19873610, 16619249, 20562898, 21641414, 21091431 19440586, 22757364, 22175564, 21241052, 20725343, 19561643, 20736227 19399918, 19195895, 20830459, 20017509, 25790353, 21828126, 21665897 25555252, 20746251, 25764020, 25612095, 25357142, 23096938, 19067244 18043064, 21329301, 18885870, 26243698, 26187943, 20324049, 19536415 23709062, 28174827, 20446883, 27314206, 21299490, 25313154, 21744290 18254023, 20591183, 27847259, 19185876, 27207110, 22465352, 24326444 20402832, 19627012, 27441326, 27620950, 16863642, 19639483, 19315691 21479753, 19174521, 20401975, 18306996, 18851894, 27034890, 20581111 20318889, 20936731, 21060755, 26828994, 22256560, 19188927, 27229389 24570598, 25475853, 21172913, 17655240, 21266085, 19028800, 19035573 19366375, 24523374, 25034396, 19289642, 21291274, 18007682, 23521523 20475845, 22148226, 22528741, 25417958, 24652769, 26088426, 19326908 19597583, 17414008, 23019710, 20897759, 26822620, 22046677, 20938170 24825843, 19891090, 21960504, 26318627, 24509056, 19054077, 26262953 22657942, 20428621, 21899588, 19723336, 19835133, 17532734, 19333670 21842017, 19285025, 21373473, 23260854, 19687159, 23061453, 14643995 20977794, 20734332, 17551063, 27548131, 21977392, 24461826, 19676012 20588502, 23315889, 19520602, 23053606, 19841800, 20245930, 19001359 21476308, 26546754, 19393542, 23533524, 21099555, 25429959, 19141838 19644859, 21915719, 19908836, 21421886, 19358317, 19524158, 23548817 25861398, 20803014, 23025340, 19335438, 19058490, 19207117, 23642282 18799993, 25919622, 26569225, 20835241, 24662775, 19475971, 18967382 20347562, 20348653, 19896336, 24812585, 20048359, 21896069, 19524384 25392535, 21147908, 20440930, 25789277, 19171086, 24718260, 17867700 19791273, 21241829, 19591608, 22707244, 18419520, 22296366, 18914624 19571367, 22654475, 21522582, 19501299, 20425790, 19708342, 27997875 16870214, 18202441, 24415926, 18743542, 19001390, 21875360, 25091141 28000269, 19149990, 20382309, 22855193, 16777441, 19606174, 20848335 25495682, 19382851, 20528052, 22762046, 24563422, 23125826, 22503297 25192729, 23338911, 22730454, 19176326, 19048007, 18849970, 21532755 20860659, 22905130, 21263635, 22160989, 18499088, 21059919, 18952989 22894949, 22518784, 25856821, 25484507, 20794034, 19468347, 17533661 19883092, 20657441, 24401351, 21285458, 18051556, 25330273, 19699191 24437510, 20669434, 18964978, 22972770, 20828947, 21373076, 25551676 25492379, 14283239, 25766822, 22922076, 25575628, 20368850, 21239530 20437153, 24848928, 20880215, 20798891, 25606091, 19013183, 21133343 22695831, 24365589, 25634317, 19587324, 20273319, 18542562, 26758193 21063322, 22062026, 20134339, 22077517, 22815955, 24690216, 22507210 20101006, 16354467, 21795111, 27938623, 23501901, 18797519, 25879984 21260397, 25029423, 19354335, 19730508, 22366558, 26658759, 6599380 20717359, 24321547, 21297872, 18964939, 26366517, 21913183, 22366322 20171986, 20603431, 21132297, 25957038, 21542577, 22507234, 23170620 24719736, 25600342, 18868646, 20627866, 26637824, 18110491, 16923858 24642295, 19518079, 20466322, 25823754, 25110233, 24908321, 20842388 17274537, 26575788, 20474192, 21644640, 21794615, 18899974, 20471920 22806698, 19052488, 19503821, 24350620, 20074391, 19157754, 21220620 24316947, 19865345, 19065556, 22816287, 25947799, 20878790, 23492665 21322887, 22305887, 20879889, 24350831, 19578350, 19363645, 21072646 20898391, 19291380, 27060167, 27086138, 22536802, 22087683, 21656630 20373598, 19248799, 22707866, 19155797, 19279273, 18886413, 25490238 20922010, 19990037, 25150925, 20509482, 24717859, 20703000, 22862134 21526048, 24929210, 24560906, 20144308, 21620471, 19670108, 19068610 20267166, 25123585, 20476175, 18549238, 22950945, 19385656, 23528412 19684504, 21174504, 20899461, 20557786, 21911701, 19143550, 20118035 19024808, 25760195, 20009833, 19604659, 16359751, 26039623, 19928926 23314180, 20212067, 24737403, 20480209, 26430737, 20856766,

27169796 21668627, 20877664, 19487147, 23149541, 24577566, 19430401, 19676905 20925795, 21296029, 21629064, 23229229, 22865673, 20708701, 25353983 19280225, 21315084, 19213447, 19989009, 18191823, 27314390, 25775213 24393981, 25639019, 17319928, 19703301, 21626377, 20122715, 6418158 23105538, 26198926, 19258504, 21188532, 23151677, 17890099, 21649497 26446098, 16887946, 26024732, 18791688, 19721304, 19490948, 27012701 19619732, 21164318, 18090142, 21641760, 19818513, 20139391, 24693382 19978542, 23543183, 22165897, 22359063, 19409212, 23035249, 18990693 20470877, 21422580, 21632821, 22351572, 20235511, 23220453, 18604493 23008056, 22901797, 18610915, 20832516, 24801152, 26089440, 20907061 20505778, 19183343, 21787056, 21273804, 25093739, 17835294, 24413809 18371441, 26714910, 24385983, 20413820, 24421668, 25897615, 25643931 23195445, 21281607, 20513399, 20558005, 20093776, 18909599, 20618595 23572982, 19211433, 20331945, 19512341, 22256431, 19637186, 19022470 18607546, 24573817, 19649152, 23115139, 19201867, 21294938, 20898997 18510194, 21842740, 22454326, 24683149, 19534363, 25489607

Version 12.1.0.2.v13

Version 12.1.0.2.v13 adds support for the following:

- Patch 27547329: Oracle Database Patch Set Update 12.1.0.2.180717
- Patch 27923320: Oracle JVM Patch Set Update 12.1.0.2.180717
- Patch 28125601: DSTv32 for RDBMS (TZDATA2018E)
- Patch 28127287: DSTv32 for OJVM (TZDATA2018E)
- Patch 17969866: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 20394750: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 21171382: DBMS_STATS Patch
- Patch 28307069: JSON Database Patch
- Patch 20033733: KGL heap size patch

Oracle patch 27547329, released July 2018

Bugs fixed: 19309466, 19902195, 18250893, 25437699, 19383839, 16756406, 18456643 26546664, 18845653, 19915271, 20172151, 18417036, 23713236, 24796092 19243521, 19658708, 21153266, 19174430, 22243719, 20688221, 21387964 13542050, 22734547, 21623164, 19012119, 19932634, 19869255, 22232606 18681056, 23324000, 25427662, 22068305, 24589081, 19439759, 19303936 22916353, 24835538, 22353346, 21106027, 26444887, 23088803, 22529728 26256131, 19134173, 20447445, 21188584, 19390567, 26513709, 19769480 21097043, 21225209, 20677396, 19284031, 26245237, 19450314, 19016730 20919320, 22075064, 22551446, 22721409, 18440095, 22496904, 16439813 18354830, 20596234, 22022760, 20936905, 23197103, 21514877, 26111842 18990023, 22492533, 20173897, 24624166, 17210525, 21260431, 20181030 25056052, 19370504, 21868720, 23068169, 19124589, 19402853, 19888853 24341675, 17722075, 20882568, 23026585, 25653109, 20717081, 25546608 19081128, 27370965, 22173980, 19178851, 20951038, 22168163, 25161298 20569094, 24308635, 19791377, 19050649, 20920911, 19189525, 19469538 20598042, 22458049, 18988834, 17409174, 22729345, 22842151, 19238590 16941434, 20387265, 24397438, 20673810, 23108128, 20356733, 22380919 18436647, 23065323, 20825533, 19124336, 22294260, 24790914, 20284155 25539063, 17365043, 20952966, 22961508, 19176223, 21300341, 23237313 18288842, 22353199, 22083366, 21419850, 26898563, 19577410, 23294548 19931709, 25423453, 25547060, 23533807, 24600330, 25600421, 18122373 20043616, 23124895, 18856999, 21450666, 18893947, 20076781, 26029780 21196809, 21354456, 20464614, 23725036, 19562381, 24808595, 19189317 18307021, 25669791, 21917884, 19708632, 27213224, 25633101, 20711718 18973548, 25982666, 22826718, 25655390, 21773465, 20250147, 19197175 19597439, 26263721, 21387128, 19180770, 19879746, 21785691, 20424183 24285405, 26544823, 20322560, 22228324, 22520320, 23172924, 21575362 22365117, 22645009, 25165496, 18774543, 20124446, 21429602, 19371175 21863727, 18940497, 19074147, 22923409, 21380789, 19154375, 19044962 19532017, 19662635, 22374754, 20560611, 25654936, 21492036, 18705806 19578247, 22024071, 22238921, 22809871, 21184223,

23089357, 19404068 18921743, 19065677, 19018447, 19018206, 18308268, 19777862, 22223463, 19304354, 22519146, 27199245, 20890311, 21142837, 20869721, 24555417 22179537, 21756699, 20217801, 18819908, 22760595, 25483815, 23007241 19593445, 21080143, 20031873, 18618122, 26784509, 24739928, 18966843 19077215, 20704450, 19068970, 20543011, 19023822, 24713381, 20432873 21756677, 20328248, 18674047, 18849537, 25459958, 20315311, 22897344 27534509, 25178179, 19308965, 18948177, 19468991, 20868862, 21780146 20466628, 21756661, 20397490, 19706965, 24831514, 23240358, 22178855 20302006, 19032777, 20862087, 19329654, 18974476, 20603378, 20859910 19307662, 21847223, 20281121, 19075256, 19076343, 18866977, 20844426, 20904530, 20441797, 21442094, 25079710, 24674955, 18840932, 18740837 20294666, 25602488, 21517440, 22062517, 27337759, 19174942, 20671094 21889720, 18411216, 20117253, 24386767, 20641666, 25264559, 22092979 21625179, 20879709, 23003979, 20165574, 19272708, 19547370, 22624709 23084507, 20228093, 21281532, 19805359, 19461270, 19434529, 18799063 20378086, 17008068, 21246723, 20831538, 20424899, 20361671, 18674024 19689979, 24411921, 19873610, 16619249, 20562898, 21091431, 21641414 19440586, 22757364, 22175564, 21241052, 19561643, 19399918, 19195895 20830459, 20017509, 25790353, 21828126, 21665897, 20746251, 25764020, 25612095, 25357142, 23096938, 19067244, 18043064, 21329301, 18885870 26187943, 20324049, 19536415, 20446883, 21299490, 27314206, 25313154 21744290, 18254023, 20591183, 27847259, 19185876, 22465352, 27207110 20402832, 19627012, 27441326, 27620950, 16863642, 19639483, 19315691 21479753, 19174521, 20401975, 18306996, 18851894, 27034890, 20581111 20318889, 20936731, 21060755, 22256560, 19188927, 24570598, 25475853 21172913, 17655240, 21266085, 19028800, 19035573, 19366375, 24523374 25034396, 19289642, 21291274, 18007682, 23521523, 20475845, 22148226 22528741, 25417958, 24652769, 26088426, 19326908, 19597583, 17414008, 23019710, 20897759, 22046677, 20938170, 24825843, 21960504, 24509056 19054077, 22657942, 26262953, 20428621, 21899588, 19723336, 19835133 17532734, 19333670, 21842017, 19285025, 21373473, 23260854, 19687159 14643995, 20977794, 20734332, 17551063, 27548131, 21977392, 24461826 19676012, 20588502, 23315889, 19520602, 23053606, 19841800, 20245930 19001359, 21476308, 26546754, 19393542, 23533524, 21099555, 25429959 19141838, 19644859, 21915719, 19908836, 21421886, 19358317, 19524158 23548817, 25861398, 20803014, 23025340, 19335438, 19058490, 19207117 18799993, 26569225, 25919622, 20835241, 24662775, 19475971, 18967382, 20347562, 20348653, 19896336, 24812585, 20048359, 21896069, 19524384 25392535, 20440930, 25789277, 19171086, 24718260, 17867700, 19791273 21241829, 19591608, 22707244, 18419520, 22296366, 18914624, 19571367 19501299, 20425790, 19708342, 27997875, 16870214, 18202441, 24415926 18743542, 19001390, 21875360, 25091141, 19149990, 20382309, 22855193 16777441, 19606174, 20848335, 25495682, 19382851, 20528052, 22762046 24563422, 23125826, 22503297, 25192729, 23338911, 22730454, 19176326 19048007, 18849970, 21532755, 20860659, 22905130, 21263635, 22160989 18499088, 21059919, 18952989, 22518784, 25856821, 25484507, 20794034, 19468347, 17533661, 19883092, 20657441, 24401351, 21285458, 18051556 25330273, 19699191, 24437510, 20669434, 18964978, 20828947, 21373076 25551676, 14283239, 25766822, 22922076, 25575628, 20368850, 21239530 20437153, 20880215, 20798891, 25606091, 19013183, 21133343, 22695831 24365589, 19587324, 18542562, 26758193, 22062026, 20134339, 22077517 22815955, 24690216, 22507210, 20101006, 21795111, 27938623, 23501901 18797519, 21260397, 25029423, 19354335, 19730508, 22366558, 26658759 6599380, 20717359, 24321547, 21297872, 18964939, 26366517, 21913183 22366322, 20171986, 20603431, 21132297, 25957038, 21542577, 22507234, 23170620, 24719736, 25600342, 18868646, 20627866, 18110491, 16923858 24642295, 19518079, 20466322, 25823754, 25110233, 24908321, 20842388 17274537, 26575788, 20474192, 21644640, 21794615, 18899974, 20471920 22806698, 19052488, 19503821, 24350620, 20074391, 19157754, 21220620 24316947, 19865345, 19065556, 22816287, 25947799, 20878790, 23492665 21322887, 20879889, 24350831, 19578350, 19363645, 21072646, 20898391 19291380, 27060167, 27086138, 22536802, 22087683, 20373598, 19248799 22707866, 19155797, 19279273, 18886413, 25490238, 20922010, 19990037 25150925, 20509482, 24717859, 20703000, 22862134, 21526048, 24929210, 24560906, 20144308, 21620471, 19670108, 19068610, 20267166, 25123585 20476175, 18549238, 22950945, 19385656, 23528412, 19684504, 21174504 20899461, 20557786, 21911701, 19143550, 19024808, 20118035, 20009833 25760195, 19604659, 16359751, 26039623, 19928926, 23314180, 20212067 24737403, 20480209, 26430737, 27169796, 21668627, 20877664, 19487147 23149541, 24577566, 19430401, 19676905, 20925795, 21296029, 21629064 23229229, 22865673, 20708701, 19280225, 25353983, 21315084, 19213447 19989009, 18191823, 24393981, 25639019, 17319928, 19703301, 21626377 20122715, 6418158, 23105538, 26198926, 19258504, 21188532, 17890099

21649497, 26446098, 16887946, 26024732, 18791688, 19721304, 19490948 19619732, 21164318, 18090142, 21641760, 19818513, 20139391, 24693382 19978542, 23543183, 22165897, 22359063, 19409212, 23035249, 18990693 20470877, 21422580, 21632821, 22351572, 20235511, 23220453, 18604493 23008056, 18610915, 20832516, 24801152, 26089440, 20907061, 20505778 19183343, 21787056, 21273804, 25093739, 17835294, 24413809, 18371441 24385983, 20413820, 26714910, 24421668, 25897615, 25643931, 21281607 20513399, 23195445, 20558005, 20093776, 18909599, 20618595, 23572982 19211433, 20331945, 19512341, 22256431, 19637186, 19022470, 18607546 24573817, 19649152, 19201867, 21294938, 20898997, 18510194, 22454326 19534363, 24683149, 25489607

Version 12.1.0.2.v12

Version 12.1.0.2.v12 adds support for the following:

- Patch 27338041: DATABASE PATCH SET UPDATE 12.1.0.2.180417
- Patch 27475603: OJVM PATCH SET UPDATE 12.1.0.2.180417
- Patch 27015449: RDBMS - PROACTIVE DSTV31 UPDATE - TZDATA2017C
- Patch 27015468: PROACTIVE DSTV31 UPDATE - TZDATA2017C - NEED OJVM FIX
- Patch 17969866: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 20394750: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 21171382: AUTO DOP COMPUTES A HIGH DOP UNNECESSARILY
- Patch 27666699: JSON Database Patch
- Patch 20033733: PART :IMC:HIT ORA 600 [KGL-HEAP-SIZE-EXCEEDED]

Oracle patch 27338041, released April 2018

Bugs fixed: 19309466, 24570598, 25475853, 21172913, 19902195, 18250893, 17655240 25437699, 19383839, 21266085, 19028800, 19035573, 16756406, 19366375 18456643, 26546664, 24523374, 25034396, 19289642, 18845653, 19915271 21291274, 18007682, 20172151, 18417036, 23713236, 24796092, 23521523 20475845, 22148226, 22528741, 19243521, 19658708, 21153266, 24652769 26088426, 19326908, 19597583, 17414008, 20897759, 23019710, 19174430 22046677, 22243719, 20938170, 24825843, 21960504, 24509056, 19054077 22657942, 20688221, 20428621, 21899588, 21387964, 13542050, 19723336 19835133, 17532734, 19333670, 21842017, 19285025, 21373473, 22734547 23260854, 19687159, 14643995, 21623164, 20977794, 20734332, 19012119 19869255, 19932634, 17551063, 18681056, 22232606, 27548131, 21977392 23324000, 24461826, 19676012, 20588502, 25427662, 22068305, 23315889 19520602, 23053606, 19841800, 19439759, 20245930, 19303936, 19001359 21476308, 26546754, 22916353, 19393542, 23533524, 21099555, 24835538 22353346, 25429959, 19141838, 19644859, 21106027, 21915719, 26444887 23088803, 19908836, 21421886, 22529728, 26256131, 19358317, 19134173 19524158, 20447445, 23548817, 25861398, 20803014, 23025340, 21188584 19335438, 19390567, 19058490, 19207117, 26513709, 18799993, 26569225 20835241, 24662775, 19769480, 19475971, 21097043, 21225209, 20677396 19284031, 19450314, 19016730, 18967382, 20919320, 22075064, 20347562 20348653, 22551446, 19896336, 22721409, 24812585, 20048359, 21896069 18440095, 22496904, 19524384, 25392535, 16439813, 18354830, 20596234 20440930, 22022760, 20936905, 19171086, 23197103, 24718260, 17867700 19791273, 21514877, 26111842, 18990023, 21241829, 19591608, 22707244 18419520, 22492533, 22296366, 20173897, 24624166, 17210525, 18914624 19571367, 21260431, 19501299, 20181030, 25056052, 20425790, 19708342 19370504, 21868720, 23068169, 19124589, 19402853, 19888853, 16870214 24341675, 17722075, 18202441, 24415926, 18743542, 19001390, 20882568 23026585, 20717081, 25546608, 19081128, 22173980, 21875360, 25091141 19178851, 19149990, 20382309, 20951038, 22855193, 22168163, 16777441 25161298, 19606174, 20569094, 24308635, 20848335, 19791377, 19050649 19382851, 20920911, 20528052, 22762046, 19189525, 24563422, 23125826 22503297, 19469538, 25192729, 23338911, 20598042, 22458049, 18988834 22730454, 19176326, 19048007, 17409174, 22729345, 18849970, 21532755 20860659, 22842151, 22905130, 19238590, 16941434, 20387265, 21263635 24397438, 20673810, 23108128, 22160989, 20356733, 22380919,

18499088 18436647, 23065323, 21059919, 20825533, 18952989, 22518784, 19124336 25856821, 22294260, 25484507, 20794034, 19468347, 20284155, 17533661 19883092, 20657441, 24401351, 25539063, 17365043, 21285458, 20952966 22961508, 18051556, 25330273, 19176223, 21300341, 23237313, 18288842 19699191, 22353199, 24437510, 22083366, 21419850, 20669434, 18964978 26898563, 19577410, 23294548, 20828947, 21373076, 25551676, 14283239 25766822, 19931709, 22922076, 25423453, 25547060, 25575628, 23533807 20368850, 21239530, 20437153, 20880215, 25600421, 20798891, 25606091 18122373, 20043616, 23124895, 19013183, 18856999, 21450666, 21133343 22695831, 18893947, 24365589, 20076781, 21196809, 21354456, 19587324 20464614, 19562381, 18542562, 26758193, 24808595, 22062026, 19189317 18307021, 21917884, 19708632, 27213224, 25633101, 20711718, 20134339 22077517, 22815955, 24690216, 18973548, 25982666, 22507210, 22826718 25655390, 21773465, 20250147, 20101006, 21795111, 19197175, 23501901 18797519, 19597439, 21387128, 19180770, 19879746, 19354335, 21785691 19730508, 20424183, 22366558, 26658759, 24285405, 6599380, 20717359 26544823, 21297872, 20322560, 18964939, 22520320, 21575362, 26366517 21913183, 22366322, 20171986, 22365117, 22645009, 25165496, 20603431 21132297, 25957038, 21542577, 22507234, 18774543, 23170620, 24719736 25600342, 20627866, 20124446, 18110491, 21429602, 16923858, 24642295 19518079, 19371175, 20466322, 21863727, 18940497, 19074147, 22923409 25823754, 25110233, 24908321, 20842388, 17274537, 21380789, 26575788 19154375, 20474192, 19044962, 19532017, 21644640, 19662635, 22374754 20560611, 25654936, 21794615, 18899974, 21492036, 18705806, 20471920 22806698, 19052488, 22024071, 22238921, 19503821, 24350620, 22809871 20074391, 21184223, 23089357, 19157754, 21220620, 19404068, 24316947 18921743, 19865345, 19065677, 19065556, 22816287, 19018447, 19018206 19777862, 25947799, 22223463, 19304354, 20878790, 22519146, 23492665 21322887, 20879889, 24350831, 20890311, 19578350, 21142837, 20869721 24555417, 22179537, 21756699, 20217801, 18819908, 19363645, 25483815 21072646, 20898391, 19291380, 27060167, 27086138, 23007241, 19593445 21080143, 22536802, 22087683, 20373598, 19248799, 20031873, 22707866 19155797, 19279273, 18886413, 18618122, 25490238, 20922010, 19990037 25150925, 20509482, 24739928, 20703000, 18966843, 19077215, 22862134 21526048, 24929210, 24560906, 20704450, 20144308, 19068970, 20543011 21620471, 19023822, 19670108, 19068610, 20267166, 24713381, 20432873 21756677, 20476175, 25123585, 18549238, 20328248, 18674047, 22950945 19385656, 18849537, 23528412, 19684504, 25459958, 20315311, 22897344 20899461, 25178179, 20557786, 21911701, 19308965, 19143550, 19024808 18948177, 19468991, 20009833, 20868862, 21780146, 20466628, 21756661 20397490, 19706965, 24831514, 23240358, 22178855, 19604659, 16359751 19032777, 20862087, 19329654, 19928926, 18974476, 23314180, 20212067 20603378, 24737403, 20480209, 20859910, 26430737, 19307662, 21847223 21668627, 20281121, 27169796, 19075256, 20877664, 19487147, 19076343 23149541, 18866977, 24577566, 19430401, 19676905, 20844426, 20904530 20925795, 20441797, 21296029, 21629064, 21442094, 23229229, 25079710 22865673, 20708701, 19280225, 21315084, 24674955, 19213447, 18840932 18740837, 20294666, 19989009, 25602488, 18191823, 21517440, 22062517 19174942, 27337759, 17319928, 20671094, 21889720, 19703301, 21626377 20122715, 23105538, 18411216, 6418158, 26198926, 20117253, 19258504 21188532, 24386767, 17890099, 21649497, 26446098, 16887946, 26024732 25264559, 18791688, 19721304, 22092979, 19490948, 19619732, 21164318 21625179, 20879709, 23003979, 20165574, 18090142, 19272708, 21641760 19818513, 19547370, 22624709, 20139391, 23084507, 24693382, 20228093 21281532, 19978542, 23543183, 22165897, 22359063, 19409212, 19805359 19461270, 23035249, 19434529, 18799063, 18990693, 20470877, 20378086 17008068, 21246723, 21422580, 21632821, 20831538, 22351572, 20424899 20361671, 18674024, 19689979, 20235511, 23220453, 24411921, 19873610 16619249, 18604493, 20562898, 21091431, 19440586, 22757364, 18610915 22175564, 21241052, 19561643, 19399918, 19195895, 20832516, 20830459 20017509, 24801152, 21828126, 20907061, 21665897, 20746251, 20505778 19183343, 25764020, 25612095, 25357142, 23096938, 21787056, 21273804 19067244, 18043064, 21329301, 18885870, 20324049, 26187943, 19536415 25093739, 17835294, 20446883, 21299490, 25313154, 24413809, 21744290 18254023, 20591183, 18371441, 24385983, 20413820, 24421668, 25897615 19185876, 25643931, 21281607, 20513399, 22465352, 20558005, 20402832 19627012, 20093776, 18909599, 20618595, 27441326, 27620950, 23572982 16863642, 19639483, 19315691, 19211433, 20331945, 19512341, 22256431 21479753, 19637186, 19174521, 19022470, 18607546, 20401975, 18306996 24573817, 18851894, 19649152, 27034890, 20581111, 19201867, 20318889 20936731, 21060755, 21294938, 20898997, 18510194, 22256560, 22454326 19534363, 25489607, 19188927

Version 12.1.0.2.v11

Version 12.1.0.2.v11 adds support for the following:

- Patch 26925311: DATABASE PATCH SET UPDATE 12.1.0.2.180116
- Patch 27001733: OJVM PATCH SET UPDATE 12.1.0.2.180116
- Patch 27015449: RDBMS - PROACTIVE DSTV31 UPDATE - TZDATA2017C
- Patch 27015468: PROACTIVE DSTV31 UPDATE - TZDATA2017C - NEED OJVM FIX
- Patch 17969866: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 20394750: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 21171382: AUTO DOP COMPUTES A HIGH DOP UNNECESSARILY
- Patch 27315904: JSON Database Patch
- Patch 20033733: ORA 600 [KGL-HEAP-SIZE-EXCEEDED]

Oracle patch 26925311, released January 2018

Bugs fixed: 21099555, 22175564, 19141838, 22083366, 20842388, 19865345, 20117253 20830459, 19791273, 20671094, 21542577, 23105538, 19243521, 20951038 22165897, 19238590, 21281532, 17008068, 19908836, 24401351, 24577566 21184223, 25427662, 20717359, 19134173, 20569094, 20031873, 20387265 20322560, 21575362, 19149990, 21263635, 18886413, 17551063, 24719736 22160989, 22519146, 21623164, 22507210, 19703301, 23338911, 19366375 18007682, 19001390, 18202441, 24285405, 25655390, 20267166, 19358317 19706965, 19068970, 24739928, 18549238, 22148226, 18797519, 26544823 20825533, 23521523, 21196809, 18940497, 19670108, 19649152, 18866977 18948177, 19404068, 22496904, 22826718, 18964978, 19176326, 19035573 20413820, 20717081, 19176223, 21106027, 20904530, 20134339, 19074147 20868862, 18411216, 23035249, 25475853, 21072646, 21322887, 22507234 20425790, 20862087, 18966843, 25861398, 24929210, 24624166, 21329301 20562898, 19333670, 19468991, 20124446, 19883092, 23543183, 20878790 22855193, 18510194, 19658708, 19591608, 19402853, 23149541, 24796092 20618595, 22238921, 21795111, 21787056, 22380919, 19469538, 21266085 17835294, 19721304, 19068610, 19791377, 22178855, 16777441, 22173980 20746251, 20048359, 21896069, 19185876, 20898391, 20281121, 20907061 22950945, 21281607, 6599380, 19577410, 22092979, 19001359, 20603378 23089357, 23572982, 19490948, 21387964, 22294260, 20832516, 17532734 22351572, 18849970, 19309466, 19081128, 20627866, 20844426, 24908321 21188532, 18791688, 21442094, 20890311, 20596234, 20368850, 26366517 18973548, 19303936, 21296029, 22536802, 20882568, 21479753, 19461270 20235511, 20936905, 22077517, 21220620, 18964939, 19430401, 22806698 22296366, 21153266, 19409212, 20703000, 22657942, 20657441, 19879746 20557786, 26758193, 23237313, 26198926, 19684504, 26088426, 21294938 19024808, 24693382, 20528052, 20977794, 18799993, 20466322, 24642295 18740837, 19662635, 18440095, 21794615, 20382309, 20228093, 19065556 20212067, 25547060, 21868720, 22905130, 20938170, 19524384, 25459958 24350831, 17722075, 20446883, 20144308, 25056052, 18952989, 24523374 16870214, 21773465, 19928926, 19835133, 21629064, 21354456, 20466628 23007241, 24386767, 25490238, 19931709, 19730508, 18819908, 20250147 23124895, 25643931, 23220453, 19188927, 20074391, 18307021, 23533807 20356733, 14643995, 26430737, 18090142, 19065677, 19547370, 26024732 21225209, 21960504, 18371441, 20397490, 26575788, 23315889, 20172151 18967382, 22729345, 19174430, 22068305, 25654936, 18419520, 21241829 19536415, 26546664, 19171086, 21889720, 21132297, 20470877, 22465352 22168163, 19335438, 24397438, 20076781, 20447445, 18856999, 20471920 19869255, 21620471, 18990693, 23096938, 17890099, 19124336, 24812585 18990023, 20101006, 21300341, 20848335, 21744290, 21241052, 20897759 21668627, 19304354, 19052488, 20543011, 20794034, 23025340, 25606091 23260854, 18681056, 19562381, 24570598, 20952966, 19896336, 20828947 25539063, 18618122, 20328248, 24365589, 20440930, 18456643, 19699191 23065323, 22865673, 19201867, 22816287, 21514877, 22022760, 18743542 20798891, 20347562, 25161298, 23294548, 19777862, 24560906, 22551446 19687159, 21373076, 19174942, 20424899, 24461826, 21641760, 21899588 22862134, 18899974, 21476308, 20598042, 21297872, 24308635, 19058490 19032777, 20171986, 22815955,

25150925, 19399918, 24718260, 19434529 22492533, 19018447, 21273804, 18051556, 22757364, 18851894, 23125826 20424183, 21842017, 19022470, 19284031, 18043064, 26898563, 20173897 23713236, 22062026, 20475845, 17274537, 19440586, 16887946, 22374754 18974476, 22961508, 24825843, 17319928, 20401975, 20708701, 22062517 24674955, 17655240, 22809871, 19805359, 16439813, 19155797, 20859910 19393542, 17210525, 22024071, 19189525, 21847223, 21649497, 19075256 25079710, 25823754, 19370504, 20315311, 22762046, 22075064, 20936731 20437153, 25165496, 18845653, 19280225, 19248799, 20560611, 18988834 21756699, 22256431, 18921743, 20245930, 21532755, 18799063, 22454326 20373598, 20476175, 19571367, 20925795, 19018206, 25264559, 24385983 20509482, 20711718, 24509056, 20588502, 20181030, 21911701, 18849537 23501901, 25034396, 19183343, 22842151, 21917884, 21142837, 20603431 19189317, 23003979, 19644859, 19390567, 19279273, 26546754, 20669434 16863642, 22528741, 22707244, 25546608, 19619732, 20348653, 18607546 19315691, 19676905, 20165574, 17867700, 23528412, 20558005, 20734332 19532017, 20922010, 19818513, 19450314, 22353346, 16941434, 20361671 25423453, 20009833, 22366558, 20294666, 23197103, 18191823, 20860659 22707866, 19195895, 19371175, 19307662, 19154375, 20043616, 20324049 21977392, 18914624, 22529728, 22256560, 25330273, 19708342, 20139391 19593445, 21291274, 19382851, 19520602, 19174521, 21875360, 19676012 19326908, 20217801, 20093776, 18840932, 21097043, 21246723, 20803014 21665897, 19143550, 23026585, 20428621, 19627012, 24415926, 22087683 23548817, 14283239, 21422580, 19213447, 19518079, 26446098, 18610915 23492665, 18674024, 24831514, 21863727, 24413809, 18306996, 19915271 21626377, 19524158, 20122715, 20513399, 18110491, 22366322, 20284155 25091141, 21080143, 20017509, 22359063, 19363645, 19597439, 21239530 23108128, 19888853, 19383839, 20880215, 21756677, 22458049, 19534363 19354335, 19044962, 19639483, 25982666, 19475971, 22353199, 21060755 22243719, 22916353, 20378086, 21260431, 21756661, 24808595, 22923409 19028800, 20877664, 22518784, 21059919, 20879889, 21380789, 19723336 19077215, 21421886, 19604659, 21285458, 23533524, 26569225, 23170620 22365117, 18288842, 19048007, 19308965, 19689979, 17409174, 19503821 23068169, 24662775, 21526048, 25429959, 19197175, 19180770, 24555417 24573817, 19902195, 26444887, 25313154, 24835538, 23324000, 20318889 21492036, 19013183, 20591183, 19012119, 20464614, 22645009, 21625179 19067244, 25178179, 23053606, 21632821, 19841800, 19512341, 19211433 22695831, 20331945, 19587324, 24316947, 19578350, 19637186, 19054077 18674047, 19708632, 20898997, 21091431, 19285025, 19289642, 25947799 21133343, 20835241, 20869721, 21172913, 25602488, 19258504, 17365043 21419850, 21644640, 19468347, 21373473, 25093739, 22721409, 16359751 24421668, 21164318, 25484507, 25489607, 22520320, 19769480, 19439759 19272708, 23088803, 19978542, 19329654, 20402832, 19873610, 23229229 21517440, 13542050, 25897615, 19291380, 21915719, 25600342, 25192729 20879709, 20677396, 19076343, 19561643, 19990037, 18909599, 19487147 22897344, 20831538, 25600421, 19016730, 18250893, 23240358, 22179537 16619249, 18354830, 24411921, 25764020, 18254023, 16756406, 21188584 19989009, 25766822, 17414008, 20688221, 20441797, 20704450, 21780146 25612095, 25957038, 24652769, 25483815, 19157754, 19207117, 24437510 18885870, 21785691, 20673810, 24341675, 21450666, 18893947, 18705806 22223463, 18417036, 16923858, 23084507, 23314180, 20919320, 22503297 20474192, 22046677, 21299490, 19501299, 19385656, 20432873, 18542562 20920911, 20899461, 21315084, 21429602, 21387128, 18122373, 20581111 22624709, 26111842, 19606174, 24690216, 18436647, 19023822, 25110233 19124589, 19178851, 19597583, 20480209, 18499088, 19050649

Version 12.1.0.2.v10

Version 12.1.0.2.v10 adds support for the following:

- Oracle October 2017 PSU, a combination of database PSU (patch 26713565) + OJVM component PSU (patch 26635845)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866)
- DBMS_STATS AUTO DOP COMPUTES A HIGH DOP UNNECESSARILY (patch 21171382)
- JSON bundle patch (patch 26750145)
- KGL heap size patch (patch 20033733)
- Timezone file DSTv30 (patch 25881255, OJVM patch 25881271)

Oracle patch 26713565, released October 2017

Bugs fixed: 21099555, 22175564, 19141838, 22083366, 20842388, 19865345, 20117253 20830459, 19791273, 20671094, 21542577, 19243521, 20951038, 22165897 19238590, 21281532, 17008068, 19908836, 24577566, 21184223, 25427662 19134173, 20569094, 20031873, 20387265, 20322560, 21575362, 19149990 21263635, 17551063, 18886413, 24719736, 22160989, 22519146, 21623164 22507210, 23338911, 19703301, 19366375, 18007682, 19001390, 18202441 24285405, 25655390, 20267166, 19358317, 19706965, 19068970, 24739928 18549238, 22148226, 18797519, 26544823, 20825533, 23521523, 21196809 18940497, 19670108, 19649152, 18866977, 18948177, 22496904, 19404068 18964978, 19176326, 19035573, 20413820, 20717081, 19176223, 21106027 20904530, 20134339, 19074147, 20868862, 23035249, 18411216, 21072646 25475853, 21322887, 22507234, 20425790, 20862087, 18966843, 25861398 21329301, 20562898, 19333670, 19468991, 20124446, 19883092, 22855193 20878790, 18510194, 19658708, 19591608, 19402853, 23149541, 20618595 22238921, 21795111, 21787056, 22380919, 19469538, 21266085, 17835294 19721304, 19068610, 19791377, 22178855, 16777441, 22173980, 20746251 20048359, 21896069, 19185876, 20898391, 20281121, 20907061, 22950945 6599380, 19577410, 22092979, 19001359, 20603378, 23089357, 21387964 19490948, 22294260, 20832516, 17532734, 22351572, 19309466, 19081128 20627866, 20844426, 24908321, 21188532, 18791688, 21442094, 20890311 20596234, 20368850, 18973548, 19303936, 21296029, 20882568, 21479753 19461270, 20235511, 22077517, 20936905, 21220620, 18964939, 19430401 22806698, 22296366, 21153266, 19409212, 22657942, 20703000, 20657441 19879746, 20557786, 26198926, 26088426, 19684504, 21294938, 19024808 24693382, 20528052, 20977794, 18799993, 20466322, 24642295, 18740837 19662635, 18440095, 21794615, 20228093, 19065556, 20212067, 25547060 21868720, 20938170, 22905130, 19524384, 25459958, 24350831, 17722075 20144308, 20446883, 25056052, 18952989, 24523374, 16870214, 19928926 19835133, 21629064, 21354456, 20466628, 24386767, 25490238, 19931709 19730508, 18819908, 20250147, 23124895, 25643931, 23220453, 19188927 20074391, 18307021, 23533807, 20356733, 26430737, 14643995, 18090142 19065677, 19547370, 21225209, 21960504, 18371441, 20397490, 26575788 23315889, 20172151, 18967382, 19174430, 22068305, 25654936, 21241829 19536415, 19171086, 26546664, 21132297, 21889720, 22465352, 22168163 19335438, 24397438, 20076781, 20447445, 18856999, 20471920, 19869255 21620471, 18990693, 23096938, 19124336, 17890099, 24812585, 18990023 21300341, 20101006, 20848335, 21744290, 21241052, 20897759, 21668627 19304354, 19052488, 20543011, 20794034, 23025340, 25606091, 23260854 18681056, 19562381, 20952966, 19896336, 20828947, 25539063, 18618122 20328248, 20440930, 18456643, 19699191, 22865673, 19201867, 22816287 22022760, 21514877, 18743542, 20798891, 20347562, 25161298, 23294548 24560906, 22551446, 19777862, 19687159, 21373076, 19174942, 20424899 21899588, 22862134, 18899974, 21476308, 20598042, 24308635, 21297872 19058490, 19032777, 20171986, 22815955, 19399918, 19434529, 19018447 18051556, 21273804, 22757364, 18851894, 23125826, 20424183, 21842017 19022470, 19284031, 18043064, 23713236, 20173897, 22062026, 20475845 17274537, 19440586, 22961508, 24825843, 18974476, 22374754, 16887946 17319928, 20401975, 20708701, 24674955, 22062517, 22809871, 17655240 19805359, 16439813, 19155797, 20859910, 19393542, 17210525, 22024071 19189525, 21847223, 21649497, 19075256, 25823754, 25079710, 20315311 22762046, 22075064, 20936731, 20437153, 18845653, 19280225, 19248799 20560611, 18988834, 21756699, 22256431, 21532755, 18921743, 20245930 22454326, 18799063, 20373598, 20476175, 19571367, 20925795, 19018206 25264559, 20711718, 20509482, 20181030, 20588502, 21911701, 18849537 23501901, 25034396, 19183343, 22842151, 21917884, 21142837, 20603431 19189317, 23003979, 19644859, 19390567, 19279273, 26546754, 20669434 16863642, 22528741, 22707244, 25546608, 19619732, 20348653, 18607546 19315691, 19676905, 20165574, 17867700, 20558005, 20734332, 19532017 20922010, 19818513, 19450314, 22353346, 16941434, 20361671, 25423453 20009833, 22366558, 20294666, 23197103, 18191823, 20860659, 19195895 19371175, 19307662, 19154375, 20043616, 21977392, 18914624, 22529728 19708342, 20139391, 25330273, 19593445, 21291274, 19382851, 19520602 19174521, 21875360, 19676012, 19326908, 20217801, 20093776, 18840932 21097043, 21246723, 20803014, 21665897, 19143550, 23026585, 20428621 19627012, 22087683, 23548817, 14283239, 21422580, 19213447, 26446098 19518079, 23492665, 18610915, 18674024, 21863727, 24413809, 18306996 19915271, 21626377, 19524158, 20122715, 20513399, 18110491, 20284155 25091141, 21080143, 20017509, 22359063, 19363645, 19597439, 21239530 23108128, 19383839, 20880215, 21756677, 19888853, 22458049, 19534363 19354335, 19044962, 19639483, 25982666, 19475971, 22353199, 21060755 22243719, 22916353, 20378086, 24808595,

21756661, 21260431, 22923409 19028800, 20877664, 21059919, 20879889, 21380789, 19723336, 19077215 21421886, 19604659, 21285458, 23533524, 23170620, 22365117, 18288842 19048007, 19308965, 19689979, 17409174, 23068169, 19503821, 24662775 25429959, 21526048, 19197175, 19180770, 24555417, 24573817, 19902195 26444887, 24835538, 23324000, 20318889, 21492036, 19013183, 20591183 19012119, 20464614, 21625179, 19067244, 23053606, 21632821, 19841800 19512341, 22695831, 20331945, 19587324, 24316947, 19578350, 19637186 19054077, 18674047, 19708632, 20898997, 19285025, 21091431, 19289642 25947799, 21133343, 20835241, 20869721, 21172913, 25602488, 19258504 17365043, 21419850, 21644640, 19468347, 21373473, 25093739, 16359751 24421668, 21164318, 25489607, 25484507, 22520320, 19769480, 19439759 19272708, 19978542, 19329654, 20402832, 19873610, 23229229, 13542050 21517440, 25897615, 19291380, 21915719, 25600342, 20879709, 20677396 19076343, 19561643, 19990037, 22897344, 18909599, 19487147, 25600421 20831538, 19016730, 18250893, 23240358, 22179537, 16619249, 18354830 24411921, 18254023, 16756406, 21188584, 19989009, 25766822, 17414008 20688221, 20441797, 20704450, 21780146, 25612095, 25957038, 24652769 25483815, 19157754, 19207117, 24437510, 18885870, 21785691, 20673810 24341675, 21450666, 18893947, 18705806, 22223463, 18417036, 16923858 23084507, 23314180, 20919320, 22503297, 20474192, 22046677, 21299490 19501299, 19385656, 20432873, 18542562, 20920911, 20899461, 21429602 21387128, 21315084, 18122373, 20581111, 26111842, 22624709, 19606174 24690216, 18436647, 19023822, 25110233, 19124589, 19178851, 19597583 18499088, 19050649

Version 12.1.0.2.v9

Version 12.1.0.2.v9 adds support for the following:

- Oracle July 2017 PSU, a combination of database PSU (patch 26609783) + OJVM component PSU (patch 26027162)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866)
- DBMS_STATS AUTO DOP COMPUTES A HIGH DOP UNNECESSARILY (patch 21171382)
- JSON bundle patch (patch 26083365)
- KGL heap size patch (patch 20033733 for 12.1.0.2)
- Timezone file DSTv30 (patch 25881255, OJVM patch 25881271)
- Adds support for [Validating DB Instance Files \(p. 859\)](#) with the RMAN logical validation utility
- Adds support for [Setting the Default Edition for a DB Instance \(p. 859\)](#)

Oracle patch 26609783, released July 2017

Bugs fixed: 21099555, 22175564, 19141838, 22083366, 20842388, 19865345, 20117253 19791273, 20671094, 21542577, 20951038, 19243521, 22165897, 19238590 21281532, 17008068, 19908836, 24577566, 21184223, 25427662, 19134173 20569094, 20031873, 20387265, 20322560, 21575362, 19149990, 21263635 17551063, 18886413, 22160989, 22507210, 19703301, 19366375, 18007682 19001390, 18202441, 24285405, 25655390, 20267166, 19358317, 19706965 19068970, 24739928, 18549238, 22148226, 18797519, 26544823, 20825533 21196809, 18940497, 19670108, 19649152, 18866977, 18948177, 22496904 19404068, 18964978, 19176326, 19035573, 20413820, 20717081, 19176223 21106027, 20904530, 20134339, 19074147, 20868862, 18411216, 21072646 25475853, 21322887, 22507234, 20425790, 20862087, 18966843, 21329301 20562898, 19333670, 19468991, 20124446, 19883092, 20878790, 18510194 19658708, 19591608, 19402853, 20618595, 21787056, 22380919, 21266085 19469538, 17835294, 19721304, 19068610, 19791377, 22178855, 16777441 22173980, 20746251, 20048359, 21896069, 19185876, 20898391, 20281121 20907061, 6599380, 19577410, 22092979, 19001359, 20603378, 23089357 21387964, 19490948, 22294260, 20832516, 17532734, 22351572, 19309466 19081128, 20627866, 20844426, 24908321, 21188532, 18791688, 21442094 20890311, 20596234, 20368850, 18973548, 19303936, 21296029, 20882568 21479753, 19461270, 20235511, 22077517, 20936905, 21220620, 18964939 19430401, 22296366, 21153266, 19409212, 22657942, 20703000, 20657441 19879746, 20557786, 19684504, 21294938, 19024808, 24693382, 20528052 20977794, 18799993, 20466322, 18740837, 19662635, 18440095, 20228093

19065556, 20212067, 25547060, 21868720, 22905130, 19524384, 25459958 24350831, 17722075, 20446883, 25056052, 18952989, 24523374, 16870214 19928926, 19835133, 21629064, 21354456, 20466628, 24386767, 25490238 19931709, 19730508, 18819908, 20250147, 23124895, 25643931, 23220453 19188927, 20074391, 18307021, 23533807, 20356733, 14643995, 18090142 19065677, 19547370, 21225209, 21960504, 26575788, 20397490, 20172151 18967382, 19174430, 21241829, 19536415, 26546664, 19171086, 21132297 21889720, 22465352, 22168163, 19335438, 24397438, 20076781, 20447445 18856999, 20471920, 19869255, 21620471, 18990693, 23096938, 19124336 17890099, 24812585, 18990023, 21300341, 20101006, 20848335, 21744290 20897759, 21668627, 19304354, 19052488, 20543011, 20794034, 23025340 25606091, 23260854, 18681056, 19562381, 20952966, 19896336, 20828947 25539063, 18618122, 20328248, 20440930, 18456643, 19699191, 22865673 19201867, 22022760, 21514877, 18743542, 20798891, 20347562, 25161298 23294548, 24560906, 22551446, 19777862, 19687159, 21373076, 19174942 20424899, 21899588, 18899974, 21476308, 20598042, 24308635, 21297872 19058490, 19032777, 20171986, 22815955, 19399918, 19434529, 19018447 18051556, 21273804, 22757364, 18851894, 19022470, 19284031, 18043064 20173897, 22062026, 20475845, 17274537, 19440586, 24825843, 18974476 22374754, 16887946, 17319928, 20401975, 20708701, 24674955, 22062517 22809871, 17655240, 19805359, 16439813, 19155797, 20859910, 19393542 17210525, 22024071, 19189525, 21847223, 21649497, 19075256, 25823754 25079710, 20315311, 22762046, 22075064, 20936731, 20437153, 18845653 19280225, 19248799, 20560611, 18988834, 21756699, 18921743, 20245930 18799063, 20373598, 20476175, 19571367, 20925795, 19018206, 25264559 20711718, 20509482, 20181030, 20588502, 21911701, 18849537, 23501901 19183343, 21917884, 21142837, 20603431, 19189317, 19644859, 19390567 26546754, 19279273, 20669434, 16863642, 22528741, 25546608, 19619732 20348653, 18607546, 19315691, 19676905, 20165574, 17867700, 20558005 20734332, 19532017, 20922010, 19818513, 19450314, 22353346, 16941434 20361671, 25423453, 20009833, 22366558, 20294666, 23197103, 18191823 19195895, 19371175, 19307662, 19154375, 20043616, 21977392, 18914624 22529728, 20139391, 25330273, 19593445, 21291274, 19382851, 19520602 19174521, 21875360, 19676012, 19326908, 20217801, 20093776, 18840932 21097043, 21246723, 20803014, 21665897, 19143550, 23026585, 20428621 19627012, 14283239, 21422580, 19213447, 19518079, 18610915, 18674024 24413809, 18306996, 19915271, 21626377, 19524158, 20122715, 20513399 20284155, 25091141, 21080143, 20017509, 22359063, 19363645, 19597439 21239530, 19383839, 20880215, 21756677, 19888853, 22458049, 19534363 19354335, 19044962, 19639483, 25982666, 19475971, 22353199, 21060755 22243719, 22916353, 20378086, 24808595, 21756661, 21260431, 22923409 19028800, 20877664, 21059919, 20879889, 21380789, 19723336, 19077215 21421886, 19604659, 21285458, 23533524, 23170620, 22365117, 18288842 19048007, 19308965, 19689979, 17409174, 19503821, 21526048, 19197175 19180770, 24573817, 19902195, 24835538, 23324000, 20318889, 19013183 20591183, 19012119, 20464614, 19067244, 21632821, 19841800, 19512341 22695831, 20331945, 19587324, 24316947, 19578350, 19637186, 19054077 18674047, 19708632, 20898997, 21091431, 19289642, 21133343, 20835241 20869721, 21172913, 19258504, 17365043, 21419850, 21644640, 19468347 21373473, 25093739, 16359751, 21164318, 25484507, 22520320, 19769480 19439759, 19272708, 19978542, 19329654, 20402832, 19873610, 23229229 13542050, 21517440, 19291380, 21915719, 25600342, 20879709, 20677396 19076343, 19561643, 19990037, 18909599, 19487147, 25600421, 20831538 19016730, 18250893, 16619249, 18354830, 24411921, 16756406, 18254023 21188584, 19989009, 25766822, 17414008, 20688221, 20441797, 20704450 21780146, 25612095, 25957038, 25483815, 19157754, 19207117, 24437510 18885870, 21785691, 20673810, 21450666, 18893947, 18705806, 22223463 18417036, 16923858, 23314180, 20919320, 20474192, 22046677, 21299490 19501299, 19385656, 20432873, 20920911, 20899461, 21387128, 21315084 18122373, 20581111, 22624709, 19606174, 24690216, 18436647, 19023822 25110233, 19124589, 19178851, 19597583, 18499088, 19050649

Version 12.1.0.2.v8

Version 12.1.0.2.v8 adds support for the following:

- Oracle patch 25433980, a combination of database PSU (patch 25171037) + OJVM component PSU (patch 25437695)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866 for 12.1.0.2)
- Oracle Forms patch 18307021 for 12.1.0.2

- DBMS_STATS Patch (patch 21171382 for 12.1.0.2)
- JSON bundle patch (patch 25531469 for 12.1.0.2)
- KGL heap size patch (patch 20033733 for 12.1.0.2)
- Fixed a bug that affected PSU apply after upgrade to 12.1.0.2.v5, v6, and v7
- Timezone file DSTv28 (patch 24701840)
- Adds support for the DBMS_CHANGE_NOTIFICATION package
- Adds support for xSTREAM packages and views (may require additional licensing)

Oracle patch 25171037, released April 2017

Bugs fixed: 21099555, 22175564, 19141838, 22083366, 20842388, 20117253, 19865345, 19791273, 21542577, 20951038, 19243521, 22165897, 17008068, 19908836, 21281532, 19238590, 24577566, 21184223, 19134173, 20569094, 20031873, 20322560, 20387265, 21575362, 19149990, 21263635, 17551063, 18886413, 22160989, 22507210, 19366375, 19703301, 19001390, 24285405, 18202441, 20267166, 19358317, 19706965, 19068970, 18549238, 24739928, 18797519, 22148226, 20825533, 21196809, 19649152, 19670108, 18940497, 18948177, 22496904, 18964978, 19176326, 19035573, 20413820, 19176223, 21106027, 20904530, 20134339, 19074147, 20868862, 18411216, 25475853, 21322887, 21072646, 22507234, 20425790, 20862087, 18966843, 21329301, 20562898, 19333670, 20124446, 19468991, 19883092, 20878790, 18510194, 19658708, 19591608, 19402853, 20618595, 21787056, 22380919, 19469538, 21266085, 17835294, 19721304, 19068610, 19791377, 22178855, 16777441, 22173980, 20048359, 20746251, 21896069, 19185876, 20898391, 20907061, 20281121, 6599380, 19577410, 22092979, 19001359, 20603378, 23089357, 21387964, 19490948, 22294260, 17532734, 20832516, 22351572, 19309466, 20627866, 19081128, 20844426, 21188532, 18791688, 20890311, 21442094, 20596234, 20368850, 18973548, 19303936, 21296029, 20882568, 19461270, 21479753, 22077517, 20936905, 20235511, 21220620, 18964939, 19430401, 22296366, 21153266, 19409212, 20703000, 22657942, 19879746, 20657441, 21294938, 19684504, 19024808, 20528052, 24693382, 20977794, 18799993, 20466322, 18740837, 19662635, 18440095, 20228093, 19065556, 20212067, 21868720, 22905130, 19524384, 24350831, 17722075, 20446883, 25056052, 18952989, 24523374, 16870214, 19928926, 19835133, 21629064, 21354456, 20466628, 24386767, 25490238, 19931709, 19730508, 18819908, 20250147, 23124895, 23220453, 19188927, 20074391, 18307021, 20356733, 14643995, 19065677, 19547370, 21960504, 21225209, 20397490, 18967382, 19174430, 21241829, 19536415, 19171086, 21889720, 22465352, 22168163, 19335438, 24397438, 20447445, 18856999, 19869255, 20471920, 21620471, 23096938, 18990693, 19124336, 17890099, 24812585, 18990023, 21300341, 20101006, 20848335, 21744290, 20897759, 21668627, 19304354, 20543011, 19052488, 20794034, 23025340, 23260854, 18681056, 20952966, 19896336, 25539063, 18618122, 20328248, 20440930, 18456643, 19699191, 19201867, 22865673, 22022760, 20798891, 18743542, 25161298, 20347562, 22551446, 19777862, 19687159, 21373076, 19174942, 20424899, 21899588, 18899974, 21476308, 20598042, 21297872, 24308635, 20171986, 19058490, 19032777, 22815955, 19399918, 19434529, 21273804, 19018447, 22757364, 18851894, 19022470, 19284031, 18043064, 20173897, 22062026, 20475845, 17274537, 19440586, 18974476, 24825843, 22374754, 16887946, 17319928, 20401975, 20708701, 22062517, 22809871, 17655240, 16439813, 19805359, 19155797, 20859910, 19393542, 22024071, 17210525, 19189525, 21847223, 21649497, 25079710, 19075256, 20315311, 22762046, 22075064, 20936731, 18845653, 19280225, 19248799, 20560611, 18988834, 21756699, 18921743, 20245930, 18799063, 20373598, 19571367, 20476175, 20925795, 19018206, 25264559, 20711718, 20509482, 20181030, 20588502, 21911701, 18849537, 23501901, 19183343, 21917884, 21142837, 19189317, 19644859, 19390567, 19279273, 20669434, 16863642, 22528741, 25546608, 19619732, 18607546, 20348653, 19315691, 19676905, 20165574, 17867700, 20558005, 20734332, 19532017, 20922010, 19818513, 19450314, 22353346, 16941434, 20361671, 20009833, 22366558, 20294666, 18191823, 23197103, 19195895, 19371175, 19307662, 19154375, 20043616, 21977392, 18914624, 22529728, 25330273, 20139391, 19593445, 21291274, 19382851, 19520602, 19174521, 21875360, 19676012, 19326908, 20217801, 20093776, 18840932, 21097043, 21246723, 20803014, 21665897, 19143550, 20428621, 19627012, 14283239, 21422580, 19213447, 19518079, 18610915, 18674024, 24413809, 18306996, 19915271, 19524158, 20122715, 20284155, 20017509, 22359063, 19363645, 19597439, 21239530, 19383839, 20880215, 21756677, 19888853, 22458049, 19534363, 19354335, 19044962, 19639483, 19475971, 22353199, 22243719, 21060755, 22916353,

20378086, 24808595 21756661, 21260431, 22923409, 19028800, 20877664, 21059919, 20879889, 21380789, 19723336, 19077215, 19604659, 21421886, 21285458, 23533524 23170620, 22365117, 18288842, 19048007, 19308965, 19689979, 19503821 21526048, 19197175, 19180770, 19902195, 23324000, 20318889, 19013183 20591183, 19012119, 20464614, 19067244, 21632821, 19841800, 19512341 22695831, 20331945, 19587324, 24316947, 19578350, 19637186, 19054077 18674047, 19708632, 20898997, 21091431, 19289642, 21133343, 20869721 21172913, 19258504, 17365043, 21419850, 19468347, 21373473, 25093739 16359751, 21164318, 22520320, 19769480, 19439759, 19272708, 19978542 19329654, 20402832, 19873610, 23229229, 13542050, 21517440, 19291380 21915719, 20879709, 20677396, 19076343, 19561643, 19990037, 19487147 18909599, 20831538, 19016730, 18250893, 16619249, 18354830, 24411921 16756406, 18254023, 21188584, 19989009, 17414008, 20688221, 20704450 20441797, 25483815, 19157754, 24437510, 18885870, 21785691, 20673810 21450666, 18893947, 18705806, 22223463, 16923858, 18417036, 23314180 20919320, 20474192, 22046677, 21299490, 19501299, 19385656, 20920911 20899461, 21387128, 21315084, 18122373, 20581111, 19606174, 24690216 18436647, 19023822, 19124589, 19178851, 19597583, 18499088, 19050649

Version 12.1.0.2.v7

Version 12.1.0.2.v7 adds support for the following:

- Oracle patch 24917069, a combination of database PSU (patch 24732082) + OJVM component PSU (patch 24917972)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866 for 12.1.0.2)
- Oracle Forms patch 18307021 for 12.1.0.2
- DBMS_STATS Patch (patch 21171382 for 12.1.0.2)
- JSON bundle patch (patch 25089615 for 12.1.0.2)
- KGL heap size patch (patch 20033733 for 12.1.0.2)

Oracle patch 24917069, released January 2017

Bugs fixed: 24917972, 25067795, 24534298, 25076732, 25076756, 24315824, 21659726 24448240, 24448282, 23177536, 22675136, 23265914, 23265965, 23727148 22674709, 22670413, 22670385, 21188537, 22139226, 22118835, 22118851 21555660, 21811517, 19623450, 21566993, 21566944, 19176885, 21068507 21047803, 21047766, 20415564, 20408829, 20408866, 19877336, 19855285 19909862, 19895362, 19895326, 19153980, 19231857, 19223010, 19245191, 19699946, 21099555, 22175564, 19141838, 22083366, 20842388, 20117253, 19865345 19791273, 21542577, 20951038, 19243521, 22165897, 19908836, 21281532 19238590, 24577566, 21184223, 19134173, 20031873, 20387265, 21575362 19149990, 21263635, 17551063, 18886413, 22160989, 22507210, 19366375 19703301, 19001390, 24285405, 18202441, 20267166, 19358317, 19706965 24739928, 19068970, 18549238, 18797519, 22148226, 20825533, 21196809 19649152, 19670108, 18940497, 18948177, 22496904, 18964978, 19035573 19176326, 20413820, 19176223, 21106027, 20904530, 20134339, 19074147 20868862, 18411216, 21072646, 21322887, 22507234, 20425790, 18966843 21329301, 20562898, 19333670, 20124446, 19468991, 19883092, 18510194 19658708, 19591608, 19402853, 20618595, 21787056, 22380919, 19469538 21266085, 17835294, 19721304, 19791377, 19068610, 22178855, 16777441 22173980, 20048359, 20746251, 21896069, 20898391, 19185876, 20907061 20281121, 6599380, 19577410, 22092979, 19001359, 20603378, 23089357 19490948, 21387964, 22294260, 20832516, 17532734, 19309466, 20627866 19081128, 20844426, 21188532, 18791688, 20890311, 21442094, 20596234 18973548, 21296029, 19303936, 2082568, 19461270, 21479753, 22077517 20936905, 20235511, 21220620, 18964939, 19430401, 22296366, 21153266 19409212, 22657942, 19879746, 20657441, 21294938, 19684504, 24693382 20528052, 19024808, 20977794, 18799993, 20466322, 18740837, 19662635 20228093, 20212067, 19065556, 19524384, 17722075, 20446883, 25056052 24523374, 18952989, 16870214, 19928926, 19835133, 21629064, 21354456 20466628, 24386767, 19931709, 19730508, 18819908, 23124895, 23220453 19188927, 20074391, 18307021, 20356733, 14643995, 19547370, 19065677 21960504, 21225209, 20397490, 18967382, 19174430, 21241829, 19536415 19171086, 22465352, 22168163, 19335438, 24397438, 20447445,

18856999 19869255, 20471920, 21620471, 18990693, 17890099, 24812585, 18990023 21300341, 20101006, 20848335, 21744290, 20897759, 21668627, 19304354 19052488, 20794034, 23025340, 23260854, 18681056, 20952966, 19896336 20328248, 18618122, 20440930, 18456643, 19699191, 19201867, 22865673 22022760, 20798891, 18743542, 25161298, 20347562, 19777862, 22551446 19687159, 21373076, 19174942, 20424899, 21899588, 18899974, 21476308 20598042, 24308635, 19032777, 19058490, 22815955, 19399918, 19434529 21273804, 19018447, 22757364, 18851894, 19022470, 19284031, 18043064 20173897, 22062026, 20475845, 17274537, 19440586, 24825843, 18974476 22374754, 16887946, 17319928, 20401975, 20708701, 22809871, 17655240 16439813, 19805359, 19155797, 20859910, 19393542, 17210525, 22024071 21847223, 19189525, 21649497, 19075256, 20315311, 22762046, 22075064 20936731, 19280225, 18845653, 20560611, 19248799, 21756699, 18988834 20245930, 18921743, 18799063, 20373598, 19571367, 20476175, 20925795 25264559, 19018206, 20711718, 20509482, 20181030, 20588502, 18849537 23501901, 19183343, 21917884, 19189317, 19644859, 19390567, 19279273 20669434, 22528741, 16863642, 19619732, 18607546, 20348653, 19315691 19676905, 20165574, 17867700, 20558005, 20734332, 19532017, 20922010 19818513, 19450314, 22353346, 20361671, 20009833, 22366558, 20294666 23197103, 18191823, 19195895, 19307662, 19371175, 20043616, 19154375 18914624, 22529728, 20139391, 21291274, 19382851, 19520602, 19174521 21875360, 19676012, 19326908, 20217801, 20093776, 18840932, 21097043 21246723, 20803014, 21665897, 19143550, 20428621, 19627012, 14283239 19518079, 18610915, 18674024, 24413809, 18306996, 19524158, 19915271 20122715, 20284155, 20017509, 22359063, 19363645, 19597439, 21239530 19888853, 21756677, 20880215, 22458049, 19534363, 19354335, 19044962 19639483, 19475971, 22353199, 21060755, 22243719, 22916353, 20378086 24808595, 21260431, 21756661, 22923409, 20877664, 19028800, 21059919 20879889, 21380789, 19723336, 19077215, 19604659, 21421886, 21285458 23533524, 23170620, 22365117, 18288842, 19308965, 19048007, 19689979 21526048, 19197175, 19180770, 19902195, 23324000, 20318889, 19013183 20591183, 19012119, 20464614, 19067244, 21632821, 19512341, 19841800 22695831, 20331945, 19587324, 24316947, 19578350, 19637186, 18674047 19054077, 20898997, 19708632, 21091431, 19289642, 21133343, 20869721 21172913, 19258504, 17365043, 19468347, 21373473, 16359751, 19769480 19439759, 19272708, 19978542, 20402832, 19329654, 19873610, 23229229 21517440, 13542050, 19291380, 21915719, 20879709, 20677396, 19076343 19561643, 19990037, 19487147, 18909599, 20831538, 18250893, 19016730 16619249, 18354830, 18254023, 21188584, 19989009, 17414008, 20688221 20704450, 20441797, 19157754, 24437510, 18885870, 21785691, 18893947 21450666, 18705806, 22223463, 16923858, 18417036, 23314180, 20919320 20474192, 22046677, 19385656, 19501299, 20920911, 20899461, 21315084 21387128, 18122373, 20581111, 19606174, 24690216, 18436647, 19023822 19178851, 19124589, 19597583, 18499088, 19050649

Version 12.1.0.2.v6

Version 12.1.0.2.v6 adds support for the following:

- Oracle patch 24433133, a combination of database PSU (patch 24006101) + OJVM component PSU (patch 24315824)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866 for 12.1.0.2)
- Oracle Forms patch 18307021 for 12.1.0.2
- DBMS_STATS Patch (patch 21171382 for 12.1.0.2)
- JSON bundle patch (patch 24568656 for 12.1.0.2)
- Fixed a bug that caused 12c upgrade scripts to drop customer directories
- Made DIAG log directory available to customers

Baseline: Oracle Database Patch Set Update 12.1.0.2.161018 (patch 24006101, released October 2016)

Bugs fixed: 21099555, 22175564, 19141838, 22083366, 20842388, 20117253, 19865345 19791273, 19243521, 20951038, 19908836, 21281532, 19238590, 24577566 21184223, 19134173, 20387265, 19149990, 21263635, 18886413, 17551063 22160989, 22507210, 19703301, 19366375, 19001390,

18202441, 20267166 19358317, 19706965, 18549238, 19068970, 18797519, 22148226, 20825533, 19649152, 19670108, 18940497, 18948177, 18964978, 19035573, 19176326 20413820, 19176223, 20904530, 20134339, 19074147, 20868862, 18411216 21322887, 22507234, 20425790, 18966843, 21329301, 19333670, 19468991 20124446, 19883092, 19658708, 19591608, 19402853, 20618595, 21787056 22380919, 21266085, 17835294, 19721304, 19791377, 19068610, 22178855 22173980, 20746251, 20048359, 20898391, 19185876, 20281121, 20907061 6599380, 19577410, 22092979, 20603378, 19001359, 19490948, 21387964 20832516, 17532734, 19309466, 19081128, 20627866, 20844426, 21188532 18791688, 21442094, 20890311, 20596234, 18973548, 21296029, 19303936, 19461270, 21479753, 20936905, 20235511, 21220620, 18964939, 19430401 22296366, 21153266, 19409212, 22657942, 20657441, 19879746, 19684504 20528052, 19024808, 20977794, 18799993, 20466322, 18740837, 19662635 20228093, 19065556, 20212067, 19524384, 17722075, 20446883, 18952989 16870214, 19928926, 19835133, 21629064, 20466628, 24386767, 19931709 19730508, 18819908, 23124895, 19188927, 20074391, 20356733, 14643995 19547370, 19065677, 21960504, 21225209, 20397490, 18967382, 19174430 21241829, 19536415, 19171086, 22465352, 22168163, 19335438, 20447445 18856999, 20471920, 19869255, 21620471, 18990693, 17890099, 18990023, 20101006, 21300341, 20848335, 21744290, 20897759, 21668627, 19304354 19052488, 20794034, 23260854, 18681056, 20952966, 19896336, 18618122 20328248, 20440930, 18456643, 19699191, 19201867, 22865673, 18743542 20798891, 20347562, 22551446, 19777862, 19687159, 21373076, 19174942 20424899, 21899588, 18899974, 20598042, 19032777, 19058490, 22815955 19399918, 19434529, 21273804, 19018447, 22757364, 18851894, 19284031 19022470, 18043064, 20173897, 22062026, 20475845, 17274537, 19440586 16887946, 22374754, 17319928, 20708701, 17655240, 16439813, 19805359 19155797, 20859910, 19393542, 22024071, 17210525, 21847223, 19189525, 21649497, 19075256, 22762046, 22075064, 19280225, 18845653, 20560611 19248799, 21756699, 18988834, 20245930, 18921743, 18799063, 20373598 20476175, 19571367, 20925795, 19018206, 20509482, 20711718, 20588502 18849537, 19183343, 21917884, 19189317, 19644859, 19390567, 19279273 20669434, 16863642, 22528741, 19619732, 18607546, 20348653, 19315691 19676905, 20165574, 17867700, 20558005, 20734332, 19532017, 20922010 19450314, 22353346, 20361671, 20009833, 22366558, 20294666, 18191823 19307662, 19371175, 19195895, 20043616, 19154375, 18914624, 20139391 21291274, 19174521, 19520602, 19382851, 21875360, 19676012, 19326908, 20217801, 20093776, 21097043, 21246723, 21665897, 19143550, 20428621 19627012, 14283239, 19518079, 18610915, 18674024, 18306996, 19524158 19915271, 20122715, 20284155, 20017509, 19363645, 19597439, 21239530 19888853, 20880215, 21756677, 19534363, 19354335, 19044962, 19639483 22353199, 22243719, 22916353, 20378086, 21756661, 21260431, 22923409 20877664, 19028800, 20879889, 19723336, 19077215, 21421886, 19604659 19308965, 19048007, 18288842, 19689979, 21526048, 19180770, 19197175 19902195, 20318889, 19013183, 19012119, 20464614, 19067244, 21632821 19512341, 19841800, 20331945, 19587324, 24316947, 19578350, 19637186, 18674047, 19054077, 20898997, 19708632, 21091431, 19289642, 20869721 19258504, 17365043, 19468347, 21373473, 16359751, 19439759, 19769480 19272708, 19978542, 20402832, 19329654, 19873610, 23229229, 21517440 13542050, 19291380, 21915719, 19076343, 19561643, 19990037, 19487147 18909599, 20831538, 18250893, 19016730, 16619249, 18354830, 21188584 19989009, 17414008, 20688221, 20704450, 20441797, 19157754, 18885870 21785691, 21450666, 18893947, 18705806, 22223463, 16923858, 18417036 20919320, 20474192, 22046677, 19385656, 19501299, 20920911, 20899461 21387128, 21315084, 18122373, 20581111, 19606174, 18436647, 19023822, 19178851, 19124589, 19597583, 18499088, 19050649

Version 12.1.0.2.v5

Version 12.1.0.2.v5 adds support for the following:

- Oracle patch 23615289, a combination of database PSU (patch 23054246) + OJVM component PSU (patch 23177536)
- Timezone file DSTv26 (patch 22873635 for 12.1.0.2)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866 for 12.1.0.2)
- Oracle Forms patch 18307021 for 12.1.0.2
- Added the ability to create custom password verify functions. For more information, see [Creating Custom Functions to Verify Passwords \(p. 850\)](#).

- Fixed a bug that prevented implicit recompilation of views owned by SYS

Baseline: Oracle Database Patch Set Update 12.1.0.2.160719 (patch 23054246, released July 2016)

Bugs fixed: 19189525, 21847223, 21099555, 21649497, 19075256, 19141838, 22762046 22075064, 20117253, 19865345, 19791273, 18845653, 19280225, 19248799 19243521, 20951038, 18988834, 21756699, 21281532, 19238590, 21184223 18921743, 20245930, 18799063, 19134173, 20373598, 19571367, 20476175 20925795, 19018206, 20509482, 20711718, 20387265, 20588502, 19149990 21263635, 18849537, 18886413, 17551063, 22507210, 19183343, 19366375 19703301, 21917884, 19001390, 18202441, 19189317, 20267166, 19644859 19390567, 19358317, 19279273, 19706965, 18549238, 16863642, 19068970 22528741, 18797519, 20825533, 19619732, 18607546, 20348653, 19649152 19670108, 18940497, 18948177, 19315691, 19676905, 18964978, 19176326 20165574, 19035573, 20413820, 17867700, 20558005, 19176223, 19532017 20904530, 20134339, 19450314, 19074147, 22353346, 20868862, 18411216 22507234, 20361671, 20425790, 18966843, 20009833, 22366558, 21329301 20294666, 18191823, 19333670, 19195895, 19371175, 19307662, 19154375 20043616, 20124446, 18914624, 19468991, 19883092, 21291274, 19382851 19520602, 19174521, 21875360, 19676012, 19326908, 19658708, 19591608 19402853, 20093776, 20618595, 21787056, 22380919, 21246723, 17835294 19721304, 19068610, 19791377, 21665897, 22178855, 22173980, 20048359 20746251, 19143550, 20898391, 19185876, 19627012, 20281121, 19577410 22092979, 19001359, 14283239, 19518079, 18610915, 19490948, 17532734 18674024, 18306996, 19309466, 19081128, 19524158, 19915271, 20122715 21188532, 18791688, 20284155, 20890311, 21442094, 20596234, 18973548 21296029, 19303936, 19597439, 20936905, 20235511, 21220620, 20880215 18964939, 21756677, 19888853, 19534363, 19430401, 19354335, 19044962 19639483, 22296366, 22353199, 21153266, 19409212, 19879746, 20657441 19684504, 20528052, 19024808, 20977794, 20378086, 18799993, 21756661 21260431, 18740837, 22923409, 19028800, 20877664, 20228093, 20879889 19065556, 19723336, 19077215, 19604659, 21421886, 19524384, 17722075 19308965, 18288842, 19048007, 19689979, 20446883, 18952989, 16870214 19928926, 19835133, 21629064, 21526048, 19197175, 19180770, 20466628 19902195, 19931709, 20318889, 19013183, 19730508, 19012119, 19067244 20074391, 20356733, 14643995, 19512341, 19841800, 20331945, 19587324 19065677, 19547370, 19578350, 21225209, 19637186, 20397490, 18967382 19174430, 21241829, 19054077, 18674047, 20898997, 19708632, 19536415 21091431, 19289642, 20869721, 22168163, 19335438, 19258504, 20447445 17365043, 18856999, 19468347, 19869255, 20471920, 21373473, 21620471 16359751, 18990693, 17890099, 19769480, 19439759, 19272708, 18990023 19978542, 19329654, 20101006, 21300341, 20402832, 19873610, 20848335 23229229, 21744290, 21668627, 21517440, 13542050, 19304354, 19052488 20794034, 19291380, 21915719, 23260854, 18681056, 20952966, 19896336 19076343, 19561643, 18618122, 19990037, 20440930, 18456643, 19699191 19201867, 19487147, 18909599, 20831538, 19016730, 18250893, 20798891 18743542, 20347562, 16619249, 18354830, 22551446, 19777862, 19687159 21373076, 19174942, 20424899, 21188584, 19989009, 17414008, 20688221 21899588, 20441797, 19157754, 19058490, 19032777, 22815955, 19399918 18885870, 19434529, 21273804, 19018447, 21450666, 18893947, 18851894 16923858, 18417036, 20919320, 19022470, 19284031, 20474192, 20173897 22046677, 22062026, 19501299, 19385656, 20920911, 17274537, 20899461 21315084, 19440586, 16887946, 22374754, 17319928, 19606174, 20708701 18436647, 17655240, 19023822, 19124589, 19178851, 16439813, 19805359 19597583, 18499088, 19155797, 19050649, 19393542

Version 12.1.0.2.v4

Version 12.1.0.2.v4 adds support for the following:

- Oracle PSU 12.1.0.2.160419 (22291127)
- Timezone file DSTv25 (patch 22037014)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866)
- Adds the ability for the master user to grant the EM_EXPRESS_BASIC and EM_EXPRESS_ALL roles

- Adds the ability for the master user to grant privileges on SYS objects with the grant option using the RDSADMIN.RDSADMIN_UTIL.GRANT_SYS_OBJECT procedure
- Adds master user privileges to support most common schemas created by the Oracle Fusion Middleware Repository Creation Utility (RCU)

Baseline: Oracle Database Patch Set Update 12.1.0.2.160419 (patch 22291127, released April 2016)

Bugs fixed: 21847223, 19189525, 19075256, 19141838, 22762046, 20117253, 19865345 19791273, 19280225, 18845653, 19248799, 20951038, 19243521, 21756699 18988834, 21281532, 19238590, 18921743, 20245930, 18799063, 19134173 20373598, 19571367, 20476175, 20925795, 19018206, 20711718, 20387265 20509482, 20588502, 19149990, 18849537, 17551063, 18886413, 19183343 19703301, 21917884, 19001390, 18202441, 19189317, 19644859, 19358317 19390567, 19279273, 19706965, 22528741, 19068970, 20825533, 19619732 18607546, 20348653, 19649152, 19670108, 18940497, 18948177, 19315691 19676905, 18964978, 19035573, 20165574, 19176326, 20413820, 20558005 19176223, 19532017, 20904530, 20134339, 19450314, 22353346, 19074147 18411216, 20361671, 20425790, 18966843, 21329301, 20294666, 19333670 19195895, 19307662, 19371175, 20043616, 19154375, 20124446, 18914624 19468991, 19883092, 19382851, 19520602, 19174521, 21875360, 19676012 19326908, 19658708, 19591608, 20093776, 20618595, 21787056, 17835294 19721304, 19791377, 19068610, 22173980, 20746251, 20048359, 19143550 19185876, 19627012, 20281121, 19577410, 22092979, 19001359, 19518079 18610915, 19490948, 18674024, 18306996, 19309466, 19081128, 19915271 20122715, 21188532, 18791688, 20284155, 20890311, 21442094, 20596234 18973548, 19303936, 19597439, 20936905, 20235511, 19888853, 21756677 18964939, 19354335, 19430401, 19044962, 19639483, 21153266, 22353199 19409212, 20657441, 19879746, 19684504, 19024808, 21260431, 21756661 18799993, 20877664, 19028800, 20879889, 19065556, 19723336, 19077215 19604659, 21421886, 19524384, 18288842, 19048007, 19689979, 20446883 18952989, 16870214, 19928926, 19835133, 21526048, 20466628, 19197175 19180770, 19902195, 20318889, 19730508, 19012119, 19067244, 20074391 20356733, 14643995, 19512341, 19841800, 20331945, 19587324, 19547370 19065677, 21225209, 19637186, 20397490, 18967382, 19174430, 19054077 18674047, 19536415, 19708632, 21091431, 19289642, 22168163, 20869721 19335438, 19258504, 20447445, 17365043, 18856999, 19468347, 20471920 19869255, 21620471, 16359751, 18990693, 17890099, 19769480, 19439759 19272708, 18990023, 19978542, 20402832, 20101006, 21300341, 19329654 19873610, 21744290, 13542050, 21517440, 21668627, 19304354, 19052488 20794034, 19291380, 21915719, 18681056, 20952966, 19896336, 19076343 19561643, 19990037, 18618122, 20440930, 18456643, 19699191, 19487147 18909599, 20831538, 18250893, 19016730, 18743542, 20347562, 16619249 18354830, 19777862, 19687159, 19174942, 20424899, 19989009, 20688221 21899588, 20441797, 19157754, 19032777, 19058490, 19399918, 18885870 19434529, 21273804, 19018447, 18893947, 16923858, 18417036, 20919320 19022470, 19284031, 20474192, 22046677, 20173897, 22062026, 19385656 19501299, 17274537, 20899461, 21315084, 19440586, 22374754, 16887946 19606174, 18436647, 17655240, 19023822, 19178851, 19124589, 16439813 19805359, 19597583, 18499088, 19155797, 19050649, 19393542

Version 12.1.0.2.v3

Version 12.1.0.2.v3 adds support for the following:

- Oracle PSU 12.1.0.2.160119 (21948354).
- Timezone file DSTv25 (patch 22037014 for 12.1.0.2). 12.1.0.1 includes DSTv24, patch 20875898 (unchanged from 12.1.0.1.v3), because a backport of DSTv25 was unavailable at build time.
- Fixed an issue that prevented customers from creating more than 10 Directory objects in the database.
- Fixed an issue that prevented customers from re-granting read privileges on the ADUMP and BDUMP Directory objects.

Baseline: Oracle Database Patch Set Update 12.1.0.2.160119 (patch 21948354, released January 2016)

Bugs fixed: 19189525, 19075256, 19141838, 19865345, 19791273, 19280225, 18845653 20951038, 19243521, 19248799, 21756699, 18988834, 19238590, 21281532 20245930, 18921743, 18799063, 19134173, 19571367, 20476175, 20925795 19018206, 20509482, 20387265, 20588502, 19149990, 18849537, 18886413 17551063, 19183343, 19703301, 19001390, 18202441, 19189317, 19644859 19358317, 19390567, 19279273, 19706965, 19068970, 19619732, 20348653 18607546, 18940497, 19670108, 19649152, 18948177, 19315691, 19676905 18964978, 19035573, 20165574, 19176326, 20413820, 20558005, 19176223 19532017, 20134339, 19074147, 18411216, 20361671, 20425790, 18966843 20294666, 19307662, 19371175, 19195895, 19154375, 19468991, 19174521 19520602, 19382851, 21875360, 19326908, 19658708, 20093776, 20618595 21787056, 17835294, 19791377, 19068610, 20048359, 20746251, 19143550 19185876, 19627012, 20281121, 19577410, 22092979, 19001359, 19518079 18610915, 19490948, 18674024, 18306996, 19309466, 19081128, 19915271 20122715, 21188532, 20284155, 18791688, 20890311, 21442094, 18973548 19303936, 19597439, 20235511, 18964939, 19430401, 19044962, 19409212 19879746, 20657441, 19684504, 19024808, 18799993, 20877664, 19028800 19065556, 19723336, 19077215, 19604659, 21421886, 19524384, 19048007 18288842, 19689979, 20446883, 18952989, 16870214, 19928926, 21526048 19180770, 19197175, 19902195, 20318889, 19730508, 19012119, 19067244 20074391, 19512341, 19841800, 14643995, 20331945, 19587324, 19547370 19065677, 19637186, 21225209, 20397490, 18967382, 19174430, 18674047 19054077, 19536415, 19708632, 19289642, 20869721, 19335438, 17365043 18856999, 19869255, 20471920, 19468347, 21620471, 16359751, 18990693 17890099, 19439759, 19769480, 19272708, 19978542, 20101006, 21300341 20402832, 19329654, 19873610, 21668627, 21517440, 19304354, 19052488 20794034, 19291380, 18681056, 19896336, 19076343, 19561643, 18618122 20440930, 18456643, 19699191, 18909599, 19487147, 18250893, 19016730 18743542, 20347562, 16619249, 18354830, 19687159, 19174942, 20424899 19989009, 20688221, 20441797, 19157754, 19032777, 19058490, 19399918 18885870, 19434529, 19018447, 18417036, 20919320, 19022470, 19284031 20474192, 20173897, 22062026, 19385656, 19501299, 17274537, 20899461 19440586, 16887946, 19606174, 18436647, 17655240, 19023822, 19178851 19124589, 19805359, 19597583, 19155797, 19393542, 19050649

Version 12.1.0.2.v2

Version 12.1.0.2.v2 adds support for the following:

- Oracle PSU 12.1.0.2.5 (21359755)
- Includes the Daylight Saving Time Patch, patch 20875898: DST-24, that came out after the April 2015 PSU.

Baseline: Oracle Database Patch Set Update 12.1.0.2.5 (patch 21359755, released October 2015)

Bugs fixed: 19189525, 19075256, 19865345, 19791273, 19280225, 18845653, 19248799 19243521, 18988834, 19238590, 21281532, 18921743, 20245930, 19134173 19571367, 20476175, 20925795, 19018206, 20387265, 19149990, 18849537 19183343, 19703301, 19001390, 18202441, 19189317, 19644859, 19390567 19358317, 19279273, 19706965, 19068970, 19619732, 18607546, 20348653 18940497, 19670108, 19649152, 18948177, 19315691, 19676905, 18964978 20165574, 19035573, 19176326, 20413820, 20558005, 19176223, 19532017 20134339, 19074147, 18411216, 20361671, 20425790, 18966843, 20294666 19371175, 19307662, 19195895, 19154375, 19468991, 19174521, 19520602 19382851, 19658708, 20093776, 17835294, 19068610, 19791377, 20746251 20048359, 19143550, 19185876, 19627012, 20281121, 19577410, 19001359 19518079, 18610915, 18674024, 18306996, 19309466, 19081128, 19915271 20122715, 20284155, 18791688, 21442094, 19303936, 19597439, 20235511 18964939, 19430401, 19044962, 19409212, 20657441, 19684504, 19024808, 19028800, 19065556, 19723336, 19077215, 21421886, 19524384, 19048007 18288842, 18952989, 16870214, 19928926, 19180770, 19197175, 19730508 19012119, 19067244, 20074391, 19841800,

19512341, 14643995, 20331945 19587324, 19065677, 19547370, 19637186, 21225209, 20397490, 18967382 19174430, 18674047, 19054077, 19708632, 19536415, 19289642, 19335438 17365043, 18856999, 20471920, 19468347, 21620471, 16359751, 18990693 19439759, 19769480, 19272708, 19978542, 19329654, 20402832, 19873610 19304354, 19052488, 19291380, 18681056, 19896336, 19076343, 19561643 18618122, 20440930, 18456643, 19699191, 18909599, 19487147, 18250893 19016730, 18743542, 20347562, 16619249, 18354830, 19687159, 19174942 20424899, 19989009, 20688221, 20441797, 19157754, 19058490, 19032777 19399918, 18885870, 19434529, 19018447, 18417036, 20919320, 19284031 19022470, 20474192, 22062026, 19385656, 19501299, 17274537, 20899461 19440586, 19606174, 18436647, 19023822, 19178851, 19124589, 19805359 19597583, 19155797, 19393542, 19050649

Version 12.1.0.2.v1

Version 12.1.0.2.v1 adds support for the following:

- Oracle PSU 12.1.0.2.3 (20299023)
- The In-Memory option allows storing a subset of data in an in-memory column format optimized for performance.
- Installs additional Oracle Text knowledge bases from Oracle Database. Examples media (English and French)
- Provides access to DBMS_REPAIR through RDSADMIN.RDSADMIN_DBMS_REPAIR
- Grants ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, and EXEMPT REDACTION POLICY to master user

Note

Version 12.1.0.2.v1 supports Enterprise Edition only.

Baseline: Oracle Database Patch Set Update 12.1.0.2.3 (patch 20299023, released April 2015)

Bugs fixed: 19189525, 19065556, 19075256, 19723336, 19077215, 19865345, 18845653 19280225, 19524384, 19248799, 18988834, 19048007, 18288842, 19238590 18921743, 18952989, 16870214, 19928926, 19134173, 19180770, 19018206 19197175, 19149990, 18849537, 19730508, 19183343, 19012119, 19001390 18202441, 19067244, 19189317, 19644859, 19358317, 19390567, 20074391 19279273, 19706965, 19068970, 19841800, 19512341, 14643995, 19619732 20348653, 18607546, 18940497, 19670108, 19649152, 19065677, 19547370 18948177, 19315691, 19637186, 19676905, 18964978, 19035573, 19176326 18967382, 19174430, 19176223, 19532017, 18674047, 19074147, 19054077 19536415, 19708632, 19289642, 20425790, 19335438, 18856999, 19371175 19468347, 19195895, 19154375, 16359751, 18990693, 19439759, 19769480 19272708, 19978542, 19329654, 19873610, 19174521, 19520602, 19382851 19658708, 19304354, 19052488, 19291380, 18681056, 19896336, 17835294 19076343, 19791377, 19068610, 19561643, 18618122, 20440930, 18456643 18909599, 19487147, 19143550, 19185876, 19016730, 18250893, 20347562 19627012, 16619249, 18354830, 19577410, 19687159, 19001359, 19174942 19518079, 18610915, 18674024, 18306996, 19309466, 19081128, 19915271 19157754, 19058490, 20284155, 18791688, 18885870, 19303936, 19434529 19018447, 18417036, 19597439, 20235511, 19022470, 18964939, 19430401 19044962, 19385656, 19501299, 17274537, 19409212, 19440586, 19606174 18436647, 19023822, 19684504, 19178851, 19124589, 19805359, 19024808 19597583, 19155797, 19393542, 19050649, 19028800

Related Topics

- [Upgrading the Oracle DB Engine \(p. 770\)](#)
- [Oracle on Amazon RDS \(p. 717\)](#)

Database Engine: 11.2.0.4

The following versions are available for database engine 11.2.0.4:

- [Version 11.2.0.4.v18 \(p. 944\)](#)
- [Version 11.2.0.4.v17 \(p. 946\)](#)
- [Version 11.2.0.4.v16 \(p. 947\)](#)
- [Version 11.2.0.4.v15 \(p. 949\)](#)
- [Version 11.2.0.4.v14 \(p. 950\)](#)
- [Version 11.2.0.4.v13 \(p. 951\)](#)
- [Version 11.2.0.4.v12 \(p. 953\)](#)
- [Version 11.2.0.4.v11 \(p. 954\)](#)
- [Version 11.2.0.4.v10 \(p. 955\)](#)
- [Version 11.2.0.4.v9 \(p. 956\)](#)
- [Version 11.2.0.4.v8 \(p. 957\)](#)
- [Version 11.2.0.4.v7 \(p. 959\)](#)
- [Version 11.2.0.4.v6 \(p. 960\)](#)
- [Version 11.2.0.4.v5 \(p. 960\)](#)
- [Version 11.2.0.4.v4 \(p. 961\)](#)
- [Version 11.2.0.4.v3 \(p. 962\)](#)
- [Version 11.2.0.4.v2 \(Deprecated\) \(p. 963\)](#)
- [Version 11.2.0.4.v1 \(p. 963\)](#)

Version 11.2.0.4.v18

Version 11.2.0.4.v18 adds support for the following:

- Patch 28204707: Oracle Database Patch Set Update 11.2.0.4.181016
- Patch 28440700: Oracle JVM Patch Set Update 11.2.0.4.181016
- Patch 28125601: DSTv32 for RDBMS (TZDATA2018E)
- Patch 27015468: DSTv32 for OJVM (TZDATA2018E)
- Patch 27216420: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patches 27659043 and 19692824 are now included in the Database Patch Set Update

Oracle patch 28204707, released October 2018

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 17205719, 18607546, 25654936 17484762, 17816865, 20506699, 24835538, 25957038, 19692824, 23330119 17922254, 17754782, 13364795, 16934803, 17311728, 18604692, 26679352 20387265, 17284817, 17441661, 20671094, 24560906, 25635149, 16992075 17446237, 14015842, 19972569, 21756677, 17375354, 21538558, 20925795 17449815, 17019086, 19463897, 26575788, 13866822, 17235750, 17982555 17478514, 18317531, 14338435, 18235390, 19461270, 20803583, 13944971 19475971, 20142975, 17811789, 16929165, 18704244, 24662775, 20506706 17359610, 17546973, 21422580, 20334344, 14054676, 25489607, 17570606 17088068, 17346091, 18264060, 17343514, 21538567, 19680952, 18471685 19211724, 21132297, 25775213, 13951456, 16315398, 21847223, 18744139 16850630, 23177648, 19049453, 18090142, 18673304, 17883081, 19915271 18641419, 18262334, 25600421, 17006183, 16065166, 18277454, 18685892 16833527, 10136473, 18051556, 17865671, 25879984, 18554871, 17852463 18774543, 17853498, 18334586, 19487147, 20879889, 17551709, 17588480 19827973, 17344412, 17842825, 18828868, 20509482, 17025461, 26039623 19429927, 13609098, 11883252, 16410570, 17239687, 23007241, 17602269 19197175, 22195457, 18316692, 17313525, 12611721, 21174504,

19544839 20294666, 18964939, 17600719, 26667015, 18191164, 17571306, 19393542 20777150, 18482502, 27086138, 19466309, 22243719, 17165204, 17040527 18098207, 24790914, 16785708, 19891090, 17465741, 16180763, 17174582 12982566, 16777840, 19463893, 22195465, 16875449, 22148226, 12816846 17237521, 6599380, 19358317, 17811438, 25505394, 17811447, 21983325 17945983, 18762750, 16912439, 17184721, 18061914, 20598042, 26631046 21380789, 17282229, 18948177, 18331850, 21142837, 18202441, 17082359 18723434, 21972320, 21532755, 19554106, 25505371, 20273319, 14034426 18339044, 19458377, 17752995, 20448824, 17891943, 17767676, 17258090 16668584, 18384391, 21063322, 17040764, 17381384, 15913355, 18356166 14084247, 20596234, 21641760, 20506715, 13853126, 21756661, 18610915 18203837, 14245531, 16043574, 21756699, 22195441, 17848897, 17877323 26667032, 21453153, 19272701, 20569094, 17468141, 17786518, 20861693 17912217, 17037130, 16956380, 18155762, 17478145, 17394950, 18641461 18189036, 17551674, 18619917, 17019356, 17027426, 21352646, 16268425 24476274, 22195492, 19584068, 26544823, 18436307, 22507210, 17265217 13498382, 17634921, 19469538, 21526048, 19258504, 23003979, 16354467 18043064, 19174430, 20004087, 17443671, 22195485, 18000422, 22321756 20004021, 17571039, 25897615, 27053456, 16832076, 21067387, 22905130 16344544, 21429602, 18009564, 14354737, 21286665, 18135678, 14521849 18614015, 20441797, 18362222, 25655390, 16472716, 17835048, 17050888 17936109, 14010183, 17325413, 18747196, 19207156, 17231779, 21842740 17761775, 16721594, 17082983, 20067212, 21179898, 17279227, 17302277 18084625, 20717359, 24624166, 15990359, 24842886, 26746894, 18203835 23026585, 17297939, 17811456, 16731148, 22380919, 21168487, 14133975 13829543, 17215560, 18740837, 17694209, 17385178, 18091059, 8322815 18259031, 28254374, 19689979, 25165496, 17586955, 17201159, 17655634 18331812, 17551699, 19730508, 17648596, 18868646, 16220077, 16069901 17393915, 17348614, 17957017, 17274537, 18096714, 17308789, 18436647 14285317, 19289642, 14764829, 17622427, 18328509, 23115139, 16943711 22195477, 22502493, 14368995, 17346671, 18996843, 17783588, 18604493 21343838, 16618694, 17672719, 18856999, 18783224, 17851160, 17546761 22168163, 17798953, 18273830, 22092979, 16596890, 19972566, 13871092 20828947, 26667023, 17726838, 16384983, 22296366, 17360606, 13645875 22321741, 25634317, 16542886, 18199537, 25879656, 21787056, 23140259 17889549, 21172913, 26245237, 14565184, 27825893, 20475845, 17071721 21281607, 17610798, 18308268, 20299015, 21343897, 22893153, 22594718 20657441, 17397545, 18230522, 16360112, 19769489, 12905058, 18641451 12747740, 18430495, 25423453, 17016369, 17042658, 14602788, 17551063 26243698, 19972568, 21517440, 23725036, 19788842, 18508861, 14657740 17332800, 13837378, 17186905, 19972564, 17019345, 19699191, 18315328 27441326, 17437634, 24570598, 22353199, 18093615, 19006849, 17392698 19013183, 17296856, 18674024, 26569225, 17232014, 16855292, 21051840 14692762, 17762296, 17705023, 23294548, 22351572, 22507234, 19121551 20324049, 21330264, 26198926, 19854503, 23315889, 26910644, 26030218 21868720, 19309466, 25764020, 18681862, 17365043, 17390160, 20031873 20558005, 18554763, 24717859, 21795111, 18456514, 13955826, 16306373 18139690, 17501491, 17752121, 17299889, 21668627, 23713236, 24652769 17889583, 18673325, 22551446, 18674465, 17242746, 19721304, 18293054 19211433, 19888853, 25914276, 24563422, 17951233, 18094246, 17649265 19615136, 17011832, 17477958, 16870214, 18522509, 20631274, 16091637 17323222, 16595641, 16524926, 17484731, 18228645, 18282562, 17596908 18272672, 18031668, 17156148, 16494615, 22683225, 20869721, 17545847 25093656, 18682983, 17655240, 24528741, 17614134, 25427662, 13558557 17341326, 22465352, 17891946, 17716305, 22657942, 27374796, 16392068 18440095, 19271443, 21351877, 20513399, 18092127, 17614227, 18440047 18849970, 16903536, 14106803, 20725343, 18973907, 18673342, 17389192 19032867, 25505382, 22809871, 17612828, 17006570, 16194160, 25369547 25505407, 16685417, 17721717, 21354456, 17390431, 17570240, 13960236 16863422, 28100487, 18325460, 17008068, 19727057, 16422541, 17267114 19972570, 18244962, 21538485, 18203838, 18765602, 16198143, 17246576 14829250, 28364007, 17835627, 20860659, 21629064, 18247991, 14458214 21051862, 17786278, 16692232, 17227277, 24348685, 24476265, 16042673 24975421, 22901797, 16314254, 19285025, 16228604, 16756406, 14176370 16837842, 20144308, 17393683, 23536835, 25823754, 18899974, 17787259 24719736, 20331945, 26078387, 19490948, 20074391, 15861775, 16399083 25555252, 25947799, 18018515, 22683212, 18260550, 21051858, 17080436 16613964, 17036973, 16579084, 24433711, 18384537, 27870645, 18280813 20296213, 16901385, 15979965, 17518652, 23330124, 20856766, 18441944 16450169, 9756271, 27534509, 22730454, 19718981, 17291347, 17892268 11733603, 16285691, 17587063, 21343775, 18180390, 26474853, 16538760 18193833, 21387964, 21051833, 17238511, 19777862, 17824637,

23065323 21656630, 17903598, 16571443, 18166013, 18306996, 19578350, 14852021 17853456,
18674047, 12364061, 24411921, 19207117, 22195448

Version 11.2.0.4.v17

Version 11.2.0.4.v17 adds support for the following:

- Patch 27734982: Oracle Database Patch Set Update 11.2.0.4.180717
- Patch 27923163: Oracle JVM Patch Set Update 11.2.0.4.180717
- Patch 28125601: DSTv32 for RDBMS (TZDATA2018E)
- Patch 27015468: DSTv32 for OJVM (TZDATA2018E)
- Patch 27216420: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 27659043: MES Bundle 405
- Patch 19692824: DBCONTROL is not coming up on OEL 7

Oracle patch 27734982, released July 2018

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 17205719, 18607546, 25654936 17816865, 20506699, 24835538, 25957038, 23330119, 17922254, 17754782 13364795, 16934803, 17311728, 20387265, 17284817, 17441661, 20671094 24560906, 16992075, 17446237, 14015842, 19972569, 21756677, 17375354 21538558, 20925795, 17449815, 19463897, 26575788, 13866822, 17235750 17982555, 17478514, 18317531, 14338435, 18235390, 19461270, 20803583 13944971, 19475971, 20142975, 17811789, 16929165, 18704244, 24662775 20506706, 17546973, 21422580, 20334344, 14054676, 25489607, 17088068 17346091, 18264060, 17343514, 21538567, 19680952, 18471685, 19211724 21132297, 13951456, 16315398, 21847223, 18744139, 16850630, 23177648 19049453, 18090142, 18673304, 17883081, 19915271, 18641419, 18262334 25600421, 17006183, 16065166, 18277454, 16833527, 10136473, 18051556 17865671, 18554871, 17852463, 18774543, 17853498, 18334586, 19487147 20879889, 17551709, 17588480, 19827973, 17344412, 17842825, 18828868 20509482, 17025461, 13609098, 11883252, 17239687, 23007241, 17602269 19197175, 22195457, 18316692, 17313525, 12611721, 21174504, 19544839 20294666, 18964939, 17600719, 26667015, 18191164, 17571306, 19393542 20777150, 18482502, 27086138, 19466309, 22243719, 17165204, 17040527 18098207, 16785708, 17465741, 16180763, 17174582, 12982566, 16777840 19463893, 22195465, 16875449, 22148226, 12816846, 17237521, 6599380 19358317, 17811438, 25505394, 17811447, 21983325, 17945983, 18762750 16912439, 17184721, 18061914, 20598042, 21380789, 17282229, 18948177 18331850, 21142837, 18202441, 17082359, 18723434, 21972320, 21532755 19554106, 25505371, 14034426, 18339044, 19458377, 17752995, 20448824 17891943, 17767676, 17258090, 16668584, 18384391, 17040764, 17381384 15913355, 18356166, 14084247, 20596234, 21641760, 20506715, 13853126 21756661, 18203837, 14245531, 16043574, 21756699, 22195441, 17848897 17877323, 21453153, 19272701, 20569094, 17468141, 17786518, 20861693 17912217, 17037130, 16956380, 18155762, 17478145, 17394950, 18641461 18189036, 18619917, 17027426, 21352646, 16268425, 24476274, 22195492 19584068, 26544823, 18436307, 22507210, 17265217, 13498382, 17634921 19469538, 21526048, 19258504, 23003979, 18043064, 19174430, 20004087 17443671, 22195485, 18000422, 20004021, 22321756, 17571039, 27053456 25897615, 21067387, 16832076, 22905130, 16344544, 21429602, 18009564 14354737, 21286665, 18135678, 14521849, 18614015, 20441797, 18362222 25655390, 16472716, 17835048, 17050888, 17936109, 14010183, 17325413 18747196, 17761775, 16721594, 17082983, 20067212, 21179898, 17302277 18084625, 20717359, 24624166, 15990359, 24842886, 26746894, 18203835 23026585, 17297939, 17811456, 16731148, 22380919, 21168487, 14133975 13829543, 17215560, 17694209, 17385178, 18091059, 8322815, 18259031 19689979, 25165496, 17586955, 17201159, 17655634, 18331812, 19730508 17648596, 18868646, 16220077, 16069901, 17393915, 17348614, 17957017 17274537, 18096714, 17308789, 18436647, 14285317, 19289642, 14764829 17622427, 18328509, 16943711, 22195477, 22502493, 14368995, 17346671 18996843, 17783588, 18604493, 21343838, 16618694, 17672719, 18856999 18783224, 17851160, 17546761, 22168163, 17798953, 18273830, 22092979 16596890, 19972566, 20828947, 13871092, 26667023, 17726838, 16384983 22296366, 17360606, 13645875, 22321741, 16542886, 18199537, 25879656 21787056, 17889549, 21172913, 14565184, 27825893,

20475845, 17071721 21281607, 18308268, 17610798, 20299015, 21343897, 22893153, 20657441
17397545, 18230522, 16360112, 19769489, 12905058, 18641451, 12747740 18430495, 25423453,
17016369, 17042658, 14602788, 17551063, 19972568 21517440, 23725036, 19788842, 18508861,
14657740, 17332800, 13837378 17186905, 19972564, 19699191, 18315328, 27441326, 17437634,
24570598 22353199, 18093615, 19006849, 17392698, 19013183, 17296856, 18674024 26569225,
17232014, 16855292, 21051840, 14692762, 17762296, 17705023 23294548, 22351572, 22507234,
19121551, 20324049, 21330264, 26198926 19854503, 23315889, 26910644, 26030218, 21868720,
19309466, 25764020 18681862, 17365043, 17390160, 20031873, 20558005, 18554763, 24717859
21795111, 18456514, 16306373, 13955826, 18139690, 17501491, 17752121 17299889, 21668627,
23713236, 24652769, 17889583, 18673325, 22551446 17242746, 19721304, 18293054, 19211433,
19888853, 24563422, 17951233 18094246, 17649265, 19615136, 17011832, 17477958, 16870214,
18522509 20631274, 16091637, 17323222, 16595641, 16524926, 18228645, 18282562 17596908,
18031668, 17156148, 16494615, 22683225, 20869721, 17545847 25093656, 17655240, 24528741,
17614134, 25427662, 13558557, 17341326 22465352, 17891946, 17716305, 22657942, 27374796,
16392068, 18440095 19271443, 21351877, 20513399, 18092127, 17614227, 18440047, 18849970
16903536, 14106803, 18973907, 18673342, 17389192, 19032867, 25505382 22809871, 17612828,
17006570, 16194160, 25369547, 25505407, 16685417 17721717, 21354456, 17390431, 17570240,
16863422, 28100487, 18325460 17008068, 19727057, 16422541, 17267114, 19972570, 18244962,
21538485 18203838, 18765602, 16198143, 17246576, 14829250, 17835627, 20860659 21629064,
18247991, 14458214, 21051862, 17786278, 16692232, 24348685 17227277, 24476265, 16042673,
16314254, 19285025, 16228604, 16756406 16837842, 20144308, 17393683, 23536835, 25823754,
18899974, 17787259 24719736, 20331945, 19490948, 20074391, 15861775, 16399083, 25947799
18018515, 22683212, 18260550, 21051858, 17080436, 16613964, 17036973 16579084, 24433711,
18384537, 27870645, 18280813, 20296213, 16901385 15979965, 23330124, 18441944, 16450169,
27534509, 9756271, 17892268 11733603, 16285691, 17587063, 21343775, 18180390, 26474853,
16538760 18193833, 21387964, 21051833, 17238511, 19777862, 17824637, 23065323 17903598,
16571443, 18306996, 19578350, 14852021, 17853456, 18674047 12364061, 24411921, 19207117,
22195448

Version 11.2.0.4.v16

Version 11.2.0.4.v16 adds support for the following:

- Patch 27338049: DATABASE PATCH SET UPDATE 11.2.0.4.180417
- Patch 27475598: OJVM PATCH SET UPDATE 11.2.0.4.180417
- Patch 27015449: RDBMS - PROACTIVE DSTV31 UPDATE - TZDATA2017C
- Patch 27015468: PROACTIVE DSTV31 UPDATE - TZDATA2017C - NEED OJVM FIX
- Patch 27216420: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 27659043: MES 405 BUNDLE ON TOP OF RDBMS 11.2.0.4.180116 PSU
- Patch 19692824: DBCONTROL is not coming up on OEL 7
- Adds support for the DBMS_ADVANCED_REWRITE package
- Fixed a bug where DBA_LOCKS and associated views available in new DB instances of 11.2.0.4.v15 were not created in upgrades to 11.2.0.4.v15. Views are now created in new and upgraded DB instances of 11.2.0.4.v16 and later.

Oracle patch 27338049, released April 2018

Bugs fixed: 21174504, 17184721, 21538558, 16091637, 18092127, 17381384, 15979965 20671094,
16731148, 16314254, 13837378, 18441944, 17835048, 13558557 17008068, 17201159, 25427662,
17853498, 20717359, 17246576, 18356166 18681862, 18440047, 20569094, 20031873, 16875449,
20387265, 19788842 17296856, 21330264, 14010183, 17648596, 17551063, 17025461, 24719736
17267114, 22507210, 17912217, 17889583, 18202441, 17040764, 17478145 16524926, 25655390,
19358317, 22148226, 18747196, 26544823, 18641419 17036973, 18948177, 17811789, 16542886,
14285317, 18009564, 16618694 8322815, 16832076, 18247991, 16692232, 22507234, 17570240,

13871092 24624166, 17848897, 17441661, 14034426, 17465741, 16596890, 17437634 21343897, 20506706, 21453153, 18339044, 22321741, 21795111, 17951233 18430495, 21787056, 22380919, 19469538, 20506715, 17811429, 19721304 17903598, 18230522, 19554106, 19458377, 21281607, 17612828, 6599380 22092979, 22321756, 17040527, 17811438, 18641461, 14657740, 13364795 21387964, 19490948, 22351572, 17346671, 17588480, 18235390, 26474853 18849970, 17889549, 19309466, 16472716, 20596234, 18331850, 18641451 17344412, 21179898, 19461270, 17546761, 24842886, 14521849, 18203835 18203838, 18964939, 18203837, 17313525, 22195457, 18139690, 16837842 22296366, 14106803, 17842825, 21352646, 22657942, 16360112, 20657441 22195441, 17389192, 26198926, 14565184, 17205719, 18440095, 14764829 22195448, 14354737, 13944971, 16571443, 21868720, 17186905, 17080436 18673342, 22905130, 17027426, 27374796, 19972569, 19972568, 20144308 19972566, 17282229, 19972564, 16870214, 21629064, 19615136, 21354456 17390431, 18762750, 23007241, 16613964, 17957017, 18098207, 18471685 19730508, 21538485, 18264060, 17323222, 17754782, 17600719, 18317531 17852463, 17596908, 17655634, 16228604, 27053456, 20074391, 19972570 18090142, 18996843, 19854503, 16042673, 17835627, 20334344, 17393683 20861693, 18000422, 17551709, 26575788, 23315889, 20506699, 19006849 18277454, 18456514, 19174430, 17258090, 17174582, 25654936, 17242746 16399083, 17824637, 21132297, 22465352, 17762296, 22168163, 17397545 16450169, 12364061, 20067212, 18856999, 19211724, 19463893, 19463897 21343775, 17853456, 18673304, 20004021, 26030218, 21668627, 16194160 17477958, 16538760, 12982566, 24570598, 20828947, 18259031, 20296213 18293054, 17610798, 19699191, 23065323, 17311728, 18135678, 18774543 23294548, 16785708, 10136473, 24560906, 22551446, 19777862, 17786518 18315328, 18334586, 12747740, 18096714, 19032867, 21641760, 18899974 17390160, 17232014, 20598042, 18673325, 16422541, 18155762, 14015842 19827973, 22683225, 17726838, 18554871, 23177648, 18051556, 20803583 21972320, 15990359, 17922254, 18282562, 16855292, 16668584, 21343838 20299015, 17446237, 18093615, 18043064, 23713236, 17694209, 17288409 20475845, 17274537, 13955826, 16934803, 17634921, 17501491, 16315398 22683212, 17006183, 13829543, 18191164, 17655240, 26746894, 22809871 18384391, 19393542, 21538567, 16198143, 21847223, 25823754, 17892268 20142975, 19584068, 17165204, 25165496, 18604493, 21756699, 18508861 16901385, 18554763, 21532755, 18189036, 17443671, 17385178, 14829250 17936109, 20925795, 20509482, 17478514, 27441326, 16850630, 13951456 16595641, 14054676, 15861775, 21142837, 16912439, 17299889, 17297939 23003979, 18619917, 16833527, 17798953, 17816865, 18607546, 17571306 21286665, 17341326, 26910644, 17851160, 20558005, 17586955, 19049453 21051840, 17587063, 16956380, 18328509, 25423453, 14133975, 18061914 18522509, 21051833, 18765602, 20860659, 20324049, 18199537, 17332800 13609098, 22502493, 18384537, 14338435, 17945983, 16392068, 21067387 17752995, 21051862, 16863422, 25505382, 17237521, 18244962, 19544839 24433711, 24717859, 17156148, 18973907, 23026585, 17877323, 17449815 18180390, 17088068, 17037130, 20004087, 21422580, 19466309, 11733603 25505371, 21051858, 18084625, 18674024, 21051852, 18091059, 25369547 16306373, 18306996, 18193833, 19915271, 17787259, 20513399, 20631274 25879656, 16344544, 14692762, 18614015, 17346091, 18228645, 17721717 18436307, 21756677, 19888853, 11883252, 17891943, 19475971, 22353199 16384983, 19121551, 12816846, 17982555, 17761775, 22243719, 17265217 25505394, 17071721, 16721594, 21756661, 18262334, 17891946, 15913355 17672719, 17602269, 17239687, 17042658, 17238511, 17811456, 17284817 17752121, 20879889, 21380789, 17394950, 17011832, 16579084, 22195465 14602788, 18325460, 24476265, 26569225, 24476274, 12611721, 16903536 17006570, 19689979, 16043574, 18783224, 24662775, 16494615, 21526048 17392698, 19197175, 16069901, 17811447, 17308789, 22195477, 24835538 17865671, 17343514, 19013183, 17325413, 18316692, 16180763, 17348614 14368995, 21983325, 17393915, 16285691, 19211433, 20331945, 17883081 17705023, 24316947, 17614227, 19578350, 22195485, 14084247, 13645875 16777840, 19727057, 14852021, 18744139, 18674047, 17716305, 19285025 18482502, 17622427, 19289642, 22195492, 25947799, 14458214, 20869721 21172913, 17767676, 18723434, 25505407, 17786278, 19258504, 17082983 21351877, 17365043, 13498382, 18331812, 16065166, 25489607, 16685417 18031668, 22893153, 16943711, 19272701, 21517440, 25897615, 17649265 13866822, 18094246, 24528741, 17783588, 14245531, 17082359, 18280813 20448824, 23330119, 16268425, 19487147, 25600421, 18018515, 17302277 17215560, 24411921, 19271443, 25764020, 17016369, 20777150, 23330124 16756406, 20441797, 19769489, 17545847, 25093656, 18260550, 13853126 17227277, 23536835, 25957038, 24652769, 19207117, 9756271, 18868646 17614134, 26667023, 17546973, 18704244, 19680952, 26667015, 17050888 18828868, 18273830, 17360606, 24563422, 16992075, 17375354,

12905058 18362222, 21429602, 27086138, 17571039, 17468141, 18436647, 17235750 21168487,
16220077, 16929165

Version 11.2.0.4.v15

Version 11.2.0.4.v15 adds support for the following:

- Patch 26925576: DATABASE PATCH SET UPDATE 11.2.0.4.180116
- Patch 26925532: OJVM PATCH SET UPDATE 11.2.0.4.180116
- Patch 27015449: RDBMS - PROACTIVE DSTV31 UPDATE - TZDATA2017C
- Patch 27015468: PROACTIVE DSTV31 UPDATE - TZDATA2017C - NEED OJVM FIX
- Patch 27216420: Oracle GoldenGate – Oracle RDBMS Server Recommended Patches
- Patch 27244661: MES 405 BUNDLE ON TOP OF RDBMS 11.2.0.4.180116 PSU
- Patch 19692824: DBCONTROL is not coming up on OEL 7
- Adds support for `DBA_LOCKS` and associated views

Oracle patch 26925576, released January 2018

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 17205719, 18607546, 25654936 17816865, 20506699, 24835538, 25957038, 23330119, 17922254, 17754782 13364795, 16934803, 17311728, 20387265, 17284817, 17441661, 20671094 24560906, 16992075, 17446237, 14015842, 19972569, 21756677, 17375354 21538558, 20925795, 17449815, 26575788, 19463897, 13866822, 17235750 17982555, 17478514, 18317531, 14338435, 18235390, 20803583, 19461270 19475971, 13944971, 20142975, 17811789, 16929165, 18704244, 24662775 20506706, 21422580, 17546973, 20334344, 14054676, 25489607, 17088068 17346091, 18264060, 17343514, 21538567, 19680952, 18471685, 19211724 21132297, 13951456, 16315398, 21847223, 18744139, 16850630, 23177648 19049453, 18090142, 18673304, 17883081, 19915271, 18641419, 18262334 25600421, 17006183, 16065166, 18277454, 16833527, 10136473, 18051556 17865671, 18554871, 17852463, 17853498, 18334586, 20879889, 17551709 17588480, 19827973, 17344412, 17842825, 18828868, 20509482, 17025461 13609098, 11883252, 17239687, 23007241, 17602269, 19197175, 18316692 22195457, 17313525, 12611721, 21174504, 19544839, 18964939, 17600719 26667015, 18191164, 17571306, 19393542, 20777150, 18482502, 19466309 22243719, 17165204, 17040527, 18098207, 16785708, 17465741, 16180763 17174582, 12982566, 16777840, 19463893, 22195465, 16875449, 22148226 12816846, 17237521, 6599380, 19358317, 17811438, 25505394, 17811447 21983325, 17945983, 18762750, 16912439, 17184721, 20598042, 18061914 21380789, 17282229, 18948177, 18331850, 21142837, 18202441, 17082359 18723434, 21972320, 21532755, 19554106, 25505371, 14034426, 18339044 19458377, 17752995, 20448824, 17891943, 17767676, 17258090, 16668584 18384391, 17040764, 17381384, 15913355, 18356166, 14084247, 20596234 21641760, 20506715, 13853126, 21756661, 18203837, 14245531, 16043574 21756699, 22195441, 17848897, 17877323, 21453153, 19272701, 20569094 17468141, 17786518, 20861693, 17912217, 17037130, 16956380, 18155762 17478145, 17394950, 18641461, 18189036, 18619917, 17027426, 21352646 16268425, 24476274, 22195492, 19584068, 26544823, 18436307, 22507210 17265217, 13498382, 17634921, 19469538, 21526048, 19258504, 23003979 19174430, 18043064, 20004087, 17443671, 22195485, 18000422, 20004021 22321756, 17571039, 25897615, 27053456, 21067387, 16832076, 22905130 16344544, 21429602, 18009564, 14354737, 21286665, 18135678, 14521849 18614015, 20441797, 18362222, 25655390, 16472716, 17835048, 17050888 17936109, 14010183, 17325413, 18747196, 17761775, 16721594, 17082983 20067212, 21179898, 17302277, 18084625, 20717359, 24624166, 15990359 26746894, 24842886, 18203835, 23026585, 17297939, 17811456, 16731148 22380919, 21168487, 14133975, 13829543, 17215560, 17694209, 17385178 18091059, 8322815, 18259031, 25165496, 19689979, 17586955, 17201159 17655634, 18331812, 19730508, 18868646, 17648596, 16220077, 16069901 17393915, 17348614, 17957017, 17274537, 18096714, 17308789, 18436647 14285317, 19289642, 14764829, 17622427, 18328509, 16943711, 22195477 22502493, 14368995, 17346671, 18996843, 17783588, 21343838, 16618694 17672719, 18856999, 18783224, 17851160, 17546761, 22168163, 17798953 18273830, 22092979, 16596890, 19972566, 20828947, 13871092, 26667023 17726838, 16384983, 22296366, 17360606, 13645875, 22321741, 16542886 25879656, 18199537, 21787056,

17889549, 21172913, 14565184, 20475845 17071721, 21281607, 17610798, 20299015, 21343897, 22893153, 20657441 17397545, 18230522, 16360112, 19769489, 12905058, 18641451, 12747740 18430495, 25423453, 17016369, 17042658, 14602788, 17551063, 19972568 21517440, 19788842, 18508861, 14657740, 17332800, 13837378, 17186905 19972564, 19699191, 18315328, 17437634, 24570598, 22353199, 18093615 19006849, 19013183, 17296856, 18674024, 26569225, 17232014, 16855292 21051840, 14692762, 17762296, 17705023, 23294548, 22507234, 19121551 20324049, 21330264, 26198926, 19854503, 23315889, 26910644, 26030218 21868720, 19309466, 25764020, 18681862, 17365043, 20031873, 20558005 18554763, 17390160, 24717859, 21795111, 18456514, 16306373, 13955826 18139690, 17501491, 17752121, 21668627, 17299889, 23713236, 24652769 17889583, 18673325, 22551446, 19721304, 18293054, 17242746, 19211433 19888853, 17951233, 18094246, 17649265, 19615136, 17011832, 17477958 16870214, 18522509, 20631274, 16091637, 17323222, 16595641, 16524926 18228645, 18282562, 17596908, 18031668, 17156148, 16494615, 22683225 20869721, 17545847, 25093656, 17655240, 24528741, 17614134, 25427662 13558557, 22465352, 17341326, 17891946, 17716305, 22657942, 16392068 18440095, 19271443, 21351877, 20513399, 18092127, 17614227, 18440047 18849970, 16903536, 14106803, 18973907, 18673342, 22809871, 17389192 19032867, 25505382, 17612828, 17006570, 16194160, 25369547, 16685417 25505407, 17721717, 21354456, 17390431, 17570240, 16863422, 18325460 17008068, 19727057, 16422541, 19972570, 17267114, 18244962, 21538485 18203838, 18765602, 16198143, 17246576, 14829250, 17835627, 20860659 21629064, 18247991, 14458214, 21051862, 17786278, 16692232, 17227277 24476265, 16042673, 16314254, 19285025, 16228604, 16756406, 16837842 20144308, 17393683, 23536835, 25823754, 18899974, 17787259, 24719736 20331945, 19490948, 20074391, 15861775, 16399083, 25947799, 18018515 22683212, 21051858, 18260550, 17080436, 16613964, 17036973, 16579084 24433711, 18384537, 18280813, 20296213, 16901385, 15979965, 23330124 18441944, 16450169, 9756271, 17892268, 11733603, 16285691, 17587063 21343775, 18180390, 26474853, 16538760, 18193833, 21387964, 21051833 17238511, 19777862, 23065323, 17824637, 16571443, 17903598, 18306996 19578350, 14852021, 17853456, 18674047, 12364061, 19207117, 24411921, 22195448

Version 11.2.0.4.v14

Version 11.2.0.4.v14 adds support for the following:

- Oracle October 2017 PSU, a combination of database PSU (patch 26392168) + OJVM component PSU (patch 26635834)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 26950781)
- RSA Micro-Edition Suite Bundle (patch 26963526)
- Timezone file DSTv30 (patch 25881255, OJVM patch 25881271)

Oracle patch 26392168, released October 2017

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 17205719, 18607546, 25654936 20506699, 17816865, 25957038, 23330119, 17922254, 17754782, 13364795 16934803, 17311728, 20387265, 17284817, 17441661, 24560906, 16992075 17446237, 14015842, 19972569, 21756677, 17375354, 21538558, 20925795 17449815, 26575788, 19463897, 13866822, 17235750, 17982555, 17478514 18317531, 14338435, 18235390, 20803583, 19461270, 13944971, 20142975 17811789, 16929165, 18704244, 24662775, 20506706, 17546973, 20334344 25489607, 14054676, 17088068, 17346091, 18264060, 17343514, 21538567 19680952, 18471685, 19211724, 21132297, 13951456, 21847223, 16315398 18744139, 16850630, 23177648, 19049453, 18673304, 17883081, 19915271 18641419, 18262334, 25600421, 17006183, 16065166, 18277454, 16833527 10136473, 18051556, 17865671, 17852463, 18554871, 17853498, 18334586 20879889, 17551709, 17588480, 19827973, 17344412, 17842825, 18828868 20509482, 17025461, 11883252, 13609098, 17239687, 17602269, 19197175 18316692, 22195457, 17313525, 12611721, 19544839, 18964939, 26667015 17600719, 18191164, 19393542, 17571306, 20777150, 18482502, 19466309 22243719, 17040527, 171165204, 18098207, 16785708, 17465741, 16180763 17174582, 12982566, 16777840, 19463893, 22195465, 16875449, 22148226 12816846, 17237521, 6599380, 19358317, 17811438, 25505394, 17811447 17945983, 21983325, 18762750, 16912439, 17184721, 18061914, 17282229 18331850, 18202441, 17082359,

18723434, 21532755, 21972320, 19554106 25505371, 14034426, 18339044, 19458377, 17752995, 20448824, 17891943 17258090, 17767676, 16668584, 18384391, 17040764, 17381384, 15913355 18356166, 14084247, 20596234, 20506715, 21756661, 13853126, 18203837 14245531, 16043574, 21756699, 22195441, 17848897, 17877323, 19272701 21453153, 20569094, 17468141, 20861693, 17786518, 17912217, 17037130 16956380, 18155762, 17478145, 17394950, 18641461, 18189036, 18619917 17027426, 21352646, 16268425, 24476274, 22195492, 19584068, 26544823 18436307, 22507210, 17265217, 17634921, 13498382, 19469538, 21526048 19258504, 18043064, 20004087, 17443671, 22195485, 18000422, 20004021 22321756, 17571039, 21067387, 16832076, 22905130, 16344544, 21429602 18009564, 14354737, 21286665, 18135678, 14521849, 18614015, 20441797 18362222, 25655390, 16472716, 17835048, 17050888, 17936109, 14010183 17325413, 18747196, 17761775, 16721594, 17082983, 20067212, 21179898 17302277, 18084625, 24624166, 15990359, 26746894, 24842886, 23026585 18203835, 17297939, 17811456, 16731148, 22380919, 21168487, 14133975 13829543, 17215560, 17694209, 17385178, 18091059, 8322815, 18259031 19689979, 17586955, 17201159, 17655634, 18331812, 19730508, 18868646 17648596, 16220077, 16069901, 17348614, 17393915, 17957017, 17274537 18096714, 17308789, 18436647, 14285317, 19289642, 14764829, 17622427 18328509, 16943711, 22195477, 14368995, 22502493, 17346671, 18996843 17783588, 21343838, 16618694, 17672719, 18856999, 18783224, 17851160 17546761, 22168163, 17798953, 18273830, 22092979, 16596890, 19972566 20828947, 13871092, 26667023, 17726838, 16384983, 22296366, 17360606 22321741, 13645875, 25879656, 18199537, 16542886, 21787056, 17889549 14565184, 20475845, 21281607, 17071721, 17610798, 20299015, 21343897 22893153, 20657441, 17397545, 18230522, 16360112, 19769489, 12905058 18641451, 12747740, 18430495, 25423453, 17016369, 17042658, 14602788 17551063, 19972568, 21517440, 19788842, 18508861, 14657740, 17332800 13837378, 17186905, 19972564, 19699191, 18315328, 17437634, 22353199 18093615, 19006849, 19013183, 17296856, 18674024, 17232014, 16855292 17762296, 14692762, 21051840, 17705023, 23294548, 22507234, 19121551 21330264, 26198926, 19854503, 23315889, 26030218, 21868720, 19309466 18681862, 17365043, 20558005, 18554763, 17390160, 18456514, 16306373 13955826, 18139690, 17501491, 17752121, 21668627, 17299889, 23713236 24652769, 17889583, 18673325, 22551446, 19721304, 18293054, 17242746 19211433, 19888853, 17951233, 18094246, 17649265, 19615136, 17011832 16870214, 17477958, 18522509, 20631274, 16091637, 17323222, 16595641 16524926, 18228645, 18282562, 17596908, 18031668, 17156148, 16494615 22683225, 20869721, 17545847, 25093656, 17655240, 24528741, 17614134 25427662, 13558557, 17341326, 17891946, 17716305, 22657942, 18440095 16392068, 19271443, 21351877, 18092127, 17614227, 18440047, 18849970 16903536, 14106803, 18973907, 18673342, 17389192, 25505382, 19032867 17612828, 16194160, 17006570, 25369547, 25505407, 16685417, 17721717 17390431, 17570240, 16863422, 18325460, 17008068, 19727057, 16422541 19972570, 17267114, 18244962, 21538485, 18203838, 18765602, 16198143 17246576, 14829250, 17835627, 18247991, 14458214, 21051862, 17786278 16692232, 17227277, 24476265, 16042673, 16314254, 19285025, 16228604 16837842, 20144308, 17393683, 23536835, 25823754, 18899974, 17787259 24719736, 20331945, 19490948, 20074391, 15861775, 16399083, 25947799 18018515, 22683212, 21051858, 18260550, 17080436, 16613964, 17036973 16579084, 24433711, 18384537, 18280813, 20296213, 16901385, 15979965 23330124, 18441944, 16450169, 9756271, 17892268, 11733603, 16285691 17587063, 21343775, 26474853, 18180390, 16538760, 18193833, 21387964 21051833, 17238511, 19777862, 23065323, 17824637, 17903598, 16571443 18306996, 19578350, 14852021, 17853456, 18674047, 12364061, 24411921 19207117, 22195448

Version 11.2.0.4.v13

Version 11.2.0.4.v13 adds support for the following:

- Oracle July 2017 PSU, a combination of database PSU (patch 26609445) + OJVM component PSU (patch 26027154)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 26554712)
- RSA Micro-Edition Suite Bundle (patch 26770426)
- Timezone file DSTv30 (patch 25881255, OJVM patch 25881271)
- Adds support for [Validating DB Instance Files \(p. 859\)](#) with the RMAN logical validation utility
- Adds support for [Setting the Default Edition for a DB Instance \(p. 859\)](#)

Oracle patch 26609445, released July 2017

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 17205719, 18607546, 20506699 17816865, 25957038, 23330119, 17922254, 17754782, 13364795, 16934803 17311728, 20387265, 17284817, 17441661, 24560906, 16992075, 17446237 14015842, 19972569, 21756677, 17375354, 21538558, 20925795, 17449815 26575788, 19463897, 13866822, 17235750, 17982555, 17478514, 18317531 14338435, 18235390, 20803583, 19461270, 13944971, 20142975, 17811789 16929165, 18704244, 20506706, 17546973, 20334344, 14054676, 17088068 17346091, 18264060, 17343514, 21538567, 19680952, 18471685, 19211724 13951456, 21847223, 16315398, 18744139, 16850630, 23177648, 19049453 18673304, 17883081, 19915271, 18641419, 18262334, 25600421, 17006183 16065166, 18277454, 16833527, 10136473, 18051556, 17865671, 17852463 18554871, 17853498, 18334586, 20879889, 17551709, 17588480, 19827973 17344412, 17842825, 18828868, 20509482, 17025461, 11883252, 13609098 17239687, 17602269, 19197175, 18316692, 22195457, 17313525, 12611721 19544839, 18964939, 17600719, 18191164, 19393542, 17571306, 20777150 18482502, 19466309, 22243719, 17040527, 17165204, 18098207, 16785708 17465741, 16180763, 17174582, 12982566, 16777840, 19463893, 22195465 16875449, 22148226, 12816846, 17237521, 6599380, 19358317, 17811438 25505394, 17811447, 17945983, 21983325, 18762750, 16912439, 17184721 18061914, 17282229, 18331850, 18202441, 17082359, 18723434, 21972320 19554106, 25505371, 14034426, 18339044, 19458377, 17752995, 20448824 17891943, 17258090, 17767676, 16668584, 18384391, 17040764, 17381384 15913355, 18356166, 14084247, 20596234, 20506715, 21756661, 13853126 18203837, 14245531, 16043574, 21756699, 22195441, 17848897, 17877323 21453153, 17468141, 20861693, 17786518, 17912217, 17037130, 16956380 18155762, 17478145, 17394950, 18641461, 18189036, 18619917, 17027426 21352646, 16268425, 24476274, 22195492, 19584068, 26544823, 18436307 22507210, 17265217, 17634921, 13498382, 19469538, 21526048, 19258504 18043064, 20004087, 17443671, 22195485, 18000422, 20004021, 22321756 17571039, 21067387, 16832076, 22905130, 16344544, 18009564, 14354737 21286665, 18135678, 14521849, 18614015, 20441797, 18362222, 25655390 16472716, 17835048, 17050888, 17936109, 14010183, 17325413, 18747196 17761775, 16721594, 17082983, 20067212, 21179898, 17302277, 18084625 15990359, 24842886, 18203835, 17297939, 17811456, 16731148, 22380919 21168487, 14133975, 13829543, 17215560, 17694209, 17385178, 18091059 8322815, 18259031, 19689979, 17586955, 17201159, 17655634, 18331812 19730508, 18868646, 17648596, 16220077, 16069901, 17348614, 17393915 17957017, 17274537, 18096714, 17308789, 18436647, 14285317, 19289642 14764829, 17622427, 18328509, 16943711, 22195477, 14368995, 22502493 17346671, 18996843, 17783588, 21343838, 16618694, 17672719, 18856999 18783224, 17851160, 17546761, 22168163, 17798953, 18273830, 22092979 16596890, 19972566, 13871092, 17726838, 16384983, 22296366, 17360606 22321741, 13645875, 25879656, 18199537, 16542886, 21787056, 17889549 14565184, 17071721, 17610798, 20299015, 21343897, 22893153, 20657441 17397545, 18230522, 16360112, 19769489, 12905058, 18641451, 12747740 18430495, 25423453, 17016369, 17042658, 14602788, 17551063, 19972568 21517440, 19788842, 18508861, 14657740, 17332800, 13837378, 17186905 19972564, 19699191, 18315328, 17437634, 22353199, 18093615, 19006849 19013183, 17296856, 18674024, 17232014, 16855292, 17762296, 14692762 21051840, 17705023, 22507234, 19121551, 21330264, 19854503, 26030218 21868720, 19309466, 18681862, 17365043, 20558005, 18554763, 17390160 18456514, 16306373, 13955826, 18139690, 17501491, 17752121, 21668627 17299889, 17889583, 18673325, 19721304, 18293054, 17242746, 19888853 17951233, 18094246, 17649265, 19615136, 17011832, 16870214, 17477958 18522509, 20631274, 16091637, 17323222, 16595641, 16524926, 18228645 18282562, 17596908, 18031668, 17156148, 16494615, 22683225, 17545847 25093656, 17655240, 24528741, 17614134, 25427662, 13558557, 17341326 17891946, 17716305, 22657942, 18440095, 16392068, 19271443, 21351877 18092127, 17614227, 18440047, 16903536, 14106803, 18973907, 18673342 17389192, 25505382, 19032867, 17612828, 16194160, 17006570, 25369547 25505407, 16685417, 17721717, 17390431, 17570240, 16863422, 18325460 19727057, 16422541, 19972570, 17267114, 18244962, 21538485, 18203838 18765602, 16198143, 17246576, 14829250, 17835627, 18247991, 14458214 21051862, 16692232, 17786278, 17227277, 24476265, 16042673, 16314254 16228604, 16837842, 17393683, 23536835, 25823754, 18899974, 17787259 20331945, 20074391, 15861775, 16399083, 18018515, 22683212, 21051858 18260550, 17080436, 16613964, 17036973, 16579084, 24433711, 18384537 18280813, 20296213, 16901385, 15979965, 23330124, 18441944, 16450169 9756271, 17892268, 11733603, 16285691, 17587063, 21343775, 18180390 16538760, 18193833,

21387964, 21051833, 17238511, 19777862, 17824637 16571443, 18306996, 19578350, 14852021, 17853456, 18674047, 12364061 24411921, 19207117, 22195448

Version 11.2.0.4.v12

Version 11.2.0.4.v12 adds support for the following:

- Oracle patch 25440428, a combination of database PSU (patch 24732075) + OJVM component PSU (patch 25434033)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 25734992)
- MES Bundle (patch 24975421 for 11.2.0.4)
- Timezone file DSTv28 (patch 24701840)
- Adds support for the `DBMS_CHANGE_NOTIFICATION` package
- Adds support for `XSTREAM` packages and views (may require additional licensing)

Oracle patch 24732075, released April 2017

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 17205719, 18607546, 20506699 17816865, 17922254, 23330119, 17754782, 16934803, 13364795, 17311728 17284817, 17441661, 24560906, 16992075, 17446237, 14015842, 19972569 21756677, 17375354, 20925795, 21538558, 17449815, 19463897, 13866822 17235750, 17982555, 17478514, 18317531, 14338435, 18235390, 20803583 13944971, 20142975, 17811789, 16929165, 18704244, 20506706, 17546973 20334344, 14054676, 17088068, 17346091, 18264060, 17343514, 21538567 19680952, 18471685, 19211724, 13951456, 21847223, 16315398, 18744139 16850630, 23177648, 19049453, 18673304, 17883081, 19915271, 18641419 18262334, 17006183, 16065166, 18277454, 16833527, 10136473, 18051556 17865671, 17852463, 18554871, 17853498, 18334586, 17551709, 17588480 19827973, 17344412, 17842825, 18828868, 17025461, 11883252, 13609098 17239687, 17602269, 19197175, 18316692, 22195457, 17313525, 12611721 19544839, 18964939, 17600719, 18191164, 19393542, 17571306, 20777150 18482502, 19466309, 22243719, 17040527, 17165204, 18098207, 16785708 17465741, 17174582, 16180763, 12982566, 16777840, 19463893, 22195465 16875449, 12816846, 22148226, 17237521, 6599380, 19358317, 25505394 17811438, 17811447, 17945983, 21983325, 18762750, 16912439, 17184721 18061914, 17282229, 18331850, 18202441, 17082359, 18723434, 21972320 19554106, 25505371, 14034426, 18339044, 19458377, 17752995, 20448824 17891943, 17258090, 17767676, 16668584, 18384391, 17040764, 17381384 15913355, 18356166, 14084247, 20596234, 20506715, 21756661, 13853126 18203837, 14245531, 16043574, 21756699, 22195441, 17848897, 17877323 21453153, 17468141, 20861693, 17786518, 17912217, 17037130, 16956380 18155762, 17478145, 17394950, 18641461, 18189036, 18619917, 17027426 21352646, 16268425, 24476274, 22195492, 19584068, 18436307, 22507210 17265217, 17634921, 13498382, 21526048, 19258504, 20004087, 17443671 22195485, 18000422, 22321756, 20004021, 17571039, 21067387, 22905130 16344544, 18009564, 14354737, 21286665, 18135678, 18614015, 20441797 18362222, 17835048, 16472716, 17936109, 17050888, 14010183, 17325413 18747196, 17761775, 16721594, 17082983, 20067212, 21179898, 17302277 18084625, 15990359, 24842886, 18203835, 17297939, 17811456, 22380919 16731148, 21168487, 14133975, 13829543, 17215560, 17694209, 17385178 18091059, 8322815, 17586955, 17201159, 17655634, 18331812, 19730508 18868646, 17648596, 16220077, 16069901, 17348614, 17393915, 17274537 17957017, 18096714, 17308789, 18436647, 14285317, 19289642, 14764829 17622427, 18328509, 16943711, 22195477, 14368995, 22502493, 17346671 18996843, 17783588, 21343838, 16618694, 17672719, 18856999, 18783224 17851160, 17546761, 17798953, 18273830, 22092979, 16596890, 19972566 16384983, 17726838, 22296366, 17360606, 22321741, 13645875, 18199537 16542886, 21787056, 17889549, 14565184, 17071721, 17610798, 20299015 21343897, 22893153, 20657441, 17397545, 18230522, 16360112, 19769489 12905058, 18641451, 12747740, 18430495, 17016369, 17042658, 14602788 17551063, 19972568, 21517440, 18508861, 19788842, 14657740, 17332800 13837378, 19972564, 17186905, 18315328, 19699191, 17437634, 22353199 18093615, 19006849, 19013183, 17296856, 18674024, 17232014, 16855292 17762296, 14692762, 21051840, 17705023, 22507234, 19121551, 21330264 19854503, 21868720, 19309466, 18681862, 20558005, 18554763, 17390160 18456514, 16306373, 13955826, 18139690, 17501491,

17752121, 21668627 17299889, 17889583, 18673325, 19721304, 18293054, 17242746, 17951233, 18094246, 17649265, 19615136, 17011832, 16870214, 17477958, 18522509 20631274, 16091637, 17323222, 16595641, 16524926, 18228645, 18282562 17596908, 18031668, 17156148, 16494615, 22683225, 17545847, 25093656 17655240, 24528741, 17614134, 13558557, 17341326, 17891946, 17716305 22657942, 18440095, 16392068, 19271443, 21351877, 18092127, 17614227 18440047, 16903536, 14106803, 18973907, 18673342, 25505382, 19032867 17389192, 17612828, 16194160, 17006570, 25369547, 25505407, 17721717 17390431, 17570240, 16863422, 18325460, 19727057, 16422541, 19972570 17267114, 18244962, 21538485, 18765602, 18203838, 16198143, 17246576 14829250, 17835627, 18247991, 14458214, 21051862, 16692232, 17786278 17227277, 24476265, 16042673, 16314254, 16228604, 16837842, 17393683 23536835, 17787259, 20331945, 20074391, 15861775, 16399083, 18018515 22683212, 18260550, 21051858, 17080436, 16613964, 17036973, 16579084 24433711, 18384537, 18280813, 20296213, 16901385, 15979965, 23330124 18441944, 16450169, 9756271, 17892268, 11733603, 16285691, 17587063 21343775, 18180390, 16538760, 18193833, 21387964, 21051833, 17238511 17824637, 16571443, 18306996, 14852021, 17853456, 18674047, 12364061 24411921, 22195448

Version 11.2.0.4.v11

Version 11.2.0.4.v11 adds support for the following:

- Oracle patch 24918033, a combination of database PSU (patch 24006111) + OJVM component PSU (patch 24917954)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 24491261)
- MES Bundle (patch 24975421 for 11.2.0.4)

Oracle patch 24918033, released January 2017

Bugs fixed: 18933818, 19176885, 17201047, 25067795, 14774730, 19153980, 21911849 23727132, 18166577, 24448240, 17056813, 21811517, 19909862, 22675136 24534298, 19895326, 22253904, 17804361, 19231857, 17528315, 19058059 19554117, 19007266, 17285560, 22670385, 18458318, 19187988, 23265914 19006757, 19374518, 19223010, 25076732, 22118835, 19852360, 20408829 21047766, 21566944, 17288409, 21051852, 24316947, 17811429, 18607546, 17205719, 20506699 17816865, 17922254, 23330119, 17754782, 16934803, 13364795, 17311728 17441661, 17284817, 16992075, 17446237, 14015842, 19972569, 21756677 17375354, 20925795, 21538558, 17449815, 19463897, 13866822, 17235750 17982555, 17478514, 18317531, 14338435, 18235390, 20803583, 13944971 20142975, 17811789, 16929165, 18704244, 20506706, 17546973, 20334344 14054676, 17088068, 17346091, 18264060, 17343514, 21538567, 19680952 18471685, 19211724, 13951456, 21847223, 16315398, 18744139, 16850630 23177648, 19049453, 18673304, 17883081, 19915271, 18641419, 18262334 17006183, 16065166, 18277454, 16833527, 10136473, 18051556, 17865671 17852463, 18554871, 17853498, 18334586, 17551709, 17588480, 19827973 17344412, 17842825, 18828868, 17025461, 11883252, 13609098, 17239687 17602269, 19197175, 22195457, 18316692, 17313525, 12611721, 19544839 18964939, 17600719, 18191164, 19393542, 17571306, 20777150, 18482502 19466309, 22243719, 17040527, 17165204, 18098207, 16785708, 17465741 17174582, 16180763, 16777840, 12982566, 19463893, 22195465, 22148226 16875449, 12816846, 17237521, 6599380, 19358317, 17811438, 17811447 17945983, 21983325, 18762750, 16912439, 17184721, 18061914, 17282229 18331850, 18202441, 17082359, 18723434, 21972320, 19554106, 14034426 18339044, 19458377, 17752995, 20448824, 17891943, 17258090, 17767676 16668584, 18384391, 17040764, 17381384, 15913355, 18356166, 14084247 20596234, 20506715, 21756661, 13853126, 18203837, 14245531, 16043574 21756699, 22195441, 17848897, 17877323, 21453153, 17468141, 20861693 17786518, 17912217, 17037130, 16956380, 18155762, 17478145, 17394950 18641461, 18189036, 18619917, 17027426, 21352646, 16268425, 24476274 22195492, 19584068, 18436307, 22507210, 17265217, 17634921, 13498382 21526048, 19258504, 20004087, 17443671, 22195485, 18000422, 22321756 20004021, 17571039, 21067387, 16344544, 18009564, 14354737, 21286665 18135678, 18614015, 20441797, 18362222, 17835048, 16472716, 17936109 17050888, 17325413, 14010183, 18747196, 17761775, 16721594, 17082983 20067212, 21179898, 17302277, 18084625, 15990359, 18203835, 17297939 17811456, 22380919, 16731148, 21168487, 14133975, 13829543,

17215560 17694209, 17385178, 18091059, 8322815, 17586955, 17201159, 17655634 18331812, 19730508, 18868646, 17648596, 16220077, 16069901, 17348614 17393915, 17274537, 17957017, 18096714, 17308789, 18436647, 14285317 19289642, 14764829, 18328509, 17622427, 16943711, 22195477, 14368995 22502493, 17346671, 18996843, 17783588, 21343838, 16618694, 17672719 18856999, 18783224, 17851160, 17546761, 17798953, 18273830, 22092979 16596890, 19972566, 16384983, 17726838, 22296366, 17360606, 22321741 13645875, 18199537, 16542886, 21787056, 17889549, 14565184, 17071721 17610798, 20299015, 21343897, 22893153, 20657441, 17397545, 18230522 16360112, 19769489, 12905058, 18641451, 12747740, 18430495, 17016369 17042658, 14602788, 17551063, 19972568, 21517440, 18508861, 19788842 14657740, 17332800, 13837378, 19972564, 17186905, 18315328, 19699191 17437634, 22353199, 18093615, 19006849, 19013183, 17296856, 18674024 17232014, 16855292, 17762296, 14692762, 21051840, 17705023, 22507234 19121551, 21330264, 19854503, 21868720, 19309466, 18681862, 20558005 18554763, 17390160, 18456514, 16306373, 13955826, 18139690, 17501491 17752121, 21668627, 17299889, 17889583, 18673325, 19721304, 18293054 17242746, 17951233, 18094246, 17649265, 19615136, 17011832, 16870214 17477958, 18522509, 20631274, 16091637, 17323222, 16595641, 16524926 18228645, 18282562, 17596908, 18031668, 17156148, 16494615, 22683225 17545847, 17655240, 24528741, 17614134, 13558557, 17341326, 17891946 17716305, 22657942, 16392068, 19271443, 21351877, 18092127, 17614227 18440047, 16903536, 14106803, 18973907, 18673342, 19032867, 17389192 17612828, 16194160, 17006570, 17721717, 17390431, 17570240, 16863422 18325460, 19727057, 16422541, 19972570, 17267114, 18244962, 21538485 18765602, 18203838, 16198143, 17246576, 14829250, 17835627, 18247991 14458214, 21051862, 16692232, 17786278, 17227277, 24476265, 16042673 16314254, 16228604, 16837842, 17393683, 23536835, 17787259, 20331945 20074391, 15861775, 16399083, 18018515, 22683212, 18260550, 21051858 17080436, 16613964, 17036973, 16579084, 24433711, 18384537, 18280813 20296213, 16901385, 15979965, 23330124, 18441944, 16450169, 9756271 17892268, 11733603, 16285691, 17587063, 21343775, 18180390, 16538760 18193833, 21387964, 21051833, 17238511, 17824637, 16571443, 18306996 14852021, 17853456, 18674047, 12364061, 22195448

Version 11.2.0.4.v10

Version 11.2.0.4.v10 adds support for the following:

- Oracle patch 24436313, a combination of database PSU (patch 24006111) + OJVM component PSU (patch 24315821)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 24491261)
- MES Bundle (patch 24975421 for 11.2.0.4)

Baseline: Oracle Database Patch Set Update 11.2.0.4.161018 (patch 24006111, released October 2016)

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 18607546, 17205719, 20506699 17816865, 17922254, 23330119, 17754782, 16934803, 13364795, 17311728 17441661, 17284817, 16992075, 17446237, 14015842, 19972569, 21756677 17375354, 20925795, 21538558, 17449815, 19463897, 13866822, 17235750 17982555, 17478514, 18317531, 14338435, 18235390, 20803583, 13944971 20142975, 17811789, 16929165, 18704244, 20506706, 17546973, 20334344 14054676, 17088068, 17346091, 18264060, 17343514, 21538567, 19680952 18471685, 19211724, 13951456, 21847223, 16315398, 18744139, 16850630 23177648, 19049453, 18673304, 17883081, 19915271, 18641419, 18262334 17006183, 16065166, 18277454, 16833527, 10136473, 18051556, 17865671 17852463, 18554871, 17853498, 18334586, 17551709, 17588480, 19827973 17344412, 17842825, 18828868, 17025461, 11883252, 13609098, 17239687 17602269, 19197175, 22195457, 18316692, 17313525, 12611721, 19544839 18964939, 17600719, 18191164, 19393542, 17571306, 20777150, 18482502 19466309, 22243719, 17040527, 17165204, 18098207, 16785708, 17465741 17174582, 16180763, 16777840, 12982566, 19463893, 22195465, 22148226 16875449, 12816846, 17237521, 6599380, 19358317, 17811438, 17811447 17945983, 21983325, 18762750, 16912439, 17184721, 18061914, 17282229 18331850, 18202441, 17082359, 18723434, 21972320, 19554106, 14034426 18339044, 19458377, 17752995, 20448824, 17891943, 17258090, 17767676 16668584, 18384391, 17040764,

17381384, 15913355, 18356166, 14084247 20596234, 20506715, 21756661, 13853126, 18203837, 14245531, 16043574 21756699, 22195441, 17848897, 17877323, 21453153, 17468141, 20861693 17786518, 17912217, 17037130, 16956380, 18155762, 17478145, 17394950 18641461, 18189036, 18619917, 17027426, 21352646, 16268425, 24476274 22195492, 19584068, 18436307, 22507210, 17265217, 17634921, 13498382 21526048, 19258504, 20004087, 17443671, 22195485, 18000422, 22321756 20004021, 17571039, 21067387, 16344544, 18009564, 14354737, 21286665 18135678, 18614015, 20441797, 18362222, 17835048, 16472716, 17936109 17050888, 17325413, 14010183, 18747196, 17761775, 16721594, 17082983 20067212, 21179898, 17302277, 18084625, 15990359, 18203835, 17297939 17811456, 22380919, 16731148, 21168487, 14133975, 13829543, 17215560 17694209, 17385178, 18091059, 8322815, 17586955, 17201159, 17655634 18331812, 19730508, 18868646, 17648596, 16220077, 16069901, 17348614 17393915, 17274537, 17957017, 18096714, 17308789, 18436647, 14285317 19289642, 14764829, 18328509, 17622427, 16943711, 22195477, 14368995 22502493, 17346671, 18996843, 17783588, 21343838, 16618694, 17672719 18856999, 18783224, 17851160, 17546761, 17798953, 18273830, 22092979 16596890, 19972566, 16384983, 17726838, 22296366, 17360606, 22321741 13645875, 18199537, 16542886, 21787056, 17889549, 14565184, 17071721 17610798, 20299015, 21343897, 22893153, 20657441, 17397545, 18230522 16360112, 19769489, 12905058, 18641451, 12747740, 18430495, 17016369 17042658, 14602788, 17551063, 19972568, 21517440, 18508861, 19788842 14657740, 17332800, 13837378, 19972564, 17186905, 18315328, 19699191 17437634, 22353199, 18093615, 19006849, 19013183, 17296856, 18674024 17232014, 16855292, 17762296, 14692762, 21051840, 17705023, 22507234 19121551, 21330264, 19854503, 21868720, 19309466, 18681862, 20558005 18554763, 17390160, 18456514, 16306373, 13955826, 18139690, 17501491 17752121, 21668627, 17299889, 17889583, 18673325, 19721304, 18293054 17242746, 17951233, 18094246, 17649265, 19615136, 17011832, 16870214 17477958, 18522509, 20631274, 16091637, 17323222, 16595641, 16524926 18228645, 18282562, 17596908, 18031668, 17156148, 16494615, 22683225 17545847, 17655240, 24528741, 17614134, 13558557, 17341326, 17891946 17716305, 22657942, 16392068, 19271443, 21351877, 18092127, 17614227 18440047, 16903536, 14106803, 18973907, 18673342, 19032867, 17389192 17612828, 16194160, 17006570, 17721717, 17390431, 17570240, 16863422 18325460, 19727057, 16422541, 19972570, 17267114, 18244962, 21538485 18765602, 18203838, 16198143, 17246576, 14829250, 17835627, 18247991 14458214, 21051862, 16692232, 17786278, 17227277, 24476265, 16042673 16314254, 16228604, 16837842, 17393683, 23536835, 17787259, 20331945 20074391, 15861775, 16399083, 18018515, 22683212, 18260550, 21051858 17080436, 16613964, 17036973, 16579084, 24433711, 18384537, 18280813 20296213, 16901385, 15979965, 23330124, 18441944, 16450169, 9756271 17892268, 11733603, 16285691, 17587063, 21343775, 18180390, 16538760 18193833, 21387964, 21051833, 17238511, 17824637, 16571443, 18306996 14852021, 17853456, 18674047, 12364061, 22195448

Version 11.2.0.4.v9

Version 11.2.0.4.v9 adds support for the following:

- Oracle patch 23615392, a combination of database PSU (patch 23054359) + OJVM component PSU (patch 23177551)
- Timezone file DSTv26 (patch 22873635 for 11.2.0.4)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 24320398 for 11.2.0.4.160719)
- MES Bundle (patch 22695784 for 11.2.0.4)
- Added the ability to create custom password verify functions. For more information, see [Creating Custom Functions to Verify Passwords \(p. 850\)](#).
- Fixed a bug that prevented implicit recompilation of views owned by SYS

Baseline: Oracle Database Patch Set Update 11.2.0.4.160719 (patch 23054359, released July 2016)

Bugs fixed: 17288409, 21051852, 17811429, 18607546, 17205719, 20506699, 17816865 23330119, 17922254, 17754782, 16934803, 13364795, 17311728, 17441661 17284817, 16992075, 17446237,

14015842, 19972569, 21756677, 17375354 21538558, 20925795, 17449815, 19463897, 13866822, 17982555, 17235750 17478514, 18317531, 14338435, 18235390, 20803583, 13944971, 20142975 17811789, 16929165, 18704244, 20506706, 17546973, 20334344, 14054676 17088068, 17346091, 18264060, 17343514, 21538567, 19680952, 18471685 19211724, 13951456, 21847223, 16315398, 18744139, 16850630, 23177648 19049453, 18673304, 17883081, 19915271, 18641419, 18262334, 17006183 16065166, 18277454, 16833527, 10136473, 18051556, 17865671, 17852463 18554871, 17853498, 18334586, 17551709, 17588480, 19827973, 17344412 17842825, 18828868, 17025461, 11883252, 13609098, 17239687, 17602269 19197175, 22195457, 18316692, 17313525, 12611721, 19544839, 18964939 17600719, 18191164, 19393542, 17571306, 18482502, 20777150, 19466309 17040527, 17165204, 18098207, 16785708, 17465741, 17174582, 16180763 16777840, 12982566, 19463893, 22195465, 16875449, 12816846, 17237521 19358317, 17811438, 17811447, 17945983, 21983325, 18762750, 16912439 17184721, 18061914, 17282229, 18331850, 18202441, 17082359, 18723434 21972320, 19554106, 14034426, 18339044, 19458377, 17752995, 20448824 17891943, 17258090, 17767676, 16668584, 18384391, 17040764, 17381384 15913355, 18356166, 14084247, 20596234, 20506715, 21756661, 13853126 18203837, 14245531, 16043574, 21756699, 22195441, 17848897, 17877323 21453153, 17468141, 20861693, 17786518, 17912217, 17037130, 16956380 18155762, 17478145, 17394950, 18641461, 18189036, 18619917, 17027426 21352646, 16268425, 22195492, 19584068, 18436307, 22507210, 17265217 17634921, 13498382, 21526048, 19258504, 20004087, 17443671, 22195485 18000422, 22321756, 20004021, 17571039, 21067387, 16344544, 18009564 14354737, 21286665, 18135678, 18614015, 20441797, 18362222, 17835048 16472716, 17936109, 17050888, 17325413, 14010183, 18747196, 17761775 16721594, 17082983, 20067212, 21179898, 17302277, 18084625, 15990359 18203835, 17297939, 22380919, 17811456, 16731148, 21168487, 13829543 17215560, 14133975, 17694209, 17385178, 18091059, 8322815, 17586955 17201159, 17655634, 18331812, 19730508, 18868646, 17648596, 16220077 16069901, 17348614, 17393915, 17274537, 17957017, 18096714, 17308789 18436647, 14285317, 19289642, 14764829, 18328509, 17622427, 16943711 22195477, 14368995, 22502493, 17346671, 18996843, 17783588, 21343838 16618694, 17672719, 18856999, 18783224, 17851160, 17546761, 17798953 18273830, 22092979, 16596890, 19972566, 16384983, 17726838, 22296366 17360606, 22321741, 13645875, 18199537, 16542886, 21787056, 17889549 14565184, 17071721, 17610798, 20299015, 21343897, 22893153, 20657441 17397545, 18230522, 16360112, 19769489, 12905058, 18641451, 12747740 18430495, 17016369, 17042658, 14602788, 17551063, 19972568, 21517440 18508861, 19788842, 14657740, 17332800, 13837378, 19972564, 17186905 18315328, 19699191, 17437634, 22353199, 18093615, 19006849, 19013183 17296856, 18674024, 17232014, 16855292, 17762296, 14692762, 21051840 17705023, 22507234, 19121551, 21330264, 19854503, 21868720, 19309466 18681862, 18554763, 20558005, 17390160, 18456514, 16306373, 13955826 18139690, 17501491, 17752121, 21668627, 17299889, 17889583, 18673325 19721304, 18293054, 17242746, 17951233, 18094246, 17649265, 19615136 17011832, 16870214, 17477958, 18522509, 20631274, 16091637, 17323222 16595641, 16524926, 18228645, 18282562, 17596908, 18031668, 17156148 16494615, 22683225, 17545847, 17655240, 17614134, 13558557, 17341326 17891946, 17716305, 16392068, 19271443, 21351877, 18092127, 17614227 18440047, 16903536, 14106803, 18973907, 18673342, 19032867, 17389192 17612828, 16194160, 17006570, 17721717, 17390431, 17570240, 16863422 18325460, 19727057, 16422541, 19972570, 17267114, 18244962, 21538485 18765602, 18203838, 16198143, 17246576, 14829250, 17835627, 18247991 14458214, 21051862, 16692232, 17786278, 17227277, 16042673, 16314254 16228604, 16837842, 17393683, 23536835, 17787259, 20331945, 20074391 15861775, 16399083, 18018515, 22683212, 18260550, 21051858, 17080436 16613964, 17036973, 16579084, 18384537, 18280813, 20296213, 16901385 15979965, 2330124, 18441944, 16450169, 9756271, 17892268, 11733603 16285691, 17587063, 21343775, 16538760, 18180390, 18193833, 21387964 21051833, 17238511, 17824637, 16571443, 18306996, 14852021, 17853456 18674047, 12364061, 22195448

Version 11.2.0.4.v8

Version 11.2.0.4.v8 adds support for the following:

- Oracle PSU 11.2.0.4.160419 (22502456)
- Timezone file DSTv25 (patch 22037014)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 22576728)

- MES Bundle (patch 22695784 for 11.2.0.4)
- Adds the ability for the master user to grant privileges on SYS objects with the grant option using the RDSADMIN.RDSADMIN_UTIL.GRANT_SYS_OBJECT procedure
- Adds master user privileges to support most common schemas created by the Oracle Fusion Middleware Repository Creation Utility (RCU)

Baseline: Oracle Database Patch Set Update 11.2.0.4.160419 (patch 22502456, released April 2016)

Bugs fixed: 17288409, 21051852, 17811429, 18607546, 17205719, 20506699, 17816865 17922254, 17754782, 16934803, 13364795, 17311728, 17441661, 17284817 16992075, 17446237, 14015842, 19972569, 21756677, 21538558, 20925795 17449815, 17375354, 19463897, 13866822, 17982555, 17235750, 17478514 18317531, 14338435, 18235390, 20803583, 13944971, 20142975, 17811789 16929165, 18704244, 20506706, 17546973, 20334344, 14054676, 17088068 17346091, 18264060, 17343514, 21538567, 19680952, 18471685, 19211724 13951456, 21847223, 16315398, 18744139, 16850630, 19049453, 18673304 17883081, 19915271, 18641419, 18262334, 17006183, 16065166, 18277454 16833527, 10136473, 18051556, 17865671, 17852463, 18554871, 17853498 18334586, 17551709, 17588480, 19827973, 17344412, 17842825, 18828868 17025461, 11883252, 13609098, 17239687, 17602269, 19197175, 22195457 18316692, 17313525, 12611721, 19544839, 18964939, 17600719, 18191164 19393542, 17571306, 18482502, 20777150, 19466309, 17040527, 17165204 18098207, 16785708, 17465741, 17174582, 16180763, 16777840, 12982566 19463893, 22195465, 16875449, 12816846, 17237521, 19358317, 17811438 17811447, 21983325, 17945983, 18762750, 16912439, 17184721, 18061914 17282229, 18331850, 18202441, 17082359, 18723434, 21972320, 19554106 14034426, 18339044, 19458377, 17752995, 20448824, 17891943, 17258090 17767676, 16668584, 18384391, 17040764, 17381384, 15913355, 18356166 14084247, 20596234, 20506715, 21756661, 13853126, 18203837, 14245531 21756699, 16043574, 22195441, 17848897, 17877323, 21453153, 17468141 20861693, 17786518, 17912217, 17037130, 18155762, 16956380, 17478145 17394950, 18641461, 18189036, 18619917, 17027426, 21352646, 16268425 22195492, 19584068, 18436307, 17265217, 17634921, 13498382, 21526048 19258504, 20004087, 17443671, 22195485, 18000422, 20004021, 22321756 17571039, 21067387, 16344544, 18009564, 14354737, 21286665, 18135678 18614015, 20441797, 18362222, 17835048, 16472716, 17936109, 17050888 17325413, 14010183, 18747196, 17761775, 16721594, 17082983, 20067212 21179898, 17302277, 18084625, 15990359, 18203835, 17297939, 17811456 16731148, 21168487, 13829543, 17215560, 14133975, 17694209, 17385178 18091059, 8322815, 17586955, 17201159, 17655634, 18331812, 19730508 18868646, 17648596, 16220077, 16069901, 17348614, 17393915, 17274537 17957017, 18096714, 17308789, 18436647, 14285317, 19289642, 14764829 18328509, 17622427, 22195477, 16943711, 22502493, 14368995, 17346671 18996843, 17783588, 21343838, 16618694, 17672719, 18856999, 18783224 17851160, 17546761, 17798953, 18273830, 22092979, 16596890, 19972566 16384983, 17726838, 17360606, 22321741, 13645875, 18199537, 16542886 21787056, 17889549, 14565184, 17071721, 17610798, 20299015, 21343897 22893153, 20657441, 17397545, 18230522, 16360112, 19769489, 12905058 18641451, 12747740, 18430495, 17016369, 17042658, 14602788, 17551063 19972568, 21517440, 18508861, 19788842, 14657740, 17332800, 13837378 19972564, 17186905, 18315328, 19699191, 17437634, 22353199, 18093615 19006849, 19013183, 17296856, 18674024, 17232014, 16855292, 17762296 14692762, 21051840, 17705023, 19121551, 21330264, 19854503, 21868720 19309466, 18681862, 18554763, 20558005, 17390160, 18456514, 16306373 13955826, 18139690, 17501491, 17752121, 21668627, 17299889, 17889583 18673325, 19721304, 18293054, 17242746, 17951233, 17649265, 18094246 19615136, 17011832, 16870214, 17477958, 18522509, 20631274, 16091637 17323222, 16595641, 16524926, 18228645, 18282562, 17596908, 17156148 18031668, 16494615, 22683225, 17545847, 17655240, 17614134, 13558557 17341326, 17891946, 17716305, 16392068, 19271443, 21351877, 18092127 18440047, 17614227, 14106803, 16903536, 18973907, 18673342, 19032867 17389192, 17612828, 16194160, 17006570, 17721717, 17390431, 17570240 16863422, 18325460, 19727057, 16422541, 19972570, 17267114, 18244962 21538485, 18765602, 18203838, 16198143, 17246576, 14829250, 17835627 18247991, 14458214, 21051862, 16692232, 17786278, 17227277, 16042673 16314254, 16228604, 16837842, 17393683, 17787259, 20331945, 20074391 15861775, 16399083, 18018515, 22683212, 18260550, 21051858, 17036973

16613964, 17080436, 16579084, 18384537, 18280813, 20296213, 16901385 15979965, 18441944, 16450169, 9756271, 17892268, 11733603, 16285691 17587063, 21343775, 16538760, 18180390, 18193833, 21387964, 21051833 17238511, 17824637, 16571443, 18306996, 14852021, 18674047, 17853456 12364061, 22195448

Version 11.2.0.4.v7

Version 11.2.0.4.v7 adds support for the following:

- Oracle PSU 11.2.0.4.160119 (21948347)
- Timezone file DSTv25 - patch 22037014 for 11.2.0.4 and 12.1.0.2 (12.1.0.1 includes DSTv24, patch 20875898 (unchanged from 12.1.0.1.v3), as a backport of DSTv25 was unavailable at build time)
- Fixed an issue that prevented customers from creating more than 10 Directory objects in the database
- Fixed an issue that prevented customers from re-granting read privileges on the ADUMP and BDUMP Directory objects

Baseline: Oracle Database Patch Set Update 11.2.0.4.160119 (patch 21948347, released January 2016)

Bugs fixed: 17288409, 21051852, 18607546, 17205719, 17811429, 17816865, 20506699 17922254, 17754782, 16934803, 13364795, 17311728, 17441661, 17284817 16992075, 17446237, 14015842, 19972569, 17449815, 21538558, 20925795 17375354, 19463897, 17982555, 17235750, 13866822, 17478514, 18317531 18235390, 14338435, 20803583, 13944971, 20142975, 17811789, 16929165 18704244, 20506706, 17546973, 20334344, 14054676, 17088068, 18264060 17346091, 17343514, 21538567, 19680952, 18471685, 19211724, 13951456 21847223, 16315398, 18744139, 16850630, 19049453, 18673304, 17883081 19915271, 18641419, 18262334, 17006183, 16065166, 18277454, 16833527 10136473, 18051556, 17865671, 17852463, 18554871, 17853498, 18334586 17588480, 17551709, 19827973, 17842825, 17344412, 18828868, 17025461 11883252, 13609098, 17239687, 17602269, 19197175, 22195457, 18316692 17313525, 12611721, 19544839, 18964939, 17600719, 18191164, 19393542 17571306, 18482502, 20777150, 19466309, 17040527, 17165204, 18098207 16785708, 17174582, 16180763, 17465741, 16777840, 12982566, 19463893 22195465, 12816846, 16875449, 17237521, 19358317, 17811438, 17811447 17945983, 18762750, 17184721, 16912439, 18061914, 17282229, 18331850 18202441, 17082359, 18723434, 21972320, 19554106, 14034426, 18339044 19458377, 17752995, 20448824, 17891943, 17258090, 17767676, 16668584 18384391, 17040764, 17381384, 15913355, 18356166, 14084247, 20506715 13853126, 18203837, 14245531, 21756699, 16043574, 22195441, 17848897 17877323, 21453153, 17468141, 20861693, 17786518, 17912217, 17037130 18155762, 16956380, 17478145, 17394950, 18189036, 18641461, 18619917 17027426, 21352646, 16268425, 22195492, 19584068, 18436307, 17265217 17634921, 13498382, 21526048, 20004087, 22195485, 17443671, 18000422 22321756, 20004021, 17571039, 21067387, 16344544, 18009564, 14354737 18135678, 18614015, 20441797, 18362222, 17835048, 16472716, 17936109 17050888, 17325413, 14010183, 18747196, 17761775, 16721594, 17082983 20067212, 21179898, 17302277, 18084625, 15990359, 18203835, 17297939 17811456, 16731148, 21168487, 17215560, 13829543, 14133975, 17694209 18091059, 17385178, 8322815, 17586955, 17201159, 17655634, 18331812 19730508, 18868646, 17648596, 16220077, 16069901, 17348614, 17393915 17274537, 17957017, 18096714, 17308789, 18436647, 14285317, 19289642 14764829, 18328509, 17622427, 22195477, 16943711, 14368995, 17346671 18996843, 17783588, 21343838, 16618694, 17672719, 18856999, 18783224 17851160, 17546761, 17798953, 18273830, 22092979, 19972566, 16384983 17726838, 17360606, 22321741, 13645875, 18199537, 16542886, 21787056 17889549, 14565184, 17071721, 17610798, 20299015, 21343897, 20657441 17397545, 18230522, 16360112, 19769489, 12905058, 18641451, 12747740 18430495, 17042658, 17016369, 14602788, 17551063, 19972568, 21517440 18508861, 19788842, 14657740, 17332800, 13837378, 19972564, 17186905 18315328, 19699191, 17437634, 19006849, 19013183, 17296856, 18674024 17232014, 16855292, 21051840, 14692762, 17762296, 17705023, 19121551 21330264, 19854503, 19309466, 18681862, 18554763, 20558005, 17390160 18456514, 16306373, 13955826, 18139690, 17501491, 21668627, 17299889 17752121, 17889583, 18673325, 18293054, 17242746, 17951233, 17649265 18094246,

19615136, 17011832, 16870214, 17477958, 18522509, 20631274 16091637, 17323222, 16595641, 16524926, 18228645, 18282562, 17596908 17156148, 18031668, 16494615, 17545847, 17655240, 17614134, 13558557 17341326, 17891946, 17716305, 16392068, 19271443, 21351877, 18092127 18440047, 17614227, 14106803, 16903536, 18973907, 18673342, 19032867 17389192, 17612828, 16194160, 17006570, 17721717, 17570240, 17390431 16863422, 18325460, 19727057, 16422541, 19972570, 17267114, 18244962 21538485, 18765602, 18203838, 16198143, 17246576, 14829250, 17835627 18247991, 14458214, 21051862, 16692232, 17786278, 17227277, 16042673 16314254, 16228604, 16837842, 17393683, 17787259, 20331945, 20074391 15861775, 16399083, 18018515, 21051858, 18260550, 17036973, 16613964 17080436, 16579084, 18384537, 18280813, 20296213, 16901385, 15979965 18441944, 16450169, 9756271, 17892268, 11733603, 16285691, 17587063 21343775, 16538760, 18180390, 18193833, 21051833, 17238511, 17824637 16571443, 18306996, 14852021, 18674047, 17853456, 12364061, 22195448

Version 11.2.0.4.v6

Version 11.2.0.4.v6 adds support for the following:

- Enable SSL encryption for Standard Edition and Standard Edition One

Version 11.2.0.4.v5

Version 11.2.0.4.v5 adds support for the following:

- Oracle PSU 11.2.0.4.8 (21352635)
- Includes the Daylight Saving Time Patch, patch 20875898: DST-24, that came out after the April 2015 PSU.

Baseline: Oracle Database Patch Set Update 11.2.0.4.8 (patch 21352635, released October 2015)

Bugs fixed: 17288409, 21051852, 18607546, 17205719, 17811429, 17816865, 20506699 17922254, 17754782, 16934803, 13364795, 17311728, 17441661, 17284817 16992075, 17446237, 14015842, 19972569, 21538558, 20925795, 17449815 17375354, 19463897, 17982555, 17235750, 13866822, 18317531, 17478514 18235390, 14338435, 20803583, 13944971, 20142975, 17811789, 16929165 18704244, 20506706, 17546973, 20334344, 14054676, 17088068, 18264060 17346091, 17343514, 21538567, 19680952, 18471685, 19211724, 13951456 16315398, 18744139, 16850630, 19049453, 18673304, 17883081, 19915271 18641419, 18262334, 17006183, 16065166, 18277454, 16833527, 10136473 18051556, 17865671, 17852463, 18554871, 17853498, 18334586, 17588480 17551709, 19827973, 17842825, 17344412, 18828868, 17025461, 11883252 13609098, 17239687, 17602269, 19197175, 18316692, 17313525, 12611721 19544839, 18964939, 17600719, 18191164, 19393542, 17571306, 18482502 20777150, 19466309, 17040527, 17165204, 18098207, 16785708, 17174582 16180763, 17465741, 16777840, 12982566, 19463893, 12816846, 16875449 17237521, 19358317, 17811438, 17811447, 17945983, 18762750, 17184721 16912439, 18061914, 17282229, 18331850, 18202441, 17082359, 18723434 19554106, 14034426, 18339044, 19458377, 17752995, 20448824, 17891943 17258090, 17767676, 16668584, 18384391, 17040764, 17381384, 15913355 18356166, 14084247, 20506715, 13853126, 18203837, 14245531, 16043574 17848897, 17877323, 17468141, 17786518, 17912217, 17037130, 18155762 16956380, 17478145, 17394950, 18189036, 18641461, 18619917, 17027426 21352646, 16268425, 19584068, 18436307, 17265217, 17634921, 13498382 20004087, 17443671, 18000422, 20004021, 17571039, 21067387, 16344544 18009564, 14354737, 18135678, 18614015, 20441797, 18362222, 17835048 16472716, 17936109, 17050888, 17325413, 14010183, 18747196, 17761775 16721594, 17082983, 20067212, 21179898, 17302277, 18084625, 15990359 18203835, 17297939, 17811456, 16731148, 17215560, 13829543, 14133975 17694209, 18091059, 17385178, 8322815, 17586955, 17201159, 17655634 18331812, 19730508, 18868646, 17648596, 16220077, 16069901, 17348614 17393915, 17274537, 17957017, 18096714, 17308789, 18436647, 14285317 19289642, 14764829, 18328509, 17622427, 16943711, 14368995, 17346671

18996843, 17783588, 16618694, 17672719, 18856999, 18783224, 17851160 17546761, 17798953, 18273830, 19972566, 16384983, 17726838, 17360606 13645875, 18199537, 16542886, 17889549, 14565184, 17071721, 20299015 17610798, 20657441, 17397545, 18230522, 16360112, 19769489, 12905058 18641451, 12747740, 18430495, 17042658, 17016369, 14602788, 19972568 18508861, 19788842, 14657740, 17332800, 13837378, 19972564, 17186905 18315328, 19699191, 17437634, 19006849, 19013183, 17296856, 18674024 17232014, 16855292, 21051840, 14692762, 17762296, 17705023, 19121551 19854503, 19309466, 18681862, 18554763, 20558005, 17390160, 18456514 16306373, 13955826, 18139690, 17501491, 17299889, 17752121, 17889583 18673325, 18293054, 17242746, 17951233, 17649265, 18094246, 19615136 17011832, 16870214, 17477958, 18522509, 20631274, 16091637, 17323222 16595641, 16524926, 18228645, 18282562, 17596908, 17156148, 18031668 16494615, 17545847, 17614134, 13558557, 17341326, 17891946, 17716305 16392068, 19271443, 18092127, 18440047, 17614227, 14106803, 16903536 18973907, 18673342, 17389192, 16194160, 17006570, 17612828, 17721717 17570240, 17390431, 16863422, 18325460, 19727057, 16422541, 19972570 17267114, 18244962, 21538485, 18765602, 18203838, 16198143, 17246576 14829250, 17835627, 18247991, 14458214, 21051862, 16692232, 17786278 17227277, 16042673, 16314254, 16228604, 16837842, 17393683, 17787259 20331945, 20074391, 15861775, 16399083, 18018515, 18260550, 21051858 17036973, 16613964, 17080436, 16579084, 18384537, 18280813, 20296213 16901385, 15979965, 18441944, 16450169, 9756271, 17892268, 11733603 16285691, 17587063, 16538760, 18180390, 18193833, 21051833, 17238511 17824637, 16571443, 18306996, 14852021, 18674047, 17853456, 12364061

Version 11.2.0.4.v4

Version 11.2.0.4.v4 adds support for the following:

- Oracle PSU 11.2.0.4.6 (20299013)
- Installs additional Oracle Text knowledge bases from Oracle Database. Examples media (English and French)
- Provides access to DBMS_REPAIR through RDSADMIN.RDSADMIN_DBMS_REPAIR
- Grants ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, and EXEMPT REDACTION POLICY to master user

Baseline: Oracle Database Patch Set Update 11.2.0.4.6 (patch 20299013, released April 2015)

Bugs fixed: 17288409, 17798953, 18273830, 18607546, 17811429, 17205719, 20506699 17816865, 19972566, 17922254, 17754782, 16384983, 17726838, 13364795 16934803, 17311728, 17284817, 17441661, 17360606, 13645875, 18199537 16992075, 16542886, 17446237, 14015842, 17889549, 14565184, 19972569 17071721, 20299015, 17610798, 17375354, 17449815, 17397545, 19463897 18230522, 138666822, 17235750, 17982555, 16360112, 18317531, 17478514 19769489, 12905058, 14338435, 18235390, 13944971, 18641451, 20142975 17811789, 16929165, 18704244, 12747740, 18430495, 20506706, 17546973 14054676, 17088068, 17346091, 18264060, 17016369, 17042658, 17343514 14602788, 19972568, 19680952, 18471685, 19788842, 18508861, 14657740 17332800, 19211724, 13837378, 13951456, 16315398, 17186905, 18744139 19972564, 16850630, 18315328, 17437634, 19049453, 18673304, 17883081 19006849, 19915271, 19013183, 18641419, 17296856, 18674024, 18262334 17006183, 18277454, 16833527, 17232014, 16855292, 10136473, 17762296 14692762, 17705023, 18051556, 17865671, 17852463, 18554871, 17853498 19121551, 18334586, 19854503, 17551709, 19309466, 17588480, 19827973 17344412, 17842825, 18828868, 18681862, 18554763, 17390160, 18456514 16306373, 17025461, 13955826, 18139690, 11883252, 13609098, 17501491 17239687, 17752121, 17299889, 17602269, 19197175, 17889583, 18316692 17313525, 18673325, 12611721, 19544839, 18293054, 17242746, 18964939 17600719, 18191164, 19393542, 17571306, 18482502, 19466309, 17951233 17649265, 18094246, 19615136, 17040527, 17011832, 17165204, 18098207 16785708, 16870214, 17465741, 16180763, 17174582, 17477958, 12982566 16777840, 18522509, 20631274, 16091637, 17323222, 19463893, 16595641 16875449, 12816846, 16524926, 17237521, 18228645, 18282562, 17596908 19358317, 17811438, 17811447, 17945983,

18762750, 17156148, 18031668 16912439, 17184721, 16494615, 18061914, 17282229, 17545847, 18331850 18202441, 17082359, 18723434, 19554106, 17614134, 13558557, 17341326 14034426, 17891946, 18339044, 17716305, 19458377, 17752995, 16392068 19271443, 17891943, 18092127, 17258090, 17767676, 16668584, 18384391 17614227, 17040764, 16903536, 17381384, 14106803, 15913355, 18973907 18356166, 18673342, 17389192, 14084247, 16194160, 17612828, 17006570 20506715, 17721717, 13853126, 17390431, 18203837, 17570240, 14245531 16043574, 16863422, 17848897, 17877323, 18325460, 19727057, 17468141 17786518, 17912217, 16422541, 19972570, 17267114, 17037130, 18244962 18765602, 18203838, 18155762, 16956380, 16198143, 17246576, 17478145 17394950, 14829250, 18189036, 18641461, 18619917, 17835627, 17027426 16268425, 18247991, 19584068, 14458214, 18436307, 17265217, 17634921 13498382, 16692232, 17786278, 17227277, 16042673, 16314254, 17443671 18000422, 16228604, 16837842, 17571039, 17393683, 16344544, 17787259 18009564, 20074391, 14354737, 15861775, 18135678, 18614015, 16399083 18362222, 18018515, 16472716, 17835048, 17050888, 17936109, 14010183 17325413, 18747196, 17080436, 16613964, 17036973, 17761775, 16579084 16721594, 17082983, 18384537, 18280813, 20296213, 17302277, 16901385 18084625, 15979965, 15990359, 18203835, 17297939, 17811456, 16731148 13829543, 14133975, 17215560, 17694209, 18091059, 17385178, 8322815 17586955, 18441944, 17201159, 16450169, 9756271, 17655634, 19730508 17892268, 18868646, 17648596, 16220077, 16069901, 11733603, 16285691 17587063, 18180390, 16538760, 18193833, 17348614, 17393915, 17957017 17274537, 18096714, 17308789, 17238511, 18436647, 17824637, 14285317 19289642, 14764829, 17622427, 18328509, 16571443, 16943711, 14368995 18306996, 17346671, 14852021, 18996843, 17783588, 16618694, 17853456 18674047, 17672719, 18856999, 12364061, 18783224, 17851160, 17546761

Version 11.2.0.4.v3

Version 11.2.0.4.v3 adds support for the following:

- Oracle PSU 11.2.0.4.4 (19121551)
- Latest DST file (DSTv23 – patch 19396455, released Oct 2014). This patch is incorporated by default in new instances only.

Baseline: Oracle Database Patch Set Update 11.2.0.4.4 (patch 19121551, released October 2014)

Bugs fixed: 19396455, 18759211, 17432124, 16799735, 17288409, 17205719, 17811429, 17754782, 17726838, 13364795, 17311728 17284817, 17441661, 13645875, 18199537, 16992075, 16542886, 17446237 14565184, 17071721, 17610798, 17375354, 17449815, 17397545, 19463897 18230522, 17235750, 16360112, 13866822, 17982555, 17478514, 12905058 14338435, 13944971, 16929165, 12747740, 17546973, 14054676, 17088068 18264060, 17343514, 17016369, 17042658, 14602788, 14657740, 17332800 19211724, 13951456, 16315398, 17186905, 18744139, 16850630, 17437634 19049453, 18673304, 17883081, 18641419, 17296856, 18262334, 17006183 18277454, 17232014, 16855292, 10136473, 17705023, 17865671, 18554871 19121551, 17588480, 17551709, 17344412, 17842825, 18681862, 17390160 13955826, 13609098, 18139690, 17501491, 17239687, 17752121, 17299889 17602269, 18673325, 17313525, 17242746, 19544839, 17600719, 18191164 17571306, 19466309, 17951233, 18094246, 17165204, 17011832, 17040527 16785708, 16180763, 17477958, 17174582, 17465741, 18522509, 17323222 19463893, 16875449, 16524926, 17237521, 17596908, 17811438, 17811447 18031668, 16912439, 16494615, 18061914, 17545847, 17082359, 19554106 17614134, 17341326, 17891946, 19458377, 17716305, 17752995, 16392068 19271443, 17767676, 17614227, 17040764, 17381384, 18973907, 18673342 14084247, 17389192, 17006570, 17612828, 17721717, 13853126, 18203837 17390431, 17570240, 14245531, 16043574, 16863422, 19727057, 17468141 17786518, 17037130, 17267114, 18203838, 16198143, 16956380, 17478145 14829250, 17394950, 17027426, 16268425, 18247991, 19584068, 14458214 18436307, 17265217, 13498382, 16692232, 17786278, 17227277, 16042673 16314254, 17443671, 16228604, 16837842, 17393683, 17787259, 18009564 15861775, 16399083, 18018515, 16472716, 17050888, 14010183, 17325413 16613964, 17080436, 17036973, 17761775, 16721594, 18280813, 15979965 18203835, 17297939, 16731148, 17811456, 14133975, 17385178, 17586955 16450169, 17655634, 9756271, 17892268,

17648596, 16220077, 16069901 11733603, 16285691, 17587063, 18180390, 17393915, 18096714, 17238511 17824637, 14285317, 19289642, 14764829, 18328509, 17622427, 16943711 17346671, 18996843, 14852021, 17783588, 16618694, 17672719, 17546761

Version 11.2.0.4.v2 (Deprecated)

Version 11.2.0.4.v2 adds support for the following:

- Oracle PSU 11.2.0.4.3 (18522509)
- User access to DBMS_TRANSACTION package to clean-up failed distributed transactions
- Latest DST file (DSTv22 – patch 18759211, released June 2014). This patch is incorporated by default only in new Oracle DB instances.
- Grants DBMS_REPUTIL to DBA role (upgrade to 11.2.0.4 revokes it from public)
- Privileges granted on DBMS_TRANSACTION, v\$pending_xatrans\$, and v\$xatrans\$
- Resolves a problem with DDL commands when user objects have "SYSTEM" in their names
- Installs schema objects to support XA Transactions, allowing transactions to be managed by an external transaction manager
- Permits truncation of temporary SYS and SYSTEM objects, allowing tools like LogMiner to function correctly

Baseline: Oracle Database Patch Set Update 11.2.0.4.3 (patch 18522509, released July 2014)

Bugs fixed: 17432124, 18759211, 18522509, 18031668, 17478514, 17752995, 17288409, 16392068, 17205719, 17811429, 17767676, 17614227 17040764, 17381384, 17754782, 17726838, 13364795, 17311728, 17389192 17006570, 17612828, 17284817, 17441661, 13853126, 17721717, 13645875 18203837, 17390431, 16542886, 16992075, 16043574, 17446237, 16863422 14565184, 17071721, 17610798, 17468141, 17786518, 17375354, 17397545 18203838, 16956380, 17478145, 16360112, 17235750, 17394950, 13866822 17478514, 17027426, 12905058, 14338435, 16268425, 13944971, 18247991 14458214, 16929165, 17265217, 13498382, 17786278, 17227277, 17546973 14054676, 17088068, 16314254, 17016369, 14602788, 17443671, 16228604 16837842, 17332800, 17393683, 13951456, 16315398, 18744139, 17186905 16850630, 17437634, 19049453, 17883081, 15861775, 17296856, 18277454 16399083, 16855292, 18018515, 10136473, 16472716, 17050888, 17865671 17325413, 14010183, 18554871, 17080436, 16613964, 17761775, 16721594 17588480, 17551709, 17344412, 18681862, 15979965, 13609098, 18139690 17501491, 17239687, 17752121, 17602269, 18203835, 17297939, 17313525 16731148, 17811456, 14133975, 17600719, 17385178, 17571306, 16450169 17655634, 18094246, 17892268, 17165204, 17011832, 17648596, 16785708 17477958, 16180763, 16220077, 17465741, 17174582, 18522509, 16069901 16285691, 17323222, 18180390, 17393915, 16875449, 18096714, 17238511

Version 11.2.0.4.v1

Version 11.2.0.4.v1 adds support for the following:

- Oracle PSU 11.2.0.4.1
- [Creating New Directories in the Main Data Storage Space \(p. 873\)](#)

Baseline: Oracle Database Patch Set Update 11.2.0.4.1 (released January 2014)

Bugs fixed: 17432124, 16850630, 17551709, 13944971, 17811447, 13866822, 17811429, 16069901 16721594, 17443671, 17478514, 17612828, 17610798, 17239687, 17501491 17446237, 16450169, 17811438, 17288409, 17811456, 12905058, 17088068 16285691, 17332800

Related Topics

- [Upgrading the Oracle DB Engine \(p. 770\)](#)
- [Oracle on Amazon RDS \(p. 717\)](#)

PostgreSQL on Amazon RDS

Amazon RDS supports DB instances running several versions of PostgreSQL. You can create DB instances and DB snapshots, point-in-time restores and backups. DB instances running PostgreSQL support Multi-AZ deployments, Read Replicas (version 9.3.5 and later), Provisioned IOPS, and can be created inside a VPC. You can also use Secure Socket Layer (SSL) to connect to a DB instance running PostgreSQL.

Before creating a DB instance, you should complete the steps in the [Setting Up for Amazon RDS \(p. 5\)](#) section of this guide.

You can use any standard SQL client application to run commands for the instance from your client computer. Such applications include *pgAdmin*, a popular Open Source administration and development tool for PostgreSQL, or *psql*, a command line utility that is part of a PostgreSQL installation. To deliver a managed service experience, Amazon RDS doesn't provide host access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application. Amazon RDS doesn't allow direct host access to a DB instance by using Telnet or Secure Shell (SSH).

Amazon RDS for PostgreSQL is compliant with many industry standards. For example, you can use Amazon RDS for PostgreSQL databases to build HIPAA-compliant applications and to store healthcare-related information, including protected health information (PHI) under an executed Business Associate Agreement (BAA) with AWS. Amazon RDS for PostgreSQL also meets Federal Risk and Authorization Management Program (FedRAMP) security requirements. Amazon RDS for PostgreSQL has received a FedRAMP Joint Authorization Board (JAB) Provisional Authority to Operate (P-ATO) at the FedRAMP HIGH Baseline within the AWS GovCloud (US-West) Region. For more information on supported compliance standards, see [AWS Cloud Compliance](#).

To import PostgreSQL data into a DB instance, follow the information in the [Importing Data into PostgreSQL on Amazon RDS \(p. 996\)](#) section.

Topics

- [Common Management Tasks for PostgreSQL on Amazon RDS \(p. 965\)](#)
- [Creating a DB Instance Running the PostgreSQL Database Engine \(p. 969\)](#)
- [Connecting to a DB Instance Running the PostgreSQL Database Engine \(p. 976\)](#)
- [Modifying a DB Instance Running the PostgreSQL Database Engine \(p. 979\)](#)
- [Upgrading the PostgreSQL DB Engine \(p. 988\)](#)
- [Working with PostgreSQL Read Replicas \(p. 992\)](#)
- [Importing Data into PostgreSQL on Amazon RDS \(p. 996\)](#)
- [Common DBA Tasks for PostgreSQL \(p. 1000\)](#)
- [Working with the Database Preview Environment \(p. 1026\)](#)
- [Amazon RDS for PostgreSQL Versions and Extensions \(p. 1029\)](#)

Common Management Tasks for PostgreSQL on Amazon RDS

The following are the common management tasks you perform with an Amazon RDS for PostgreSQL DB instance, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Setting up Amazon RDS for first-time use <p>There are prerequisites you must complete before you create your DB instance. For example, DB instances are created by default with a firewall that prevents access to it. You therefore must create a security group with the correct IP addresses and network configuration to access the DB instance.</p>	Setting Up for Amazon RDS (p. 5)
Understanding Amazon RDS DB instances <p>If you are creating a DB instance for production purposes, you should understand how instance classes, storage types, and Provisioned IOPS work in Amazon RDS.</p>	DB Instance Class (p. 82) Amazon RDS Storage Types (p. 103) Provisioned IOPS SSD Storage (p. 105)
Finding supported PostgreSQL versions <p>Amazon RDS supports several versions of PostgreSQL.</p>	Supported PostgreSQL Database Versions (p. 1030)
Setting up high availability and failover support <p>A production DB instance should use Multi-AZ deployments. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances.</p>	High Availability (Multi-AZ) for Amazon RDS (p. 109)
Understanding the Amazon Virtual Private Cloud (VPC) network <p>If your AWS account has a default VPC, then your DB instance is automatically created inside the default VPC. In some cases, your account might not have a default VPC, and you might want the DB instance in a VPC. In these cases, create the VPC and subnet groups before you create the DB instance.</p>	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412) Working with an Amazon RDS DB Instance in a VPC (p. 420)
Importing data into Amazon RDS PostgreSQL <p>You can use several different tools to import data into your PostgreSQL DB instance on Amazon RDS.</p>	Importing Data into PostgreSQL on Amazon RDS (p. 996)
Setting up read only Read Replicas (master/standby) <p>PostgreSQL on Amazon RDS supports Read Replicas in both the same AWS Region and in a different AWS Region from the master instance.</p>	Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances (p. 143) Working with PostgreSQL Read Replicas (p. 992) Creating a Read Replica in a Different AWS Region (p. 148)
Understanding security groups <p>By default, DB instances are created with a firewall that prevents access to them. You therefore must create a security group with the correct IP addresses and network configuration to access the DB instance.</p>	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 412) Controlling Access with Security Groups (p. 394)

Task Area	Relevant Documentation
<p>In general, if your DB instance is on the <i>EC2-Classic</i> platform, you need to create a DB security group. If your DB instance is on the <i>EC2-VPC</i> platform, you need to create a VPC security group.</p>	
<p>Setting up parameter groups and features</p> <p>If your DB instance is going to require specific database parameters, you should create a parameter group before you create the DB instance.</p>	<p>Working with DB Parameter Groups (p. 169)</p>
<p>Performing common DBA tasks for PostgreSQL</p> <p>Some of the more common tasks for PostgreSQL DBAs include:</p> <ul style="list-style-type: none"> • Creating Roles (p. 1000) • Managing PostgreSQL Database Access (p. 1001) • Working with PostgreSQL Parameters (p. 1001) • Working with PostgreSQL Autovacuum on Amazon RDS (p. 1009) • Audit Logging for a PostgreSQL DB Instance (p. 1017) • Working with PostGIS (p. 1020) • Using pgBadger for Log Analysis with PostgreSQL (p. 1022) • Using a Custom DNS Server for Outbound Network Access (p. 1025) 	<p>Common DBA Tasks for PostgreSQL (p. 1000)</p>
<p>Connecting to your PostgreSQL DB instance</p> <p>After creating a security group and associating it to a DB instance, you can connect to the DB instance using any standard SQL client application such as pgadmin III.</p>	<p>Connecting to a DB Instance</p> <p>Running the PostgreSQL Database Engine (p. 976)</p> <p>Using SSL with a PostgreSQL DB Instance (p. 1068)</p>
<p>Backing up and restoring your DB instance</p> <p>You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots.</p>	<p>Backing Up and Restoring Amazon RDS DB Instances (p. 207)</p>
<p>Monitoring the activity and performance of your DB instance</p> <p>You can monitor a PostgreSQL DB instance by using CloudWatch Amazon RDS metrics, events, and enhanced monitoring.</p>	<p>Viewing DB Instance Metrics (p. 254)</p> <p>Viewing Amazon RDS Events (p. 305)</p>
<p>Upgrading the PostgreSQL database version</p> <p>You can do both major and minor version upgrades for your PostgreSQL DB instance.</p>	<p>Upgrading a PostgreSQL DB Instance (p. 1068)</p> <p>Major Version Upgrades (p. 989)</p>
<p>Working with log files</p> <p>You can access the log files for your PostgreSQL DB instance.</p>	<p>PostgreSQL Database Log Files (p. 334)</p>

Task Area	Relevant Documentation
Understanding the best practices for PostgreSQL DB instances Find some of the best practices for working with PostgreSQL on Amazon RDS.	Best Practices for Working with PostgreSQL (p. 77)

Creating a DB Instance Running the PostgreSQL Database Engine

The basic building block of Amazon RDS is the DB instance. This is the environment in which you will run your PostgreSQL databases.

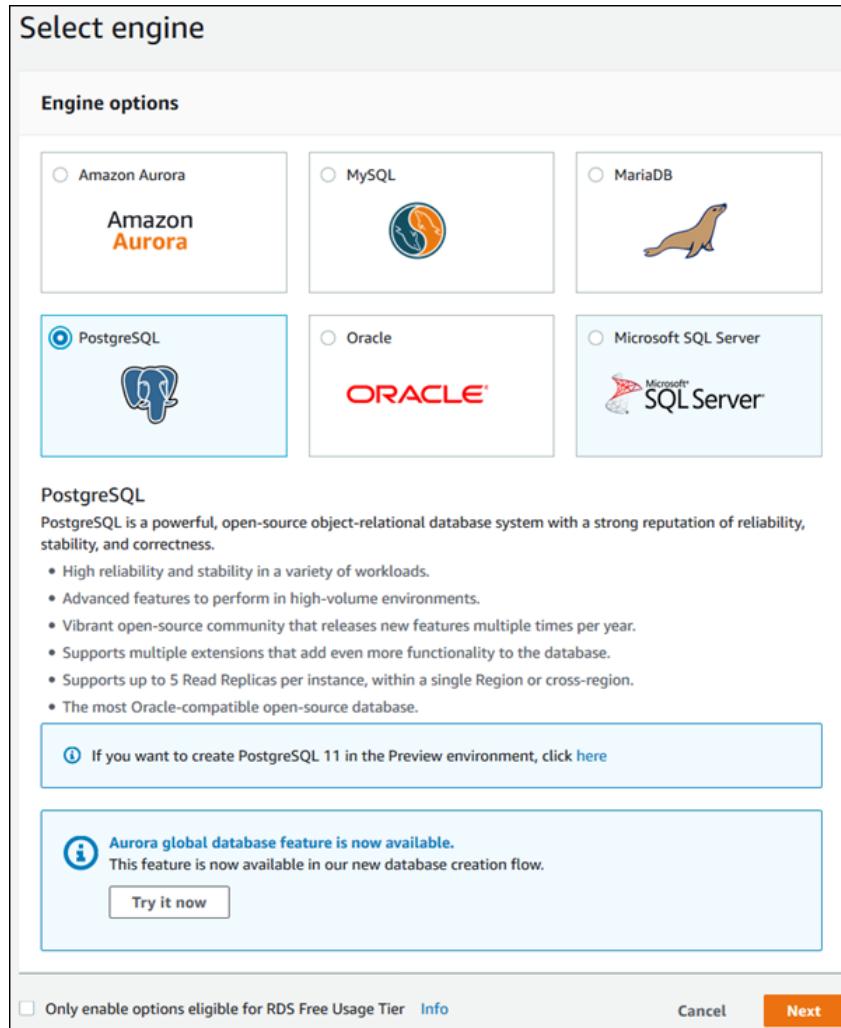
Important

You must complete the tasks in the [Setting Up for Amazon RDS \(p. 5\)](#) section before you can create or connect to a DB instance.

Create a PostgreSQL DB Instance

To launch a PostgreSQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the top right corner of the AWS Management Console, select the AWS Region where you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
If the navigation pane is closed, choose the menu icon at the top left to open it.
4. Choose **Create database** to start open on the **Select engine** page.



5. On the **Select engine** page, choose the PostgreSQL icon, and then choose **Next**.
6. Next, the **Use case** page asks if you are planning to use the DB instance you are creating for production. If you are, choose **Production**. If you choose this option, the following are preselected in a later step:
 - **Multi-AZ failover** option
 - **Provisioned IOPS** storage option
 - **Enable deletion protection** option

Choose **Next** when you are finished.

7. On the **Specify DB Details** page, specify your DB instance information. Choose **Next** when you are finished.

For This Parameter	Do This
License Model	PostgreSQL has only one license model. Choose postgresql-license to use the general license agreement for PostgreSQL.

For This Parameter	Do This
DB Engine Version	Choose the version of PostgreSQL you want to use.
DB Instance Class	Choose db.t2.small for a configuration that equates to 2 GiB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity. For more information about all the DB instance class options, see DB Instance Class (p. 82) .
Multi-AZ Deployment	<p>Choose Yes to have a standby replica of your DB instance created in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose No.</p> <p>For more information, see High Availability (Multi-AZ) for Amazon RDS (p. 109).</p>
Storage Type	Choose the storage type General Purpose (SSD) . For more information about storage, see DB instance storage (p. 103) .
Allocated Storage	Type 20 to allocate 20 GiB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon Relational Database Service Features .
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the AWS Region you chose. You can add some intelligence to the name, such as including the AWS Region and DB engine you chose, for example postgresSQL-test .
Master Username	Type a name using alphanumeric characters to use as the master user name to log on to your DB instance. For information on the default privileges granted to the master user name, see Amazon RDS for PostgreSQL Versions and Extensions (p. 1029)
Master Password and Confirm Password	Type a password that contains from 8 to 128 printable ASCII characters (excluding /, ", and @) for your master password, then type the password again in the Confirm Password box.

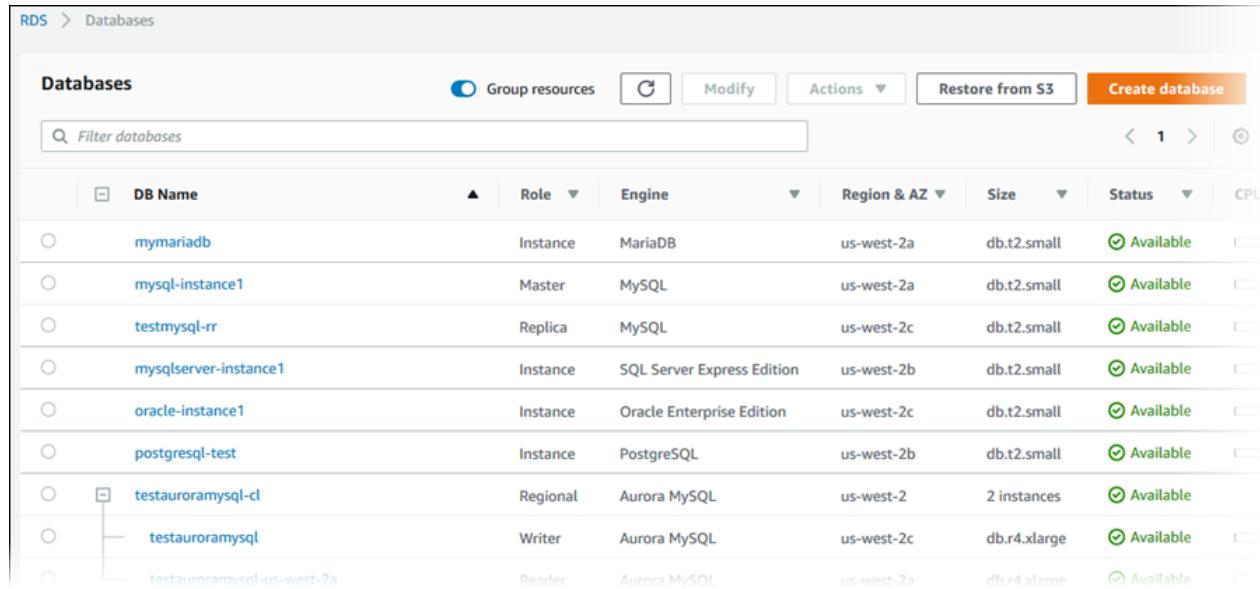
- On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the PostgreSQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then choose **Create database**.

For This Parameter	Do This
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, choose Not Specified .

For This Parameter	Do This
	in VPC. For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 412) .
Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, choose default , which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC. For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 412) .
Publicly Accessible	Choose Yes to give the DB instance a public IP address, meaning that it is accessible outside the VPC; otherwise, choose No , so the DB instance is only accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB Instance in a VPC from the Internet (p. 422) .
Availability Zone	Use the default value of No Preference unless you want to specify an Availability Zone.
VPC Security Group	If you are a new customer to AWS, choose the default VPC. If you created a VPC security group, choose the VPC security group you previously created. When you choose Create new VPC security group in the RDS console, a new security group is created with an inbound rule that allows access to the DB instance from the IP address detected in your browser.
Database Name	Type a name for your database of up to 63 alpha-numeric characters. If you do not provide a name, the default "postgres" database is created. To create additional databases, connect to the DB instance and use the SQL command <code>CREATE DATABASE</code> . For more information about connecting to the DB instance, see Connecting to a DB Instance Running the PostgreSQL Database Engine (p. 976) .
Database Port	Specify a port you want to use to access the database. PostgreSQL installations default to port 5432.
DB Parameter Group	Use the default value unless you have created your own parameter group.
Option Group	Use the default value unless you have created your own option group.
Copy Tags To Snapshots	Choose this option to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .
Enable Encryption	Choose Yes to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 389) .

For This Parameter	Do This
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to 1 .
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of No Preference .
Enable Enhanced Monitoring	Choose Yes to enable real-time OS monitoring. Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You are only charged for Enhanced Monitoring that exceeds the free tier provided by Amazon CloudWatch Logs.
Monitoring Role	Choose Default to use the default IAM role.
Granularity	Choose 60 to monitor the instance every minute.
Auto minor version upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.
Maintenance Window	Choose the 30-minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose No Preference .
Enable deletion protection	Enable deletion protection to prevent your DB instance from being deleted. If you create a production DB instance with the AWS Management Console, deletion protection is enabled by default. For more information, see Deleting a DB Instance (p. 135) .

9. On the final page, choose **Create database**.
10. On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.



The screenshot shows the AWS RDS Databases console. At the top, there's a navigation bar with 'RDS' and 'Databases'. Below it is a header row with columns: DB Name, Role, Engine, Region & AZ, Size, Status, and CPU. A 'Group resources' button and a 'Modify' button are also at the top. A search bar labeled 'Filter databases' is present. The main table lists several database instances:

DB Name	Role	Engine	Region & AZ	Size	Status	CPU
mymariadb	Instance	MariaDB	us-west-2a	db.t2.small	Available	
mysql-instance1	Master	MySQL	us-west-2a	db.t2.small	Available	
testmysql-rr	Replica	MySQL	us-west-2c	db.t2.small	Available	
mysqlserver-instance1	Instance	SQL Server Express Edition	us-west-2b	db.t2.small	Available	
oracle-instance1	Instance	Oracle Enterprise Edition	us-west-2c	db.t2.small	Available	
postgres-test	Instance	PostgreSQL	us-west-2b	db.t2.small	Available	
testauroramysql-cl	Regional	Aurora MySQL	us-west-2	2 instances	Available	
testauroramysql	Writer	Aurora MySQL	us-west-2c	db.r4.xlarge	Available	
testauroramysql-us-west-2a	Reader	Aurora MySQL	us-west-2a	db.r4.xlarge	Available	

CLI

To create a PostgreSQL DB instance, use the AWS CLI [create-db-instance](#) command with the following parameters:

- `--db-instance-identifier`
- `--allocated-storage`
- `--db-instance-class`
- `--engine`
- `--master-username`
- `--master-user-password`

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance
--db-instance-identifier pgdbinstance \
--allocated-storage 20 \
--db-instance-class db.t2.small \
--engine postgres \
--master-username masterawsuser \
--master-user-password masteruserpassword
```

For Windows:

```
aws rds create-db-instance
--db-instance-identifier pgdbinstance ^
--allocated-storage 20 ^
--db-instance-class db.t2.small ^
--engine postgres ^
--master-username masterawsuser ^
--master-user-password masteruserpassword
```

This command should produce output similar to the following:

```
DBINSTANCE pgdbinstance db.t2.small postgres 20 sa creating 3 **** n 9.3
SECGROUP default active
PARAMGRP default.PostgreSQL9.3 in-sync
```

API

To create a PostgreSQL DB instance, use the Amazon RDS API[CreateDBInstance](#) command with the following parameters:

- Engine = *postgres*
- DBInstanceIdentifier = *pgdbinstance*
- DBInstanceClass = *db.t2.small*
- AllocatedStorage = *20*
- BackupRetentionPeriod = *3*
- MasterUsername = *masterawsuser*
- MasterUserPassword = *masteruserpassword*

Example

```
https://rds.amazonaws.com/
?Action=CreateDBInstance
&AllocatedStorage=20
&BackupRetentionPeriod=3
&DBInstanceClass=db.t2.small
&DBInstanceIdentifier=pgdbinstance
&DBName=mydatabase
&DBSecurityGroups.member.1=mysecuritygroup
&DBSubnetGroup=mydbsubnetgroup
&Engine=postgres
&MasterUserPassword=<masteruserpassword>
&MasterUsername=<masterawsuser>
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2013-09-09
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140212/us-west-2/rds/aws4_request
&X-Amz-Date=20140212T190137Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=60d520ca0576c191b9eac8dbfe5617ebb6a6a9f3994d96437a102c0c2c80f88d
```

Connecting to a DB Instance Running the PostgreSQL Database Engine

After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. To list the details of an Amazon RDS DB instance, you can use the AWS Management Console, the AWS CLI [describe-db-instances](#) command, or the Amazon RDS API [DescribeDBInstances](#) action. You need the following information to connect:

- The host or host name for the DB instance, for example:

```
myinstance.123456789012.us-east-1.rds.amazonaws.com
```

- The port on which the DB instance is listening. For example, the default PostgreSQL port is 5432.
- The user name and password for the DB instance.

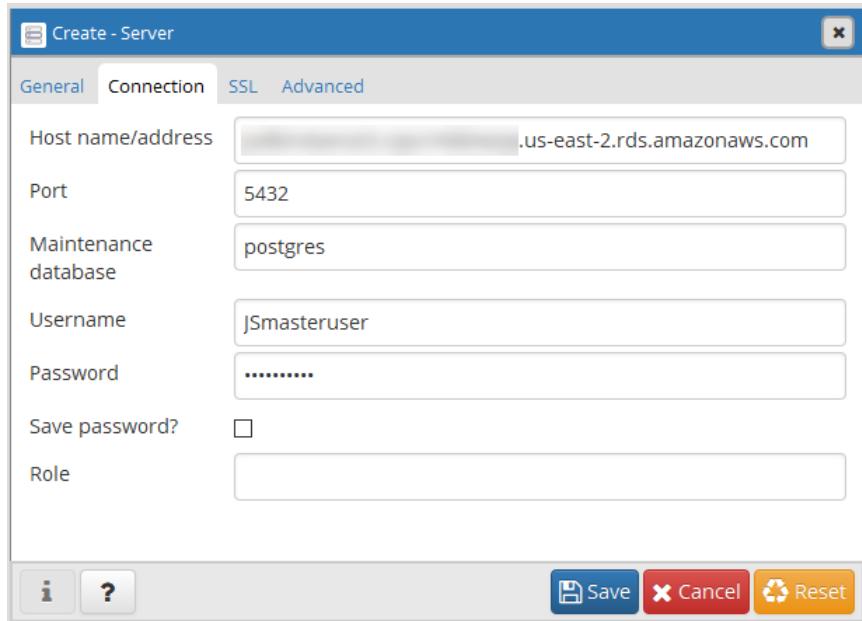
Following are two ways to connect to a PostgreSQL DB instance. The first example uses pgAdmin, a popular open-source administration and development tool for PostgreSQL. The second example uses psql, a command line utility that is part of a PostgreSQL installation.

Using pgAdmin to Connect to a PostgreSQL DB Instance

You can use the open-source tool pgAdmin to connect to a PostgreSQL DB instance.

To connect to a PostgreSQL DB instance using pgAdmin

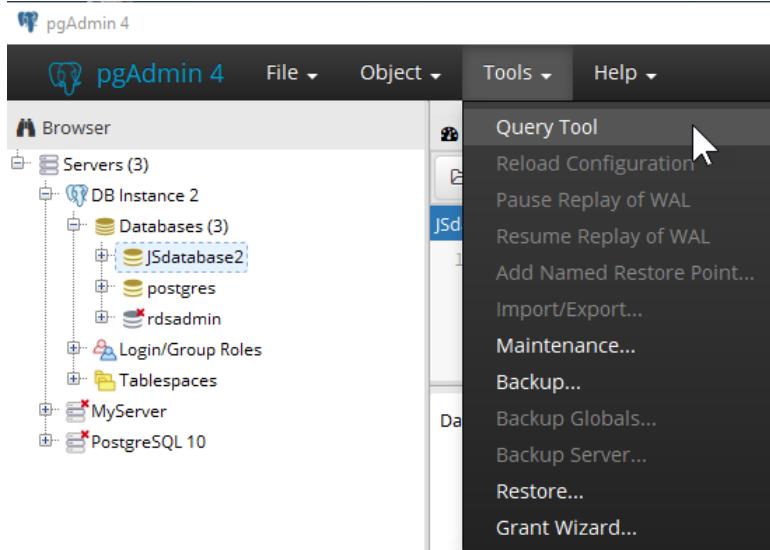
1. Install pgAdmin from <http://www.pgadmin.org/>. You can download and use pgAdmin without having a local instance of PostgreSQL on your client computer.
2. Launch the pgAdmin application on your client computer.
3. On the **Dashboard** tab, choose **Add New Server**.
4. In the **Create - Server** dialog box, type a name on the **General** tab to identify the server in pgAdmin.
5. On the **Connection** tab, type the following information from your DB instance:
 - For **Host**, type the endpoint, for example `mypostgresql.c6c8dntfzzhgv0.us-east-2.rds.amazonaws.com`.
 - For **Port**, type the assigned port.
 - For **Username**, type the user name that you entered when you created the DB instance.
 - For **Password**, type the password that you entered when you created the DB instance.



6. Choose **Save**.

If you have any problems connecting, see [Troubleshooting Connection Issues \(p. 978\)](#).

7. To access a database in the pgAdmin browser, expand **Servers**, the DB instance, and **Databases**. Choose the DB instance's database name.



8. To open a panel where you can enter SQL commands, choose **Tools, Query Tool**.

Using psql to Connect to a PostgreSQL DB Instance

You can use a local instance of the psql command line utility to connect to a PostgreSQL DB instance. You need either PostgreSQL or the psql client installed on your client computer. To connect to your PostgreSQL DB instance using psql, you need to provide host information and access credentials.

Use one of the following formats to connect to a PostgreSQL DB instance on Amazon RDS. When you connect, you're prompted for a password. For batch jobs or scripts, use the `--no-password` option.

For Unix, use the following format.

```
psql \
--host=<DB instance endpoint> \
--port=<port> \
--username=<master user name> \
--password \
--dbname=<database name>
```

For Windows, use the following format.

```
psql ^
--host=<DB instance endpoint> ^
--port=<port> ^
--username=<master user name> ^
--password ^
--dbname=<database name>
```

For example, the following command connects to a database called `mypgdb` on a PostgreSQL DB instance called `mypostgresql` using fictitious credentials.

```
psql --host=mypostgresql.c6c8mwvfdgv0.us-west-2.rds.amazonaws.com --port=5432 --
username=awsuser --password --dbname=mypgdb
```

Troubleshooting Connection Issues

If you can't connect to the DB instance, the most common error is `Could not connect to server: Connection timed out`. If you receive this error, do the following:

- Check that the host name used is the DB instance endpoint and that the port number used is correct.
- Make sure that the DB instance's public accessibility is set to **Yes**.
- Check that the security group assigned to the DB instance has rules to allow access through any firewall your connection might go through. For example, if the DB instance was created using the default port of 5432, your company might have firewall rules blocking connections to that port from company devices.

To fix this failure, modify the DB instance to use a different port. Also, make sure that the security group applied to the DB instance allows connections to the new port.

- Check whether the DB instance was created using a security group that doesn't authorize connections from the device or Amazon EC2 instance where the application is running. For the connection to work, the security group you assigned to the DB instance at its creation must allow access to the DB instance. For example, if the DB instance was created in a VPC, it must have a VPC security group that authorizes connections. Alternatively, if the DB instance was created outside of a VPC, it must have a database security group that authorizes those connections.

By far the most common connection problem is with the security group's access rules assigned to the DB instance. If you used the default DB security group when you created the DB instance, the security group likely didn't have access rules that allow you to access the instance. For more information about Amazon RDS security groups, see [Controlling Access with Security Groups \(p. 394\)](#).

Modifying a DB Instance Running the PostgreSQL Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS PostgreSQL DB instance, and describes the settings for PostgreSQL instances. For information about additional tasks, such as renaming, rebooting, deleting, tagging, or upgrading an Amazon RDS DB instance, see [Amazon RDS DB Instance Lifecycle \(p. 112\)](#). We recommend that you test any changes on a test instance before modifying a production instance so you better understand the impact of a change. This is especially important when upgrading database versions.

You can have the changes apply immediately or have them applied during the DB instance's next maintenance window. Applying changes immediately can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option when modifying a DB instance, see [Modifying an Amazon RDS DB Instance \(p. 115\)](#).

AWS Management Console

To modify a PostgreSQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want. For information about each setting, see [Settings for PostgreSQL DB Instances \(p. 980\)](#).
5. When all the changes are as you want them, choose **Continue**.
6. To apply the changes immediately, choose **Apply Immediately**. Choosing this option can cause an outage in some cases. For more information, see [Using the Apply Immediately Parameter \(p. 115\)](#).
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify a PostgreSQL DB instance, use the AWS CLI command [modify-db-instance](#).

Example

The following code modifies pgdbinstance by setting the backup retention period to 1 week (7 days) and disabling automatic minor version upgrades. These changes are applied during the next maintenance window.

Parameters

- **--db-instance-identifier**—the name of the DB instance
- **--backup-retention-period**—the number of days to retain automatic backups.
- **--auto-minor-version-upgrade**—allow automatic minor version upgrades. To disallow automatic minor version upgrades, use **--no-auto-minor-version-upgrade**.
- **--no-apply-immediately**—apply changes during the next maintenance window. To apply changes immediately, use **--apply-immediately**.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier pgdbinstance \
--backup-retention-period 7 \
--auto-minor-version-upgrade \
--no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier pgdbinstance ^
--backup-retention-period 7 ^
--auto-minor-version-upgrade ^
--no-apply-immediately
```

API

To modify a PostgreSQL DB instance, use the [ModifyDBInstance](#) action.

Example

The following code modifies pgdbinstance by setting the backup retention period to 1 week (7 days) and disabling automatic minor version upgrades. These changes are applied during the next maintenance window.

Parameters

- **DBInstanceIdentifier**—the name of the DB instance
- **BackupRetentionPeriod**—the number of days to retain automatic backups.
- **AutoMinorVersionUpgrade=true**—allow automatic minor version upgrades. To disallow automatic minor version upgrades, set the value to **false**.
- **ApplyImmediately=false**—apply changes during the next maintenance window. To apply changes immediately, set the value to **true**.

```
https://rds.us-east-1.amazonaws.com/
?Action=ModifyDBInstance
&ApplyImmediately=false
&AutoMinorVersionUpgrade=true
&BackupRetentionPeriod=7
&DBInstanceIdentifier=mydbinstance
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2013-09-09
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-east-1/rds/aws4_request
&X-Amz-Date=20131016T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757fffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for PostgreSQL DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated Storage	<p>The storage, in gigabytes, that you want to allocate for your DB instance. You can only increase the allocated storage, you can't reduce the allocated storage.</p> <p>You can't modify allocated storage if the DB instance status is <i>storage-optimization</i> or if the allocated storage for the DB instance has been modified in the last six hours.</p> <p>The maximum storage allowed depends on the storage type. For more information, see DB instance storage (p. 103).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	No downtime. Performance may be degraded during the change.
Auto Minor Version Upgrade	Choose Enable auto minor version upgrade to enable your DB instance to receive preferred minor DB engine version upgrades automatically when they become available. Amazon RDS performs automatic minor version upgrades in the maintenance window.	–	–
Backup Retention Period	<p>The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0.</p> <p>For more information, see Working With Backups (p. 208).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false and you change the setting from a non-zero value to another non-zero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.</p>	An outage occurs if you change from 0 to a non-zero value, or from a non-zero value to 0.
Backup Window	<p>The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours.</p> <p>For more information, see Working With Backups (p. 208).</p>	The change is applied asynchronously, as soon as possible.	–
Certificate Authority	The certificate that you want to use.	–	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Copy Tags to Snapshots	If you have any DB instance tags, this option copies them when you create a DB snapshot. For more information, see Tagging Amazon RDS Resources (p. 138) .	The change occurs immediately. This setting ignores the Apply immediately setting.	–
Database Port	The port that you want to use to access the database. The port value must not match any of the port values specified for options in the option group for the DB instance.	The change occurs immediately. This setting ignores the Apply Immediately setting.	The DB instance is rebooted immediately.
DB Engine Version	The version of the PostgreSQL database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications.	If Apply Immediately is set to true, the change occurs immediately. If Apply Immediately is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Class	The DB instance class that you want to use. For more information, see DB Instance Class (p. 82)	If Apply Immediately is set to true, the change occurs immediately. If Apply Immediately is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Identifier	The DB instance identifier. This value is stored as a lowercase string. For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 126) .	If Apply Immediately is set to true, the change occurs immediately. If Apply Immediately is set to false, the change occurs during the next maintenance window.	An outage occurs during this change. The DB instance is rebooted.

Setting	Setting Description	When the Change Occurs	Downtime Notes
DB Parameter Group	<p>The parameter group that you want associated with the DB instance.</p> <p>For more information, see Working with DB Parameter Groups (p. 169).</p>	The parameter group change occurs immediately.	<p>An outage doesn't occur during this change. When you change the parameter group, changes to some parameters are applied to the DB instance immediately without a reboot. Changes to other parameters are applied only after the DB instance is rebooted.</p> <p>For more information, see Rebooting a DB Instance (p. 129).</p>
Enable deletion protection	Enable deletion protection to prevent your DB instance from being deleted. For more information, see Deleting a DB Instance (p. 135) .	–	–
Enable Enhanced Monitoring	<p>Yes to enable gathering metrics in real time for the operating system that your DB instance runs on.</p> <p>For more information, see Enhanced Monitoring (p. 256).</p>	–	–
IAM DB authentication	<p>Enable IAM DB authentication to enable IAM database authentication for this DB instance.</p> <p>For more information, see IAM Database Authentication for MySQL and PostgreSQL (p. 372).</p>	–	–
License Model	Select the PostgreSQL License.	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change.
Log exports	<p>Select the types of PostgreSQL database log files to publish to Amazon CloudWatch Logs.</p> <p>For more information, see PostgreSQL Database Log Files (p. 334).</p>	The change occurs immediately. This setting ignores the Apply immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Maintenance Window	<p>The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours.</p> <p>If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied.</p> <p>For more information, see The Amazon RDS Maintenance Window (p. 120).</p>	The change occurs immediately. This setting ignores the Apply Immediately setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.
Multi-AZ Deployment	<p>Yes to deploy your DB instance in multiple Availability Zones; otherwise, No.</p> <p>For more information, see Regions and Availability Zones (p. 101).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	–
New Master Password	The password for your master user. The password must contain from 8 to 30 alphanumeric characters.	The change is applied asynchronously, as soon as possible. This setting ignores the Apply Immediately setting.	–
Option Group	<p>No options are available for PostgreSQL DB instances.</p> <p>For more information, see Working with Option Groups (p. 156).</p>	–	–
Publicly Accessible	<p>Yes to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. No to make the DB instance accessible only from inside the VPC.</p> <p>For more information, see Hiding a DB Instance in a VPC from the Internet (p. 422).</p>	The change occurs immediately. This setting ignores the Apply Immediately setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Security Group	<p>The security group you want associated with the DB instance.</p> <p>For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 399).</p>	<p>The change is applied asynchronously, as soon as possible. This setting ignores the Apply Immediately setting.</p>	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Storage Type	<p>The storage type that you want to use.</p> <p>For more information, see Amazon RDS Storage Types (p. 103).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	<p>The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place.</p> <ul style="list-style-type: none"> • From General Purpose (SSD) to Magnetic. • From General Purpose (SSD) to Provisioned IOPS (SSD), if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ DB instance or for the source DB instance of a read replica. • From Magnetic to General Purpose (SSD). • From Magnetic to Provisioned IOPS (SSD). • From Provisioned IOPS (SSD) to Magnetic. • From Provisioned IOPS (SSD) to General Purpose (SSD), if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ

Setting	Setting Description	When the Change Occurs	Downtime Notes
			DB instance or for the source DB instance of a read replica.
Subnet Group	<p>The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC.</p> <p>For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 426).</p>	<p>If Apply Immediately is set to true, the change occurs immediately.</p> <p>If Apply Immediately is set to false, the change occurs during the next maintenance window.</p>	An outage occurs during this change. The DB instance is rebooted.

Related Topics

- the section called “Rebooting a DB Instance” (p. 129) (p. 129)
- the section called “Connecting to a DB Instance Running the PostgreSQL Database Engine” (p. 976) (p. 976)
- the section called “Upgrading the PostgreSQL DB Engine” (p. 988)

Upgrading the PostgreSQL DB Engine

When Amazon RDS supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades: major version upgrades and minor version upgrades.

Amazon RDS supports major and minor version upgrades for PostgreSQL DB instances.

Major version upgrades can contain database changes that are not backward-compatible with existing applications. As a result, Amazon RDS doesn't apply major version upgrades automatically; you must manually modify your DB instance. You can initiate a major version upgrade manually by modifying your instance. However, there are recommended steps to follow when performing a major version upgrade. For details, see [Major Version Upgrades \(p. 989\)](#).

You can initiate a minor version upgrade manually by modifying your instance. Alternatively, you can enable the auto minor version upgrades option when creating or modifying a DB instance. Doing so means that your instance is automatically upgraded after the new version is tested and approved by Amazon RDS.

AWS RDS does not automatically upgrade PostgreSQL extensions. To upgrade an extension, you must use the `ALTER EXTENSION UPDATE` command. For example, to upgrade PostGIS when you upgrade the PostgreSQL DB engine from 9.4.x to 9.5.x, you would run the following command:

```
ALTER EXTENSION POSTGIS UPDATE TO '2.2.2'
```

Note

If you are running the PostGIS extension in your Amazon RDS PostgreSQL instance, make sure and follow the [PostGIS upgrade instructions](#) before you upgrade PostgreSQL.

Topics

- [Overview of Upgrading \(p. 988\)](#)
- [Major Version Upgrades \(p. 989\)](#)
- [Automatic Minor Version Upgrades for PostgreSQL \(p. 991\)](#)
- [Upgrading a PostgreSQL DB Instance Manually \(p. 991\)](#)

Overview of Upgrading

If your backup retention period is greater than 0, Amazon RDS takes two DB snapshots during both the major and minor upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken after the upgrade completes.

Note

Amazon RDS only takes DB snapshots if you have set the backup retention period for your DB instance to a number greater than 0. To change your backup retention period, see [Modifying a DB Instance Running the PostgreSQL Database Engine \(p. 979\)](#).

After an upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the DB snapshot that was taken before the upgrade to create a new DB instance.

If your DB instance is in a Multi-AZ deployment, both the primary and standby DB instances are upgraded. The primary and standby DB instances are upgraded at the same time, and you experience an outage until the upgrade is complete.

Major Version Upgrades

Major version upgrades can contain database changes that are not backward-compatible with previous versions of the database. This functionality can cause your existing applications to stop working correctly. As a result, Amazon RDS doesn't apply major version upgrades automatically; you must modify your DB instance manually to perform a major version upgrade. You should thoroughly test any upgrade to verify that your applications work correctly before applying the upgrade to your production DB instances. A best practice we recommend is to perform the major version upgrade on a restored instance that you create from a DB snapshot.

Amazon RDS supports an in-place upgrade from the following:

- A PostgreSQL 9.3.x DB instance to a PostgreSQL 9.4.x DB instance
- A PostgreSQL 9.4.x DB instance to a PostgreSQL 9.5.x DB instance
- A PostgreSQL 9.5.x DB instance to a PostgreSQL 9.6.x DB instance
- A PostgreSQL 9.6.x DB instance to a PostgreSQL 10.x DB instance

Amazon RDS uses the `pg_upgrade` utility found at <http://www.postgresql.org/docs/9.4/static/pgupgrade.html> to safely upgrade your instance.

Because some PostgreSQL minor versions updates for 9.3 were released after major version 9.4 was released, you cannot upgrade from version 9.3.9 to 9.4.1, and you cannot upgrade from version 9.3.10 to 9.4.1 or 9.4.4.

Read Replicas cannot undergo a major version upgrade. The source instance can undergo a major version upgrade, but all Read Replicas remain as readable nodes on the previous engine version. After a source instance is upgraded, its Read Replicas can no longer replicate changes performed on the source instance. We recommend that you either promote your Read Replicas, or delete and recreate them after the source instance has upgraded to a different major version.

Major Version Upgrade Process

We recommend the following process when upgrading an Amazon RDS PostgreSQL DB instance:

1. **Have a version-compatible parameter group ready** – If you are using a custom parameter group, you must specify either a default parameter group for the new DB engine version or create your own custom parameter group for the new DB engine version. Associating the new parameter group with the DB instance requires a customer-initiated database reboot after the upgrade completes. The instance's parameter group status will show pending-reboot if the instance needs to be rebooted to apply the parameter group changes. An instance's parameter group status can be viewed in the AWS console or by using a "describe" call such as `describe-db-instances`.
2. **Check for unsupported usage:**
 - a. **Prepared transactions** – Commit or roll back all open prepared transactions before attempting an upgrade.

You can use the following query to verify that there are no open prepared transactions on your instance:

```
SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
```

- b. **The line data type** – If you are upgrading an RDS PostgreSQL 9.3 instance, you must remove all uses of the `line` data type before attempting an upgrade, because the `line` data type was not fully implemented in PostgreSQL until version 9.4.

You can use the following query on each database to be upgraded to verify that there are no uses of the `line` data type in each database:

```
SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
pg_catalog.pg_attribute a
WHERE c.oid = a.attrelid
AND NOT a.attisdropped
AND a.atttypid = 'pg_catalog.line'::pg_catalog.regtype
AND c.relnamespace = n.oid
AND n.nspname !~ '^pg_temp_'
AND n.nspname !~ '^pg_toast_temp_'
AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

Note

To list all databases on an instance, use the following query:

```
SELECT d.datname FROM pg_catalog.pg_database d WHERE d.datallowconn = true;
```

- c. **Reg* data types** – Remove all uses of the `reg*` data types before attempting an upgrade, because these data types contain information that cannot be persisted with `pg_upgrade`. Uses of `reg*` data types cannot be upgraded, except for `regtype` and `regclass`. Remove all usages before attempting an upgrade.

You can use the following query to verify that there are no uses of unsupported `reg*` data types in each database:

```
SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
pg_catalog.pg_attribute a
WHERE c.oid = a.attrelid
AND NOT a.attisdropped
AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
'pg_catalog.regprocedure'::pg_catalog.regtype,
'pg_catalog.regoper'::pg_catalog.regtype,
'pg_catalog.regoperator'::pg_catalog.regtype,
'pg_catalog.regconfig'::pg_catalog.regtype,
'pg_catalog.regdictionary'::pg_catalog.regtype)
AND c.relnamespace = n.oid
AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

Perform a `VACUUM` operation before upgrading your instance. The `pg_upgrade` utility vacuums each database when you upgrade to a different major version. If you haven't performed a `VACUUM` operation, the upgrade process can take much longer, causing increased downtime for your RDS instance.

3. Perform a dry run of your major version upgrade. We highly recommend testing major version upgrade on a duplicate of your production database before attempting it on your production database. To create a duplicate test instance, you can either restore your database from a recent snapshot or point-in-time restore your database to its latest restorable time. After you have completed the major version upgrade, consider testing your application on the upgraded database with a similar workload in order to verify that everything works as expected. After the upgrade is verified, you can delete this test instance.
4. We recommend that you perform a backup before performing the major version upgrade so that you have a known restore point for your database. Note that we create a DB snapshot of your DB instance before and after upgrading.

5. Upgrade your production instance. If the dry-run major version upgrade was successful, you should now be able to upgrade your production database with confidence.

You can use Amazon RDS to view two logs that the `pg_upgrade` utility produces: `pg_upgrade_internal.log` and `pg_upgrade_server.log`. Amazon RDS appends a timestamp to the file name for these logs. You can view these logs as you can any other log.

You cannot perform a point-in-time restore of your instance to a point in time during the upgrade process. During the upgrade process, RDS takes an automatic backup of the instance after the upgrade has been performed. You can perform a point-in-time restore to times before the upgrade began and after the automatic backup of your instance has completed.

The `public` and `template1` databases and the `public` schema in every database on the instance are renamed during the major version upgrade. These objects will appear in the logs with their original name and a random string appended. The string is appended so that custom settings such as the `locale` and `owner` are preserved during the major version upgrade. Once the upgrade completes, the objects are renamed back to their original names.

Note

After you have completed the upgrade, you should run the `ANALYZE` operation to refresh the `pg_statistic` table.

Automatic Minor Version Upgrades for PostgreSQL

Minor version upgrades occur automatically if a minor upgrade has been tested and approved by Amazon RDS and you enable the **Auto minor version upgrade** option. In all other cases, you must modify the DB instance manually to perform a minor version upgrade. If you enable the **Auto minor version upgrade** option when creating or modifying a DB instance, you can have your instance automatically upgraded after the new version is tested and approved by Amazon RDS.

If your PostgreSQL DB instance is using read replication, you must upgrade all of the Read Replicas before upgrading the source instance. If the DB instance is in a Multi-AZ deployment, both the primary and standby replicas are upgraded, and the instance might not be available until the upgrade is complete.

Upgrading a PostgreSQL DB Instance Manually

For information about manually or automatically upgrading a PostgreSQL DB instance, see [Upgrading a DB Instance Engine Version \(p. 123\)](#).

Working with PostgreSQL Read Replicas

You usually use Read Replicas to configure replication between Amazon RDS DB instances. For general information about Read Replicas, see [Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances \(p. 143\)](#).

This section contains specific information about working with Read Replicas on PostgreSQL.

Topics

- [Read Replica Configuration with PostgreSQL \(p. 992\)](#)
- [Monitoring PostgreSQL Read Replicas \(p. 993\)](#)
- [Read Replica Limitations with PostgreSQL \(p. 993\)](#)
- [Replication Interruptions with PostgreSQL Read Replicas \(p. 993\)](#)
- [Troubleshooting a PostgreSQL Read Replica Problem \(p. 993\)](#)

Read Replica Configuration with PostgreSQL

Amazon RDS PostgreSQL 9.3.5 and later uses PostgreSQL native streaming replication to create a read-only copy of a source (a "master" in PostgreSQL terms) DB instance. This Read Replica (a "standby" in PostgreSQL terms) DB instance is an asynchronously created physical replication of the master DB instance. It's created by a special connection that transmits write ahead log (WAL) data between the source DB instance and the Read Replica where PostgreSQL asynchronously streams database changes as they are made.

PostgreSQL uses a "replication" role to perform streaming replication. The role is privileged, but can't be used to modify any data. PostgreSQL uses a single process for handling replication.

Before a DB instance can serve as a source DB instance, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0.

Creating a PostgreSQL Read Replica doesn't require an outage for the master DB instance. Amazon RDS sets the necessary parameters and permissions for the source DB instance and the Read Replica without any service interruption. A snapshot is taken of the source DB instance, and this snapshot becomes the Read Replica. No outage occurs when you delete a Read Replica.

You can create up to five Read Replicas from one source DB instance. For replication to operate effectively, each Read Replica should have the same amount of compute and storage resources as the source DB instance. If you scale the source DB instance, you should also scale the Read Replicas.

Amazon RDS overrides any incompatible parameters on a Read Replica if it prevents the Read Replica from starting. For example, suppose that the `max_connections` parameter value is higher on the source DB instance than on the Read Replica. In that case, Amazon RDS updates the parameter on the Read Replica to be the same value as that on the source DB instance.

PostgreSQL DB instances use a secure connection that you can encrypt by setting the `ssl` parameter to 1 for both the source and the Read Replica instances.

You can create a Read Replica from either single-AZ or Multi-AZ DB instance deployments. You use Multi-AZ deployments to improve the durability and availability of critical data, but you can't use the Multi-AZ secondary to serve read-only queries. Instead, you can create Read Replicas from high-traffic Multi-AZ DB instances to offload read-only queries. If the source instance of a Multi-AZ deployment fails over to the secondary, any associated Read Replicas automatically switch to use the secondary (now primary) as their replication source. For more information, see [High Availability \(Multi-AZ\) for Amazon RDS \(p. 109\)](#).

You can create a Read Replica as a Multi-AZ DB instance. Amazon RDS creates a standby of your replica in another Availability Zone for failover support for the replica. Creating your Read Replica as a Multi-AZ DB instance is independent of whether the source database is a Multi-AZ DB instance.

If you use the [postgres_fdw](#) extension to access data from a remote server, the Read Replica will also have access to the remote server. For more information about using `postgres_fdw`, see [Accessing External Data with the postgres_fdw Extension \(p. 1024\)](#).

Monitoring PostgreSQL Read Replicas

For PostgreSQL Read Replicas, you can monitor replication lag in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric. The `ReplicaLag` metric reports the value of `SELECT extract(epoch from now() - pg_last_xact_replay_timestamp()) AS slave_lag.`

Read Replica Limitations with PostgreSQL

The following are limitations for PostgreSQL Read Replicas:

- Each PostgreSQL Read Replica is read-only and can't be made a writable Read Replica.
- You can't create a Read Replica from another Read Replica (that is, you can't create cascading Read Replicas).
- You can promote a PostgreSQL Read Replica to be a new source DB instance. However, the Read Replica doesn't become the new source DB instance automatically. The Read Replica, when promoted, stops receiving WAL communications and is no longer a read-only instance. You must set up any replication you intend to have going forward because the promoted Read Replica is now a new source DB instance.
- A PostgreSQL Read Replica reports a replication lag of up to five minutes if there are no user transactions occurring on the source DB instance.

Replication Interruptions with PostgreSQL Read Replicas

In several situations, a PostgreSQL source DB instance can unintentionally break replication with a Read Replica. These situations include the following:

- The `max_wal_senders` parameter is set too low to provide enough data to the number of Read Replicas. This situation causes replication to stop.
- The PostgreSQL parameter `wal_keep_segments` dictates how many WAL files are kept to provide data to the Read Replicas. The parameter value specifies the number of logs to keep. If you set the parameter value too low, you can cause a Read Replica to fall so far behind that streaming replication stops. In this case, Amazon RDS reports a replication error and begins recovery on the Read Replica by replaying the source DB instance's archived WAL logs. This recovery process continues until the Read Replica has caught up enough to continue streaming replication. For more information, see [Troubleshooting a PostgreSQL Read Replica Problem \(p. 993\)](#).
- A PostgreSQL Read Replica requires a reboot if the source DB instance endpoint changes.

When the WAL stream that provides data to a Read Replica is broken, PostgreSQL switches into recovery mode to restore the Read Replica by using archived WAL files. When this process is complete, PostgreSQL attempts to re-establish streaming replication.

Troubleshooting a PostgreSQL Read Replica Problem

PostgreSQL uses replication slots for cross-region replication, so the process for troubleshooting same-region replication problems and cross-region replication problems is different.

Troubleshooting PostgreSQL Read Replica Problems Within an AWS Region

The PostgreSQL parameter, `wal_keep_segments`, dictates how many Write Ahead Log (WAL) files are kept to provide data to the Read Replicas. The parameter value specifies the number of logs to keep. If you set the parameter value too low, you can cause a Read Replica to fall so far behind that streaming replication stops. In this case, Amazon RDS reports a replication error and begins recovery on the Read Replica by replaying the source DB instance's archived WAL logs. This recovery process continues until the Read Replica has caught up enough to continue streaming replication.

The PostgreSQL log on the Read Replica shows when Amazon RDS is recovering a Read Replica that is this state by replaying archived WAL files.

```
2014-11-07 19:01:10 UTC:@:[23180]:DEBUG: switched WAL source from archive to stream
after
failure 2014-11-07 19:01:10 UTC:@:[11575]:LOG: started streaming WAL from primary
at
1A/D3000000 on timeline 1 2014-11-07 19:01:10 UTC:@:[11575]:FATAL: could not
receive
data from WAL stream: ERROR: requested WAL segment 00000001000001A00000D3 has
already been
removed 2014-11-07 19:01:10 UTC:@:[23180]:DEBUG: could not restore file
"00000002.history" from archive: return code 0 2014-11-07 19:01:15
UTC:@:[23180]:DEBUG: switched WAL source from stream to archive after failure
recovering 00000001000001A00000D3 2014-11-07 19:01:16 UTC:@:[23180]:LOG: restored
log file "00000001000001A00000D3"
from archive
```

After a certain amount of time, Amazon RDS replays enough archived WAL files on the replica to catch up and allow the Read Replica to begin streaming again. At this point, PostgreSQL resumes streaming and writes a similar line to the following to the log file.

```
2014-11-07 19:41:36 UTC:@:[24714]:LOG: started streaming WAL from primary at 1B/
B6000000
on timeline 1
```

You can determine how many WAL files you should keep by looking at the checkpoint information in the log. The PostgreSQL log shows the following information at each checkpoint. By looking at the "# recycled" transaction log files of these log statements, you can understand how many transaction files will be recycled during a time range and use this information to tune the `wal_keep_segments` parameter.

```
2014-11-07 19:59:35 UTC:@:[26820]:LOG: checkpoint complete: wrote 376 buffers (0.2%); 0
transaction log file(s) added, 0 removed, 1 recycled; write=35.681 s, sync=0.013 s,
total=35.703 s; sync files=10, longest=0.013 s, average=0.001 s
```

For example, suppose that the PostgreSQL log shows that 35 files are recycled from the "checkpoint completed" log statements within a 5-minute time frame. In that case, we know that with this usage pattern a Read Replica relies on 35 transaction files in five minutes. A Read Replica can't survive five minutes in a nonstreaming state if the source DB instance is set to the default `wal_keep_segments` parameter value of 32.

Troubleshooting PostgreSQL Read Replica Problems Across AWS Regions

PostgreSQL (versions 9.4.7 and 9.5.2 exclusively) uses physical replication slots to manage Write Ahead Log (WAL) retention on the source DB instance. For each cross-region Read Replica instance, Amazon

RDS creates and associates a physical replication slot. You can use two Amazon CloudWatch metrics, Oldest Replication Slot Lag and Transaction Logs Disk Usage, to see how far behind the most lagging replica is in terms of WAL data received and to see how much storage is being used for WAL data. The Transaction Logs Disk Usage value can substantially increase when a cross-region Read Replica is lagging significantly.

If the workload on your DB instance generates a large amount of WAL data, you might need to change the DB instance class of your source DB instance and Read Replica. In that case, you change it to one with high (10 Gbps) network performance for the replica to keep up. The Amazon CloudWatch metric `Transaction Logs Generation` can help you understand the rate at which your workload is generating WAL data.

To determine the status of a cross-region Read Replica, you can query `pg_replication_slots` on the source instance, as in the following example:

```

postgres=# select * from pg_replication_slots;
          slot_name           | plugin | slot_type | datoid | database |
active | active_pid | xmin | catalog_xmin | restart_lsn
-----+-----+-----+-----+-----+
rds_us_east_1_db_uzwlh0lddgpb1ksce6hgw4nkte |       | physical |       |       | t
| 12598 |       |           | 4E/95000060
(1 row)

```

Importing Data into PostgreSQL on Amazon RDS

Suppose that you have an existing PostgreSQL deployment that you want to move to Amazon RDS. The complexity of your task depends on the size of your database and the types of database objects that you're transferring. For example, consider a database that contains datasets on the order of gigabytes, along with stored procedures and triggers. Such a database is going to be more complicated than a simple database with only a few megabytes of test data and no triggers or stored procedures.

We recommend that you use native PostgreSQL database migration tools under the following conditions:

- You have a homogeneous migration, where you are migrating from a database with the same database engine as the target database.
- You are migrating an entire database.
- The native tools allow you to migrate your system with minimal downtime.

In most other cases, performing a database migration using AWS Database Migration Service (AWS DMS) is the best approach. AWS DMS can migrate databases without downtime and, for many database engines, continue ongoing replication until you are ready to switch over to the target database. You can migrate to either the same database engine or a different database engine using AWS DMS. If you are migrating to a different database engine than your source database, you can use the AWS Schema Conversion Tool (AWS SCT). You use AWS SCT to migrate schema objects that are not migrated by AWS DMS. For more information about AWS DMS, see [What Is AWS Database Migration Service?](#)

Modify your DB parameter group to include the following settings *for your import only*. You should test the parameter settings to find the most efficient settings for your DB instance size. You also need to revert back to production values for these parameters after your import completes.

Modify your DB instance settings to the following:

- Disable DB instance backups (set backup_retention to 0).
- Disable Multi-AZ.

Modify your DB parameter group to include the following settings. You should only use these settings when importing data. You should test the parameter settings to find the most efficient settings for your DB instance size. You also need to revert back to production values for these parameters after your import completes.

Parameter	Recommended Value When Importing	Description
<code>maintenance_work_mem</code>	524288, 1048576, 2097152 or 4194304 (in KB). These settings are comparable to 512 MB, 1 GB, 2 GB, and 4 GB.	The value for this setting depends on the size of your host. This parameter is used during CREATE INDEX statements and each parallel command can use this much memory. Calculate the best value so that you don't set this value so high that you run out of memory.
<code>checkpoint_segments</code>	256	The value for this setting consumes more disk space, but gives you less contention on your WAL logs. For PostgreSQL versions 9.5.x and 9.6.x, this value would be <code>max_wal_size</code> .
<code>checkpoint_timeout</code>	1800	The value for this setting allows for less frequent WAL rotation.

Parameter	Recommended Value When Importing	Description
synchronous_commit	Off	Disable this setting to speed up writes. Turning this parameter off can increase the risk of data loss in the event of a server crash (do not turn off FSYNC)
wal_buffers	8192	This value is in 8 KB units. This again helps your WAL generation speed
autovacuum	Off	Disable the PostgreSQL auto vacuum parameter while you are loading data so that it doesn't use resources

Use the `pg_dump -Fc` (compressed) or `pg_restore -j` (parallel) commands with these settings.

Note

The PostgreSQL command `pg_dumpall` requires super_user permissions that are not granted when you create a DB instance, so it cannot be used for importing data.

Importing a PostgreSQL Database from an Amazon EC2 Instance

If you have data in a PostgreSQL server on an Amazon EC2 instance and want to move it to a PostgreSQL DB instance, you can use the following process. The following list shows the steps to take. Each step is discussed in more detail in the following sections.

1. Create a file using `pg_dump` that contains the data to be loaded
2. Create the target DB instance
3. Use `psql` to create the database on the DB instance and load the data
4. Create a DB snapshot of the DB instance

Step 1: Create a File Using pg_dump That Contains the Data to Load

The `pg_dump` utility uses the `COPY` command to create a schema and data dump of a PostgreSQL database. The dump script generated by `pg_dump` loads data into a database with the same name and recreates the tables, indexes, and foreign keys. You can use the `pg_restore` command and the `-d` parameter to restore the data to a database with a different name.

Before you create the data dump, you should query the tables to be dumped to get a row count so you can confirm the count on the target DB instance.

The following command creates a dump file called `mydb2dump.sql` for a database called `mydb2`.

```
prompt>pg_dump dbname=mydb2 -f mydb2dump.sql
```

Step 2: Create the Target DB Instance

Create the target PostgreSQL DB instance using either the Amazon RDS console, AWS CLI, or API. Create the instance with the backup retention setting set to 0 and disable Multi-AZ. Doing so allows faster data import. You must create a database on the instance before you can dump the data. The database can

have the same name as the database that is contained the dumped data. Alternatively, you can create a database with a different name. In this case, you use the `pg_restore` command and the `-d` parameter to restore the data into the newly named database.

For example, the following commands can be used to dump, restore, and rename a database.

```
pg_dump -Fc -v -h [endpoint of instance] -U [master username] [database] > [database].dump
createdb [new database name]
pg_restore -v -h [endpoint of instance] -U [master username] -d [new database
name] [database].dump
```

Step 3: Use `psql` to Create the Database on the DB Instance and Load Data

You can use the same connection you used to execute the `pg_dump` command to connect to the target DB instance and recreate the database. Using `psql`, you can use the master user name and master password to create the database on the DB instance

The following example uses `psql` and a dump file named `mydb2dump.sql` to create a database called `mydb2` on a PostgreSQL DB instance called `mypginstance`:

For Linux, OS X, or Unix:

```
psql \
-f mydb2dump.sql \
--host mypginstance.c6c8mntzhgv0.us-west-2.rds.amazonaws.com \
--port 8199 \
--username myawsuser \
--password password \
--dbname mydb2
```

For Windows:

```
psql ^
-f mydb2dump.sql ^
--host mypginstance.c6c8mntzhgv0.us-west-2.rds.amazonaws.com ^
--port 8199 ^
--username myawsuser ^
--password password ^
--dbname mydb2
```

Step 4: Create a DB Snapshot of the DB Instance

Once you have verified that the data was loaded into your DB instance, we recommend that you create a DB snapshot of the target PostgreSQL DB instance. DB snapshots are complete backups of your DB instance that can be used to restore your DB instance to a known state. A DB snapshot taken immediately after the load protects you from having to load the data again in case of a mishap. You can also use such a snapshot to seed new DB instances. For information about creating a DB snapshot, see [Creating a DB Snapshot \(p. 216\)](#).

Using the `\copy` Command to Import Data to a Table on a PostgreSQL DB Instance

You can run the `\copy` command from the `psql` prompt to import data into a table on a PostgreSQL DB instance. The table must already exist on the DB instance. For more information on the `\copy` command, see the [PostgreSQL documentation](#).

Note

The `\copy` command doesn't provide confirmation of actions, such as a count of rows inserted. PostgreSQL does provide error messages if the copy command fails due to an error.

Create a .csv file from the data in the source table, log on to the target database on the PostgreSQL instance using `psql`, and then run the following command. This example uses *source-table* as the source table name, *source-table.csv* as the .csv file, and *target-db* as the target database:

```
target-db=> \copy source-table from 'source-table.csv' with DELIMITER ',';
```

You can also run the following command from your client computer command prompt. This example uses *source-table* as the source table name, *source-table.csv* as the .csv file, and *target-db* as the target database:

For Linux, OS X, or Unix:

```
$psql target-db \
-U <admin user> \
-p <port> \
-h <DB instance name> \
-c "\copy source-table from 'source-table.csv' with DELIMITER ','"
```

For Windows:

```
$psql target-db ^
-U <admin user> ^
-p <port> ^
-h <DB instance name> ^
-c "\copy source-table from 'source-table.csv' with DELIMITER ','"
```

Common DBA Tasks for PostgreSQL

This section describes the Amazon RDS implementations of some common DBA tasks for DB instances running the PostgreSQL database engine. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.

For information about working with PostgreSQL log files on Amazon RDS, see [PostgreSQL Database Log Files \(p. 334\)](#).

Topics

- [Creating Roles \(p. 1000\)](#)
- [Managing PostgreSQL Database Access \(p. 1001\)](#)
- [Working with PostgreSQL Parameters \(p. 1001\)](#)
- [Working with PostgreSQL Autovacuum on Amazon RDS \(p. 1009\)](#)
- [Audit Logging for a PostgreSQL DB Instance \(p. 1017\)](#)
- [Working with the pgaudit Extension \(p. 1018\)](#)
- [Working with the pg_repack Extension \(p. 1019\)](#)
- [Working with PostGIS \(p. 1020\)](#)
- [Using pgBadger for Log Analysis with PostgreSQL \(p. 1022\)](#)
- [Viewing the Contents of pg_config \(p. 1022\)](#)
- [Working with the orafce Extension \(p. 1023\)](#)
- [Accessing External Data with the postgres_fdw Extension \(p. 1024\)](#)
- [Using a Custom DNS Server for Outbound Network Access \(p. 1025\)](#)
- [Restricting Password Management \(p. 1026\)](#)

Creating Roles

When you create a DB instance, the master user system account that you create is assigned to the `rds_superuser` role. The `rds_superuser` role is a predefined Amazon RDS role similar to the PostgreSQL superuser role (customarily named `postgres` in local instances), but with some restrictions. As with the PostgreSQL superuser role, the `rds_superuser` role has the most privileges for your DB instance. You should not assign this role to users unless they need the most access to the DB instance.

The `rds_superuser` role can do the following:

- Add extensions that are available for use with Amazon RDS. For more information, see [Supported PostgreSQL Features \(p. 1063\)](#) and the [PostgreSQL documentation](#).
- Manage tablespaces, including creating and deleting them. For more information, see the [Tablespaces](#) section in the PostgreSQL documentation.
- View all users not assigned the `rds_superuser` role using the `pg_stat_activity` command and kill their connections using the `pg_terminate_backend` and `pg_cancel_backend` commands.
- Grant and revoke the `rds_replication` role for all roles that are not the `rds_superuser` role. For more information, see the [GRANT](#) section in the PostgreSQL documentation.

The following example shows how to create a user and then grant the user the `rds_superuser` role. User-defined roles, such as `rds_superuser`, have to be granted.

```
create role testuser with password 'testuser' login;
CREATE ROLE
grant rds_superuser to testuser;
```

GRANT ROLE

Managing PostgreSQL Database Access

In Amazon RDS for PostgreSQL, you can manage which users have privileges to connect to which databases. In other PostgreSQL environments, you sometimes perform this kind of management by modifying the `pg_hba.conf` file. In Amazon RDS, you can use database grants instead.

New databases in PostgreSQL are always created with a default set of privileges. The default privileges allow `PUBLIC` (all users) to connect to the database and to create temporary tables while connected.

To control which users are allowed to connect to a given database in Amazon RDS, first revoke the default `PUBLIC` privileges. Then grant back the privileges on a more granular basis. The following example code shows how.

```
psql> revoke all on database <database-name> from public;
psql> grant connect, temporary on database <database-name> to <user/role name>;
```

For more information about privileges in PostgreSQL databases, see the [GRANT](#) command in the PostgreSQL documentation.

Working with PostgreSQL Parameters

PostgreSQL parameters that you set for a local PostgreSQL instance in the `postgresql.conf` file are maintained in the DB parameter group for your DB instance. If you create a DB instance using the default parameter group, the parameter settings are in the parameter group called `default.postgres9.6`.

When you create a DB instance, the parameters in the associated DB parameter group are loaded. You can modify parameter values by changing values in the parameter group. You can also change parameter values, if you have the security privileges to do so, by using the `ALTER DATABASE`, `ALTER ROLE`, and `SET` commands. You can't use the command line `postgres` command or the `env PGOPTIONS` command, because you have no access to the host.

Keeping track of PostgreSQL parameter settings can occasionally be difficult. Use the following command to list current parameter settings and the default value.

```
select name, setting, boot_val, reset_val, unit
from pg_settings
order by name;
```

For an explanation of the output values, see the [pg_settings](#) topic in the PostgreSQL documentation.

If you set the memory settings too large for `max_connections`, `shared_buffers`, or `effective_cache_size`, you will prevent the PostgreSQL instance from starting up. Some parameters use units that you might not be familiar with; for example, `shared_buffers` sets the number of 8-KB shared memory buffers used by the server.

The following error is written to the `postgres.log` file when the instance is attempting to start up, but incorrect parameter settings are preventing it from starting.

```
2013-09-18 21:13:15 UTC::@[8097]:FATAL:  could not map anonymous shared
memory: Cannot allocate memory
2013-09-18 21:13:15 UTC::@[8097]:HINT:  This error usually means that
PostgreSQL's request for a shared memory segment exceeded available memory or
swap space. To reduce the request size (currently 3514134274048 bytes), reduce
PostgreSQL's shared memory usage, perhaps by reducing shared_buffers or
max_connections.
```

There are two types of PostgreSQL parameters, static and dynamic. Static parameters require that the DB instance be rebooted before they are applied. Dynamic parameters can be applied immediately. The following table shows parameters that you can modify for a PostgreSQL DB instance and each parameter's type.

Parameter Name	Apply_Type	Description
application_name	Dynamic	Sets the application name to be reported in statistics and logs.
array_nulls	Dynamic	Enables input of NULL elements in arrays.
authentication_timeout	Dynamic	Sets the maximum allowed time to complete client authentication.
autovacuum	Dynamic	Starts the autovacuum subprocess.
autovacuum_analyze_scale_factor	Dynamic	Number of tuple inserts, updates, or deletes before analyze as a fraction of reltuples.
autovacuum_analyze_threshold	Dynamic	Minimum number of tuple inserts, updates, or deletes before analyze.
autovacuum_naptime	Dynamic	Time to sleep between autovacuum runs.
autovacuum_vacuum_cost_delay	Dynamic	Vacuum cost delay, in milliseconds, for autovacuum.
autovacuum_vacuum_cost_limit	Dynamic	Vacuum cost amount available before napping, for autovacuum.
autovacuum_vacuum_scale_factor	Dynamic	Number of tuple updates or deletes before vacuum as a fraction of reltuples.
autovacuum_vacuum_threshold	Dynamic	Minimum number of tuple updates or deletes before vacuum.
backslash_quote	Dynamic	Sets whether a backslash (\) is allowed in string literals.
bgwriter_delay	Dynamic	Background writer sleep time between rounds.
bgwriter_lru_maxpages	Dynamic	Background writer maximum number of LRU pages to flush per round.
bgwriter_lru_multiplier	Dynamic	Multiple of the average buffer usage to free per round.
bytea_output	Dynamic	Sets the output format for bytes.
check_function_bodies	Dynamic	Checks function bodies during CREATE FUNCTION.
checkpoint_completion_target	Dynamic	Time spent flushing dirty buffers during checkpoint, as a fraction of the checkpoint interval.
checkpoint_segments	Dynamic	Sets the maximum distance in log segments between automatic WAL checkpoints.
checkpoint_timeout	Dynamic	Sets the maximum time between automatic WAL checkpoints.

Parameter Name	Apply_Type	Description
checkpoint_warning	Dynamic	Enables warnings if checkpoint segments are filled more frequently than this.
client_encoding	Dynamic	Sets the client's character set encoding.
client_min_messages	Dynamic	Sets the message levels that are sent to the client.
commit_delay	Dynamic	Sets the delay in microseconds between transaction commit and flushing WAL to disk.
commit_siblings	Dynamic	Sets the minimum concurrent open transactions before performing commit_delay.
constraint_exclusion	Dynamic	Enables the planner to use constraints to optimize queries.
cpu_index_tuple_cost	Dynamic	Sets the planner's estimate of the cost of processing each index entry during an index scan.
cpu_operator_cost	Dynamic	Sets the planner's estimate of the cost of processing each operator or function call.
cpu_tuple_cost	Dynamic	Sets the planner's estimate of the cost of processing each tuple (row).
cursor_tuple_fraction	Dynamic	Sets the planner's estimate of the fraction of a cursor's rows that will be retrieved.
datestyle	Dynamic	Sets the display format for date and time values.
deadlock_timeout	Dynamic	Sets the time to wait on a lock before checking for deadlock.
debug_pretty_print	Dynamic	Indents parse and plan tree displays.
debug_print_parse	Dynamic	Logs each query's parse tree.
debug_print_plan	Dynamic	Logs each query's execution plan.
debug_print_rewritten	Dynamic	Logs each query's rewritten parse tree.
default_statistics_target	Dynamic	Sets the default statistics target.
default_tablespace	Dynamic	Sets the default tablespace to create tables and indexes in.
default_transaction_deferrable	Dynamic	Sets the default deferrable status of new transactions.
default_transaction_isolation	Dynamic	Sets the transaction isolation level of each new transaction.
default_transaction_read_only	Dynamic	Sets the default read-only status of new transactions.
default_with_oids	Dynamic	Creates new tables with OIDs by default.
effective_cache_size	Dynamic	Sets the planner's assumption about the size of the disk cache.

Parameter Name	Apply_Type	Description
<code>effective_io_concurrency</code>	Dynamic	Number of simultaneous requests that can be handled efficiently by the disk subsystem.
<code>enable_bitmapscan</code>	Dynamic	Enables the planner's use of bitmap-scan plans.
<code>enable_hashagg</code>	Dynamic	Enables the planner's use of hashed aggregation plans.
<code>enable_hashjoin</code>	Dynamic	Enables the planner's use of hash join plans.
<code>enable_indexscan</code>	Dynamic	Enables the planner's use of index-scan plans.
<code>enable_material</code>	Dynamic	Enables the planner's use of materialization.
<code>enable_mergejoin</code>	Dynamic	Enables the planner's use of merge join plans.
<code>enable_nestloop</code>	Dynamic	Enables the planner's use of nested-loop join plans.
<code>enable_seqscan</code>	Dynamic	Enables the planner's use of sequential-scan plans.
<code>enable_sort</code>	Dynamic	Enables the planner's use of explicit sort steps.
<code>enable_tidscan</code>	Dynamic	Enables the planner's use of TID scan plans.
<code>escape_string_warning</code>	Dynamic	Warns about backslash (\) escapes in ordinary string literals.
<code>extra_float_digits</code>	Dynamic	Sets the number of digits displayed for floating-point values.
<code>fromCollapse_limit</code>	Dynamic	Sets the FROM-list size beyond which subqueries are not collapsed.
<code>fsync</code>	Dynamic	Forces synchronization of updates to disk.
<code>full_page_writes</code>	Dynamic	Writes full pages to WAL when first modified after a checkpoint.
<code>geqo</code>	Dynamic	Enables genetic query optimization.
<code>geqo_effort</code>	Dynamic	GEQO: effort is used to set the default for other GEQO parameters.
<code>geqo_generations</code>	Dynamic	GEQO: number of iterations of the algorithm.
<code>geqo_pool_size</code>	Dynamic	GEQO: number of individuals in the population.
<code>geqo_seed</code>	Dynamic	GEQO: seed for random path selection.
<code>geqo_selection_bias</code>	Dynamic	GEQO: selective pressure within the population.
<code>geqo_threshold</code>	Dynamic	Sets the threshold of FROM items beyond which GEQO is used.
<code>gin_fuzzy_search_limit</code>	Dynamic	Sets the maximum allowed result for exact search by GIN.
<code>hot_standby_feedback</code>	Dynamic	Determines whether a hot standby sends feedback messages to the primary or upstream standby.

Parameter Name	Apply_Type	Description
intervalstyle	Dynamic	Sets the display format for interval values.
joinCollapse_limit	Dynamic	Sets the FROM-list size beyond which JOIN constructs are not flattened.
lc_messages	Dynamic	Sets the language in which messages are displayed.
lc_monetary	Dynamic	Sets the locale for formatting monetary amounts.
lc_numeric	Dynamic	Sets the locale for formatting numbers.
lc_time	Dynamic	Sets the locale for formatting date and time values.
log_autovacuum_min_duration	Dynamic	Sets the minimum execution time above which autovacuum actions will be logged.
log_checkpoints	Dynamic	Logs each checkpoint.
log_connections	Dynamic	Logs each successful connection.
log_disconnections	Dynamic	Logs end of a session, including duration.
log_duration	Dynamic	Logs the duration of each completed SQL statement.
log_error_verbosity	Dynamic	Sets the verbosity of logged messages.
log_executor_stats	Dynamic	Writes executor performance statistics to the server log.
log_filename	Dynamic	Sets the file name pattern for log files.
log_hostname	Dynamic	Logs the host name in the connection logs.
log_lock_waits	Dynamic	Logs long lock waits.
log_min_duration_statement	Dynamic	Sets the minimum execution time above which statements will be logged.
log_min_error_statement	Dynamic	Causes all statements generating an error at or above this level to be logged.
log_min_messages	Dynamic	Sets the message levels that are logged.
log_parser_stats	Dynamic	Writes parser performance statistics to the server log.
log_planner_stats	Dynamic	Writes planner performance statistics to the server log.
log_rotation_age	Dynamic	Automatic log file rotation will occur after N minutes.
log_rotation_size	Dynamic	Automatic log file rotation will occur after N kilobytes.
log_statement	Dynamic	Sets the type of statements logged.
log_statement_stats	Dynamic	Writes cumulative performance statistics to the server log.
log_temp_files	Dynamic	Logs the use of temporary files larger than this number of kilobytes.

Parameter Name	Apply_Type	Description
<code>maintenance_work_mem</code>	Dynamic	Sets the maximum memory to be used for maintenance operations.
<code>max_stack_depth</code>	Dynamic	Sets the maximum stack depth, in kilobytes.
<code>max_standby_archive_delay</code>	Dynamic	Sets the maximum delay before canceling queries when a hot standby server is processing archived WAL data.
<code>max_standby_streaming_delay</code>	Dynamic	Sets the maximum delay before canceling queries when a hot standby server is processing streamed WAL data.
<code>quote_all_identifiers</code>	Dynamic	Adds quotes ("") to all identifiers when generating SQL fragments.
<code>random_page_cost</code>	Dynamic	Sets the planner's estimate of the cost of a non-sequentially fetched disk page.
<code>rds.log_retention_period</code>	Dynamic	Amazon RDS will delete PostgreSQL logs that are older than N minutes.
<code>rds.restrict_password_comment</code>	Static	Restricts who can manage passwords to users with the <code>rds_password</code> role. Set this parameter to 1 to enable password restriction. The default is 0.
<code>search_path</code>	Dynamic	Sets the schema search order for names that are not schema-qualified.
<code>seq_page_cost</code>	Dynamic	Sets the planner's estimate of the cost of a sequentially fetched disk page.
<code>session_replication_role</code>	Dynamic	Sets the sessions behavior for triggers and rewrite rules.
<code>sql_inheritance</code>	Dynamic	Causes subtables to be included by default in various commands.
<code>ssl_renegotiation_limit</code>	Dynamic	Sets the amount of traffic to send and receive before renegotiating the encryption keys.
<code>standard_conforming_strings</code>	Dynamic	Causes ... strings to treat backslashes literally.
<code>statement_timeout</code>	Dynamic	Sets the maximum allowed duration of any statement.
<code>synchronize_seqscans</code>	Dynamic	Enables synchronized sequential scans.
<code>synchronous_commit</code>	Dynamic	Sets the current transactions synchronization level.
<code>tcp_keepalives_count</code>	Dynamic	Maximum number of TCP keepalive retransmits.
<code>tcp_keepalives_idle</code>	Dynamic	Time between issuing TCP keepalives.
<code>tcp_keepalives_interval</code>	Dynamic	Time between TCP keepalive retransmits.
<code>temp_buffers</code>	Dynamic	Sets the maximum number of temporary buffers used by each session.

Parameter Name	Apply_Type	Description
temp_tablespaces	Dynamic	Sets the tablespaces to use for temporary tables and sort files.
timezone	Dynamic	Sets the time zone for displaying and interpreting time stamps.
track_activities	Dynamic	Collects information about executing commands.
track_counts	Dynamic	Collects statistics on database activity.
track_functions	Dynamic	Collects function-level statistics on database activity.
track_io_timing	Dynamic	Collects timing statistics on database I/O activity.
transaction_deferrable	Dynamic	Indicates whether to defer a read-only serializable transaction until it can be executed with no possible serialization failures.
transaction_isolation	Dynamic	Sets the current transactions isolation level.
transaction_read_only	Dynamic	Sets the current transactions read-only status.
transform_null_equals	Dynamic	Treats expr=NULL as expr IS NULL.
update_process_title	Dynamic	Updates the process title to show the active SQL command.
vacuum_cost_delay	Dynamic	Vacuum cost delay in milliseconds.
vacuum_cost_limit	Dynamic	Vacuum cost amount available before napping.
vacuum_cost_page_dirty	Dynamic	Vacuum cost for a page dirtied by vacuum.
vacuum_cost_page_hit	Dynamic	Vacuum cost for a page found in the buffer cache.
vacuum_cost_page_miss	Dynamic	Vacuum cost for a page not found in the buffer cache.
vacuum_defer_cleanup_age	Dynamic	Number of transactions by which vacuum and hot cleanup should be deferred, if any.
vacuum_freeze_min_age	Dynamic	Minimum age at which vacuum should freeze a table row.
vacuum_freeze_table_age	Dynamic	Age at which vacuum should scan a whole table to freeze tuples.
wal_writer_delay	Dynamic	WAL writer sleep time between WAL flushes.
work_mem	Dynamic	Sets the maximum memory to be used for query workspaces.
xmlbinary	Dynamic	Sets how binary values are to be encoded in XML.
xmloption	Dynamic	Sets whether XML data in implicit parsing and serialization operations is to be considered as documents or content fragments.
autovacuum_freeze_max_age	Static	Age at which to autovacuum a table to prevent transaction ID wraparound.

Parameter Name	Apply_Type	Description
autovacuum_max_workers	Static	Sets the maximum number of simultaneously running autovacuum worker processes.
max_connections	Static	Sets the maximum number of concurrent connections.
max_files_per_process	Static	Sets the maximum number of simultaneously open files for each server process.
max_locks_per_transaction	Static	Sets the maximum number of locks per transaction.
max_pred_locks_per_transaction	Static	Sets the maximum number of predicate locks per transaction.
max_prepared_transactions	Static	Sets the maximum number of simultaneously prepared transactions.
shared_buffers	Static	Sets the number of shared memory buffers used by the server.
ssl	Static	Enables SSL connections.
temp_file_limit	Static	Sets the maximum size in KB to which the temporary files can grow.
track_activity_query_size	Static	Sets the size reserved for pg_stat_activity.current_query, in bytes.
wal_buffers	Static	Sets the number of disk-page buffers in shared memory for WAL.

Amazon RDS uses the default PostgreSQL units for all parameters. The following table shows the PostgreSQL default unit and value for each parameter.

Parameter Name	Unit
effective_cache_size	8 KB
segment_size	8 KB
shared_buffers	8 KB
temp_buffers	8 KB
wal_buffers	8 KB
wal_segment_size	8 KB
log_rotation_size	KB
log_temp_files	KB
maintenance_work_mem	KB
max_stack_depth	KB
ssl_renegotiation_limit	KB

Parameter Name	Unit
temp_file_limit	KB
work_mem	KB
log_rotation_age	minutes
autovacuum_vacuum_cost_delay	ms
bgwriter_delay	ms
deadlock_timeout	ms
lock_timeout	ms
log_autovacuum_min_duration	ms
log_min_duration_statement	ms
max_standby_archive_delay	ms
max_standby_streaming_delay	ms
statement_timeout	ms
vacuum_cost_delay	ms
wal_receiver_timeout	ms
wal_sender_timeout	ms
wal_writer_delay	ms
archive_timeout	s
authentication_timeout	s
autovacuum_naptime	s
checkpoint_timeout	s
checkpoint_warning	s
post_auth_delay	s
pre_auth_delay	s
tcp_keepalives_idle	s
tcp_keepalives_interval	s
wal_receiver_status_interval	s

Working with PostgreSQL Autovacuum on Amazon RDS

We strongly recommend that you use the autovacuum feature for PostgreSQL databases to maintain the health of your PostgreSQL DB instance. Because autovacuum checks for tables that have had a large number of inserted, updated, or deleted tuples, you can use autovacuum to prevent transaction

ID wraparound. Autovacuum automates the execution of the VACUUM and the ANALYZE command. Using autovacuum is required by PostgreSQL, not imposed by Amazon RDS, and its use is critical to good performance. The feature is enabled by default for all new Amazon RDS PostgreSQL DB instances, and the related configuration parameters are appropriately set by default. Since our defaults are somewhat generic, you can benefit from tuning parameters to your specific workload. This section can help you perform the needed autovacuum tuning.

For information on creating a process that warns you about transaction ID wraparound, see the AWS Database Blog entry [Implement an Early Warning System for Transaction ID Wraparound in Amazon RDS for PostgreSQL](#).

Topics

- [Maintenance Work Memory \(p. 1010\)](#)
- [Determining if the Tables in Your Database Need Vacuuming \(p. 1010\)](#)
- [Determining Which Tables Are Currently Eligible for Autovacuum \(p. 1011\)](#)
- [Determining if Autovacuum Is Currently Running and For How Long \(p. 1012\)](#)
- [Performing a Manual Vacuum Freeze \(p. 1014\)](#)
- [Reindexing a Table When Autovacuum Is Running \(p. 1015\)](#)
- [Other Parameters That Affect Autovacuum \(p. 1016\)](#)
- [Autovacuum Logging \(p. 1017\)](#)

Maintenance Work Memory

One of the most important parameters influencing autovacuum performance is the `maintenance_work_mem` parameter. This parameter determines how much memory you allocate for autovacuum to use to scan a database table and to hold all the row IDs that are going to be vacuumed. If you set the value of the `maintenance_work_mem` parameter too low, the vacuum process might have to scan the table multiple times to complete its work, possibly impacting performance.

When doing calculations to determine the `maintenance_work_mem` parameter value, keep in mind two things:

- The default unit is KB for this parameter.
- The `maintenance_work_mem` parameter works in conjunction with the `autovacuum_max_workers` parameter. If you have many small tables, allocate more `autovacuum_max_workers` and less `maintenance_work_mem`. If you have large tables (say, larger than 100 GB), allocate more memory and fewer workers. You need to have enough memory allocated to succeed on your biggest table. Each `autovacuum_max_workers` can use the memory you allocate, so you should make sure the combination of workers and memory equal the total memory you want to allocate.

In general terms, for large hosts, set the `maintenance_work_mem` parameter to a value between one and two gigabytes. For extremely large hosts, set the parameter to a value between two and four gigabytes. The value you set for this parameter should depend on the workload. Amazon RDS has updated its default for this parameter to be `GREATEST({DBInstanceClassMemory/63963136*1024}, 65536)`.

Determining if the Tables in Your Database Need Vacuuming

A PostgreSQL database can have two billion "in-flight" unvacuumed transactions before PostgreSQL takes dramatic action to avoid data loss. If the number of unvacuumed transactions reaches ($2^{31} - 10,000,000$), the log will start warning that vacuuming is needed. If the number of unvacuumed transactions reaches ($2^{31} - 1,000,000$), PostgreSQL sets the database to read only and requires an offline, single-user, standalone vacuum. This requires multiple hours or days (depending on size).

of downtime. A very detailed explanation of [TransactionID wraparound](#) is found in the PostgreSQL documentation.

The following query can be used to show the number of unvacuumed transactions in a database. The datfrozenxid column of a database's pg_database row is a lower bound on the normal XIDs appearing in that database; it is the minimum of the per-table relfrozenxid values within the database.

```
select datname, age(datfrozenxid) from pg_database order by age(datfrozenxid) desc limit 20;
```

For example, the results of running the preceding query might be the following:

datname	age
mydb	1771757888
template0	1721757888
template1	1721757888
rdsadmin	1694008527
postgres	1693881061
(5 rows)	

When the age of a database hits two billion, TransactionID (XID) wraparound occurs and the database will go into read only. This query can be used to produce a metric and run a few times a day. By default, autovacuum is set to keep the age of transactions to no more than 200,000,000 ([autovacuum_freeze_max_age](#)).

A sample monitoring strategy might look like this:

- Autovacuum_freeze_max_age is set to 200 million.
- If a table hits 500 million unvacuumed transactions, a low-severity alarm is triggered. This isn't an unreasonable value, but it could indicate that autovacuum isn't keeping up.
- If a table ages to one billion, this should be treated as an actionable alarm. In general, you want to keep ages closer to autovacuum_freeze_max_age for performance reasons. Investigation using the following steps is recommended.
- If a table hits 1.5 billion unvacuumed transactions, a high-severity alarm is triggered. Depending on how quickly your database uses XIDs, this alarm can indicate that the system is running out of time to run autovacuum and that you should consider immediate resolution.

If a table is constantly breaching these thresholds, you need further modify your autovacuum parameters. By default, VACUUM (which has cost-based delays disabled) is more aggressive than default autovacuum, but, also more intrusive to the system as a whole.

We have the following recommendations:

- Be aware and enable a monitoring mechanism so that you are aware of the age of your oldest transactions.
- For busier tables, perform a manual vacuum freeze regularly during a maintenance window in addition to relying on autovacuum. For information on performing a manual vacuum freeze, see [Performing a Manual Vacuum Freeze \(p. 1014\)](#).

Determining Which Tables Are Currently Eligible for Autovacuum

Often, it is one or two tables in need of vacuuming. Tables whose relfrozenxid value is more than autovacuum_freeze_max_age transactions old are always targeted by autovacuum. Otherwise, if the

number of tuples made obsolete since the last VACUUM exceeds the "vacuum threshold", the table is vacuumed.

The [autovacuum threshold](#) is defined as:

```
Vacuum threshold = vacuum base threshold + vacuum scale factor * number of tuples
```

While you are connected to your database, run the following query to see a list of tables that autovacuum sees as eligible for vacuuming:

```

WITH vbt AS (SELECT setting AS autovacuum_vacuum_threshold FROM
pg_settings WHERE name = 'autovacuum_vacuum_threshold')
     , vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor FROM
pg_settings WHERE name = 'autovacuum_vacuum_scale_factor')
     , fma AS (SELECT setting AS autovacuum_freeze_max_age FROM
pg_settings WHERE name = 'autovacuum_freeze_max_age')
     , sto AS (select opt_oid, split_part(setting, '=', 1) as param,
split_part(setting, '=', 2) as value from (select oid opt_oid,
unnest(reloptions) setting from pg_class) opt)
SELECT
      '''||ns.nspname||'.'||c.relname||''' as relation
     , pg_size.pretty(pg_table_size(c.oid)) as table_size
     , age(relfrozenxid) as xid_age
     , coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
autovacuum_freeze_max_age
     , (coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float)
+ coalesce(cvsf.value::float, autovacuum_vacuum_scale_factor::float) *
c.reltuples) as autovacuum_vacuum_tuples
     , n_dead_tup as dead_tuples
FROM pg_class c join pg_namespace ns on ns.oid = c.relnamespace
join pg_stat_all_tables stat on stat.relid = c.oid
join vbt on (1=1) join vsf on (1=1) join fma on (1=1)
left join sto cvbt on cvbt.param = 'autovacuum_vacuum_threshold' and
c.oid = cvbt.opt_oid
left join sto cvsf on cvsf.param = 'autovacuum_vacuum_scale_factor' and
c.oid = cvsf.opt_oid
left join sto cfma on cfma.param = 'autovacuum_freeze_max_age' and
c.oid = cfma.opt_oid
WHERE c.relkind = 'r' and nspname <> 'pg_catalog'
and (
      age(relfrozenxid) >= coalesce(cfma.value::float,
autovacuum_freeze_max_age::float)
      or
      coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float, autovacuum_vacuum_scale_factor::float) *
c.reltuples <= n_dead_tup
      -- or 1 = 1
)
ORDER BY age(relfrozenxid) DESC LIMIT 50;
```

Determining if Autovacuum Is Currently Running and For How Long

If you need to manually vacuum a table, you need to determine if autovacuum is currently running. If it is, you might need to adjust parameters to make it run more efficiently, or terminate autovacuum so you can manually run VACUUM.

Use the following query to determine if autovacuum is running, how long it has been running, and if it is waiting on another session.

If you are using Amazon RDS PostgreSQL 9.6+ or higher, use this query:

```
SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
xact_runtime, query
FROM pg_stat_activity
WHERE upper(query) like '%VACUUM%'
ORDER BY xact_start;
```

After running the query, you should see output similar to the following.

datname	usename	pid	state	wait_event	xact_runtime	query
mydb	rdsadmin	16473	active		33 days 16:32:11.600656	autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
mydb	rdsadmin	22553	active		14 days 09:15:34.073141	autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
mydb	rdsadmin	41909	active		3 days 02:43:54.203349	autovacuum: VACUUM ANALYZE public.mytable3
mydb	rdsadmin	618	active		00:00:00	SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS xact_runtime, query+ FROM pg_stat_activity + WHERE query like '%VACUUM%' + ORDER BY xact_start; +

If you are using a version less than Amazon RDS PostgreSQL 9.6, but, 9.3.12 or later, 9.4.7 or later, or 9.5.2+, use this query:

```
SELECT datname, usename, pid, waiting, current_timestamp - xact_start AS xact_runtime,
query
FROM pg_stat_activity
WHERE upper(query) like '%VACUUM%'
ORDER BY xact_start;
```

After running the query, you should see output similar to the following.

datname	usename	pid	waiting	xact_runtime	query
mydb	rdsadmin	16473	f	33 days 16:32:11.600656	autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
mydb	rdsadmin	22553	f	14 days 09:15:34.073141	autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
mydb	rdsadmin	41909	f	3 days 02:43:54.203349	autovacuum: VACUUM ANALYZE public.mytable3
mydb	rdsadmin	618	f	00:00:00	SELECT datname, usename, pid, waiting, current_timestamp - xact_start AS xact_runtime, query+ FROM pg_stat_activity + WHERE query like '%VACUUM%' %

```
| ORDER BY xact_start;  
+
```

Several issues can cause long running (multiple days) autovacuum session. The most common issue is that your `maintenance_work_mem` parameter value is set too low for the size of the table or rate of updates.

We recommend that you use the following formula to set the `maintenance_work_mem` parameter value.

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

Short running autovacuum sessions can also indicate problems:

- It can indicate that there aren't enough autovacuum_max_workers for your workload. You will need to indicate the number of workers.
- It can indicate that there is an index corruption (autovacuum will crash and restart on the same relation but make no progress). You will need to run a manual vacuum freeze verbose `__table__` to see the exact cause.

Performing a Manual Vacuum Freeze

You might want to perform a manual vacuum on a table that has a vacuum process already running. This is useful if you have identified a table with an "XID age" approaching 2 billion (or above any threshold you are monitoring).

The following steps are a guideline, and there are several variations to the process. For example, during testing, you find that the `maintenance_work_mem` parameter value was set too small and that you need to take immediate action on a table but don't want to bounce the instance at the moment. Using the queries listed above, you determine which table is the problem and notice a long running autovacuum session. You know you need to change the `maintenance_work_mem` parameter setting, but you also need to take immediate action and vacuum the table in question. The following procedure shows what you would do in this situation:

To manually perform a vacuum freeze

1. Open two sessions to the database containing the table you want to vacuum. For the second session, use "screen" or another utility that maintains the session if your connection is dropped.
2. In session one, get the PID of the autovacuum session running on the table. This action requires that you are running Amazon RDS PostgreSQL 9.3.12 or later, 9.4.7 or later, or 9.5.2 or later to have full visibility into the running rdsadmin processes.

Run the following query to get the PID of the autovacuum session.

```
SELECT datname, username, pid, waiting, current_timestamp - xact_start  
AS xact_runtime, query  
FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY  
xact_start;
```

3. In session two, calculate the amount of memory you will need for this operation. In this example, we determine that we can afford to use up to 2 GB of memory for this operation, so we set `maintenance_work_mem` for the current session to 2 GB.

```
set maintenance_work_mem='2 GB';  
SET
```

4. In session two, issue a vacuum freeze verbose for the table. The verbose setting is useful because, although there is no progress report for this in PostgreSQL currently, you can see activity.

```
\timing on
Timing is on.
vacuum freeze verbose pgbench_branches;
INFO:  vacuuming "public.pgbench_branches"
INFO:  index "pgbench_branches_pkey" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  index "pgbench_branches_test_index" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  "pgbench_branches": found 0 removable, 50 nonremovable row versions
      in 43 out of 43 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 9347 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
Time: 2.765 ms
```

5. In session one, if autovacuum was blocking, you will see in pg_stat_activity that waiting is "T" for your vacuum session. In this case, you need to terminate the autovacuum process.

```
select pg_terminate_backend('the_pid');
```

6. At this point, your session begins. It's important to note that autovacuum will restart immediately as this table is probably the highest on its list of work. You will need to initiate your command in session 2 and then terminate the autovacuum process in session one.

Reindexing a Table When Autovacuum Is Running

If an index has become corrupt, autovacuum will continue to process the table and fail. If you attempt a manual vacuum in this situation, you will receive an error message similar to the following:

```
mydb=# vacuum freeze pgbench_branches;
ERROR: index "pgbench_branches_test_index" contains unexpected
      zero page at block 30521
HINT: Please REINDEX it.
```

When the index is corrupted and autovacuum is attempting to run against the table, you will contend with an already running autovacuum session. When you issue a "[REINDEX](#)" command, you will be taking out an exclusive lock on the table and write operations will be blocked as well as reads that use that specific index.

To reindex a table when autovacuum is running on the table

1. Open two sessions to the database containing the table you want to vacuum. For the second session, use "screen" or another utility that maintains the session if your connection is dropped.
2. In session one, get the PID of the autovacuum session running on the table. This action requires that you are running Amazon RDS PostgreSQL 9.3.12 or later, 9.4.7 or later, or 9.5.2 or later to have full visibility into the running rdsadmin processes.

Run the following query to get the PID of the autovacuum session:

```
SELECT datname, username, pid, waiting, current_timestamp - xact_start
```

```
AS xact_runtime, query
FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
xact_start;
```

3. In session two, issue the reindex command.

```
\timing on
Timing is on.
reindex index pgbench_branches_test_index;
REINDEX
Time: 9.966 ms
```

4. In session one, if autovacuum was blocking, you will see in *pg_stat_activity* that waiting is "T" for your vacuum session. In this case, you will need to terminate the autovacuum process.

```
select pg_terminate_backend('the_pid');
```

5. At this point, your session begins. It's important to note that autovacuum will restart immediately as this table is probably the highest on its list of work. You will need to initiate your command in session 2 and then terminate the autovacuum process in session one.

Other Parameters That Affect Autovacuum

This query will show the values of some of the parameters that directly impact autovacuum and its behavior. The [autovacuum parameters](#) are described fully in the PostgreSQL documentation.

```
select name, setting, unit, short_desc
from pg_settings
where name in (
'autovacuum_max_workers',
'autovacuum_analyze_scale_factor',
'autovacuum_naptime',
'autovacuum_analyze_threshold',
'autovacuum_analyze_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_cost_delay',
'autovacuum_vacuum_cost_limit',
'vacuum_cost_limit',
'autovacuum_freeze_max_age',
'maintenance_work_mem',
'vacuum_freeze_min_age');
```

While these all affect autovacuum, some of the most important ones are:

- [Maintenance_Work_Mem](#)
- [Autovacuum_freeze_max_age](#)
- [Autovacuum_max_workers](#)
- [Autovacuum_vacuum_cost_delay](#)
- [Autovacuum_vacuum_cost_limit](#)

Table-Level Parameters

Autovacuum related [storage parameters](#) can be set at a table level, which can be better than altering the behavior of the entire database. For large tables, you might need to set aggressive settings and you might not want to make autovacuum behave that way for all tables.

This query will show which tables currently have table level options in place:

```
select relname, reloptions
from pg_class
where reloptions is not null;
```

An example where this might be useful is on tables that are much larger than the rest of your tables. If you have one 300-GB table and 30 other tables less than 1 GB, you might set some specific parameters for your large table so you don't alter the behavior of your entire system.

```
alter table mytable set (autovacuum_vacuum_cost_delay=0);
```

Doing this disables the cost-based autovacuum delay for this table at the expense of more resource usage on your system. Normally, autovacuum pauses for autovacuum_vacuum_cost_delay each time autovacuum_cost_limit is reached. You can find more details in the PostgreSQL documentation about [cost-based vacuuming](#).

Autovacuum Logging

By default, the *postgresql.log* doesn't contain information about the autovacuum process. If you are using PostgreSQL 9.4.5 or later, you can see output in the PostgreSQL error log from the autovacuum worker operations by setting the `rds.force_autovacuum_logging_level` parameter. Allowed values are `disabled`, `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, and `panic`. The default value is `disabled` because the other allowable values can add significant amount of information to your logs.

We recommend that you set the value of the `rds.force_autovacuum_logging_level` parameter to `log` and that you set the `log_autovacuum_min_duration` parameter to a value from 1000 or 5000. If you set this value to 5000, Amazon RDS writes activity to the log that takes more than five seconds and shows "vacuum skipped" messages when application locking is causing autovacuum to intentionally skip tables. If you are troubleshooting a problem and need more detail, you can use a different logging level value, such as `debug1` or `debug3`. Use these debug parameters for a short period of time because these settings produce extremely verbose content written to the error log file. For more information about these debug settings, see the [PostgreSQL documentation](#).

NOTE: PostgreSQL version 9.4.7 and later includes improved visibility of autovacuum sessions by allowing the `rds_superuser` account to view autovacuum sessions in `pg_stat_activity`. For example, you can identify and terminate an autovacuum session that is blocking a command from running, or executing slower than a manually issued `vacuum` command.

Audit Logging for a PostgreSQL DB Instance

There are several parameters you can set to log activity that occurs on your PostgreSQL DB instance. These parameters include the following:

- The `log_statement` parameter can be used to log user activity in your PostgreSQL database. For more information, see [PostgreSQL Database Log Files \(p. 334\)](#).
- The `rds.force_admin_logging_level` parameter logs actions by the RDS internal user (`rdsadmin`) in the databases on the DB instance, and writes the output to the PostgreSQL error log. Allowed values are `disabled`, `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, and `panic`. The default value is `disabled`.
- The `rds.force_autovacuum_logging_level` parameter logs autovacuum worker operations in all databases on the DB instance, and writes the output to the PostgreSQL error log. Allowed values are `disabled`, `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, and `panic`. The default value is `disabled`. The Amazon RDS recommended setting for

rds.force_autovacuum_logging_level: is LOG. Set log_autovacuum_min_duration to a value from 1000 or 5000. Setting this value to 5000 will write activity to the log that takes more than 5 seconds and will show "vacuum skipped" messages. For more information on this parameter, see [Best Practices for Working with PostgreSQL \(p. 77\)](#).

Working with the pgaudit Extension

The pgaudit extension provides detailed session and object audit logging for Amazon RDS for PostgreSQL version 9.6.3 and later and version 9.5.7 version and later. You can enable session auditing or object auditing using this extension.

With session auditing, you can log audit events from various sources and includes the fully qualified command text when available. For example, you can use session auditing to log all READ statements that connect to a database by setting pgaudit.log to 'READ'.

With object auditing, you can refine the audit logging to work with specific commands. For example, you can specify that you want audit logging for READ operations on a specific number of tables.

To use object based logging with the pgaudit extension

1. Create a specific database role called rds_pgaudit. Use the following command to create the role.

```
CREATE ROLE rds_pgaudit;
CREATE ROLE
```

2. Modify the parameter group that is associated with your DB instance to use the shared preload libraries that contain pgaudit and set the parameter pgaudit.role. The pgaudit.role must be set to the role rds_pgaudit.

The following command modifies a custom parameter group.

```
aws rds modify-db-parameter-group
  --db-parameter-group-name rds-parameter-group-96
  --parameters
  "ParameterName=pgaudit.role,ParameterValue=rds_pgaudit,ApplyMethod=pending-reboot"
    --parameters
  "ParameterName=shared_preload_libraries,ParameterValue=pgaudit,ApplyMethod=pending-reboot"
  --region us-west-2
```

3. Reboot the instance so that the DB instance will pick up the changes to the parameter group. The following command reboots a DB instance.

```
aws rds reboot-db-instance --db-instance-identifier rds-test-instance --region us-west-2
```

4. Run the following command to confirm that pgaudit has been initialized.

```
show shared_preload_libraries;
shared_preload_libraries
-----
```

```
rdsutils,pgaudit  
(1 row)
```

5. Run the following command to create the pgaudit extension.

```
CREATE EXTENSION pgaudit;  
CREATE EXTENSION
```

6. Run the following command to confirm pgaudit.role is set to *rds_pgaudit*.

```
show pgaudit.role;  
pgaudit.role  
-----  
rds_pgaudit
```

To test the audit logging, run several commands that you have chosen to audit. For example, you might run the following commands.

```
CREATE TABLE t1 (id int);  
CREATE TABLE  
GRANT SELECT ON t1 TO rds_pgaudit;  
GRANT  
select * from t1;  
id  
----  
(0 rows)
```

The database logs will contain an entry similar to the following:

```
...  
2017-06-12 19:09:49 UTC:...:rds_test@postgres:[11701]:LOG: AUDIT:  
OBJECT,1,1,READ,SELECT,TABLE,public.t1,select * from t1;  
...
```

For information on viewing the logs, see [Amazon RDS Database Log Files \(p. 307\)](#).

Working with the pg_repack Extension

You can use the pg_repack extension to remove bloat from tables and indexes. This extension is supported on Amazon RDS for PostgreSQL versions 9.6.3 and later. For more information on the pg_repack extension, see the [GitHub project documentation](#).

To use the pg_repack extension

1. Install the pg_repack extension on your Amazon RDS for PostgreSQL DB instance by running the following command.

```
CREATE EXTENSION pg_repack;
```

2. Use the `pg_repack` client utility to connect to a database. Use a database role that has `rds_superuser` privileges to connect to the database. In the following connection example, the `rds_test` role has `rds_superuser` privileges, and the database endpoint used is `rds-test-instance.cw7jjfgdr4on8.us-west-2.rds.amazonaws.com`.

```
pg_repack -h rds-test-instance.cw7jjfgdr4on8.us-west-2.rds.amazonaws.com -U rds_test -k postgres
```

Connect using the `-k` option. The `-a` option is not supported.

3. The response from the `pg_repack` client provides information on the tables on the DB instance that are repacked.

```
INFO: repacking table "pgbench_tellers"  
INFO: repacking table "pgbench_accounts"  
INFO: repacking table "pgbench_branches"
```

Working with PostGIS

PostGIS is an extension to PostgreSQL for storing and managing spatial information. If you are not familiar with PostGIS, you can get a good general overview at [PostGIS Introduction](#).

You need to perform a bit of setup before you can use the PostGIS extension. The following list shows what you need to do; each step is described in greater detail later in this section.

- Connect to the DB instance using the master user name used to create the DB instance.
- Load the PostGIS extensions.
- Transfer ownership of the extensions to the `rds_superuser` role.
- Transfer ownership of the objects to the `rds_superuser` role.
- Test the extensions.

Step 1: Connect to the DB Instance Using the Master User Name Used to Create the DB Instance

First, you connect to the DB instance using the master user name that was used to create the DB instance. That name is automatically assigned the `rds_superuser` role. You need the `rds_superuser` role that is needed to do the remaining steps.

The following example uses `SELECT` to show you the current user; in this case, the current user should be the master username you chose when creating the DB instance.

```
select current_user;  
current_user  
-----  
myawsuser  
(1 row)
```

Step 2: Load the PostGIS Extensions

Use the CREATE EXTENSION statements to load the PostGIS extensions. You must also load the extension. You can then use the \dn *psql* command to list the owners of the PostGIS schemas.

```
create extension postgis;
CREATE EXTENSION
create extension fuzzystrmatch;
CREATE EXTENSION
create extension postgis_tiger_geocoder;
CREATE EXTENSION
create extension postgis_topology;
CREATE EXTENSION
\dn
    List of schemas
   Name   |  Owner
-----+-----
 public  | myawsuser
 tiger   | rdsadmin
 tiger_data | rdsadmin
 topology | rdsadmin
(4 rows)
```

Step 3: Transfer Ownership of the Extensions to the *rds_superuser* Role

Use the ALTER SCHEMA statements to transfer ownership of the schemas to the *rds_superuser* role.

```
alter schema tiger owner to rds_superuser;
ALTER SCHEMA
alter schema tiger_data owner to rds_superuser;
ALTER SCHEMA
alter schema topology owner to rds_superuser;
ALTER SCHEMA
\dn
    List of schemas
   Name   |  Owner
-----+-----
 public  | myawsuser
 tiger   | rds_superuser
 tiger_data | rds_superuser
 topology | rds_superuser
(4 rows)
```

Step 4: Transfer Ownership of the Objects to the *rds_superuser* Role

Use the following function to transfer ownership of the PostGIS objects to the *rds_superuser* role. Run the following statement from the *psql* prompt to create the function.

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $$ BEGIN EXECUTE $1;
 RETURN $1; END; $$;
```

Next, run this query to run the *exec* function that in turn executes the statements and alters the permissions.

```
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' || quote_ident(s.relname) || '  
OWNER TO rds_superuser;')  
FROM (  
    SELECT nspname, relname  
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)  
    WHERE nspname in ('tiger','topology') AND  
    relkind IN ('r','S','v') ORDER BY relkind = 'S'  
s;
```

Step 5: Test the Extensions

Add `tiger` to your search path using the following command.

```
SET search_path=public,tiger;
```

Test `tiger` by using the following SELECT statement.

```
select na.address, na.streetname, na.streettypeabbrev, na.zip  
from normalize_address('1 Devonshire Place, Boston, MA 02109') as na;  
address | streetname | streettypeabbrev | zip  
-----+-----+-----+-----  
1 | Devonshire | Pl | 02109  
(1 row)
```

Test `topology` by using the following SELECT statement.

```
select topology.createtopology('my_new_topo',26986,0.5);  
createtopology  
-----  
1  
(1 row)
```

Using pgBadger for Log Analysis with PostgreSQL

You can use a log analyzer such as [pgbadger](#) to analyze PostgreSQL logs. The *pgbadger* documentation states that the `%l` pattern (log line for session/process) should be a part of the prefix. However, if you provide the current rds `log_line_prefix` as a parameter to *pgbadger* it should still produce a report.

For example, the following command correctly formats an Amazon RDS PostgreSQL log file dated 2014-02-04 using *pgbadger*.

```
./pgbadger -p '%t:%r:%u@%d:[%p]:' postgresql.log.2014-02-04-00
```

Viewing the Contents of pg_config

In PostgreSQL version 9.6.1, you can see the compile-time configuration parameters of the currently installed version of PostgreSQL using the new view `pg_config`. You can use the view by calling the `pg_config` function as shown in the following sample.

```
select * from pg_config();  
name | setting
```

```

-----+
-----+
BINDIR          | /rdsdbbin/postgres-9.6.1.R1/bin
DOCDIR          | /rdsdbbin/postgres-9.6.1.R1/share/doc
HTMLDIR         | /rdsdbbin/postgres-9.6.1.R1/share/doc
INCLUDEDIR       | /rdsdbbin/postgres-9.6.1.R1/include
PKGINCLUEDIR    | /rdsdbbin/postgres-9.6.1.R1/include
INCLUDEDIR-SERVER | /rdsdbbin/postgres-9.6.1.R1/include/server
LIBDIR          | /rdsdbbin/postgres-9.6.1.R1/lib
PKGLIBDIR       | /rdsdbbin/postgres-9.6.1.R1/lib
LOCALEDIR        | /rdsdbbin/postgres-9.6.1.R1/share/locale
MANDIR          | /rdsdbbin/postgres-9.6.1.R1/share/man
SHAREDIR         | /rdsdbbin/postgres-9.6.1.R1/share
SYSCONFDIR      | /rdsdbbin/postgres-9.6.1.R1/etc
PGXS            | /rdsdbbin/postgres-9.6.1.R1/lib/pgxs/src/makefiles/pgxs.mk
CONFIGURE        | '--prefix=/rdsdbbin/postgres-9.6.1.R1' '--with-openssl' '--with-perl'
                  '--with-tcl' '--with-uuid' '--with-libxml' '--with-libraries=/rdsdbbin
/postgres-9.6.1.R1/lib' '--with-includes=/rdsdbbin/postgres-9.6.1.R1/include' '--enable-
debug'
CC               | gcc
CPPFLAGS         | -D_GNU_SOURCE -I/usr/include/libxml2 -I/rdsdbbin/postgres-9.6.1.R1/
include
CFLAGS           | -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-
statement
-Wendif-labels   -Wmissing-format-attribute -Wformat-security -fno-strict-
aliasing -fwrapv -fexcess-precision=standard -g -O2
CFLAGS_SL        | -fpic
LDFLAGS          | -L../../src/common -L/rdsdbbin/postgres-9.6.1.R1/lib -Wl,--as-needed -
Wl,
-rpath,'/rdsdbbin/postgres-9.6.1.R1/lib',--enable-new-dtags
LDFLAGS_EX       |
LDFLAGS_SL       |
LIBS             | -lpgcommon -lpgport -lxmll -lssl -lcrypto -lz -lreadline -lrt -lcrypt
-ldl -lm
VERSION          | PostgreSQL 9.6.1
(23 rows)

```

If you attempt to access the view directly, the request fails.

```

select * from pg_config;
ERROR: permission denied for relation pg_config

```

Working with the orafce Extension

The `orafce` extension provides functions that are common in commercial databases, and can make it easier for you to port a commercial database to PostgreSQL. Amazon RDS for PostgreSQL versions 9.6.6 and later support this extension. For more information about `orafce`, see the [orafce project on GitHub](#).

Note

Amazon RDS for PostgreSQL doesn't support the `utl_file` package that is part of the `orafce` extension. This is because the `utl_file` schema functions provide read and write operations on operating-system text files, which requires superuser access to the underlying host.

To use the orafce extension

1. Connect to the DB instance with the master user name that you used to create the DB instance.

Note

If you want to enable `orafce` on a different database in the same instance, use the `/c dbname psql` command to change from the master database after initiating the connection.

2. Enable the `orafce` extension with the `CREATE EXTENSION` statement.

```
CREATE EXTENSION orafce;
```

3. Transfer ownership of the oracle schema to the `rds_superuser` role with the `ALTER SCHEMA` statement.

```
ALTER SCHEMA oracle OWNER TO rds_superuser;
```

Note

If you want to see the list of owners for the oracle schema, use the `\dn` `psql` command.

Accessing External Data with the `postgres_fdw` Extension

You can access data in a table on a remote database server with the `postgres_fdw` extension. If you set up a remote connection from your PostgreSQL DB instance, access is also available to your Read Replica.

To use `postgres_fdw` to access a remote database server

1. Install the `postgres_fdw` extension.

```
CREATE EXTENSION postgres_fdw;
```

2. Create a foreign data server using `CREATE SERVER`.

```
CREATE SERVER foreign_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'xxx.xx.xxx.xx', port '5432', dbname 'foreign_db');
```

3. Create a user mapping to identify the role to be used on the remote server.

```
CREATE USER MAPPING FOR local_user
SERVER foreign_server
OPTIONS (user 'foreign_user', password 'password');
```

4. Create a table that maps to the table on the remote server.

```
CREATE FOREIGN TABLE foreign_table (
    id integer NOT NULL,
    data text)
SERVER foreign_server
OPTIONS (schema_name 'some_schema', table_name 'some_table');
```

Using a Custom DNS Server for Outbound Network Access

Amazon RDS for PostgreSQL supports outbound network access on your DB instances and allows Domain Name Service (DNS) resolution from a custom DNS server owned by the customer. You can resolve only fully qualified domain names from your Amazon RDS DB instance through your custom DNS server.

Topics

- [Enabling Custom DNS Resolution \(p. 1025\)](#)
- [Disabling Custom DNS Resolution \(p. 1025\)](#)
- [Setting Up a Custom DNS Server \(p. 1025\)](#)

Enabling Custom DNS Resolution

To enable the DNS resolution in your customer VPC, you need to associate a custom DB parameter group to your RDS PostgreSQL instance, turn on the parameter **rds.custom_dns_resolution** by setting it to 1, and restart the DB instance for the changes to take place.

Disabling Custom DNS Resolution

In order to disable the DNS resolution in your customer VPC, you need to turn off the parameter **rds.custom_dns_resolution** of your custom DB parameter group by setting it to 0, then restart the DB instance for the changes to take place.

Setting Up a Custom DNS Server

After you set up your custom DNS name server, it takes up to 30 minutes to propagate the changes to your DB instance. After the changes are propagated to your DB instance, all outbound network traffic requiring a DNS lookup queries your DNS server over port 53.

To set up a custom DNS server for your Amazon RDS PostgreSQL DB instance, do the following:

1. From the DHCP options set attached to your VPC, set the `domain-name-servers` option to the IP address of your DNS name server. For more information, see [DHCP Options Sets](#).

Note
The `domain-name-servers` option accepts up to four values, but your Amazon RDS DB instance uses only the first value.
2. Ensure that your DNS server can resolve all lookup queries, including public DNS names, Amazon EC2 private DNS names, and customer-specific DNS names. If the outbound network traffic contains any DNS lookups that your DNS server can't handle, your DNS server must have appropriate upstream DNS providers configured.
3. Configure your DNS server to produce User Datagram Protocol (UDP) responses of 512 bytes or less.
4. Configure your DNS server to produce Transmission Control Protocol (TCP) responses of 1024 bytes or less.
5. Configure your DNS server to allow inbound traffic from your Amazon RDS DB instances over port 53. If your DNS server is in an Amazon VPC, the VPC must have a security group that contains inbound rules that allow UDP and TCP traffic on port 53. If your DNS server is not in an Amazon VPC, it must have appropriate firewall whitelisting to allow UDP and TCP inbound traffic on port 53.
For more information, see [Security Groups for Your VPC](#) and [Adding and Removing Rules](#).
6. Configure the VPC of your Amazon RDS DB instance to allow outbound traffic over port 53. Your VPC must have a security group that contains outbound rules that allow UDP and TCP traffic on port 53.

For more information, see [Security Groups for Your VPC](#) and [Adding and Removing Rules](#).

7. The routing path between the Amazon RDS DB instance and the DNS server has to be configured correctly to allow DNS traffic.

If the Amazon RDS DB instance and the DNS server are not in the same VPC, a peering connection has to be set up between them. For more information, see [What is VPC Peering?](#)

Restricting Password Management

You can restrict who can manage database user passwords to a special role. By doing this, you can have more control over password management on the client side.

You enable restricted password management with the static parameter `rds.restrict_password_commands` and use a role called `rds_password`. When the parameter `rds.restrict_password_commands` is set to 1, only users that are members of the `rds_password` role can run certain SQL commands. The restricted SQL commands are commands that modify database user passwords and password expiration time.

To use restricted password management, your DB instance must be running Amazon RDS for PostgreSQL 10.6 or higher. Because the `rds.restrict_password_commands` parameter is static, changing this parameter requires a database restart.

When a database has restricted password management enabled, if you try to run restricted SQL commands you get the following error: ERROR: must be a member of `rds_password` to alter passwords.

Following are some examples of SQL commands that are restricted when restricted password management is enabled.

```
postgres=> CREATE ROLE myrole WITH PASSWORD 'mypassword';
postgres=> CREATE ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2020-01-01';
postgres=> ALTER ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2020-01-01';
postgres=> ALTER ROLE myrole WITH PASSWORD 'mypassword';
postgres=> ALTER ROLE myrole VALID UNTIL '2020-01-01';
postgres=> ALTER ROLE myrole RENAME TO myrole2;
```

Some `ALTER ROLE` commands that include `RENAME TO` might also be restricted. They might be restricted because renaming a PostgreSQL role that has an MD5 password clears the password.

The `rds_superuser` role has membership for the `rds_password` role by default, and you can't change this. You can give other roles membership for the `rds_password` role by using the `GRANT` SQL command. We recommend that you give membership to `rds_password` to only a few roles that you use solely for password management. These roles require the `CREATEROLE` attribute to modify other roles.

Make sure that you verify password requirements such as expiration and needed complexity on the client side. We recommend that you restrict password-related changes by using your own client-side utility. This utility should have a role that is a member of `rds_password` and has the `CREATEROLE` role attribute.

Working with the Database Preview Environment

When you create a DB instance in Amazon RDS, you know that the PostgreSQL version it's based on has been tested and is fully supported by Amazon. The PostgreSQL community releases new versions and new extensions continuously. You can try out new PostgreSQL versions and extensions before they are fully supported. To do that, you can create a new DB instance in the Database Preview Environment.

DB instances in the Database Preview Environment are similar to DB instances in a production environment. However, keep in mind several important factors:

- All DB instances are deleted 60 days after you create them, along with any backups and snapshots.
- You can only create a DB instance in a virtual private cloud (VPC) based on the Amazon VPC service.
- You can only create M4, T2, and R4 instance types. For more information about RDS instance classes, see [DB Instance Class \(p. 82\)](#).
- You can't get help from AWS Support with DB instances. You can post your questions in the [RDS Database Preview Environment Forum](#).
- You can only use General Purpose SSD and Provisioned IOPS SSD storage.
- You can't copy a snapshot of a DB instance to a production environment.
- Some Amazon RDS features aren't available in the preview environment, as described following.

Topics

- [Features Not Supported in the Preview Environment \(p. 1027\)](#)
- [PostgreSQL Extensions Supported in the Preview Environment \(p. 1027\)](#)
- [Creating a New DB Instance in the Preview Environment \(p. 1029\)](#)

Features Not Supported in the Preview Environment

The following features are not available in the preview environment:

- Cross-region snapshot copy
- Cross-region Read Replicas
- Extensions not in the following table of supported extensions

PostgreSQL Extensions Supported in the Preview Environment

The PostgreSQL extensions supported in the Database Preview Environment are listed following.

Extension	Version
amcheck	1.1
bloom	1.0
btree_gin	1.3
btree_gist	1.5
citext	1.5
cube	1.4
dblink	1.2
dict_int	1.0
dict_xsyn	1.0
earthdistance	1.1

Amazon Relational Database Service User Guide
PostgreSQL Extensions Supported
in the Preview Environment

Extension	Version
fuzzystrmatch	1.1
hstore	1.5
hstore_plper	1.0
intagg	1.1
antarray	1.2
isn	1.2
log_fdw	1.0
ltree	1.1
pg_buffercache	1.3
pg_freespacemap	1.2
pg_prewarm	1.2
pg_stat_statements	1.5
pg_trgm	1.4
pg_visibility	1.2
pgcrypto	1.3
pgrowlocks	1.2
pgstattuple	1.5
plperl	1.0
plpgsql	1.0
pltcl	1.0
postgres_fdw	1.0
sslinfo	1.2
tablefunc	1.0
test_parser	1.0
tsm_system_rows	1.0
tsm_system_time	1.0
unaccent	1.1
uuid-ossp	1.1

Creating a New DB Instance in the Preview Environment

Use the following procedure to create a DB instance in the preview environment.

To create a DB instance in the preview environment

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. Choose **Dashboard** from the navigation pane.
3. Choose **Switch to database preview environment**.

Database Preview Environment

Get early access to new DB engine versions, before they're generally available. The RDS database preview environment lets you work with upcoming beta, release candidate, and early production versions of PostgreSQL engines. Preview environment instances are fully functional, so you can easily test new features and functionality with your applications.

[Info](#)

[Preview PostgreSQL in US EAST \(Ohio\)](#)

You also can navigate directly to the [Database Preview Environment](#).

Note

If you want to create an instance in the Database Preview Environment with the API or CLI the endpoint is `rds-preview.us-east-2.amazonaws.com`.

4. Continue with the procedure as described in [Create a PostgreSQL DB Instance \(p. 969\)](#).

Amazon RDS for PostgreSQL Versions and Extensions

Amazon RDS supports DB instances running several editions of PostgreSQL. Use this section to see how to work with PostgreSQL on Amazon RDS. You should also be aware of the limits for PostgreSQL DB instances.

For information about importing PostgreSQL data into a DB instance, see [Importing Data into PostgreSQL on Amazon RDS \(p. 996\)](#).

Topics

- [Supported PostgreSQL Database Versions \(p. 1030\)](#)
- [Supported PostgreSQL Features and Extensions \(p. 1043\)](#)

Supported PostgreSQL Database Versions

Amazon RDS supports the following PostgreSQL versions.

Topics

- [PostgreSQL Version 11 on Amazon RDS in the Database Preview Environment \(p. 1031\)](#)
- [PostgreSQL Version 10.6 on Amazon RDS \(p. 1031\)](#)
- [PostgreSQL Version 10.5 on Amazon RDS \(p. 1031\)](#)
- [PostgreSQL Version 10.4 on Amazon RDS \(p. 1032\)](#)
- [PostgreSQL Version 10.3 on Amazon RDS \(p. 1032\)](#)
- [PostgreSQL Version 10.1 on Amazon RDS \(p. 1033\)](#)
- [PostgreSQL Version 9.6.11 on Amazon RDS \(p. 1033\)](#)
- [PostgreSQL Version 9.6.10 on Amazon RDS \(p. 1033\)](#)
- [PostgreSQL Version 9.6.9 on Amazon RDS \(p. 1034\)](#)
- [PostgreSQL Version 9.6.8 on Amazon RDS \(p. 1034\)](#)
- [PostgreSQL Version 9.6.6 on Amazon RDS \(p. 1034\)](#)
- [PostgreSQL Version 9.6.5 on Amazon RDS \(p. 1035\)](#)
- [PostgreSQL Version 9.6.3 on Amazon RDS \(p. 1035\)](#)
- [PostgreSQL Version 9.6.2 on Amazon RDS \(p. 1035\)](#)
- [PostgreSQL Version 9.6.1 on Amazon RDS \(p. 1036\)](#)
- [PostgreSQL Version 9.5.15 on Amazon RDS \(p. 1037\)](#)
- [PostgreSQL Version 9.5.14 on Amazon RDS \(p. 1037\)](#)
- [PostgreSQL Version 9.5.13 on Amazon RDS \(p. 1037\)](#)
- [PostgreSQL Version 9.5.12 on Amazon RDS \(p. 1037\)](#)
- [PostgreSQL Version 9.5.10 on Amazon RDS \(p. 1037\)](#)
- [PostgreSQL Version 9.5.9 on Amazon RDS \(p. 1037\)](#)
- [PostgreSQL Version 9.5.7 on Amazon RDS \(p. 1038\)](#)
- [PostgreSQL Version 9.5.6 on Amazon RDS \(p. 1038\)](#)
- [PostgreSQL Version 9.5.4 on Amazon RDS \(p. 1038\)](#)
- [PostgreSQL Version 9.5.2 on Amazon RDS \(p. 1039\)](#)
- [PostgreSQL Version 9.4.20 on Amazon RDS \(p. 1040\)](#)
- [PostgreSQL Version 9.4.19 on Amazon RDS \(p. 1040\)](#)
- [PostgreSQL Version 9.4.18 on Amazon RDS \(p. 1040\)](#)
- [PostgreSQL Version 9.4.17 on Amazon RDS \(p. 1040\)](#)
- [PostgreSQL Version 9.4.15 on Amazon RDS \(p. 1040\)](#)
- [PostgreSQL Version 9.4.14 on Amazon RDS \(p. 1040\)](#)
- [PostgreSQL Version 9.4.12 on Amazon RDS \(p. 1040\)](#)
- [PostgreSQL Version 9.4.11 on Amazon RDS \(p. 1041\)](#)
- [PostgreSQL Version 9.4.9 on Amazon RDS \(p. 1041\)](#)
- [PostgreSQL Version 9.4.7 on Amazon RDS \(p. 1041\)](#)
- [PostgreSQL Version 9.3.25 on Amazon RDS \(p. 1041\)](#)
- [PostgreSQL Version 9.3.24 on Amazon RDS \(p. 1042\)](#)
- [PostgreSQL Version 9.3.23 on Amazon RDS \(p. 1042\)](#)
- [PostgreSQL Version 9.3.22 on Amazon RDS \(p. 1042\)](#)
- [PostgreSQL Version 9.3.20 on Amazon RDS \(p. 1042\)](#)
- [PostgreSQL Version 9.3.19 on Amazon RDS \(p. 1042\)](#)

- [PostgreSQL Version 9.3.17 on Amazon RDS \(p. 1042\)](#)
- [PostgreSQL Version 9.3.16 on Amazon RDS \(p. 1043\)](#)
- [PostgreSQL Version 9.3.14 on Amazon RDS \(p. 1043\)](#)
- [PostgreSQL Version 9.3.12 on Amazon RDS \(p. 1043\)](#)

PostgreSQL Version 11 on Amazon RDS in the Database Preview Environment

PostgreSQL version 11 contains several improvements that are described in [PostgreSQL 11 Released!](#)

For information on the Database Preview Environment, see [the section called “Working with the Database Preview Environment” \(p. 1026\)](#). To access the Preview Environment from the console, select <https://console.aws.amazon.com/rds-preview/>.

PostgreSQL Version 10.6 on Amazon RDS

PostgreSQL version 10.6 contains several bug fixes for issues in release 10.5. For more information on the fixes in PostgreSQL 10.6, see the [PostgreSQL documentation](#).

This version also includes the following changes:

- A new `rds.restrict_password_commands` parameter and a new `rds_password` role have been introduced. When the `rds.restrict_password_commands` parameter is enabled, only users who have the `rds_password` role can make user password and password expiration changes. By restricting password-related operations to a limited set of roles, you can implement policies such as password complexity requirements from the client side. The `rds.restrict_password_commands` parameter is static, so it requires a database restart to change it. For more information, see [Restricting Password Management \(p. 1026\)](#).
- The logical decoding plugin `wal2json` has been updated to commit 9e962ba.

For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

Note

Amazon RDS for PostgreSQL has announced the removal of the `tsearch2` extension in the next major release. We encourage customers still using pre-8.3 text search to migrate to the equivalent built-in features. For more information about migrating, see the [PostgreSQL documentation](#).

PostgreSQL Version 10.5 on Amazon RDS

PostgreSQL version 10.5 contains several bug fixes for issues in release 10.4. For more information on the fixes in 10.5, see the [PostgreSQL documentation](#).

This version also includes the following changes:

- Support for the `pglogical` extension version 2.2.0. Prerequisites for using this extension are the same as the prerequisites for using logical replication for PostgreSQL as described in [Logical Replication for PostgreSQL on Amazon RDS \(p. 1064\)](#).
- Support for the `pg_similarity` extension version 1.0.
- Support for the `pageinspect` extension version 1.6.

- Support for the `libprotobuf` extension version 1.3.0 for the PostGIS component.
- An update for the `pg_hint_plan` extension to version 1.3.1.
- An update for the `wal2json` extension to version 01c5c1e.

For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 10.4 on Amazon RDS

PostgreSQL version 10.4 contains several bug fixes for issues in release 10.3. For more information on the fixes in 10.4, see the [PostgreSQL documentation](#).

This version also includes the following changes:

- Support for PostgreSQL 10 Logical Replication using the native publication and subscription framework. RDS PostgreSQL databases can function as both publishers and subscribers. You can specify replication to other PostgreSQL databases at the database-level or at the table-level. With logical replication, the publisher and subscriber databases need not be physically identical (block-to-block) to each other. This allows for use cases such as data consolidation, data distribution, and data replication across different database versions for 10.4 and above. For more details, refer to [Logical Replication for PostgreSQL on Amazon RDS \(p. 1064\)](#).
- The temporary file size limitation is user-configurable. You require the `rds_superuser` role to modify the `temp_file_limit` parameter.
- Update of the GDAL library, which is used by the PostGIS extension. See [Working with PostGIS \(p. 1020\)](#).
- Update of the `ip4r` extension to version 2.1.1.
- Update of the `pg_repack` extension to version 1.4.3. See [Working with the pg_repack Extension \(p. 1019\)](#).
- Update of the `plv8` extension to version 2.1.2.

For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

Note

The `tsearch2` extension is to be removed in the next major release. We encourage customers still using pre-8.3 text search to migrate to the equivalent built-in features. For more information about migrating, see the [PostgreSQL documentation](#).

PostgreSQL Version 10.3 on Amazon RDS

PostgreSQL version 10.3 contains several bug fixes for issues in release 10. For more information on the fixes in 10.3, see the [PostgreSQL documentation](#).

Version 2.1.0 of PL/v8 is now available. If you use PL/v8 and upgrade PostgreSQL to a new PL/v8 version, you immediately take advantage of the new extension but the catalog metadata doesn't reflect this fact. For the steps to synchronize your catalog metadata with the new version of PL/v8, see [Upgrade PL/v8 \(p. 1061\)](#).

For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 10.1 on Amazon RDS

PostgreSQL version 10.1 contains several bug fixes for issues in release 10. For more information on the fixes in 10.1, see the [PostgreSQL documentation](#) and the [PostgreSQL 10 community announcement](#).

For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL version 10.1 includes the following changes:

- **Declarative table partitioning** – PostgreSQL 10 adds table partitioning to SQL syntax and native tuple routing.
- **Parallel queries** – When you create a new PostgreSQL 10.1 instance, parallel queries are enabled for the `default.postgres10` parameter group. The parameter `max_parallel_workers_per_gather` is set to 2 by default, but you can modify it to support your specific workload requirements.
- **Support for the International Components for Unicode (ICU)** – You can use the ICU library to provide explicitly versioned collations. Amazon RDS for PostgreSQL 10.1 is compiled with ICU version 60.2. For more information about ICU implementation in PostgreSQL, see [Collation Support](#).
- **Huge pages** – Huge pages is a feature of the Linux kernel that uses multiple page size capabilities of modern hardware architectures. Amazon RDS for PostgreSQL supports huge pages with a global configuration parameter. When you create a new PostgreSQL 10.1 instance with RDS, the `huge_pages` parameter is set to "on" for the `default.postgres10` parameter group. You can modify this setting to support your specific workload requirements.
- **PL/v8 update** – PL/v8 is a procedural language that allows you to write functions in JavaScript that you can then call from SQL. This release of PostgreSQL supports version 2.1.0 of PL/v8.
- **Renaming of xlog and location** – In PostgreSQL version 10 the abbreviation "xlog" has changed to "wal", and the term "location" has changed to "lsn". For more information, see <https://www.postgresql.org/docs/10/static/release-10.html#id-1.11.6.8.4>.
- **tsearch2 module** – Amazon RDS continues to provide the `tsearch2` module in PostgreSQL version 10, but is to remove it in the next major version release. If your application uses `tsearch2` functions update it to use the equivalent functions the core engine provides. For more information about using `tsearch2`, see [tsearch2 module](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.6.11 on Amazon RDS

PostgreSQL version 9.6.11 contains several bug fixes for issues in release 9.6.10. For more information on the fixes in PostgreSQL 9.6.11, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

With this version, the logical decoding plugin `wal2json` has been updated to commit 9e962ba.

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.6.10 on Amazon RDS

PostgreSQL version 9.6.10 contains several bug fixes for issues in release 9.6.9. For more information on the fixes in 9.6.10, see the [PostgreSQL documentation](#).

This version includes the following changes:

- Support for the `pglogical` extension version 2.2.0. Prerequisites for using this extension are the same as the prerequisites for using logical replication for PostgreSQL as described in [Logical Replication for PostgreSQL on Amazon RDS \(p. 1064\)](#).
- Support for the `pg_similarty` extension version 2.2.0.
- An update for the `wal2json` extension to version 01c5c1e.
- An update for the `pg_hint_plan` extension to version 1.2.3.

For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.6.9 on Amazon RDS

PostgreSQL version 9.6.9 contains several bug fixes for issues in release 9.6.8. For more information on the fixes in 9.6.9, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

This version includes the following changes:

- The temporary file size limitation is user-configurable. You require the `rds_superuser` role to modify the `temp_file_limit` parameter.
 - Update of the `GDAL` library, which is used by the PostGIS extension. See [Working with PostGIS \(p. 1020\)](#).
 - Update of the `ip4r` extension to version 2.1.1.
 - Update of the `pgaudit` extension to version 1.1.1. See [Working with the pgaudit Extension \(p. 1018\)](#).
- Update of the `pg_repack` extension to version 1.4.3. See [Working with the pg_repack Extension \(p. 1019\)](#).
- Update of the `plv8` extension to version 2.1.2.

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.6.8 on Amazon RDS

PostgreSQL version 9.6.8 contains several bug fixes for issues in release 9.6.6. For more information on the fixes in 9.6.8, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.6.6 on Amazon RDS

PostgreSQL version 9.6.6 contains several bug fixes for issues in release 9.6.5. For more information on the fixes in 9.6.6, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

This version includes the following features:

- Supports the `orafce` extension, version 3.6.1. This extension contains functions that are native to commercial databases, and can be helpful if you are porting a commercial database to

PostgreSQL. For more information about using orafce with Amazon RDS, see [Working with the orafce Extension \(p. 1023\)](#).

- Supports the `prefix` extension, version 1.2.6. This extension provides an operator for text prefix searches. For more information about `prefix`, see the [prefix project on GitHub](#).
- Supports version 2.3.4 of PostGIS, version 2.4.2 of pgrouting, and an updated version of wal2json.

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.6.5 on Amazon RDS

PostgreSQL version 9.6.5 contains several bug fixes for issues in release 9.6.4. For more information on the fixes in 9.6.5, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

This version also includes support for the `pgrouting` and `postgresql-hll` extensions, and the `decoder_raw` optional module.

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.6.3 on Amazon RDS

PostgreSQL version 9.6.3 contains several new features and bug fixes. This version includes the following features:

- Supports the extension `pg_repack` version 1.4.0. You can use this extension to remove bloat from tables and indexes. For more information on using `pg_repack` with Amazon RDS, see [Working with the pg_repack Extension \(p. 1019\)](#).
- Supports the extension `pgaudit` version 1.1.0. This extension provides detailed session and object audit logging. For more information on using `pgaudit` with Amazon RDS, see [Working with the pgaudit Extension \(p. 1018\)](#).
- Supports `wal2json`, an output plugin for logical decoding.
- Supports the `auto_explain` module. You can use this module to log execution plans of slow statements automatically. The following example shows how to use `auto_explain` from within an Amazon RDS PostgreSQL session:

```
LOAD '$libdir/plugins/auto_explain';
```

For more information on using `auto_explain`, see the [PostgreSQL documentation](#).

PostgreSQL Version 9.6.2 on Amazon RDS

PostgreSQL version 9.6.2 contains several new features and bug fixes. The new version also includes the following extension versions:

- PostGIS version 2.3.2
- `pg_freespacemap` version 1.1—Provides a way to examine the free space map (FSM). This extension provides an overloaded function called `pg_freespace`. The functions show the value recorded in the free space map for a given page, or for all pages in the relation.
- `pg_hint_plan` version 1.1.3—Provides control of execution plans by using hinting phrases at the beginning of SQL statements.

- **log_fdw** version 1.0—Using this extension from Amazon RDS, you can load and query your database engine log from within the database. For more information, see [Using the log_fdw Extension \(p. 1060\)](#).
- With this version release, you can now edit the `max_worker_processes` parameter in a DB parameter group.

PostgreSQL version 9.6.2 on Amazon RDS also supports altering enum values. For more information, see [ALTER ENUM for PostgreSQL \(p. 1067\)](#).

For more information on the fixes in 9.6.2, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.6.1 on Amazon RDS

PostgreSQL version 9.6.1 contains several new features and improvements. For more information about the fixes and improvements in PostgreSQL 9.6.1, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#). For information about performing parallel queries and phrase searching using Amazon RDS for PostgreSQL 9.6.1, see the [AWS Database Blog](#).

PostgreSQL version 9.6.1 includes the following changes:

- **Parallel query execution:** Supports parallel execution of large read-only queries, allowing sequential scans, hash joins, nested loops, and aggregates to be run in parallel. By default, parallel query execution is not enabled. To enable parallel query execution, set the parameter `max_parallel_workers_per_gather` to a value larger than zero.
- **Updated postgres_fdw extension:** Supports remote JOINs, SORTs, UPDATEs, and DELETE operations.
- **PL/v8 update:** Provides version 1.5.3 of the PL/v8 language.
- **PostGIS version update:** Supports POSTGIS="2.3.0 r15146" GEOS="3.5.0-CAPI-1.9.0 r4084" PROJ="Rel. 4.9.2, 08 September 2015" GDAL="GDAL 2.1.1, released 2016/07/07" LIBXML="2.9.1" LIBJSON="0.12" RASTER
- **Vacuum improvement:** Avoids scanning pages unnecessarily during vacuum freeze operations.
- **Full-text search support for phrases:** Supports the ability to specify a phrase-search query in tsquery input using the new operators <-> and <N>.
- **Two new extensions are supported:**
 - `bloom`, an index access method based on [Bloom filters](#)
 - `pg_visibility`, which provides a means for examining the visibility map and page-level visibility information of a table.
- With the release of version 9.6.2, you can now edit the `max_worker_processes` parameter in a PostgreSQL version 9.6.1 DB parameter group.

You can create a new PostgreSQL 9.6.1 database instance using the AWS Management Console, AWS CLI, or RDS API. You can also upgrade an existing PostgreSQL 9.5 instance to version 9.6.1 using major version upgrade. If you want to upgrade a DB instance from version 9.3 or 9.4 to 9.6, you must perform a point-and-click upgrade to the next major version first. Each upgrade operation involves a short period of unavailability for your DB instance.

PostgreSQL Version 9.5.15 on Amazon RDS

PostgreSQL version 9.5.15 contains several bug fixes for issues in release 9.5.14. For more information on the fixes in 9.5.15, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.5.14 on Amazon RDS

PostgreSQL version 9.5.14 contains several bug fixes for issues in release 9.5.13. For more information on the fixes in 9.5.14, see the [PostgreSQL documentation](#).

For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.5.13 on Amazon RDS

PostgreSQL version 9.5.13 contains several bug fixes for issues in release 9.5.12. For more information on the fixes in 9.5.13, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

This version includes the following extension updates:

- Update of the `pgaudit` extension to version 1.0.6. See [Working with the pgaudit Extension \(p. 1018\)](#).
- Update of the `pg_hint_plan` extension to version 1.1.5.
- Update of the `plv8` extension to version 2.1.2.

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.5.12 on Amazon RDS

PostgreSQL version 9.5.12 contains several bug fixes for issues in release 9.5.10. For more information on the fixes in 9.5.12, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.5.10 on Amazon RDS

PostgreSQL version 9.5.10 contains several bug fixes for issues in version 9.5.9. For more information on the fixes in 9.5.10, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.5.9 on Amazon RDS

PostgreSQL version 9.5.9 contains several bug fixes for issues in version 9.5.8. For more information on the fixes in 9.5.9, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.5.7 on Amazon RDS

PostgreSQL version 9.5.7 contains several new features and bug fixes. This version includes the following features:

- Supports the extension `pgaudit` version 1.0.5. This extension provides detailed session and object audit logging. For more information on using `pgaudit` with Amazon RDS, see [Working with the pgaudit Extension \(p. 1018\)](#).
- Supports `wal2json`, an output plugin for logical decoding.
- Supports the `auto_explain` module. You can use this module to log execution plans of slow statements automatically. The following example shows how to use `auto_explain` from within an Amazon RDS PostgreSQL session.

```
LOAD '$libdir/plugins/auto_explain';
```

For more information on using `auto_explain`, see the [PostgreSQL documentation](#).

PostgreSQL Version 9.5.6 on Amazon RDS

PostgreSQL version 9.5.6 contains several new features and bug fixes. The new version also includes the following extension versions:

- PostGIS version 2.2.5
- `pg_freespacemap` version 1.1—Provides a way to examine the free space map (FSM). This extension provides an overloaded function called `pg_freespace`. This function shows the value recorded in the free space map for a given page, or for all pages in the relation.
- `pg_hint_plan` version 1.1.3—Provides control of execution plans by using hinting phrases at the beginning of SQL statements.

PostgreSQL version 9.5.6 on Amazon RDS also supports altering enum values. For more information, see [ALTER ENUM for PostgreSQL \(p. 1067\)](#).

For more information on the fixes in 9.5.6, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.5.4 on Amazon RDS

PostgreSQL version 9.5.4 contains several fixes to issue found in previous versions. For more information on the fixes in 9.5.4, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

Beginning with PostgreSQL version 9.4, PostgreSQL supports the streaming of WAL changes using logical replication decoding. Amazon RDS supports logical replication for PostgreSQL version 9.4.9 and higher and 9.5.4 and higher. For more information about PostgreSQL logical replication on Amazon RDS, see [Logical Replication for PostgreSQL on Amazon RDS \(p. 1064\)](#).

Beginning with PostgreSQL version 9.5.4 for Amazon RDS, the command `ALTER USER WITH BYPASSRLS` is supported.

PostgreSQL versions 9.4.9 and later and version 9.5.4 and later support event triggers, and Amazon RDS supports event triggers for these versions. You can use the master user account can be used to create, modify, rename, and delete event triggers. Event triggers are at the DB instance level, so they can apply

to all databases on an instance. For more information about PostgreSQL event triggers on Amazon RDS, see [Event Triggers for PostgreSQL on Amazon RDS \(p. 1065\)](#).

PostgreSQL Version 9.5.2 on Amazon RDS

PostgreSQL version 9.5.2 contains several fixes to issues found in previous versions. For more information on the features in 9.5.2, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL version 9.5.2 doesn't support the db.m1 or db.m2 DB instance classes. If you need to upgrade a DB instance running PostgreSQL version 9.4 to version 9.5.2 to one of these instance classes, you need to scale compute. To do that, you need a comparable db.t2 or db.m3 DB instance class before you can upgrade a DB instance running PostgreSQL version 9.4 to version 9.5.2. For more information on DB instance classes, see [DB Instance Class \(p. 82\)](#).

Native PostgreSQL version 9.5.2 introduced the command ALTER USER WITH BYPASSRLS.

This release includes updates from previous versions, including the following:

- **CVE-2016-2193:** Fixes an issue where a query plan might be reused for more than one ROLE in the same session. Reusing a query plan can cause the query to use the wrong set of Row Level Security (RLS) policies.
- **CVE-2016-3065:** Fixes a server crash bug triggered by using pageinspect with BRIN index pages. Because an attacker might be able to expose a few bytes of server memory, this crash is being treated as a security issue.

Major enhancements in RDS PostgreSQL 9.5 include the following:

- UPSERT: Allow INSERTs that would generate constraint conflicts to be turned into UPDATEs or ignored
- Add the GROUP BY analysis features GROUPING SETS, CUBE, and ROLLUP
- Add row-level security control
- Create mechanisms for tracking the progress of replication, including methods for identifying the origin of individual changes during logical replication
- Add Block Range Indexes (BRIN)
- Add substantial performance improvements for sorting
- Add substantial performance improvements for multi-CPU machines
- PostGIS 2.2.2 - To use this latest version of PostGIS, use the ALTER EXTENSION UPDATE statement to update after you upgrade to version 9.5.2. Example:

```
ALTER EXTENSION POSTGIS UPDATE TO '2.2.2'
```

- Improved visibility of autovacuum sessions by allowing the rds_superuser account to view autovacuum sessions in pg_stat_activity. For example, you can identify and terminate an autovacuum session that is blocking a command from running, or executing slower than a manually issued vacuum command.

RDS PostgreSQL version 9.5.2 includes the following new extensions:

- **address_standardizer** – A single-line address parser that takes an input address and normalizes it based on a set of rules stored in a table, helper lex, and gaz tables.
- **hstore_plperl** – Provides transforms for the hstore type for PL/Perl.
- **tsm_system_rows** – Provides the table sampling method SYSTEM_ROWS, which can be used in the TABLESAMPLE clause of a SELECT command.
- **tsm_system_time** – Provides the table sampling method SYSTEM_TIME, which can be used in the TABLESAMPLE clause of a SELECT command.

PostgreSQL Version 9.4.20 on Amazon RDS

PostgreSQL version 9.4.20 contains several bug fixes for issues in release 9.4.19. For more information on the fixes in 9.4.20, the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.4.19 on Amazon RDS

PostgreSQL version 9.4.19 contains several bug fixes for issues in release 9.4.18. For more information on the fixes in 9.4.19, see the [PostgreSQL documentation](#).

For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.4.18 on Amazon RDS

PostgreSQL version 9.4.18 contains several bug fixes for issues in release 9.4.17. For more information on the fixes in 9.4.18, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

The plv8 extension has been updated to version 2.1.2. For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.4.17 on Amazon RDS

PostgreSQL version 9.4.17 contains several bug fixes for issues in release 9.4.15. For more information on the fixes in 9.4.17, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.4.15 on Amazon RDS

PostgreSQL version 9.4.15 contains several bug fixes for issues in release 9.4.14. For more information on the fixes in 9.4.15, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.4.14 on Amazon RDS

PostgreSQL version 9.4.14 contains several bug fixes for issues in release 9.4.12. For more information on the fixes in 9.4.14, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.4.12 on Amazon RDS

PostgreSQL version 9.4.12 contains several fixes to issue found in previous versions.

For more information on the fixes in 9.4.12, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.4.11 on Amazon RDS

PostgreSQL version 9.4.11 contains several fixes to issue found in previous versions.

For more information on the fixes in 9.4.11, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

Beginning with PostgreSQL version 9.4, PostgreSQL supports the streaming of WAL changes using logical replication decoding. Amazon RDS supports logical replication for PostgreSQL version 9.4.9 and higher and 9.5.4 and higher. For more information about PostgreSQL logical replication on Amazon RDS, see [Logical Replication for PostgreSQL on Amazon RDS \(p. 1064\)](#).

PostgreSQL versions 9.4.9 and later and version 9.5.4 and later support event triggers, and Amazon RDS supports event triggers for these versions. The master user account can be used to create, modify, rename, and delete event triggers. Event triggers are at the DB instance level, so they can apply to all databases on an instance. For more information about PostgreSQL event triggers on Amazon RDS, see [Event Triggers for PostgreSQL on Amazon RDS \(p. 1065\)](#).

PostgreSQL Version 9.4.9 on Amazon RDS

PostgreSQL version 9.4.9 contains several fixes to issue found in previous versions. For more information on the fixes in 9.4.9, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

Beginning with PostgreSQL version 9.4, PostgreSQL supports the streaming of WAL changes using logical replication decoding. Amazon RDS supports logical replication for PostgreSQL version 9.4.9 and higher and 9.5.4 and higher. For more information about PostgreSQL logical replication on Amazon RDS, see [Logical Replication for PostgreSQL on Amazon RDS \(p. 1064\)](#).

PostgreSQL versions 9.4.9 and later and version 9.5.4 and later support event triggers, and Amazon RDS supports event triggers for these versions. The master user account can be used to create, modify, rename, and delete event triggers. Event triggers are at the DB instance level, so they can apply to all databases on an instance. For more information about PostgreSQL event triggers on Amazon RDS, see [Event Triggers for PostgreSQL on Amazon RDS \(p. 1065\)](#).

PostgreSQL Version 9.4.7 on Amazon RDS

PostgreSQL version 9.4.7 contains several fixes to issue found in previous versions. For more information on the fixes in 9.4.7, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL version 9.4.7 includes improved visibility of autovacuum sessions by allowing the rds_superuser account to view autovacuum sessions in pg_stat_activity. For example, you can identify and terminate an autovacuum session that is blocking a command from running, or executing slower than a manually issued vacuum command.

PostgreSQL Version 9.3.25 on Amazon RDS

Note

Amazon RDS for PostgreSQL announced retirement of PostgreSQL 9.3 in September 2018 and has stopped support for PostgreSQL version 9.3. We encourage you to upgrade to PostgreSQL 9.4 or a higher version as early as possible. For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL version 9.3.25 contains several bug fixes for issues in release 9.3.24. For more information on the fixes in 9.3.25, see the [PostgreSQL documentation](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.3.24 on Amazon RDS

PostgreSQL version 9.3.24 contains several bug fixes for issues in release 9.3.23. For more information on the fixes in 9.3.24, see the [PostgreSQL documentation](#).

Note

Amazon RDS for PostgreSQL has deprecated PostgreSQL version 9.3.x.

For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.3.23 on Amazon RDS

PostgreSQL version 9.3.23 contains several bug fixes for issues in release 9.3.22. For more information on the fixes in 9.3.23, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

The plv8 extension has been updated to version 2.1.2. For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

Note

Amazon RDS for PostgreSQL has announced version 9.3.x is scheduled for retirement in September, 2018. We encourage you to upgrade your 9.3.x databases to the latest version at your earliest convenience.

PostgreSQL Version 9.3.22 on Amazon RDS

PostgreSQL version 9.3.22 contains several bug fixes for issues in release 9.3.20. For more information on the fixes in 9.3.22, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

For the complete list of extensions supported by Amazon RDS for PostgreSQL, see [Supported PostgreSQL Features and Extensions \(p. 1043\)](#).

PostgreSQL Version 9.3.20 on Amazon RDS

PostgreSQL version 9.3.20 contains several bug fixes for issues in version 9.3.19. For more information on the fixes in 9.3.20, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.3.19 on Amazon RDS

PostgreSQL version 9.3.19 contains several bug fixes for issues in version 9.3.18. For more information on the fixes in 9.3.19, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.3.17 on Amazon RDS

PostgreSQL version 9.3.17 contains several fixes for bugs found in previous versions. This version contains the same extension components as version 9.3.16. For a list of fixes in version 9.3.17, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.3.16 on Amazon RDS

PostgreSQL version 9.3.16 contains several fixes for bugs found in previous versions. This version contains the same extension components as version 9.3.14. For a list of fixes in version 9.3.16, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.3.14 on Amazon RDS

PostgreSQL version 9.3.14 contains several fixes for bugs found in previous versions. For a list of fixes in version 9.3.14, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL Version 9.3.12 on Amazon RDS

PostgreSQL version 9.3.12 contains several fixes for bugs found in previous versions. For a list of fixes in version 9.3.12, see the [PostgreSQL documentation](#). For information on upgrading the engine version for your PostgreSQL DB instance, see [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#).

PostgreSQL version 9.3.12 includes improved visibility of autovacuum sessions by allowing the rds_superuser account to view autovacuum sessions in pg_stat_activity. For example, you can identify and terminate an autovacuum session that is blocking a command from running, or executing slower than a manually issued vacuum command.

Supported PostgreSQL Features and Extensions

Amazon RDS supports many of the most common PostgreSQL extensions and features.

Topics

- [PostgreSQL Extensions and Modules Supported on Amazon RDS \(p. 1043\)](#)
- [Upgrade PL/v8 \(p. 1061\)](#)
- [Supported PostgreSQL Features \(p. 1063\)](#)
- [Limits for PostgreSQL DB Instances \(p. 1068\)](#)
- [Upgrading a PostgreSQL DB Instance \(p. 1068\)](#)
- [Using SSL with a PostgreSQL DB Instance \(p. 1068\)](#)

PostgreSQL Extensions and Modules Supported on Amazon RDS

PostgreSQL supports many PostgreSQL extensions and modules. Extensions and modules expand on the functionality provided by the PostgreSQL engine. The following sections show the extensions and modules supported by Amazon RDS for the major PostgreSQL versions.

Topics

- [PostgreSQL Version 11 Extensions and Modules Supported on Amazon RDS \(p. 1044\)](#)
- [PostgreSQL Version 10.x Extensions and Modules Supported on Amazon RDS \(p. 1046\)](#)
- [PostgreSQL Version 9.6.x Extensions and Modules Supported on Amazon RDS \(p. 1049\)](#)
- [PostgreSQL Version 9.5.x Extensions Supported on Amazon RDS \(p. 1051\)](#)
- [PostgreSQL Version 9.4.x Extensions and Modules Supported on Amazon RDS \(p. 1054\)](#)
- [PostgreSQL Version 9.3.x Extensions Supported on Amazon RDS \(p. 1056\)](#)
- [PostgreSQL Extension Support for PostGIS on Amazon RDS \(p. 1058\)](#)
- [Using the log_fdw Extension \(p. 1060\)](#)

You can find a list of extensions supported by Amazon RDS in the default DB parameter group for that PostgreSQL version. You can also see the current extensions list using `psql` by showing the `rds.extensions` parameter as in the following example.

```
SHOW rds.extensions;
```

Note

Parameters added in a minor version release might display inaccurately when using the `rds.extensions` parameter in `psql`.

PostgreSQL Version 11 Extensions and Modules Supported on Amazon RDS

The following tables show PostgreSQL extensions and modules for PostgreSQL version 11 that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension or module is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see [Packaging Related Objects into an Extension](#).

Extension	Version 11
address_standardizer	2.4.2
address_standardizer_data_us	2.4.2
bloom	1.0
btree_gin	1.2
btree_gist	1.5
chkpass	1.0
citext	1.4
cube	1.2
dblink	1.2
dict_int	1.0
dict_xsyn	1.0
earthdistance	1.1
fuzzystrmatch	1.1
hstore	1.4
hstore_plperl	1.0
intagg	1.1
intarray	1.2
ip4r	2.1.1
isn	1.1
log_fdw—see Using the log_fdw Extension (p. 1060)	1.0
libprotobuf	1.3.0

Extension	Version 11
ltree	1.1
orafce	3.6.1
pgaudit	1.2.0
pg_buffercache	1.3
pg_freespacemap	1.2
pg_hint_plan	1.3.1
pg_prewarm	1.1
pg_repack	1.4.3
pg_similarity	1.0
pg_stat_statements	1.5
pg_trgm	1.3
pg_visibility	1.2
pgcrypto	1.3
pageinspect	1.6
pglogical	2.2.0
pgrowlocks	1.2
pgrouting	2.5.2
pgstattuple	1.5
plcoffee	2.1.0
plls	2.1.0
plperl	1.0
plpgsql	1.0
pltcl	1.0
plv8	2.1.2
PostGIS	2.5.0
postgis_tiger_geocoder	2.5.0
postgis_topology	2.5.0
postgres_fdw	1.0
postgresql-hll	2.10.2
prefix	1.2.0
sslinfo	1.2

Extension	Version 11
tablefunc	1.0
test_parser	1.0
tsearch2 (deprecated in version 10)	1.0
tsm_system_rows	1.0
tsm_system_time	1.0
unaccent	1.1
uuid-ossp	1.1

The following modules are supported as shown for PostgreSQL version 11.

Module	Version 11
amcheck	Supported
auto_explain	Supported
decoder_raw	Supported
ICU	Version 60.2 supported
test_decoder	Supported
wal2json	Commit hash 01c5c1e

PostgreSQL Version 10.x Extensions and Modules Supported on Amazon RDS

The following tables show PostgreSQL extensions and modules for PostgreSQL version 10 that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension or module is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see [Packaging Related Objects into an Extension](#).

Extension	10.1	10.3	10.4	10.5	10.6
address_standard	2.4.2	2.4.2	2.4.2	2.4.2	2.4.2
address_standard	2.4.2	2.4.2	2.4.2	2.4.2	2.4.2
bloom	1.0	1.0	1.0	1.0	1.0
btree_gin	1.2	1.2	1.2	1.2	1.2
btree_gist	1.5	1.5	1.5	1.5	1.5
chkpass	1.0	1.0	1.0	1.0	1.0
citext	1.4	1.4	1.4	1.4	1.4
cube	1.2	1.2	1.2	1.2	1.2

Extension	10.1	10.3	10.4	10.5	10.6
dblink	1.2	1.2	1.2	1.2	1.2
dict_int	1.0	1.0	1.0	1.0	1.0
dict_xsyn	1.0	1.0	1.0	1.0	1.0
earthdistance	1.1	1.1	1.1	1.1	1.1
fuzzystrmatch	1.1	1.1	1.1	1.1	1.1
hstore	1.4	1.4	1.4	1.4	1.4
hstore_plperl	1.0	1.0	1.0	1.0	1.0
intagg	1.1	1.1	1.1	1.1	1.1
intarray	1.2	1.2	1.2	1.2	1.2
ip4r	2.0	2.0	2.1. 12.1.1	2.1.1	
isn	1.1	1.1	1.1	1.1	1.1
log_fdw— see Using the log_fdw Extension (p. 1060)	1.0	1.0	1.0	1.0	1.0
libprotobuf	N/A	N/ A	N/ A	1.3.0	1.3.0
ltree	1.1	1.1	1.1	1.1	1.1
orafce	3.6.1	3.6.1	3.6.1	3.6.1	3.6.1
pgaudit	1.2.0	1.2.0	1.2.0	1.2.0	1.2.0
pg_buffercache	1.3	1.3	1.3	1.3	1.3
pg_freespacemap	1.2	1.2	1.2	1.2	1.2
pg_hint_plan	1.3.0	1.3.0	1.3.0	1.3.1	1.3.1
pg_prewarm	1.1	1.1	1.1	1.1	1.1
pg_repack	1.4.2	1.4.2	1.4. 31.4.3	1.4.3	1.4.3
pg_similarity	N/A	N/ A	N/ A	1.0	1.0
pg_stat_statements	1.5	1.5	1.5	1.5	1.5
pg_trgm	1.3	1.3	1.3	1.3	1.3
pg_visibility	1.2	1.2	1.2	1.2	1.2
pgcrypto	1.3	1.3	1.3	1.3	1.3
pageinspect	N/A	N/ A	N/ A	1.6	1.6

Extension	10.1	10.3	10.4	10.5	10.6
pglogical	N/A	N/ A	N/ A	2.2.0	2.2.0
pgrowlocks	1.2	1.2	1.2	1.2	1.2
pgrouting	2.5.2	2.5.2	2.5.2	2.5.2	2.5.2
pgstattuple	1.5	1.5	1.5	1.5	1.5
plcoffee	2.1.0	2.1.0	2.1.2	2.1.2	
plls	2.1.0	2.1.0	2.1.2	2.1.2	
plperl	1.0	1.0	1.0	1.0	1.0
plpgsql	1.0	1.0	1.0	1.0	1.0
pltcl	1.0	1.0	1.0	1.0	1.0
plv8	2.1.0	2.1.0	2.1.2	2.1.2	
PostGIS	2.4.2	2.4.2	2.4.4	2.4.4	
postgis_tiger_geometry	2.4.2	2.4.2	2.4.2	2.4.2	
postgis_topology	2.4.2	2.4.2	2.4.2	2.4.2	
postgres_fdw	1.0	1.0	1.0	1.0	1.0
postgresql-hll		2.10.22.10.2	2.10.2.10.2	2.10.22.10.2	
prefix	1.2.0	1.2.0	1.2.0	1.2.0	1.2.0
sslinfo	1.2	1.2	1.2	1.2	1.2
tablefunc	1.0	1.0	1.0	1.0	1.0
test_parser	1.0	1.0	1.0	1.0	1.0
tsearch2 (deprecated in version 10)	1.0	1.0	1.0	1.0	1.0
tsm_system_rows	1.0	1.0	1.0	1.0	1.0
tsm_system_time	1.0	1.0	1.0	1.0	1.0
unaccent	1.1	1.1	1.1	1.1	1.1
uuid-ossp	1.1	1.1	1.1	1.1	1.1

The `tsearch2` extension is deprecated in version 10. The PostgreSQL team plans to remove `tsearch2` from the next major release of PostgreSQL.

The following modules are supported as shown for versions of PostgreSQL 10.

Module	Version 10.1	10.3		10.4		10.5		10.6
amcheck	Not Supported	Supported		Supported		Supported		Supported
auto_explain	Supported	Supported		Supported		Supported		Supported
decoder	Supported	Supported		Supported		Supported		Supported
ICU	Version 60.2 supported	Version 60.2 supported		Version 60.2 supported		Version 60.2 supported		Version 60.2 supported
test_decoder	Supported	Supported		Supported		Supported		Supported
wal2json	Commit hash 5352cc4	Commit hash 5352cc4		Commit hash 5352cc4		Commit hash 01c5c1e		Commit hash 9e962ba

PostgreSQL Version 9.6.x Extensions and Modules Supported on Amazon RDS

The following tables show PostgreSQL extensions and modules for PostgreSQL version 9.6.x that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension or module is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see [Packaging Related Objects into an Extension](#).

Extension	9.6.1	9.6.2	9.6.3	9.6.4	9.6.6	9.6.8	9.6.9	9.6.10	9.6.11
address_standard	2.3.0	2.3.2	2.3.2	2.3.2	2.3.4	2.3.4	2.3.4	2.3.4	2.3.4
address_standard	2.3.0	2.3.2	2.3.2	2.3.2	2.3.4	2.3.4	2.3.4	2.3.4	2.3.4
bloom	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
btree_gin	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
btree_gist	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
chkpass	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
citext	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
cube	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
dblink	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
dict_int	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
dict_xsyn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
earthdistance	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
fuzzystrmatch	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
hstore	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
hstore_plperl	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Extension	9.6.1	9.6.2	9.6.3	9.6.4	9.6.6	9.6.8	9.6.9	9.6.10	9.6.11
intagg	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
intarray	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
ip4r	2.0	2.0	2.0	2.0	2.0	2.0	2.1.1	2.1.1	2.1.1
isn	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
log_fdw— see Using the log_fdw Extension (p. 1060)	N/ A	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
ltree	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
orafce	N/ A	N/A	N/ A	N/ A	3.6.1	3.6.1	3.6.1	3.6.1	3.6.1
pgaudit	N/ A	N/A	1.1	1.1	1.1	1.1	1.1.1	1.1.1	1.1.1
pg_buffercache	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
pg_freespacemap	N/ A	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pg_hint_plan	N/ A	1.1.3	1.1.3	1.1.3	1.1.3	1.2.2	1.2.2	1.2.3	1.2.3
pg_prewarm	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pg_repack	N/ A	N/A	1.4.0	1.4.1	1.4.2	1.4.2	1.4.3	1.4.3	1.4.3
pg_similarity	N/ A	N/A	N/ A	N/ A	N/A	N/A	N/ A	1.0	1.0
pg_stat_statement	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
pg_trgm	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
pg_visibility	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pgcrypto	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
pglogical	N/ A	N/A	N/ A	N/ A	N/A	N/A	N/ A	2.2.0	2.2.0
pgrowlocks	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
pgrouting	N/ A	N/A	N/ A	2.3.2	2.4.2	2.4.2	2.4.2	2.4.2	2.4.2
pgstattuple	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
plcoffee	1.5.3	1.5.3	1.5.3	1.5.3	2.1.2	2.1.2	2.1.2	2.1.2	2.1.2
plls	1.5.3	1.5.3	1.5.3	1.5.3	2.1.2	2.1.2	2.1.2	2.1.2	2.1.2
plperl	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Extension	9.6.1	9.6.2	9.6.3	9.6.4	9.6.6	9.6.8	9.6.9	9.6.10	9.6.11
plpgsql	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pltcl	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
plv8	1.5.3	1.5.3	1.5.3	1.5.3	1.5.3	2.1.0	2.1.2	2.1.2	2.1.2
PostGIS	2.3.0	2.3.2	2.3.2	2.3.2	2.3.4	2.3.4	2.3.7	2.3.7	2.3.7
postgis_tiger_geotab	2.3.0	2.3.2	2.3.2	2.3.2	2.3.4	2.3.4	2.3.4	2.3.4	2.3.4
postgis_topology	2.3.0	2.3.2	2.3.2	2.3.2	2.3.4	2.3.4	2.3.4	2.3.4	2.3.4
postgres_fdw	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
postgresql-hll	N/A	N/A	N/A	N/A	2.10.2	2.10.2	2.10.2	2.10.2	2.10.2
prefix	N/A	N/A	N/A	N/A	1.2.6	1.2.6	1.2.6	1.2.6	1.2.6
sslinfo	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
tablefunc	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
test_parser	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tsearch2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tsm_system_row	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tsm_system_time	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
unaccent	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
uuid-ossp	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1

The following modules are supported as shown for versions of PostgreSQL 9.6.

Module	9.6.1	9.6.2	9.6.3	9.6.5	9.6.8	9.6.9	9.6.10	9.6.11
auto_explain	N/A	N/A	Supported	Supported	Supported	Supported	Supported	Supported
decoder_raw	N/A	N/A	N/A	Supported	Supported	Supported	Supported	Supported
test_decoder	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported
wal2json	N/A	N/a	Commit hash 2828409	Commit hash 645ab69	Commit hash 5352cc4	Commit hash 5352cc4	Commit hash 01c5c1e	Commit hash 9e962ba

PostgreSQL Version 9.5.x Extensions Supported on Amazon RDS

The following tables show PostgreSQL extensions and modules for PostgreSQL version 9.5.x that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension or module is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see [Packaging Related Objects into an Extension](#).

Extension	9.5	9.5	9.5	9.5	9.5	9.5.1	9.5.1	9.5.1	9.5.1	9.5.15
address_standard	2.1	2.2	2.2	2.2	2.5	2.5	2.5	2.5	2.5	2.5
address_standard	2.1	2.2	2.2	2.2	2.5	2.5	2.5	2.5	2.5	2.5
bloom	N/ A	N/A	N/A	N/A						
btree_gin	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
btree_gist	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
chkpass	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
citext	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
cube	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
dblink	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
dict_int	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
dict_xsyn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
earthdistance	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
fuzzystrmatch	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
hstore	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
hstore_plperl	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
intagg	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
intarray	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
ip4r	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
isn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
log_fdw— see Using the log_fdw Extension (p. 1060)	N/ A	N/A	N/A	N/A						
ltree	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pgaudit	N/ A	N/ A	N/ A	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pg_buffercache	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pg_freespacemap	N/ A	N/ A	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pg_hint_plan	N/ A	N/ A	1.1	1.3	1.3	1.3	1.3	1.5	1.5	1.5
pg_prewarm	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pg_stat_statements	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3

The following modules are supported as shown for versions of PostgreSQL 9.5.

Module	9.5.2	9.5.4	9.5.6	9.5.7	9.5.9	9.5.12	9.5.13	9.5.14	9.5.15
wal2json	N/A	N/a	N/A	Commit hash 2828409					

PostgreSQL Version 9.4.x Extensions and Modules Supported on Amazon RDS

The following tables show the PostgreSQL extensions and modules for PostgreSQL version 9.4.x that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension or module is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see [Packaging Related Objects into an Extension](#).

Extension	9.4	9.4	9.4	9.4.	9.4.	9.4.	9.4.	9.4.	9.4.1	9.4.20
address_standardizer	N/ A	N/A								
address_standardizer_data	N/ A	N/A								
bloom	N/ A	N/A								
btree_gin	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
btree_gist	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
chkpass	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
citext	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
cube	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
dblink	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
dict_int	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
dict_xsyn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
earthdistance	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
fuzzystrmatch	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
hstore	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
hstore_plperl	N/ A	N/A								
intagg	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
intarray	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
ip4r	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
isn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Extension	9.4	9.4	9.4	9.4	9.4	9.4	9.4	9.4	9.4	9.4.20
log_fdw —see Using the log_fdw Extension (p. 1060)	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/A
ltree	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pg_buffercache	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pg_freespacemap	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/A
pg_hint_plan	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/A
pg_prewarm	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pg_stat_statements	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
pg_trgm	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pg_visibility	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/A
pgcrypto	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pgrowlocks	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pgstattuple	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
plcoffee	1.4.4.4.4.4.41.4.41.4.41.4.41.4.41.4.41.4.41.4.41.4.41.4.41.4.4									
plls	1.4.4.4.4.4.41.4.41.4.41.4.41.4.41.4.41.4.41.4.41.4.41.4.4									
plperl	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
plpgsql	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pltcl	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
plv8	1.4.4.4.4.4.41.4.41.4.41.4.41.4.42.1.02.1.22.1.2 2.1.2									
PostGIS	2.1.8.1.8.1.82.1.82.1.82.1.82.1.82.1.82.1.82.1.8 2.1.8									
postgis_tiger_geocoder	2.1.8.1.8.1.82.1.82.1.82.1.82.1.82.1.82.1.82.1.8 2.1.8									
postgis_topology	2.1.8.1.8.1.82.1.82.1.82.1.82.1.82.1.82.1.82.1.8 2.1.8									
postgres_fdw	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
sslinfo	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tablefunc	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
test_parser	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tsearch2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tsm_system_rows	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/ A	N/A

Extension	9.4	9.4	9.4	9.4	9.4	9.4	9.4	9.4	9.4	9.4.20
tsm_system_time	N/ A	N/A								
unaccent	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
uuid-ossp	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

The following modules are supported as shown for versions of PostgreSQL 9.4.

Module	9.4.7	9.4.9	9.4.11	9.4.12	9.4.14	9.4.17
test_decoder	N/A	N/A	N/A	Supported	Supported	Supported

PostgreSQL Version 9.3.x Extensions Supported on Amazon RDS

Note

Amazon RDS for PostgreSQL has deprecated PostgreSQL version 9.3.x. We strongly recommend that you upgrade to a major version, preferably version 9.6.x or 10.x. See [Upgrading the PostgreSQL DB Engine \(p. 988\)](#).

The following table shows PostgreSQL extensions for PostgreSQL version 9.3.x that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see [Packaging Related Objects into an Extension](#).

Extension	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.2	9.3.25
address_standard	N/ A	N/A	N/A							
address_standard_dataloss	N/ A	N/A	N/A							
bloom	N/ A	N/A	N/A							
btree_gin	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
btree_gist	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
chkpass	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
citext	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
cube	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
dblink	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
dict_int	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
dict_xsyn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
earthdistance	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
fuzzystrmatch	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Extension	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.2	9.3.25
hstore	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
hstore_plperl	N/A	N/A	N/A							
intagg	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
intarray	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
ip4r	N/A	N/A	N/A							
isn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
log_fdw—see Using the log_fdw Extension (p. 1060)	N/A	N/A	N/A							
ltree	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pg_buffercache	N/A	N/A	N/A							
pg_freespace	N/A	N/A	N/A							
pg_hint_plan	N/A	N/A	N/A							
pg_prewarm	N/A	N/A	N/A							
pg_stat_statements	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pg_trgm	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pg_visibility	N/A	N/A	N/A							
pgcrypto	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pgrowlocks	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
pgstattuple	N/A	N/A	N/A							
plcoffee	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4
plls	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4
plperl	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
plpgsql	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pltcl	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
plv8	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4	2.1.02	2.1.22	2.1.2
PostGIS	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8

Extension	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.1	9.3.2	9.3.25
postgis_tiger_geod	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8
postgis_topology	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8	2.1.8
postgres_fdw	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
sslinfo	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tablefunc	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
test_parser	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tsearch2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tsm_system_ronly	N/A									
tsm_system_trusted	N/A									
unaccent	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
uuid-ossp	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

PostgreSQL Extension Support for PostGIS on Amazon RDS

Before you can use the PostGIS extension, you must create it by running the following command.

```
CREATE EXTENSION POSTGIS;
```

The following table shows the PostGIS component versions that ship with the Amazon RDS for PostgreSQL versions.

Version	PostGIS	GEOS	GDAL	PROJ
9.3.12	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.4, released 2016/01/25	Rel. 4.9.2, 08 September 2015
9.3.14	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.3.16	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.3.17	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.4.7	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.4, released 2016/01/25	Rel. 4.9.2, 08 September 2015

Version	PostGIS	GEOS	GDAL	PROJ
9.4.9	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.4.11	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.4.12	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.5.2	2.2.2 r14797	3.5.0-CAPI-1.9.0 r4084	GDAL 2.0.2, released 2016/01/26	Rel. 4.9.2, 08 September 2015
9.5.4	2.2.2 r14797	3.5.0-CAPI-1.9.0 r4084	GDAL 2.0.3, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.5.6	2.2.5 r15298	3.5.1-CAPI-1.9.1 r4246	GDAL 2.0.3, released 2016/07/01	Rel. 4.9.3, 15 August 2016
9.5.7	2.2.5 r15298	3.5.1-CAPI-1.9.1 r4246	GDAL 2.0.3, released 2016/07/01	Rel. 4.9.3, 15 August 2016
9.6.1	2.3.0 r15146	3.5.0-CAPI-1.9.0 r4084	GDAL 2.1.1, released 2016/07/07	Rel. 4.9.2, 08 September 2015
9.6.2	2.3.2 r15302	3.5.1-CAPI-1.9.1 r4246	GDAL 2.1.3, released 2017/20/01	Rel. 4.9.3, 15 August 2016
9.6.3	2.3.2 r15302	3.5.1-CAPI-1.9.1 r4246	GDAL 2.1.3, released 2017/20/01	Rel. 4.9.3, 15 August 2016
9.6.6	2.3.4 r16009	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.3, released 2017/20/01	Rel. 4.9.3, 15 August 2016
9.6.8	2.3.4 r16009	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.3, released 2017/20/01	Rel. 4.9.3, 15 August 2016
9.6.9	2.3.7 r16523	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.4, released 2017/06/23	Rel. 4.9.3, 15 August 2016
9.6.10	2.3.7 r16523	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.4, released 2017/06/23	Rel. 4.9.3, 15 August 2016
9.6.11	2.3.7 r16523	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.4, released 2017/06/23	Rel. 4.9.3, 15 August 2016
10.1	2.4.2	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.3, released 2017/20/01	Rel. 4.9.3, 15 August 2016
10.3	2.4.2	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.3, released 2017/20/01	Rel. 4.9.3, 15 August 2016
10.4	2.4.4 r16526	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.4, released 2017/06/23	Rel. 4.9.3, 15 August 2016

Version	PostGIS	GEOS	GDAL	PROJ
10.5	2.4.4 r16526	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.4, released 2017/06/23	Rel. 4.9.3, 15 August 2016
10.6	2.4.4 r16526	3.6.2-CAPI-1.10.2 4d2925d6	GDAL 2.1.4, released 2017/06/23	Rel. 4.9.3, 15 August 2016

Note

PostgreSQL 10.5 added support for the `libprotobuf` extension version 1.3.0 to the PostGIS component.

Using the `log_fdw` Extension

The `log_fdw` extension is new for Amazon RDS for PostgreSQL version 9.6.2 and later. Using this extension, you can access your database engine log using a SQL interface. In addition to viewing the `stderr` log files that are generated by default on RDS, you can view CSV logs (set the `log_destination` parameter to `csvlog`) and build foreign tables with the data neatly split into several columns.

This extension introduces two new functions that make it easy to create foreign tables for database logs:

- `list_postgres_log_files()` – Lists the files in the database log directory and the file size in bytes.
- `create_foreign_table_for_log_file(table_name text, server_name text, log_file_name text)` – Builds a foreign table for the specified file in the current database.

All functions created by `log_fdw` are owned by `rds_superuser`. Members of the `rds_superuser` role can grant access to these functions to other database users.

The following example shows how to use the `log_fdw` extension.

To use the `log_fdw` extension

1. Get the `log_fdw` extension.

```
postgres=> CREATE EXTENSION log_fdw;
CREATE EXTENSION
```

2. Create the log server as a foreign data wrapper.

```
postgres=> CREATE SERVER log_server FOREIGN DATA WRAPPER log_fdw;
CREATE SERVER
```

3. Select all from a list of log files.

```
postgres=> SELECT * from list_postgres_log_files() order by 1;
```

A sample response is as follows.

file_name	file_size_bytes
postgresql.log.2016-08-09-22.csv	1111
postgresql.log.2016-08-09-23.csv	1172
postgresql.log.2016-08-10-00.csv	1744
postgresql.log.2016-08-10-01.csv	1102
(4 rows)	

4. Create a table with a single 'log_entry' column for non-CSV files.

```
postgres=> SELECT create_foreign_table_for_log_file('my_postgres_error_log',
    'log_server', 'postgresql.log.2016-08-09-22.csv');
```

A sample response is as follows.

```
-----  
(1 row)
```

5. Select a sample of the log file. The following code retrieves the log time and error message description.

```
postgres=> SELECT log_time, message from my_postgres_error_log order by 1;
```

A sample response is as follows.

log_time	message
Tue Aug 09 15:45:18.172 2016 PDT	ending log output to stderr
Tue Aug 09 15:45:18.175 2016 PDT	database system was interrupted; last known up at 2016-08-09 22:43:34 UTC
Tue Aug 09 15:45:18.223 2016 PDT	checkpoint record is at 0/90002E0
Tue Aug 09 15:45:18.223 2016 PDT	redo record is at 0/90002A8; shutdown FALSE
Tue Aug 09 15:45:18.223 2016 PDT	next transaction ID: 0/1879; next OID: 24578
Tue Aug 09 15:45:18.223 2016 PDT	next MultiXactId: 1; next MultiXactOffset: 0
Tue Aug 09 15:45:18.223 2016 PDT	oldest unfrozen transaction ID: 1822, in database 1
(7 rows)	

Upgrade PL/v8

If you use PL/v8 and upgrade PostgreSQL to a new PL/v8 version, you immediately take advantage of the new extension but the catalog metadata doesn't reflect this fact. The following steps synchronize your catalog metadata with the new version of PL/v8. These steps are optional but we highly recommended you complete them to avoid metadata mismatch warnings.

1. Verify that you need to update.

Run the following command while connected to your instance.

```
select * from pg_available_extensions where name in
('plv8','plls','plcoffee');
```

If your results contain values for an installed version that is a lower number than the default version, you should continue with this procedure to update your extensions.

For example, the following result set indicates you should update:

name	default_version	installed_version	comment
plls	2.1.0	1.5.3	PL/LiveScript (v8) trusted procedural language
plcoffee	2.1.0	1.5.3	PL/CoffeeScript (v8) trusted procedural language
plv8	2.1.0	1.5.3	PL/JavaScript (v8) trusted procedural language
(3 rows)			

2. Take a snapshot of your instance.

The upgrade drops all your PL/v8 functions. Take a snapshot of your instance as a precaution. You can continue with the following steps while the snapshot is being created.

For steps to create a snapshot see, [Creating a DB Snapshot \(p. 216\)](#)

3. Get a count of the functions you need to drop and recreate.

Obtain the count of the number of PL/v8 functions in your instance so you can validate that they are all in place after the upgrade.

The following code returns the number of functions written in PL/v8, plcoffee, or plls:

```
select proname, nspname, lanname
from pg_proc p, pg_language l, pg_namespace n
where p.prolang = l.oid
and n.oid = p.pronamespace
and lanname in ('plv8','plcoffee','plls');
```

4. Use pg_dump to create a schema-only dump file.

The following code creates a file on your client machine in the /tmp directory.

```
./pg_dump -Fc --schema-only -U master postgres > /tmp/test.dmp
```

This example uses the following flags:

- -FC "format custom"
- --schema-only "will only dump commands necessary to create schema (functions in our case)"
- -U "rds master username"
- database "the database name in our instance"

For more information on pg_dump see, [pg_dump](#).

5. Extract the "CREATE FUNCTION" DDL statement that is present in the dump file.

The following code extracts the DDL statement needed to create the functions. You use this in subsequent steps to recreate the functions. The code uses the grep command to extract the statements to a file.

```
./pg_restore -l /tmp/test.dmp | grep FUNCTION > /tmp/function_list/
```

For more information on pg_restore see, [pg_restore](#).

6. Drop the functions and extensions.

The following code drops any PL/v8 based objects. The cascade option ensures that any dependent are dropped.

```
drop extension plv8 cascade;
```

If your PostgreSQL instance contains objects based on plcoffee or pll, repeat this step for those extensions.

7. Create the extensions.

The following code creates the PL/v8, plcoffee, and pll extensions:

```
create extension plv8;  
  
create extension plcoffee;  
  
create extension pll;
```

8. Create the functions using the dump file and "driver" file.

The following code re-creates the functions that you extracted previously.

```
./pg_restore -U master -d postgres -Fc -L /tmp/function_list /tmp/test.dmp
```

9. Verify your functions count.

Validate that your functions have all been re-creating by re-running the following code:

```
select * from pg_available_extensions where name in  
('plv8','pll','plcoffee');
```

Note

PL/v8 version 2 adds the following extra row to your result set:

proname	nspname	lanname
plv8_version	pg_catalog	plv8

Supported PostgreSQL Features

Amazon RDS supports many of the most common PostgreSQL features. These include:

Topics

- [Logical Replication for PostgreSQL on Amazon RDS \(p. 1064\)](#)
- [Event Triggers for PostgreSQL on Amazon RDS \(p. 1065\)](#)
- [Huge Pages for Amazon RDS for PostgreSQL \(p. 1066\)](#)
- [Tablespaces for PostgreSQL on Amazon RDS \(p. 1066\)](#)
- [Autovacuum for PostgreSQL on Amazon RDS \(p. 1067\)](#)
- [RAM Disk for the stats_temp_directory \(p. 1067\)](#)
- [ALTER ENUM for PostgreSQL \(p. 1067\)](#)

Logical Replication for PostgreSQL on Amazon RDS

Beginning with PostgreSQL version 10.4, RDS supports the publication and subscription SQL Syntax for PostgreSQL 10 Logical Replication.

To enable logical replication for an Amazon RDS for PostgreSQL DB instance

1. The AWS user account requires the **rds_superuser** role to perform logical replication for the PostgreSQL database on Amazon RDS.
2. Set the **rds.logical_replication** parameter to 1.
3. Modify the inbound rules of the security group for the publisher instance (production) to allow the subscriber instance (replica) to connect. This is usually done by including the IP address of the subscriber in the security group.

For more information on PostgreSQL logical replication, see the [PostgreSQL documentation](#).

Logical Decoding and Logical Replication

Beginning with PostgreSQL version 9.4, RDS supports the streaming of WAL changes using logical replication slots. Amazon RDS supports logical decoding for a PostgreSQL DB instance version 9.4.9 and higher and 9.5.4 and higher. You can set up logical replication slots on your instance and stream database changes through these slots to a client such as `pg_recvlogical`. Logical replication slots are created at the database level and support replication connections to a single database.

The most common clients for PostgreSQL logical replication are the AWS Database Migration Service or a custom-managed host on an AWS EC2 instance. The logical replication slot knows nothing about the receiver of the stream, and there is no requirement that the target be a replica database. If you set up a logical replication slot and don't read from the slot, data can be written and quickly fill up your DB instance's storage.

PostgreSQL logical replication and logical decoding on Amazon RDS are enabled with a parameter, a replication connection type, and a security role. The client for logical decoding can be any client that is capable of establishing a replication connection to a database on a PostgreSQL DB instance.

To enable logical decoding for an Amazon RDS for PostgreSQL DB instance

1. The user account requires the **rds_superuser** role to enable logical replication. The user account also requires the **rds_replication** role to grant permissions to manage logical slots and to stream data using logical slots.
2. Set the **rds.logical_replication static** parameter to 1. As part of applying this parameter, we also set the parameters `wal_level`, `max_wal_senders`, `max_replication_slots`, and `max_connections`. These parameter changes can increase WAL generation, so you should only set the **rds.logical_replication** parameter when you are using logical slots.
3. Reboot the DB instance for the static **rds.logical_replication** parameter to take effect.
4. Create a logical replication slot as explained in the next section. This process requires that you specify a decoding plugin. Currently we support the `test_decoding` output plugin that ships with PostgreSQL.

For more information on PostgreSQL logical decoding, see the [PostgreSQL documentation](#).

Working with Logical Replication Slots

You can use SQL commands to work with logical slots. For example, the following command creates a logical slot named `test_slot` using the default PostgreSQL output plugin `test_decoding`.

```
SELECT * FROM pg_create_logical_replication_slot('test_slot', 'test_decoding');
```

The output should be similar to the following.

```
slot_name      | xlog_position
-----+-----
regression_slot | 0/16B1970
(1 row)
```

To list logical slots, use the following command.

```
SELECT * FROM pg_replication_slots;
```

To drop a logical slot, use the following command.

```
SELECT pg_drop_replication_slot('test_slot');
```

The output should be similar to the following.

```
pg_drop_replication_slot
-----
(1 row)
```

For more examples on working with logical replication slots, see [Logical Decoding Examples in the PostgreSQL documentation](#).

Once you create the logical replication slot, you can start streaming. The following example shows how logical decoding is controlled over the streaming replication protocol, using the program `pg_recvlogical` included in the PostgreSQL distribution. This requires that client authentication is set up to allow replication connections.

```
pg_recvlogical -d postgres --slot test_slot -U master
               --host sg-postgresql1.c6c8mresaghv0.us-west-2.rds.amazonaws.com
               -f - --start
```

Event Triggers for PostgreSQL on Amazon RDS

PostgreSQL versions 9.4.9 and later and version 9.5.4 and later support event triggers, and Amazon RDS supports event triggers for these versions. The master user account can be used to create, modify, rename, and delete event triggers. Event triggers are at the DB instance level, so they can apply to all databases on an instance.

For example, the following code creates an event trigger that prints the current user at the end of every DDL command.

```
CREATE OR REPLACE FUNCTION raise_notice_func()
```

```
RETURNS event_trigger
LANGUAGE plpgsql AS
$$
BEGIN
    RAISE NOTICE 'In trigger function: %', current_user;
END;
$$;

CREATE EVENT TRIGGER event_trigger_1
    ON ddl_command_end
EXECUTE PROCEDURE raise_notice_func();
```

For more information about PostgreSQL event triggers, see [Event Triggers](#) in the PostgreSQL documentation.

There are several limitations to using PostgreSQL event triggers on Amazon RDS. These include:

- You cannot create event triggers on read replicas. You can, however, create event triggers on a read replica master. The event triggers are then copied to the read replica. The event triggers on the read replica don't fire on the read replica when changes are pushed from the master. However, if the read replica is promoted, the existing event triggers fire when database operations occur.
- To perform a major version upgrade to a PostgreSQL DB instance that uses event triggers, you must delete the event triggers before you upgrade the instance.

Huge Pages for Amazon RDS for PostgreSQL

Amazon RDS for PostgreSQL supports multiple page sizes for PostgreSQL versions 9.4.11 and later, 9.5.6 and later, and 9.6.2 and later. This support includes 4 K and 2 MB page sizes.

Huge pages reduce overhead when using large contiguous chunks of memory. You allocate huge pages for your application by using calls to *mmap* or SYSV shared memory. You enable huge pages on an Amazon RDS for PostgreSQL database by using the `huge_pages` parameter. Set this parameter to "on" to enable huge pages.

For PostgreSQL versions 10 and above, huge pages are enabled for all instance classes. For PostgreSQL versions below 10, huge pages are enabled by default for db.r4.* , db.m4.16xlarge, and db.m5.* instance classes. For other instance classes, huge pages are disabled by default.

When you set the `huge_pages` parameter to "on," Amazon RDS uses huge pages based on the available shared memory. If the DB instance is unable to use huge pages due to shared memory constraints, Amazon RDS prevents the instance from starting and sets the status of the DB instance to an incompatible parameters state. In this case, you can set the `huge_pages` parameter to "off" to allow Amazon RDS to start the DB instance.

The `shared_buffers` parameter is key to setting the shared memory pool that is required for using huge pages. The default value for the `shared_buffers` parameter is set to a percentage of the total 8K pages available for that instance's memory. When you use huge pages, those pages are allocated in the huge pages collocated together. Amazon RDS puts a DB instance into an incompatible parameters state if the shared memory parameters are set to require more than 90 percent of the DB instance memory. For more information about setting shared memory for PostgreSQL, see the [PostgreSQL documentation](#).

Note

Huge pages are not supported for the db.m1, db.m2, and db.m3 DB instance classes.

Tablespaces for PostgreSQL on Amazon RDS

Tablespaces are supported in PostgreSQL on Amazon RDS for compatibility; since all storage is on a single logical volume, tablespaces cannot be used for IO splitting or isolation. We have benchmarks and practical experience that shows that a single logical volume is the best setup for most use cases.

Autovacuum for PostgreSQL on Amazon RDS

The PostgreSQL auto-vacuum is an optional, but highly recommended, parameter that by default is turned on for new PostgreSQL DB instances. Do not turn this parameter off. For more information on using auto-vacuum with Amazon RDS for PostgreSQL, see [Working with PostgreSQL Autovacuum on Amazon RDS \(p. 1009\)](#).

RAM Disk for the stats_temp_directory

The Amazon RDS for PostgreSQL parameter, `rds.pg_stat_ramdisk_size`, can be used to specify the system memory allocated to a RAM disk for storing the PostgreSQL `stats_temp_directory`. The RAM disk parameter is available for all PostgreSQL versions on Amazon RDS.

Under certain workloads, setting this parameter can improve performance and decrease IO requirements. For more information about the `stats_temp_directory`, see [the PostgreSQL documentation](#).

To enable a RAM disk for your `stats_temp_directory`, set the `rds.pg_stat_ramdisk_size` parameter to a non-zero value in the parameter group used by your DB instance. The parameter value is in MB. You must reboot the DB instance before the change takes effect.

For example, the following AWS CLI command sets the RAM disk parameter to 256 MB.

```
postgres=>aws rds modify-db-parameter-group \
    --db-parameter-group-name pg-95-ramdisk-testing \
    --parameters "ParameterName=rds.pg_stat_ramdisk_size, ParameterValue=256,
    ApplyMethod=pending-reboot"
```

After you reboot, run the following command to see the status of the `stats_temp_directory`:

```
postgres=>show stats_temp_directory;
```

The command should return the following:

```
stats_temp_directory
-----
/rdsdbramdisk/pg_stat_tmp
(1 row)
```

ALTER ENUM for PostgreSQL

Amazon RDS for PostgreSQL versions 9.6.2 and 9.5.6 and later support the ability to alter enumerations. This feature is not available in other versions on Amazon RDS.

The following code shows an example of altering an enum value.

```
postgres=> CREATE TYPE rainbow AS ENUM ('red', 'orange', 'yellow', 'green', 'blue',
    'purple');
CREATE TYPE
postgres=> CREATE TABLE t1 (colors rainbow);
CREATE TABLE
postgres=> INSERT INTO t1 VALUES ('red'), ('orange');
INSERT 0 2
postgres=> SELECT * from t1;
```

```
colors
-----
red
orange
(2 rows)
postgres=> ALTER TYPE rainbow RENAME VALUE 'red' TO 'crimson';
ALTER TYPE
postgres=> SELECT * from t1;
colors
-----
crimson
orange
(2 rows)
```

Limits for PostgreSQL DB Instances

You can have up to 40 PostgreSQL DB instances. The following is a list of limitations for PostgreSQL on Amazon RDS:

- The maximum storage size for PostgreSQL DB instances is the following:
 - General Purpose (SSD) storage: 32 TiB (16 TiB for db.t2.micro and db.t2.small instance classes)
 - Provisioned IOPS storage: 32 TiB (16 TiB for db.t2.micro and db.t2.small instance classes)
 - Magnetic storage: 3 TiB
- The minimum storage size for PostgreSQL DB instances is the following:
 - General Purpose (SSD) storage: 5 GiB
 - Provisioned IOPS storage: 100 GiB
 - Magnetic storage: 5 GiB
- Amazon RDS reserves up to 3 connections for system maintenance. If you specify a value for the user connections parameter, you need to add 3 to the number of connections that you expect to use.

Upgrading a PostgreSQL DB Instance

There are two types of upgrades you can manage for your PostgreSQL DB instance:

- OS Updates – Occasionally, Amazon RDS might need to update the underlying operating system of your DB instance to apply security fixes or OS changes. You can decide when Amazon RDS applies OS updates by using the RDS console, AWS Command Line Interface (AWS CLI), or RDS API.
For more information about OS updates, see [Applying Updates for a DB Instance \(p. 118\)](#).
- Database Engine Upgrades – When Amazon RDS supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades: major version upgrades and minor version upgrades. Amazon RDS supports both major and minor version upgrades for PostgreSQL DB instances.

For more information about PostgreSQL DB engine upgrades, see [Upgrading the PostgreSQL DB Engine \(p. 988\)](#).

Using SSL with a PostgreSQL DB Instance

Amazon RDS supports Secure Socket Layer (SSL) encryption for PostgreSQL DB instances. Using SSL, you can encrypt a PostgreSQL connection between your applications and your PostgreSQL DB instances. You can also force all connections to your PostgreSQL DB instance to use SSL.

Topics

- [Requiring an SSL Connection to a PostgreSQL DB Instance \(p. 1069\)](#)
- [Determining the SSL Connection Status \(p. 1069\)](#)

SSL support is available in all AWS regions for PostgreSQL. Amazon RDS creates an SSL certificate for your PostgreSQL DB instance when the instance is created. If you enable SSL certificate verification, then the SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks.

To connect to a PostgreSQL DB instance over SSL

1. Download the certificate stored at <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>.
2. Import the certificate into your operating system.
3. Connect to your PostgreSQL DB instance over SSL by appending `sslmode=verify-full` to your connection string. When you use `sslmode=verify-full`, the SSL connection verifies the DB instance endpoint against the endpoint in the SSL certificate.

Use the `sslrootcert` parameter to reference the certificate, for example, `sslrootcert=rds-ssl-ca-cert.pem`.

The following is an example of using the `psql` program to connect to a PostgreSQL DB instance :

```
$ psql -h testpg.cdhmuqifdpib.us-east-1.rds.amazonaws.com -p 5432 \
"dbname=testpg user=testuser sslrootcert=rds-ca-2015-root.pem sslmode=verify-full"
```

Requiring an SSL Connection to a PostgreSQL DB Instance

You can require that connections to your PostgreSQL DB instance use SSL by using the `rds.force_ssl` parameter. By default, the `rds.force_ssl` parameter is set to 0 (off). You can set the `rds.force_ssl` parameter to 1 (on) to require SSL for connections to your DB instance. Updating the `rds.force_ssl` parameter also sets the PostgreSQL `ssl` parameter to 1 (on) and modifies your DB instance's `pg_hba.conf` file to support the new SSL configuration.

You can set the `rds.force_ssl` parameter value by updating the parameter group for your DB instance. If the parameter group for your DB instance isn't the default one, and the `ssl` parameter is already set to 1 when you set `rds.force_ssl` to 1, you don't need to reboot your DB instance. Otherwise, you must reboot your DB instance for the change to take effect. For more information on parameter groups, see [Working with DB Parameter Groups \(p. 169\)](#).

When the `rds.force_ssl` parameter is set to 1 for a DB instance, you see output similar to the following when you connect, indicating that SSL is now required:

```
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql (9.3.12, server 9.4.4)
WARNING: psql major version 9.3, server major version 9.4.
Some psql features might not work.
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

Determining the SSL Connection Status

The encrypted status of your connection is shown in the logon banner when you connect to the DB instance:

```
Password for user master:  
psql (9.3.12)  
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)  
Type "help" for help.  
  
postgres=>
```

You can also load the `sslinfo` extension and then call the `ssl_is_used()` function to determine if SSL is being used. The function returns `t` if the connection is using SSL, otherwise it returns `f`.

```
postgres=> create extension sslinfo;  
CREATE EXTENSION  
  
postgres=> select ssl_is_used();  
ssl_is_used  
-----  
t  
(1 row)
```

You can use the `select ssl_cipher()` command to determine the SSL cipher:

```
postgres=> select ssl_cipher();  
ssl_cipher  
-----  
DHE-RSA-AES256-SHA  
(1 row)
```

If you enable `set rds.force_ssl` and restart your instance, non-SSL connections are refused with the following message:

```
$ export PGSSLMODE=disable  
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser  
psql: FATAL: no pg_hba.conf entry for host "host.ip", user "someuser", database "postgres",  
      SSL off  
$
```

Limits for Amazon RDS

This topic describes the resource limits and naming constraints for Amazon RDS.

Topics

- [Limits in Amazon RDS \(p. 1071\)](#)
- [Naming Constraints in Amazon RDS \(p. 1072\)](#)
- [File Size Limits in Amazon RDS \(p. 1074\)](#)

Limits in Amazon RDS

Each AWS account has limits, for each AWS Region, on the number of Amazon RDS resources that can be created. Once a limit for a resource has been reached, additional calls to create that resource fail with an exception.

The following table lists the resources and their limits per region.

Resource	Default Limit
Clusters	40
Cluster parameter groups	50
Cross-region snapshots copy requests	5
DB Instances ¹	40
Event subscriptions	20
Manual snapshots	100
Manual cluster snapshots	100
Option groups	20
Parameter groups	50
Read replicas per master	5
Reserved instances	40
Rules per DB security group	20
Rules per VPC security group	50 inbound 50 outbound
DB Security groups	25
VPC Security groups	5
Subnet groups	50
Subnets per subnet group	20
Tags per resource	50
Total storage for all DB instances	100 TiB

1. By default, you can have up to a total of 40 Amazon RDS DB instances. Of those 40, up to 10 can be Oracle or SQL Server DB instances under the "License Included" model. All 40 can be MySQL, MariaDB, PostgreSQL, or Oracle under the "BYOL" model. If your application requires more DB instances, you can request additional DB instances via this request form [Request RDS DB instance limit](#).

Naming Constraints in Amazon RDS

The following table describes naming constraints in Amazon RDS.

DB instance identifier	<ul style="list-style-type: none">Must contain 1 to 63 alphanumeric characters or hyphens.First character must be a letter.Cannot end with a hyphen or contain two consecutive hyphens.Must be unique for all DB instances per AWS account, per region.
Database name	<p>Database name constraints differ for each database engine.</p> <p>MySQL and MariaDB</p> <ul style="list-style-type: none">Must contain 1 to 64 alphanumeric characters.Cannot be a word reserved by the database engine. <p>Oracle</p> <ul style="list-style-type: none">Cannot be longer than 8 characters. <p>PostgreSQL</p> <ul style="list-style-type: none">Must contain 1 to 63 alphanumeric characters.Must begin with a letter or an underscore. Subsequent characters can be letters, underscores, or digits (0-9).Cannot be a word reserved by the database engine. <p>SQL Server</p> <ul style="list-style-type: none">Not applicable. For SQL Server, you create your databases after you create your DB instance. Database names follow the usual SQL Server naming rules.
Master user name	<p>Master user name constraints differ for each database engine.</p> <p>MariaDB</p> <ul style="list-style-type: none">Must contain 1 to 16 alphanumeric characters.Cannot be a word reserved by the database engine. <p>MySQL</p> <ul style="list-style-type: none">Must contain 1 to 16 alphanumeric characters.First character must be a letter.

	<ul style="list-style-type: none"> • Cannot be a word reserved by the database engine. <p>Oracle</p> <ul style="list-style-type: none"> • Must contain 1 to 30 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine. <p>PostgreSQL</p> <ul style="list-style-type: none"> • Must contain 1 to 63 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine. <p>SQL Server</p> <ul style="list-style-type: none"> • Must contain 1 to 64 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine.
Master password	<p>The password for the master database user can be any printable ASCII character except "/", "", or "@". Master password constraints differ for each database engine.</p> <p>MySQL and MariaDB</p> <ul style="list-style-type: none"> • Must contain 8 to 41 characters. <p>Oracle</p> <ul style="list-style-type: none"> • Must contain 8 to 30 characters. <p>PostgreSQL</p> <ul style="list-style-type: none"> • Must contain 8 to 128 characters. <p>SQL Server</p> <ul style="list-style-type: none"> • Must contain 8 to 128 characters.
DB parameter group name	<ul style="list-style-type: none"> • Must contain from 1 to 255 alphanumeric characters. • First character must be a letter. • Hyphens are allowed, but the name cannot end with a hyphen or contain two consecutive hyphens.
DB subnet group name	<ul style="list-style-type: none"> • Must contain from 1 to 255 characters. • Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.

File Size Limits in Amazon RDS

File size limits apply to Amazon RDS DB instances.

MySQL File Size Limits in Amazon RDS

For Amazon RDS MySQL DB instances, the maximum provisioned storage limit constrains the size of a table to a maximum size of 16 TB when using InnoDB file-per-table tablespaces. This limit also constrains the system tablespace to a maximum size of 16 TB. InnoDB file-per-table tablespaces (with tables each in their own tablespace) are set by default for Amazon RDS MySQL DB instances. For more information, see [DB instance storage \(p. 103\)](#).

Note

Some existing DB instances have a lower limit. For example, MySQL DB instances created prior to April 2014 have a file and table size limit of 2 TB. This 2-TB file size limit also applies to DB instances or Read Replicas created from DB snapshots taken before April 2014, regardless of when the DB instance was created.

There are advantages and disadvantages to using InnoDB file-per-table tablespaces, depending on your application. To determine the best approach for your application, go to [InnoDB File-Per-Table Mode](#) in the MySQL documentation.

We don't recommend allowing tables to grow to the maximum file size. In general, a better practice is to partition data into smaller tables, which can improve performance and recovery times.

One option that you can use for breaking a large table up into smaller tables is partitioning. Partitioning distributes portions of your large table into separate files based on rules that you specify. For example, if you store transactions by date, you can create partitioning rules that distribute older transactions into separate files using partitioning. Then periodically, you can archive the historical transaction data that doesn't need to be readily available to your application. For more information, see [Partitioning](#) in the MySQL documentation.

To determine the file size of a table

Use the following SQL command to determine if any of your tables are too large and are candidates for partitioning. To update table statistics, issue an `ANALYZE TABLE` command on each table. For more information, see [ANALYZE TABLE](#) in the MySQL documentation.

```
SELECT TABLE_SCHEMA, TABLE_NAME,
       round(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) As "Approximate size (MB)",
       DATA_FREE
  FROM information_schema.TABLES
 WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema');
```

To enable InnoDB file-per-table tablespaces

- To enable InnoDB file-per-table tablespaces, set the `innodb_file_per_table` parameter to 1 in the parameter group for the DB instance.

To disable InnoDB file-per-table tablespaces

- To disable InnoDB file-per-table tablespaces, set the `innodb_file_per_table` parameter to 0 in the parameter group for the DB instance.

For information on updating a parameter group, see [Working with DB Parameter Groups \(p. 169\)](#).

When you have enabled or disabled InnoDB file-per-table tablespaces, you can issue an `ALTER TABLE` command. You can use this command to move a table from the global tablespace to its own tablespace, or from its own tablespace to the global tablespace as shown in the following example.

```
ALTER TABLE table_name ENGINE=InnoDB, ALGORITHM=COPY;
```

MariaDB File Size Limits in Amazon RDS

For Amazon RDS MariaDB DB instances, the maximum provisioned storage limit constrains the size of a table to a maximum size of 16 TB when using InnoDB file-per-table tablespaces. This limit also constrains the system tablespace to a maximum size of 16 TB. InnoDB file-per-table tablespaces (with tables each in their own tablespace) is set by default for Amazon RDS MariaDB DB instances. For more information, see [DB instance storage \(p. 103\)](#).

There are advantages and disadvantages to using InnoDB file-per-table tablespaces, depending on your application. To determine the best approach for your application, go to [InnoDB File-Per-Table Mode](#) in the MySQL documentation.

We don't recommend allowing tables to grow to the maximum file size. In general, a better practice is to partition data into smaller tables, which can improve performance and recovery times.

One option that you can use for breaking a large table up into smaller tables is partitioning. Partitioning distributes portions of your large table into separate files based on rules that you specify. For example, if you store transactions by date, you can create partitioning rules that distribute older transactions into separate files using partitioning. Then periodically, you can archive the historical transaction data that doesn't need to be readily available to your application. For more information, go to <https://dev.mysql.com/doc/refman/5.6/en/partitioning.html> in the MySQL documentation.

To determine the file size of a table

Use the following SQL command to determine if any of your tables are too large and are candidates for partitioning. To update table statistics, issue an `ANALYZE TABLE` command on each table. For more information, see [ANALYZE TABLE](#) in the MySQL documentation.

```
SELECT TABLE_SCHEMA, TABLE_NAME,
       round(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) As "Approximate size (MB)",
       DATA_FREE
  FROM information_schema.TABLES
 WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema');
```

To enable InnoDB file-per-table tablespaces

- To enable InnoDB file-per-table tablespaces, set the `innodb_file_per_table` parameter to 1 in the parameter group for the DB instance.

To disable InnoDB file-per-table tablespaces

- To disable InnoDB file-per-table tablespaces, set the `innodb_file_per_table` parameter to 0 in the parameter group for the DB instance.

For information on updating a parameter group, see [Working with DB Parameter Groups \(p. 169\)](#).

When you have enabled or disabled InnoDB file-per-table tablespaces, you can issue an `ALTER TABLE` command. You can use this command to move a table from the global tablespace to its own tablespace, or from its own tablespace to the global tablespace as shown in the following example.

```
ALTER TABLE table_name ENGINE=InnoDB, ALGORITHM=COPY;
```

Troubleshooting

Use the following sections to help troubleshoot problems you have with Amazon RDS.

Topics

- [Cannot Connect to Amazon RDS DB Instance \(p. 1077\)](#)
- [Amazon RDS Security Issues \(p. 1078\)](#)
- [Resetting the DB Instance Owner Role Password \(p. 1079\)](#)
- [Amazon RDS DB Instance Outage or Reboot \(p. 1079\)](#)
- [Amazon RDS DB Parameter Changes Not Taking Effect \(p. 1080\)](#)
- [Amazon RDS DB Instance Running Out of Storage \(p. 1080\)](#)
- [Amazon RDS Insufficient DB Instance Capacity \(p. 1081\)](#)
- [Amazon RDS MySQL and MariaDB Issues \(p. 1081\)](#)
- [Amazon RDS Oracle GoldenGate Issues \(p. 1087\)](#)
- [Cannot Connect to Amazon RDS SQL Server DB Instance \(p. 1087\)](#)
- [Cannot Connect to Amazon RDS PostgreSQL DB Instance \(p. 1088\)](#)
- [Cannot Set Backup Retention Period to 0 \(p. 1088\)](#)

Cannot Connect to Amazon RDS DB Instance

When you cannot connect to a DB instance, the following are common causes:

- The access rules enforced by your local firewall and the ingress IP addresses that you authorized to access your DB instance in the instance's security group are not in sync. The problem is most likely the ingress rules in your security group. By default, DB instances do not allow access; access is granted through a security group. To grant access, you must create your own security group with specific ingress and egress rules for your situation. If necessary, add rules to the security group associated with the VPC that allow traffic related to the source in and out of the DB instance. You can specify an IP address, a range of IP addresses, or another VPC security group.

For more information about setting up a security group, see [Provide Access to Your DB Instance in Your VPC by Creating a Security Group \(p. 8\)](#).

- The port you specified when you created the DB instance cannot be used to send or receive communications due to your local firewall restrictions. In this case, check with your network administrator to determine if your network allows the specified port to be used for inbound and outbound communication.
- Your DB instance is still being created and is not yet available. Depending on the size of your DB instance, it can take up to 20 minutes before an instance is available.

Testing a Connection to an Amazon RDS DB Instance

You can test your connection to a DB instance using common Linux or Windows tools.

From a Linux or Unix terminal, you can test the connection by typing the following (replace <DB-instance-endpoint> with the endpoint and <port> with the port of your DB instance):

```
$nc -zv <DB-instance-endpoint> <port>
```

For example, the following shows a sample command and the return value:

```
$nc -zv postgresql1.c6c8mn7tsdgv0.us-west-2.rds.amazonaws.com 8299
Connection to postgresql1.c6c8mn7tsdgv0.us-west-2.rds.amazonaws.com 8299 port [tcp/vvr-data] succeeded!
```

Windows users can use Telnet to test the connection to a DB instance. Note that Telnet actions are not supported other than for testing the connection. If a connection is successful, the action returns no message. If a connection is not successful, you receive an error message such as the following:

```
C:\>telnet sg-postgresql1.c6c8mntzhgv0.us-west-2.rds.amazonaws.com 819
Connecting To sg-postgresql1.c6c8mntzhgv0.us-west-2.rds.amazonaws.com...Could not open
connection to the host, on port 819: Connect failed
```

If Telnet actions return success, your security group is properly configured.

Note

Amazon RDS does not accept internet control message protocol (ICMP) traffic, including ping.

Troubleshooting Connection Authentication

If you can connect to your DB instance but you get authentication errors, you might want to reset the master user password for the DB instance. You can do this by modifying the RDS instance.

For more information about modifying a DB instance, see one of the following topics:

- [Modifying a DB Instance Running the MySQL Database Engine \(p. 610\)](#)
- [Modifying a DB Instance Running the Oracle Database Engine \(p. 758\)](#)
- [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#)
- [Modifying a DB Instance Running the PostgreSQL Database Engine \(p. 979\)](#)

Amazon RDS Security Issues

To avoid security issues, never use your master AWS user name and password for a user account. Best practice is to use your master AWS account to create IAM users and assign those to DB user accounts. You can also use your master account to create other user accounts, if necessary.

For more information on creating IAM users, see [Create an IAM User \(p. 5\)](#).

Error Message "Failed to retrieve account attributes, certain console functions may be impaired."

There are several reasons you would get this error; it could be because your account is missing permissions, or your account has not been properly set up. If your account is new, you may not have

waited for the account to be ready. If this is an existing account, you could lack permissions in your access policies to perform certain actions such as creating a DB instance. To fix the issue, your IAM administrator needs to provide the necessary roles to your account. For more information, see the IAM documentation.

Resetting the DB Instance Owner Role Password

You can reset the assigned permissions for your DB instance by resetting the master password. For example, if you lock yourself out of the `db_owner` role on your SQL Server database, you can reset the `db_owner` role password by modifying the DB instance master password. By changing the DB instance password, you can regain access to the DB instance, access databases using the modified password for the `db_owner`, and restore privileges for the `db_owner` role that may have been accidentally revoked. You can change the DB instance password by using the Amazon RDS console, the AWS CLI command [modify-db-instance](#), or by using the [ModifyDBInstance](#) action.

For more information about modifying a SQL Server DB instance, see [Modifying a DB Instance Running the Microsoft SQL Server Database Engine \(p. 521\)](#).

Amazon RDS DB Instance Outage or Reboot

A DB instance outage can occur when a DB instance is rebooted, when the DB instance is put into a state that prevents access to it, and when the database is restarted. A reboot can occur when you manually reboot your DB instance or when you change a DB instance setting that requires a reboot before it can take effect.

When you modify a setting for a DB instance, you can determine when the change is applied by using the **Apply Immediately** setting.

To see a table that shows DB instance actions and the effect that setting the **Apply Immediately** value has, see [Modifying an Amazon RDS DB Instance \(p. 115\)](#).

A DB instance reboot only occurs when you change a setting that requires a reboot, or when you manually cause a reboot. A reboot can occur immediately if you change a setting and request that the change take effect immediately or it can occur during the DB instance's maintenance window.

A DB instance reboot occurs immediately when one of the following occurs:

- You change the backup retention period for a DB instance from 0 to a nonzero value or from a nonzero value to 0 and set **Apply Immediately** to *true*.
- You change the DB instance class, and **Apply Immediately** is set to *true*.
- You change the storage type from **Magnetic (Standard)** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)**, or from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic (Standard)**. from standard to PIOPS.

A DB instance reboot occurs during the maintenance window when one of the following occurs:

- You change the backup retention period for a DB instance from 0 to a nonzero value or from a nonzero value to 0, and **Apply Immediately** is set to *false*.
- You change the DB instance class, and **Apply Immediately** is set to *false*.

When you change a static parameter in a DB parameter group, the change will not take effect until the DB instance associated with the parameter group is rebooted. The change requires a manual reboot; the DB instance will not automatically be rebooted during the maintenance window.

Amazon RDS DB Parameter Changes Not Taking Effect

If you change a parameter in a DB parameter group but you don't see the changes take effect, you most likely need to reboot the DB instance associated with the DB parameter group. When you change a dynamic parameter, the change takes effect immediately; when you change a static parameter, the change won't take effect until you reboot the DB instance associated with the parameter group.

You can reboot a DB instance using the RDS console or explicitly calling the `RebootDbInstance` API action (without failover, if the DB instance is in a Multi-AZ deployment). The requirement to reboot the associated DB instance after a static parameter change helps mitigate the risk of a parameter misconfiguration affecting an API call, such as calling `ModifyDBInstance` to change DB instance class. For more information, see [Modifying Parameters in a DB Parameter Group \(p. 171\)](#).

Amazon RDS DB Instance Running Out of Storage

If your DB instance runs out of storage space, it might no longer be available. We highly recommend that you constantly monitor the `FreeStorageSpace` metric published in CloudWatch to ensure that your DB instance has enough free storage space.

If your database instance runs out of storage, its status will change to *storage-full*. For example, a call to the `DescribeDBInstances` action for a DB instance that has used up its storage will output the following:

```
aws rds describe-db-instances --db-instance-identifier mydbinstance

DBINSTANCE mydbinstance 2009-12-22T23:06:11.915Z db.m3.large mysql5.6 50 sa
storage-full mydbinstance.clla4j4jgyp.us-east-1.rds.amazonaws.com 3306
us-east-1b 3
SECGROUP default active
PARAMGRP default.mysql5.6 in-sync
```

To recover from this scenario, add more storage space to your instance using the `ModifyDBInstance` action or the following AWS CLI command:

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--allocated-storage 60 \
--apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--allocated-storage 60 ^
--apply-immediately
```

```
DBINSTANCE mydbinstance 2009-12-22T23:06:11.915Z db.m3.large mysql5.6 50 sa
storage-full mydbinstance.clla4j4jgyp.us-east-1.rds.amazonaws.com 3306
us-east-1b 3 60
SECGROUP default active
PARAMGRP default.mysql5.6 in-sync
```

Now, when you describe your DB instance, you will see that your DB instance will have *modifying* status, which indicates the storage is being scaled.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

```
DBINSTANCE mydbinstance 2009-12-22T23:06:11.915Z db.m3.large mysql5.6 50 sa
modifying mydbinstance.clla4j4jgyp.us-east-1.rds.amazonaws.com
3306 us-east-1b 3 60
SECGROUP default active
PARAMGRP default.mysql5.6 in-sync
```

Once storage scaling is complete, your DB instance status will change to *available*.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

```
DBINSTANCE mydbinstance 2009-12-22T23:06:11.915Z db.m3.large mysql5.6 60 sa
available mydbinstance.clla4j4jgyp.us-east-1.rds.amazonaws.com 3306
us-east-1b 3
SECGROUP default active
PARAMGRP default.mysql5.6 in-sync
```

Note that you can receive notifications when your storage space is exhausted using the `DescribeEvents` action. For example, in this scenario, if you do a `DescribeEvents` call after these operations you will see the following output:

```
aws rds describe-events --source-type db-instance --source-identifier mydbinstance
```

```
2009-12-22T23:44:14.374Z mydbinstance Allocated storage has been exhausted db-instance
2009-12-23T00:14:02.737Z mydbinstance Applying modification to allocated storage db-
instance
2009-12-23T00:31:54.764Z mydbinstance Finished applying modification to allocated storage
```

Amazon RDS Insufficient DB Instance Capacity

If you get an `InsufficientDBInstanceCapacity` error when you try to modify a DB instance class, it might be because the DB instance is on the EC2-Classic platform and is therefore not in a VPC. Some DB instance classes require a VPC. For example, if you are on the EC2-Classic platform and try to increase capacity by switching to a DB instance class that requires a VPC, this error results. For information about Amazon Elastic Compute Cloud instance types that are only available in a VPC, see [Instance Types Available Only in a VPC](#) in the *Amazon Elastic Compute Cloud User Guide*.

To correct the problem, you can move the DB instance into a VPC. For more information, see [Moving a DB Instance Not in a VPC into a VPC \(p. 426\)](#).

For information about modifying a DB instance, see [Modifying an Amazon RDS DB Instance \(p. 115\)](#). For information about troubleshooting instance capacity issues for Amazon EC2, see [Troubleshooting Instance Capacity](#) in the *Amazon Elastic Compute Cloud User Guide*.

Amazon RDS MySQL and MariaDB Issues

You can diagnose and correct problems with MySQL and MariaDB DB instances.

Index Merge Optimization Returns Wrong Results

This issue applies only to MySQL DB instances.

Queries that use index merge optimization might return wrong results due to a bug in the MySQL query optimizer that was introduced in MySQL 5.5.37. When you issue a query against a table with multiple indexes the optimizer scans ranges of rows based on the multiple indexes, but does not merge the results together correctly. For more information on the query optimizer bug, go to <http://bugs.mysql.com/bug.php?id=72745> and <http://bugs.mysql.com/bug.php?id=68194> in the MySQL bug database.

For example, consider a query on a table with two indexes where the search arguments reference the indexed columns.

```
SELECT * FROM table1
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

In this case, the search engine searches both indexes. However, due to the bug, the merged results are incorrect.

To resolve this issue, you can do one of the following:

- Set the `optimizer_switch` parameter to `index_merge=off` in the DB parameter group for your MySQL DB instance. For information on setting DB parameter group parameters, see [Working with DB Parameter Groups \(p. 169\)](#).
- Upgrade your MySQL DB instance to MySQL version 5.6, 5.7, or 8.0. For more information, see [Upgrading a MySQL DB Snapshot \(p. 623\)](#).
- If you cannot upgrade your instance or change the `optimizer_switch` parameter, you can work around the bug by explicitly identifying an index for the query, for example:

```
SELECT * FROM table1
  USE INDEX covering_index
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

For more information, go to [Index Merge Optimization](#).

Diagnosing and Resolving Lag Between Read Replicas

After you create a MySQL or MariaDB Read Replica and the Read Replica is available, Amazon RDS first replicates the changes made to the source DB instance from the time the create Read Replica operation was initiated. During this phase, the replication lag time for the Read Replica will be greater than 0. You can monitor this lag time in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric.

The `ReplicaLag` metric reports the value of the `Seconds_Behind_Master` field of the MySQL or MariaDB `SHOW SLAVE STATUS` command. For more information, see [SHOW SLAVE STATUS](#). When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance. If the `ReplicaLag` metric returns -1, replication might not be active. To troubleshoot a replication error, see [Diagnosing and Resolving a MySQL or MariaDB Read Replication Failure \(p. 1083\)](#). A `ReplicaLag` value of -1 can also mean that the `Seconds_Behind_Master` value cannot be determined or is `NULL`.

The `ReplicaLag` metric returns -1 during a network outage or when a patch is applied during the maintenance window. In this case, wait for network connectivity to be restored or for the maintenance window to end before you check the `ReplicaLag` metric again.

Because the MySQL and MariaDB read replication technology is asynchronous, you can expect occasional increases for the `BinLogDiskUsage` metric on the source DB instance and for the `ReplicaLag` metric

on the Read Replica. For example, a high volume of write operations to the source DB instance can occur in parallel, while write operations to the Read Replica are serialized using a single I/O thread, can lead to a lag between the source instance and Read Replica. For more information about Read Replicas and MySQL, go to [Replication Implementation Details](#) in the MySQL documentation. For more information about Read Replicas and MariaDB, go to [Replication Overview](#) in the MariaDB documentation.

You can reduce the lag between updates to a source DB instance and the subsequent updates to the Read Replica by doing the following:

- Set the DB instance class of the Read Replica to have a storage size comparable to that of the source DB instance.
- Ensure that parameter settings in the DB parameter groups used by the source DB instance and the Read Replica are compatible. For more information and an example, see the discussion of the `max_allowed_packet` parameter in the next section.
- Disable the query cache. For tables that are modified often, using the query cache can increase replica lag because the cache is locked and refreshed often. If this is the case, you might see less replica lag if you disable the query cache. You can disable the query cache by setting the `query_cache_type` parameter to 0 in the DB parameter group for the DB instance. For more information on the query cache, see [Query Cache Configuration](#).
- Warm the buffer pool on the Read Replica for InnoDB for MySQL, InnoDB for MariaDB 10.2 or higher, or XtraDB for MariaDB 10.1 or lower. If you have a small set of tables that are being updated often, and you are using the InnoDB or XtraDB table schema, then dump those tables on the Read Replica. Doing this causes the database engine to scan through the rows of those tables from the disk and then cache them in the buffer pool, which can reduce replica lag. The following shows an example.

For Linux, OS X, or Unix:

```
PROMPT> mysqldump \
-h <endpoint> \
--port=<port> \
-u=<username> \
-p <password> \
database_name table1 table2 > /dev/null
```

For Windows:

```
PROMPT> mysqldump ^
-h <endpoint> ^
--port=<port> ^
-u=<username> ^
-p <password> ^
database_name table1 table2 > /dev/null
```

Diagnosing and Resolving a MySQL or MariaDB Read Replication Failure

Amazon RDS monitors the replication status of your Read Replicas and updates the **Replication State** field of the Read Replica instance to **Error** if replication stops for any reason. You can review the details of the associated error thrown by the MySQL or MariaDB engines by viewing the **Replication Error** field. Events that indicate the status of the Read Replica are also generated, including [RDS-EVENT-0045 \(p. 291\)](#), [RDS-EVENT-0046 \(p. 292\)](#), and [RDS-EVENT-0047 \(p. 291\)](#). For more information about events and subscribing to events, see [Using Amazon RDS Event Notification \(p. 287\)](#). If a MySQL error message is returned, review the error in the [MySQL error message documentation](#). If a MariaDB error message is returned, review the error in the [MariaDB error message documentation](#).

Common situations that can cause replication errors include the following:

- The value for the `max_allowed_packet` parameter for a Read Replica is less than the `max_allowed_packet` parameter for the source DB instance.

The `max_allowed_packet` parameter is a custom parameter that you can set in a DB parameter group that is used to specify the maximum size of data manipulation language (DML) that can be executed on the database. If the `max_allowed_packet` parameter value for the source DB instance is smaller than the `max_allowed_packet` parameter value for the Read Replica, the replication process can throw an error and stop replication. The most common error is `packet bigger than 'max_allowed_packet' bytes`. You can fix the error by having the source and Read Replica use DB parameter groups with the same `max_allowed_packet` parameter values.

- Writing to tables on a Read Replica. If you are creating indexes on a Read Replica, you need to have the `read_only` parameter set to `0` to create the indexes. If you are writing to tables on the Read Replica, it can break replication.
- Using a non-transactional storage engine such as MyISAM. Read replicas require a transactional storage engine. Replication is only supported for the following storage engines: InnoDB for MySQL, InnoDB for MariaDB 10.2 or higher, or XtraDB for MariaDB 10.1 or lower.

You can convert a MyISAM table to InnoDB with the following command:

```
alter table <schema>.<table_name> engine=innodb;
```

- Using unsafe non-deterministic queries such as `SYSDATE()`. For more information, see [Determination of Safe and Unsafe Statements in Binary Logging](#).

The following steps can help resolve your replication error:

- If you encounter a logical error and you can safely skip the error, follow the steps described in [Skipping the Current Replication Error \(p. 685\)](#). Your MySQL or MariaDB DB instance must be running a version that includes the `mysql_rds_skip_repl_error` procedure. For more information, see [mysql.rds_skip_repl_error \(p. 707\)](#).
- If you encounter a binlog position issue, you can change the slave replay position with the `mysql_rds_next_master_log` command. Your MySQL or MariaDB DB instance must be running a version that supports the `mysql_rds_next_master_log` command in order to change the slave replay position. For version information, see [mysql.rds_next_master_log \(p. 708\)](#).
- If you encounter a temporary performance issue due to high DML load, you can set the `innodb_flush_log_at_trx_commit` parameter to `2` in the DB parameter group on the Read Replica. Doing this can help the Read Replica catch up, though it temporarily reduces atomicity, consistency, isolation, and durability (ACID).
- You can delete the Read Replica and create an instance using the same DB instance identifier so that the endpoint remains the same as that of your old Read Replica.

If a replication error is fixed, the **Replication State** changes to **replicating**. For more information, see [Troubleshooting a MySQL Read Replica Problem \(p. 661\)](#).

Creating Triggers with Binary Logging Enabled Requires SUPER Privilege

When trying to create triggers in an RDS MySQL or MariaDB DB instance, you might receive the following error:

```
"You do not have the SUPER privilege and binary logging is enabled"
```

To use triggers when binary logging is enabled requires the SUPER privilege, which is restricted for RDS MySQL and MariaDB DB instances. You can create triggers when binary logging is enabled without the SUPER privilege by setting the `log_bin_trust_function_creators` parameter to true. To set the `log_bin_trust_function_creators` to true, create a new DB parameter group or modify an existing DB parameter group.

To create a new DB parameter group that allows you to create triggers in your RDS MySQL or MariaDB DB instance with binary logging enabled, use the following CLI commands. To modify an existing parameter group, start with step 2.

To create a new parameter group to allow triggers with binary logging enabled using the CLI

1. Create a new parameter group.

For Linux, OS X, or Unix:

```
aws rds create-db-parameter-group \
--db-parameter-group-name allow-triggers \
--db-parameter-group-family mysql5.5 \
--description "parameter group allowing triggers"
```

For Windows:

```
aws rds create-db-parameter-group ^
--db-parameter-group-name allow-triggers ^
--db-parameter-group-family mysql5.5 ^
--description "parameter group allowing triggers"
```

2. Modify the DB parameter group to allow triggers.

For Linux, OS X, or Unix:

```
aws rds modify-db-parameter-group \
--db-parameter-group-name allow-triggers \
--parameters "name=log_bin_trust_function_creators,value=true, method=pending-reboot"
```

For Windows:

```
aws rds modify-db-parameter-group ^
--db-parameter-group-name allow-triggers ^
--parameters "name=log_bin_trust_function_creators,value=true, method=pending-reboot"
```

3. Modify your DB instance to use the new DB parameter group.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--db-parameter-group-name allow-triggers \
--apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--db-parameter-group-name allow-triggers ^
```

```
--apply-immediately
```

4. In order for the changes to take effect, manually reboot the DB instance.

```
aws rds reboot-db-instance mydbinstance
```

Diagnosing and Resolving Point-In-Time Restore Failures

Restoring a DB Instance That Includes Temporary Tables

When attempting a Point-In-Time Restore (PITR) of your MySQL or MariaDB DB instance, you might encounter the following error:

```
Database instance could not be restored because there has been incompatible database activity for restore functionality. Common examples of incompatible activity include using temporary tables, in-memory tables, or using MyISAM tables. In this case, use of Temporary table was detected.
```

PITR relies on both backup snapshots and binlogs from MySQL or MariaDB to restore your DB instance to a particular time. Temporary table information can be unreliable in binlogs and can cause a PITR failure. If you use temporary tables in your MySQL or MariaDB DB instance, you can minimize the possibility of a PITR failure by performing more frequent backups. A PITR failure is most probable in the time between a temporary table's creation and the next backup snapshot.

Restoring a DB Instance That Includes In-Memory Tables

You might encounter a problem when restoring a database that has in-memory tables. In-memory tables are purged during a restart. As a result, your in-memory tables might be empty after a reboot. We recommend that when you use in-memory tables, you architect your solution to handle empty tables in the event of a restart. If you are using in-memory tables with replicated DB instances, you might need to recreate the Read Replicas after a restart if a Read Replica reboots and is unable to restore data from an empty in-memory table.

For more information about backups and PITR, see [Working With Backups \(p. 208\)](#) and [Restoring a DB Instance to a Specified Time \(p. 237\)](#).

Slave Down or Disabled Error

When you call the `mysql.rds_skip_repl_error` command, you might receive the following error message: `Slave is down or disabled`.

This error message appears because replication has stopped and could not be restarted.

If you need to skip a large number of errors, the replication lag can increase beyond the default retention period for binary log files. In this case, you might encounter a fatal error due to binary log files being purged before they have been replayed on the replica. This purge causes replication to stop, and you can no longer call the `mysql.rds_skip_repl_error` command to skip replication errors.

You can mitigate this issue by increasing the number of hours that binary log files are retained on your replication master. After you have increased the binlog retention time, you can restart replication and call the `mysql.rds_skip_repl_error` command as needed.

To set the binlog retention time, use the [`mysql.rds_set_configuration` \(p. 711\)](#) procedure and specify a configuration parameter of 'binlog retention hours' along with the number of hours to retain binlog files

on the DB cluster, up to 720 (30 days). The following example sets the retention period for binlog files to 48 hours:

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

Read Replica Create Fails or Replication Breaks With Fatal Error 1236

After changing default parameter values for a MySQL or MariaDB DB instance, you might encounter one of the following problems:

- You are unable to create a Read Replica for the DB instance.
- Replication fails with `fatal error 1236`.

Some default parameter values for MySQL or MariaDB DB instances help to ensure the database is ACID compliant and Read Replicas are crash-safe by making sure that each commit is fully synchronized by writing the transaction to the binary log before it is committed. Changing these parameters from their default values to improve performance can cause replication to fail when a transaction has not been written to the binary log.

To resolve this issue, set the following parameter values:

- `sync-binlog = 1`
- `innodb_support_xa = 1`
- `innodb_flush_log_at_trx_commit = 1`

Amazon RDS Oracle GoldenGate Issues

Retaining Logs for Sufficient Time

The source database must retain archived redo logs. The duration for log retention is specified in hours. The duration should exceed any potential downtime of the source instance or any potential period of communication or networking issues for the source instance, so that Oracle GoldenGate can recover logs from the source instance as needed. The absolute minimum value required is one (1) hour of logs retained. If you don't have log retention enabled, or if the retention value is too small, you will receive the following message:

```
2014-03-06 06:17:27  ERROR  OGG-00446  error 2 (No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log
for sequence 1306Not able to establish initial position for begin time 2014-03-06
06:16:55.
```

Cannot Connect to Amazon RDS SQL Server DB Instance

When you have problems connecting to a DB instance using SQL Server Management Studio, the following are some common causes:

- The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the instance's security group are not in sync. If you use your DB instance's endpoint and port with Microsoft SQL Server Management Studio and cannot connect, the problem is most likely the egress or ingress rules on your firewall. To grant access, you must create your own security group with specific ingress and egress rules for your situation. For more information about security groups, see [Controlling Access with Security Groups \(p. 394\)](#).
- The port you specified when you created the DB instance cannot be used to send or receive communications due to your local firewall restrictions. In this case, check with your network administrator to determine if your network allows the specified port to be used for inbound and outbound communication.
- Your DB instance is still being created and is not yet available. Depending on the size of your DB instance, it can take up to 20 minutes before an instance is available.

If you can send and receive communications through the port you specified, check for the following SQL Server errors:

- **Could not open a connection to SQL Server - Microsoft SQL Server, Error: 53** – You must include the port number when you specify the server name when using Microsoft SQL Server Management Studio. For example, the server name for a DB instance (including the port number) might be: `sqlsvr-pdz.c6c8mdfntzgv0.region.rds.amazonaws.com,1433`.
- **No connection could be made because the target machine actively refused it - Microsoft SQL Server, Error: 10061** – In this case, you reached the DB instance but the connection was refused. This error is often caused by an incorrect user name or password.

Cannot Connect to Amazon RDS PostgreSQL DB Instance

The most common problem when attempting to connect to a PostgreSQL DB instance is that the security group assigned to the DB instance has incorrect access rules. By default, DB instances do not allow access; access is granted through a security group. To grant access, you must create your own security group with specific ingress and egress rules for your situation. For more information about creating a security group for your DB instance, see [Provide Access to Your DB Instance in Your VPC by Creating a Security Group \(p. 8\)](#).

The most common error is `could not connect to server: Connection timed out`. If you receive this error, check that the host name is the DB instance endpoint and that the port number is correct. Check that the security group assigned to the DB instance has the necessary rules to allow access through your local firewall.

Cannot Set Backup Retention Period to 0

There are several reasons why you may need to set the backup retention period to 0. For example, you can disable automatic backups immediately by setting the retention period to 0. If you set the value to 0 and receive a message saying that the retention period must be between 1 and 35, check to make sure you haven't setup a read replica for the instance. Read replicas require backups for managing read replica logs, thus, you can't set the retention period of 0.

Amazon RDS Application Programming Interface (API) Reference

In addition to the AWS Management Console, and the AWS Command Line Interface (AWS CLI), Amazon Relational Database Service (Amazon RDS) also provides an application programming interface (API). You can use the API to automate tasks for managing your DB instances and other objects in Amazon RDS.

- For an alphabetical list of API actions, see [API Actions](#).
- For an alphabetical list of data types, see [Data Types](#).
- For a list of common query parameters, see [Common Parameters](#).
- For descriptions of the error codes, see [Common Errors](#).

For more information about the AWS CLI, see [AWS Command Line Interface Reference for Amazon RDS](#).

Topics

- [Using the Query API \(p. 1089\)](#)
- [Troubleshooting Applications on Amazon RDS \(p. 1090\)](#)

Using the Query API

The following sections discuss the parameters and request authentication used with the Query API.

Query Parameters

HTTP Query-based requests are HTTP requests that use the HTTP verb GET or POST and a Query parameter named `Action`.

Each Query request must include some common parameters to handle authentication and selection of an action.

Some operations take lists of parameters. These lists are specified using the `param.n` notation. Values of *n* are integers starting from 1.

For information about Amazon RDS regions and endpoints, go to [Amazon Relational Database Service \(RDS\)](#) in the Regions and Endpoints section of the *Amazon Web Services General Reference*.

Query Request Authentication

You can only send Query requests over HTTPS, and you must include a signature in every Query request. You must use either AWS signature version 4 or signature version 2. For more information, see [Signature Version 4 Signing Process](#) and [Signature Version 2 Signing Process](#).

Troubleshooting Applications on Amazon RDS

Topics

- [Retrieving Errors \(p. 1090\)](#)
- [Troubleshooting Tips \(p. 1090\)](#)

Amazon RDS provides specific and descriptive errors to help you troubleshoot problems while interacting with the Amazon RDS API.

Retrieving Errors

Typically, you want your application to check whether a request generated an error before you spend any time processing results. The easiest way to find out if an error occurred is to look for an `Error` node in the response from the Amazon RDS API.

XPath syntax provides a simple way to search for the presence of an `Error` node, as well as an easy way to retrieve the error code and message. The following code snippet uses Perl and the `XML::XPath` module to determine if an error occurred during a request. If an error occurred, the code prints the first error code and message in the response.

```
use XML::XPath;
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xp->findvalue("//Error[1]/Code"), "\n", " ",
$xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Troubleshooting Tips

We recommend the following processes to diagnose and resolve problems with the Amazon RDS API.

- Verify that Amazon RDS is operating normally in the AWS Region you are targeting by visiting <http://status.aws.amazon.com>.
- Check the structure of your request

Each Amazon RDS operation has a reference page in the *Amazon RDS API Reference*. Double-check that you are using parameters correctly. In order to give you ideas regarding what might be wrong, look at the sample requests or user scenarios to see if those examples are doing similar operations.

- Check the forum

Amazon RDS has a development community forum where you can search for solutions to problems others have experienced along the way. To view the forum, go to

<https://forums.aws.amazon.com/>

Document History

- **Latest documentation update:** December 19, 2018
- **Current API version:** 2014-10-31

The following table describes important changes in each release of the *Amazon RDS User Guide* after May 2018. For notification about updates to this documentation, you can subscribe to an RSS feed.

update-history-change	update-history-description	update-history-date
Amazon RDS for PostgreSQL supports new minor versions (p. 1091)	Amazon RDS for PostgreSQL now supports the following new minor versions: 10.6, 9.6.11, 9.5.15, 9.4.20, and 9.3.25. For more information, see Amazon RDS for PostgreSQL Versions and Extensions .	December 19, 2018
Amazon RDS for PostgreSQL supports db.r5 DB instance classes (p. 1091)	You can now create Amazon RDS DB instances running PostgreSQL that use the db.r5 DB instance classes. For more information, see DB Instance Class .	December 19, 2018
Amazon RDS for PostgreSQL now supports restricted password management (p. 1091)	Amazon RDS for PostgreSQL enables you to restrict who can manage user passwords and password expiration changes by using the parameter <code>rds.restrict_password_commands</code> and the role <code>rds_password</code> . For more information, see Restricting Password Management .	December 19, 2018
Amazon RDS for PostgreSQL supports uploading database logs to Amazon CloudWatch Logs (p. 1091)	Amazon RDS for PostgreSQL supports uploading database logs to CloudWatch Logs. For more information, see Publishing PostgreSQL Logs to CloudWatch Logs .	December 10, 2018
Amazon RDS for Oracle supports db.r5 DB instance classes (p. 1091)	You can now create Amazon RDS DB instances running Oracle that use the db.r5 DB instance classes. For more information, see DB Instance Class .	November 20, 2018
Retain backups when deleting a DB instance (p. 1091)	Amazon RDS supports retaining automated backups when you delete a DB instance. For more information, see Working with Backups .	November 15, 2018

Amazon RDS for PostgreSQL supports db.m5 DB instance classes (p. 1091)	You can now create Amazon RDS DB instances running PostgreSQL that use the db.m5 DB instance classes. For more information, see DB Instance Class .	November 15, 2018
Amazon RDS for Oracle supports a new major version (p. 1091)	You can now create Amazon RDS DB instances running Oracle version 12.2. For more information, see Oracle 12c Version 12.2.0.1 with Amazon RDS .	November 13, 2018
Amazon RDS for Oracle October 2018 PSU (p. 1091)	Amazon RDS for Oracle has released database engine versions 12.1.0.2.v14 and 11.2.0.4.v18 to support the October 2018 Oracle Database Patch Set Update (PSU). For more information, see Oracle Database Engine Release Notes .	November 13, 2018
Amazon RDS for SQL Server supports Always On (p. 1091)	Amazon RDS for SQL Server supports Always On. For more information, see Multi-AZ Deployments for Microsoft SQL Server .	November 8, 2018
Amazon RDS for PostgreSQL supports outbound network access using custom DNS servers (p. 1091)	Amazon RDS for PostgreSQL supports outbound network access using custom DNS servers. For more information, see Using a Custom DNS Server for Outbound Network Access .	November 8, 2018
Amazon RDS for MariaDB, MySQL, and PostgreSQL supports 32 TiB of storage (p. 1091)	You can now create Amazon RDS DB instances with up to 32 TiB of storage for MySQL, MariaDB, and PostgreSQL. For more information, see DB Instance storage .	November 7, 2018
Amazon RDS for Oracle supports extended data types (p. 1091)	You can now enable extended data types on Amazon RDS DB instances running Oracle. With extended data types, the maximum size is 32,767 bytes for the VARCHAR2, NVARCHAR2, and RAW data types. For more information, see Using Extended Data Types .	November 6, 2018
Amazon RDS for Oracle supports db.m5 DB instance classes (p. 1091)	You can now create Amazon RDS DB instances running Oracle that use the db.m5 DB instance classes. For more information, see DB Instance Class .	November 2, 2018

Amazon RDS for Oracle migration from SE, SE1, or SE2 to EE (p. 1091)	You can now migrate from any Oracle Database Standard Edition (SE, SE1, or SE2) to Oracle Database Enterprise Edition (EE). For more information, see Migrating Between Oracle Editions .	October 31, 2018
Amazon RDS can now stop Multi-AZ instances (p. 1091)	Amazon RDS can now stop a DB instance that is part of a Multi-AZ deployment. Formerly, the stop instance feature had a limitation for multi-AZ instances. For more information, see Stopping an Amazon RDS DB Instance Temporarily .	October 29, 2018
Amazon RDS Performance Insights is available for Amazon RDS Oracle (p. 1091)	Amazon RDS Performance Insights is now available for Amazon RDS Oracle. For more information, see Using Amazon RDS Performance Insights .	October 29, 2018
Amazon RDS for PostgreSQL supports PostgreSQL version 11 in the Database Preview Environment (p. 1091)	Amazon RDS for PostgreSQL now supports PostgreSQL version 11 in the Database Preview Environment. For more information, see PostgreSQL Version 11 on Amazon RDS in the Database Preview Environment .	October 25, 2018
MySQL supports a new major version (p. 1091)	You can now create Amazon RDS DB instances running MySQL version 8.0. For more information, see MySQL on Amazon RDS Versions .	October 23, 2018
MariaDB supports a new major version (p. 1091)	You can now create Amazon RDS DB instances running MariaDB version 10.3. For more information, see MariaDB on Amazon RDS Versions .	October 23, 2018
Amazon RDS for Oracle supports Oracle JVM (p. 1091)	Amazon RDS for Oracle now supports the Oracle Java Virtual Machine (JVM) option. For more information, see Oracle Java Virtual Machine .	October 16, 2018
Custom parameter group for restore and point in time recovery (p. 1091)	You can now specify a custom parameter group when you restore a snapshot or perform a point in time recovery operation. For more information, see Restoring from a DB Snapshot and Restoring a DB Instance to a Specified Time .	October 15, 2018

Amazon RDS for Oracle supports 32 TiB storage (p. 1091)	You can now create Oracle RDS DB instances with up to 32 TiB of storage. For more information, see DB instance storage .	October 15, 2018
Amazon RDS for MySQL supports GTIDs (p. 1091)	Amazon RDS for MySQL now supports global transaction identifiers (GTIDs), which are unique across all DB instances and in a replication configuration. For more information, see Using GTID-Based Replication for Amazon RDS MySQL .	October 10, 2018
MySQL 5.7.23, 5.6.41, and 5.5.61 (p. 1091)	You can now create Amazon RDS DB instances running MySQL versions 5.7.23, 5.6.41, and 5.5.61. For more information, see MySQL on Amazon RDS Versions .	October 8, 2018
Amazon RDS for PostgreSQL supports new minor versions (p. 1091)	Amazon RDS for PostgreSQL now supports the following new minor versions: 10.5, 9.6.10, 9.5.14, 9.4.19, and 9.3.24. For more information, see Amazon RDS for PostgreSQL Versions and Extensions .	October 4, 2018
Amazon RDS for Oracle supports a new version of SQLT (p. 1091)	Amazon RDS for Oracle now supports SQLT version 12.2.180331. For more information, see Oracle SQLT .	October 4, 2018
Amazon RDS for Oracle July 2018 PSU (p. 1091)	Amazon RDS for Oracle has released database engine versions 12.1.0.2.v13 and 11.2.0.4.v17 to support the July 2018 Oracle Database Patch Set Update (PSU). For more information, see Oracle Database Engine Release Notes .	October 3, 2018
Amazon RDS for PostgreSQL now supports IAM authentication (p. 1091)	Amazon RDS for PostgreSQL now supports IAM authentication. For more information see IAM Database Authentication for MySQL and PostgreSQL .	September 27, 2018
You can enable deletion protection for your Amazon RDS DB instances (p. 1091)	When you enable deletion protection for a DB instance, the database cannot be deleted by any user. For more information, see Deleting a DB Instance .	September 26, 2018

Amazon RDS for MySQL and Amazon RDS for MariaDB support db.m5 DB instance classes (p. 1091)	You can now create Amazon RDS DB instances running MySQL or MariaDB that use the db.m5 DB instance classes. For more information, see DB Instance Class .	September 18, 2018
Amazon RDS now supports upgrades to SQL Server 2017 (p. 1091)	You can upgrade your existing DB instance to SQL Server 2017 from any version except SQL Server 2008. To upgrade from SQL Server 2008, first upgrade to one of the other versions first. For information, see Upgrading the Microsoft SQL Server DB Engine .	September 11, 2018
Amazon RDS for PostgreSQL now supports PostgreSQL Version 11 Beta 3 in the Database Preview Environment (p. 1091)	In this release, the Write-Ahead Log (WAL) segment size (<code>wal_segment_size</code>) is now set to 64MB. For more about PostgreSQL version 11 Beta 3, see PostgreSQL 11 Beta 3 Released . For information on the Database Preview Environment, see Working with the Database Preview Environment .	September 7, 2018
Amazon Aurora User Guide (p. 1091)	The Amazon Aurora User Guide describes all Amazon Aurora concepts and provides instructions on using the various features with both the console and the command line interface. The Amazon RDS User Guide now covers non-Aurora database engines.	August 31, 2018
Amazon RDS Performance Insights is available for Amazon RDS MySQL (p. 1091)	Amazon RDS Performance Insights is now available for Amazon RDS MySQL. For more information, see Using Amazon RDS Performance Insights .	August 28, 2018
Aurora with PostgreSQL compatibility now supports Aurora Auto Scaling (p. 1091)	Auto Scaling of Aurora replicas is now available for Aurora with PostgreSQL compatibility. For more information, see Using Amazon Aurora Auto Scaling with Aurora Replicas .	August 16, 2018
Aurora Serverless for Aurora MySQL (p. 1091)	Aurora Serverless is an on-demand, autoscaling configuration for Amazon Aurora. For more information, see Using Amazon Aurora Serverless .	August 9, 2018

MySQL 5.7.22 and 5.6.40 (p. 1091)	You can now create Amazon RDS DB instances running MySQL versions 5.7.22 and 5.6.40. For more information, see MySQL on Amazon RDS Versions .	August 6, 2018
Aurora is now available in the China (Ningxia) region (p. 1091)	Aurora MySQL and Aurora PostgreSQL are now available in the China (Ningxia) region. For more information, see Availability for Amazon Aurora MySQL and Availability for Amazon Aurora PostgreSQL .	August 6, 2018
Amazon RDS for MySQL supports delayed replication (p. 1091)	Amazon RDS for MySQL now supports delayed replication as a strategy for disaster recovery. For more information, see Configuring Delayed Replication with MySQL .	August 6, 2018
Amazon RDS Performance Insights is available for Aurora MySQL (p. 1091)	Amazon RDS Performance Insights is now available for Aurora MySQL. For more information, see Using Amazon RDS Performance Insights .	August 6, 2018
Amazon RDS Performance Insights integration with Amazon CloudWatch (p. 1091)	Amazon RDS Performance Insights automatically publishes metrics to Amazon CloudWatch. For more information, see Performance Insights Metrics Published to CloudWatch .	August 6, 2018
Amazon RDS recommendations (p. 1091)	Amazon RDS now provides automated recommendations for database resources. For more information, see Using Amazon RDS Recommendations .	July 25, 2018
Amazon RDS for PostgreSQL supports new minor versions (p. 1091)	Amazon RDS for PostgreSQL now supports the following new minor versions: 10.4, 9.6.9, 9.5.13, 9.4.18, and 9.3.23. For more information, see Amazon RDS for PostgreSQL Versions and Extensions .	July 25, 2018
Incremental snapshot copies across AWS Regions (p. 1091)	Amazon RDS supports incremental snapshot copies across AWS Regions for both unencrypted and encrypted instances. For more information, see Copying Snapshots Across AWS Regions .	July 24, 2018

Amazon RDS Performance Insights is available for Amazon RDS for PostgreSQL (p. 1091)	Amazon RDS Performance Insights is now available for Amazon RDS for PostgreSQL. For more information, see Using Amazon RDS Performance Insights .	July 18, 2018
Amazon RDS for Oracle supports Oracle APEX version 5.1.4.v1 (p. 1091)	Amazon RDS for Oracle now supports Oracle Application Express (APEX) version 5.1.4.v1. For more information, see Oracle Application Express .	July 10, 2018
Amazon RDS for Oracle supports publishing logs to Amazon CloudWatch Logs (p. 1091)	Amazon RDS for Oracle now supports publishing alert, audit, trace, and listener log data to a log group in CloudWatch Logs. For more information, see Publishing Oracle Logs to Amazon CloudWatch Logs .	July 9, 2018
MariaDB 10.2.15, 10.1.34, and 10.0.35 (p. 1091)	You can now create Amazon RDS DB instances running MariaDB versions 10.2.15, 10.1.34, and 10.0.35. For more information, see MariaDB on Amazon RDS Versions .	July 5, 2018
Aurora PostgreSQL 1.2 is available and compatible with PostgreSQL 9.6.8 (p. 1091)	Aurora PostgreSQL 1.2 is now available and is compatible with PostgreSQL 9.6.8. For more information, see Version 1.2 .	June 27, 2018
Read Replicas for Amazon RDS PostgreSQL support Multi-AZ deployments (p. 1091)	RDS Read Replicas in Amazon RDS PostgreSQL now support multiple Availability Zones. For more information, see Working with PostgreSQL Read Replicas .	June 25, 2018
Performance Insights available for Aurora PostgreSQL (p. 1091)	Performance Insights is generally available for Aurora PostgreSQL, with support for extended retention of performance data. For more information, see Using Amazon RDS Performance Insights .	June 21, 2018
Aurora PostgreSQL available in western US (Northern California) region (p. 1091)	Aurora PostgreSQL is now available in the western United States (Northern California) region. For more information, see Availability for Amazon Aurora PostgreSQL .	June 11, 2018

<p>Amazon RDS for Oracle now supports CPU configuration (p. 1091)</p> <p>Amazon RDS for Oracle April 2018 PSU (p. 1091)</p>	<p>Amazon RDS for Oracle supports June 5, 2018 configuring the number of CPU cores and the number of threads for each core for the processor of a DB instance class. For more information, see Configuring the Processor of the DB Instance Class.</p> <p>Amazon RDS for Oracle has released database engine versions 12.1.0.2.v12 and 11.2.0.4.v16 to support the April 2018 Oracle Database Patch Set Update (PSU). For more information, see Oracle Database Engine Release Notes.</p>	
---	---	--

Earlier Updates

The following table describes the important changes in each release of the *Amazon RDS User Guide* before June 2018.

Change	Description	Date Changed
Amazon RDS for PostgreSQL now supports PostgreSQL Version 11 Beta 1 in the Database Preview Environment	PostgreSQL version 11 Beta 1 contains several improvements that are described in PostgreSQL 11 Beta 1 Released! For information on the Database Preview Environment, see Working with the Database Preview Environment (p. 1026) .	May 31, 2018
Amazon RDS for Oracle now supports TLS versions 1.0 and 1.2	Amazon RDS for Oracle supports Transport Layer Security (TLS) versions 1.0 and 1.2. For more information, see TLS Versions for the Oracle SSL Option (p. 818) .	May 30, 2018
Aurora MySQL supports publishing logs to Amazon CloudWatch Logs	Aurora MySQL now supports publishing general, slow, audit, and error log data to a log group in CloudWatch Logs. For more information, see Publishing Aurora MySQL to CloudWatch Logs .	May 23, 2018
Database Preview Environment for Amazon RDS PostgreSQL	You can now launch a new instance of Amazon RDS PostgreSQL in a preview mode. For more information about the Database Preview Environment see, Working with the Database Preview Environment (p. 1026) .	May 22, 2018
Amazon RDS for Oracle DB instances support new DB instance classes	Oracle DB instances now support the db.x1e and db.x1 DB instance classes. For more information, see DB Instance Class (p. 82) and DB Instance Class Support for Oracle (p. 720) .	May 22, 2018
Amazon RDS PostgreSQL now supports	You can now use <code>postgres_fdw</code> to connect to a remote server from a Read Replica. For more information	May 17, 2018

Change	Description	Date Changed
postgres_fdw on a Read Replica.	see, Accessing External Data with the postgres_fdw Extension (p. 1024) .	
Amazon RDS for Oracle now supports setting sqlnet.ora parameters	You can now set sqlnet.ora parameters with Amazon RDS for Oracle. For more information, see Modifying Oracle sqlnet.ora Parameters (p. 767) .	May 10, 2018
Aurora PostgreSQL available in Asia Pacific (Seoul) region.	Aurora PostgreSQL is now available in the Asia Pacific (Seoul) region. For more information, see Availability for Amazon Aurora PostgreSQL .	May 9, 2018
Aurora MySQL supports backtracking	Aurora MySQL now supports "rewinding" a DB cluster to a specific time, without restoring data from a backup. For more information, see Backtracking an Aurora DB Cluster .	May 9, 2018
Aurora MySQL supports encrypted migration and replication from external MySQL	Aurora MySQL now supports encrypted migration and replication from an external MySQL database. For more information, see Migrating Data from an External MySQL Database to an Amazon Aurora MySQL DB Cluster and Replication Between Aurora and MySQL or Between Aurora and Another Aurora DB Cluster .	April 25, 2018
Aurora with PostgreSQL compatibility support for the Copy-on-Write protocol.	You can now clone databases in an Aurora PostgreSQL database cluster. For more information see, Cloning Databases in an Aurora DB Cluster .	April 10, 2018
MariaDB 10.2.12, 10.1.31, and 10.0.34	You can now create Amazon RDS DB instances running MariaDB versions 10.2.12, 10.1.31, and 10.0.34. For more information, see MariaDB on Amazon RDS Versions (p. 434) .	March 21, 2018
Aurora PostgreSQL Support for new regions	Aurora PostgreSQL is now available in the EU (London) and Asia Pacific (Singapore) regions. For more information, see Availability for Amazon Aurora PostgreSQL .	March 13, 2018
MySQL 5.7.21, 5.6.39, and 5.5.59	You can now create Amazon RDS DB instances running MySQL versions 5.7.21, 5.6.39, and 5.5.59. For more information, see MySQL on Amazon RDS Versions (p. 589) .	March 9, 2018
Amazon RDS for Oracle now supports Oracle REST Data Services	Amazon RDS for Oracle supports Oracle REST Data Services as part of the APEX option. For more information, see Oracle Application Express (p. 788) .	March 9, 2018

Change	Description	Date Changed
Amazon Aurora with MySQL compatibility available in new AWS Region	Aurora MySQL is now available in the Asia Pacific (Singapore) region. For the complete list of AWS Regions for Aurora MySQL, see Availability for Amazon Aurora MySQL .	March 6, 2018
Support for PostgreSQL 10.1	Amazon RDS now supports version 10.1 of PostgreSQL. For more information, see PostgreSQL Version 10.1 on Amazon RDS (p. 1033)	February 27, 2018
Oracle January 2018 PSU	Amazon RDS for Oracle has released database engine versions 12.1.0.2.v11 and 11.2.0.4.v15 to support the January 2018 Oracle Database Patch Set Update (PSU). For more information, see Oracle Database Engine Release Notes (p. 921) .	February 22, 2018
Amazon RDS DB instances running Microsoft SQL Server support change data capture (CDC)	DB instances running Amazon RDS for Microsoft SQL Server now support change data capture (CDC). For more information, see Change Data Capture Support for Microsoft SQL Server DB Instances (p. 497) .	February 6, 2018
Aurora MySQL supports a new major version	You can now create Aurora MySQL DB clusters running MySQL version 5.7. For more information, see Amazon Aurora MySQL Database Engine Updates 2018-02-06 .	February 6, 2018
Support for PostgreSQL 9.6.6	Amazon RDS PostgreSQL now supports version 9.6.6. This release also includes support for the prefix and orafce extensions. For more information, see PostgreSQL Version 9.6.6 on Amazon RDS (p. 1034) .	January 19, 2018
Publish MySQL and MariaDB logs to Amazon CloudWatch Logs	You can now publish MySQL and MariaDB log data to CloudWatch Logs. For more information, see Publishing MySQL Logs to CloudWatch Logs (p. 321) and Publishing MariaDB Logs to Amazon CloudWatch Logs (p. 312) .	January 17, 2018
Multi-AZ support for Read Replicas	You can now create a Read Replica as a Multi-AZ DB instance. Amazon RDS creates a standby of your replica in another Availability Zone for failover support for the replica. Creating your Read Replica as a Multi-AZ DB instance is independent of whether the source database is a Multi-AZ DB instance. For more information, see Working with Read Replicas of MariaDB, MySQL, and PostgreSQL DB Instances (p. 143) .	January 11, 2018
Amazon RDS for MariaDB supports a new major version	You can now create Amazon RDS DB instances running MariaDB version 10.2. For more information, see MariaDB 10.2 Support on Amazon RDS (p. 435) .	January 3, 2018
Amazon Aurora with PostgreSQL compatibility available in new AWS Region	Aurora PostgreSQL is now available in the EU (Paris) region. For the complete list of AWS Regions for Aurora PostgreSQL, see Availability for Amazon Aurora PostgreSQL .	December 22, 2017

Change	Description	Date Changed
Aurora PostgreSQL supports new instance types	Aurora PostgreSQL now supports new instance types. For the complete list of instance types, see Choosing the DB Instance Class .	December 20, 2017
Oracle October 2017 PSU	Amazon RDS for Oracle has released database engine versions 12.1.0.2.v10 and 11.2.0.4.v14 to support the October 2017 Oracle Database Patch Set Update (PSU). For more information, see Oracle Database Engine Release Notes (p. 921) .	December 19, 2017
Amazon Aurora with MySQL compatibility available in new AWS Region	Aurora MySQL is now available in the EU (Paris) region. For the complete list of AWS Regions for Aurora MySQL, see Availability for Amazon Aurora MySQL .	December 18, 2017
Aurora MySQL supports hash joins	This feature can improve query performance when you need to join a large amount of data by using an equijoin. For more information, see Working with Hash Joins in Aurora MySQL .	December 11, 2017
Aurora MySQL supports native functions to invoke AWS Lambda functions	You can call the native functions <code>lambda_sync</code> and <code>lambda_async</code> when you use Aurora MySQL. For more information, see Invoking a Lambda Function from an Amazon Aurora MySQL DB Cluster .	December 11, 2017
Added Aurora PostgreSQL HIPAA compliance	Aurora PostgreSQL now supports building HIPAA compliant applications, see Working with Amazon Aurora PostgreSQL .	December 6, 2017
Additional AWS Regions available for Amazon Aurora with PostgreSQL compatibility	Amazon Aurora with PostgreSQL compatibility is now available in four new AWS Regions. For more information, see Availability for Amazon Aurora PostgreSQL .	November 22, 2017
Modify storage for Amazon RDS DB instances running Microsoft SQL Server	You can now modify the storage of your Amazon RDS DB instances running SQL Server. For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 521) .	November 21, 2017
Amazon RDS supports 16 TiB storage for Linux-based engines	You can now create MySQL, MariaDB, PostgreSQL, and Oracle RDS DB instances with up to 16 TiB of storage. For more information, see DB instance storage (p. 103) .	November 21, 2017
Amazon RDS supports fast scale up of storage	You can now add storage to MySQL, MariaDB, PostgreSQL, and Oracle RDS DB instances in a few minutes. For more information, see DB instance storage (p. 103) .	November 21, 2017
Amazon RDS supports MariaDB versions 10.1.26 and 10.0.32	You can now create Amazon RDS DB instances running MariaDB versions 10.1.26 and 10.0.32. For more information, see MariaDB on Amazon RDS Versions (p. 434) .	November 20, 2017

Change	Description	Date Changed
Amazon RDS for Microsoft SQL Server now supports new DB instance classes	You can now create Amazon RDS DB instances running SQL Server that use the db.r4 and db.m4.16xlarge DB instance classes. For more information, see DB Instance Class Support for Microsoft SQL Server (p. 491) .	November 20, 2017
Amazon RDS for MySQL and MariaDB now supports new DB instance classes	You can now create Amazon RDS DB instances running MySQL and MariaDB that use the db.r4, db.m4.16xlarge, db.t2.xlarge, and db.t2.2xlarge DB instance classes. For more information, see DB Instance Class (p. 82) .	November 20, 2017
SQL Server 2017	You can now create Amazon RDS DB instances running Microsoft SQL Server 2017. You can also create DB instances running SQL Server 2016 SP1 CU5. For more information, see Microsoft SQL Server on Amazon RDS (p. 488) .	November 17, 2017
Restore MySQL backups from Amazon S3	You can now create a backup of your on-premises database, store it on Amazon S3, and then restore the backup file onto a new Amazon RDS DB instance running MySQL. For more information, see Restoring a Backup into an Amazon RDS MySQL DB Instance (p. 631) .	November 17, 2017
Auto Scaling with Aurora Replicas	Amazon Aurora MySQL now supports Aurora Auto Scaling. Aurora Auto Scaling dynamically adjusts the number of Aurora Replicas based on increases or decreases in connectivity or workload. For more information, see Using Amazon Aurora Auto Scaling with Aurora Replicas .	November 17, 2017
Oracle default edition support	Amazon RDS for Oracle DB instances now supports setting the default edition for the DB instance. For more information, see Setting the Default Edition for a DB Instance (p. 859) .	November 3, 2017
Oracle DB instance file validation	Amazon RDS for Oracle DB instances now supports validating DB instance files with the Oracle Recovery Manager (RMAN) logical validation utility. For more information, see Validating DB Instance Files (p. 859) .	November 3, 2017
Oracle July 2017 PSU	Amazon RDS for Oracle has released database engine versions 12.1.0.2.v9 and 11.2.0.4.v13 to support the July 2017 Oracle Database Patch Set Update (PSU). For more information, see Oracle Database Engine Release Notes (p. 921) .	November 3, 2017
Management Agent for OEM 13c	Amazon RDS Oracle DB instances now support the Management Agent for Oracle Enterprise Manager (OEM) Cloud Control 13c. For more information, see Oracle Management Agent for Enterprise Manager Cloud Control (p. 799) .	November 1, 2017
PostgreSQL 9.6.5, 9.5.9, 9.4.14, and 9.3.19	You can now create Amazon RDS DB instances running PostgreSQL versions 9.6.5., 9.5.9, 9.4.14, and 9.3.19. For more information, see Supported PostgreSQL Database Versions (p. 1030) .	November 1, 2017

Change	Description	Date Changed
Storage reconfiguration for Microsoft SQL Server snapshots	You can now reconfigure the storage when you restore a snapshot to an Amazon RDS DB instance running Microsoft SQL Server. For more information, see Restoring from a DB Snapshot (p. 218) .	October 26, 2017
Asynchronous key prefetch for Aurora with MySQL compatibility	Asynchronous key prefetch (AKP) improves the performance of noncached index joins, by prefetching keys in memory ahead of when they are needed. For more information, see Working with Asynchronous Key Prefetch in Amazon Aurora .	October 26, 2017
MySQL 5.7.19, 5.6.37, and 5.5.57	You can now create Amazon RDS DB instances running MySQL versions 5.7.19, 5.6.37, and 5.5.57. For more information, see MySQL on Amazon RDS Versions (p. 589) .	October 25, 2017
General availability of Amazon Aurora with PostgreSQL compatibility	Amazon Aurora with PostgreSQL compatibility makes it simple and cost-effective to set up, operate, and scale your new and existing PostgreSQL deployments, thus freeing you to focus on your business and applications. For more information, see Working with Amazon Aurora PostgreSQL .	October 24, 2017
Amazon RDS for Oracle DB instances support new DB instance classes	Amazon RDS Oracle DB instances now support Memory Optimized Next Generation (db.r4) instance classes. Amazon RDS Oracle DB instances also now support the following new current generation instance classes: db.m4.16xlarge, db.t2.xlarge, and db.t2.2xlarge. For more information, see DB Instance Class (p. 82) and DB Instance Class Support for Oracle (p. 720) .	October 23, 2017
New feature	Your new and existing Reserved Instances can now cover multiple sizes in the same DB instance class. Size-flexible reserved instances are available for DB instances with the same AWS Region, database engine, and instance family, and across AZ configuration. Size-flexible reserved instances are available for the following database engines: Amazon Aurora, MariaDB, MySQL, Oracle (Bring Your Own License), PostgreSQL. For more information, see Size-Flexible Reserved DB Instances (p. 195) .	October 11, 2017
New feature	You can now use the Oracle SQLT option to tune a SQL statement for optimal performance. For more information, see Oracle SQLT (p. 825) .	September 22, 2017
New feature	If you have existing manual DB snapshots of your Amazon RDS Oracle DB instances, you can now upgrade them to a later version of the Oracle database engine. For more information, see Upgrading an Oracle DB Snapshot (p. 774) .	September 20, 2017
New feature	You can now use Oracle Spatial to store, retrieve, update, and query spatial data in your Amazon RDS DB instances running Oracle. For more information, see Oracle Spatial (p. 823) .	September 15, 2017

Change	Description	Date Changed
New feature	You can now use Oracle Locator to support internet and wireless service-based applications and partner-based GIS solutions with your Amazon RDS DB instances running Oracle. For more information, see Oracle Locator (p. 810) .	September 15, 2017
New feature	You can now use Oracle Multimedia to store, manage, and retrieve images, audio, video, and other heterogeneous media data in your Amazon RDS DB instances running Oracle. For more information, see Oracle Multimedia (p. 813) .	September 15, 2017
New feature	You can now export audit logs from your Amazon Aurora MySQL DB clusters to Amazon CloudWatch Logs. For more information, see Publishing Aurora MySQL Logs to Amazon CloudWatch Logs .	September 14, 2017
New feature	Amazon RDS now supports multiple versions of Oracle Application Express (APEX) for your DB instances running Oracle. For more information, see Oracle Application Express (p. 788) .	September 13, 2017
New feature	You can now use Amazon Aurora to migrate an unencrypted or encrypted DB snapshot or Amazon RDS MySQL DB instance to an encrypted Aurora MySQL DB cluster. For more information, see Migrating an RDS MySQL Snapshot to Aurora and Migrating Data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster by Using an Aurora Read Replica .	September 5, 2017
New feature	You can use Amazon RDS for Microsoft SQL Server databases to build HIPAA-compliant applications. For more information, see Compliance Program Support for Microsoft SQL Server DB Instances (p. 492) .	August 31, 2017
New feature	You can now use Amazon RDS for MariaDB databases to build HIPAA-compliant applications. For more information, see MariaDB on Amazon RDS (p. 432) .	August 31, 2017
New feature	You can now create Amazon RDS DB instances running Microsoft SQL Server with allocated storage up to 16 TiB, and Provisioned IOPS to storage ranges of 1:1–50:1. For more information, see DB instance storage (p. 103) .	August 22, 2017
New feature	You can now use Multi-AZ deployments for DB instances running Microsoft SQL Server in the EU (Frankfurt) region. For more information, see Multi-AZ Deployments for Microsoft SQL Server (p. 551) .	August 3, 2017
New feature	You can now create Amazon RDS DB instances running MariaDB versions 10.1.23 and 10.0.31. For more information, see MariaDB on Amazon RDS Versions (p. 434) .	July 17, 2017
New feature	Amazon RDS now supports Microsoft SQL Server Enterprise Edition with the License Included model in all AWS Regions. For more information, see Licensing Microsoft SQL Server on Amazon RDS (p. 503) .	July 13, 2017

Change	Description	Date Changed
New feature	Amazon RDS for Oracle now supports Linux kernel huge pages for increased database scalability. The use of huge pages results in smaller page tables and less CPU time spent on memory management, increasing the performance of large database instances. You can use huge pages with your Amazon RDS DB instances running all editions of Oracle versions 12.1.0.2 and 11.2.0.4. For more information, see Using Huge Pages with an Oracle DB Instance (p. 736) .	July 7, 2017
New feature	Updated to support encryption at rest (EAR) for db.t2.small and db.t2.medium DB instance classes for all non-Aurora DB engines. For more information, see Availability of Amazon RDS Encryption (p. 391) .	June 27, 2017
New feature	Updated to support Amazon Aurora in the EU (Frankfurt) region. For more information, see Availability for Amazon Aurora MySQL .	June 16, 2017
New feature	You can now specify an option group when you copy a DB snapshot across AWS regions. For more information, see Option Group Considerations (p. 222) .	June 12, 2017
New feature	You can now copy DB snapshots created from specialized DB instances across AWS regions. You can copy snapshots from DB instances that use Oracle TDE, Microsoft SQL Server TDE, and Microsoft SQL Server Multi-AZ with Mirroring. For more information, see Copying a DB Snapshot (p. 223) .	June 12, 2017
New feature	Amazon Aurora now allows you to quickly and cost-effectively copy all of your databases in an Amazon Aurora DB cluster. For more information, see Cloning Databases in an Aurora DB Cluster .	June 12, 2017
New feature	Amazon RDS now supports Microsoft SQL Server 2016 SP1 CU2. For more information, see Microsoft SQL Server on Amazon RDS (p. 488) .	June 7, 2017
New feature	Amazon RDS for Oracle has released database engine versions 12.1.0.2.v8 and 11.2.0.4.v12 to support the April 2017 Oracle Database Patch Set Update (PSU). For more information, see Oracle Database Engine Release Notes (p. 921) .	May 23, 2017
New Feature	Amazon RDS now supports PostgreSQL versions 9.6.2, 9.5.6, 9.4.11, and 9.3.16. For more information, see Supported PostgreSQL Database Versions (p. 1030)	May 3, 2017
Preview	Public preview of Amazon Aurora with PostgreSQL Compatibility. For more information, see Working with Amazon Aurora PostgreSQL .	April 19, 2017

Change	Description	Date Changed
New feature	Amazon Aurora now allows you to execute an ALTER TABLE <i>tbl_name</i> ADD COLUMN <i>col_name</i> <i>column_definition</i> operation nearly instantaneously. The operation completes without requiring the table to be copied and without materially impacting other DML statements. For more information, see Altering Tables in Amazon Aurora Using Fast DDL .	April 5, 2017
New feature	We have added a new monitoring command, SHOW VOLUME STATUS, to display the number of nodes and disks in a volume. For more information, see Displaying Volume Status for an Aurora DB Cluster .	April 5, 2017
New feature	Amazon RDS for Oracle now includes the January 2017 Oracle Database Patch Set Update (PSU). This adds support for database engine versions 12.1.0.2.v7 and 11.2.0.4.v11. For more information, see Oracle Database Engine Release Notes (p. 921) .	March 21, 2017
New feature	You can now use your own custom logic in your custom password verification functions for Oracle on Amazon RDS. For more information, see Creating Custom Functions to Verify Passwords (p. 850) .	March 21, 2017
New feature	You can now access your online and archived redo log files on your Oracle DB instances on Amazon RDS. For more information, see Accessing Transaction Logs (p. 872) .	March 21, 2017
New feature	You can now copy both encrypted and unencrypted DB cluster snapshots between accounts in the same region. For more information, see Copying a DB Cluster Snapshot Across Accounts .	March 7, 2017
New feature	You can now share encrypted DB cluster snapshots between accounts in the same region. For more information, see Sharing a DB Cluster Snapshot .	March 7, 2017
New feature	You can now replicate encrypted Amazon Aurora MySQL DB clusters to create cross-region Aurora Replicas. For more information, see Replicating Aurora MySQL DB Clusters Across AWS Regions .	March 7, 2017
New feature	You can now require that all connections to your DB instance running Microsoft SQL Server use Secure Sockets Layer (SSL). For more information, see Using SSL with a Microsoft SQL Server DB Instance (p. 555) .	February 27, 2017
New feature	You can now set your local time zone to one of 15 additional time zones. For more information, see Supported Time Zones (p. 500) .	February 27, 2017
New feature	You can now use the Amazon RDS procedure msdb.dbo.rds_shrink_tempdbfile to shrink the tempdb database on your DB instances running Microsoft SQL Server. For more information, see Shrinking the tempdb Database (p. 565) .	February 17, 2017

Change	Description	Date Changed
New feature	You can now compress your backup file when you export your Enterprise and Standard Edition Microsoft SQL Server database from an Amazon RDS DB instance to Amazon S3. For more information, see Compressing Backup Files (p. 539) .	February 17, 2017
New feature	Amazon RDS now supports custom DNS servers to resolve DNS names used in outbound network access on your DB instances running Oracle. For more information, see Setting Up a Custom DNS Server (p. 853) .	January 26, 2017
New feature	Amazon RDS now supports creating an encrypted Read Replica in another region. For more information, see Creating a Read Replica in a Different AWS Region (p. 148) and CreateDBInstanceReadReplica .	January 23, 2017
New feature	Amazon RDS now supports upgrading a MySQL DB snapshot from MySQL 5.1 to MySQL 5.5. For more information, see Upgrading a MySQL DB Snapshot (p. 623) and ModifyDBSnapshot .	January 20, 2017
New feature	Amazon RDS now supports copying an encrypted DB snapshot to another region for the MariaDB, MySQL, Oracle, PostgreSQL, and Microsoft SQL Server database engines. For more information, see Copying a DB Snapshot (p. 223) and CopyDBSnapshot .	December 20, 2016
New feature	Amazon RDS now supports migrating an Amazon RDS MySQL 5.6 DB snapshot to a new DB instance running MariaDB 10.1. For more information, see Migrating Data from a MySQL DB Snapshot to a MariaDB DB Instance (p. 466) .	December 20, 2016
New feature	Amazon Aurora MySQL now supports spatial indexing. Spatial indexing improves query performance on large datasets for queries that use spatial data. For more information, see Amazon Aurora MySQL and Spatial Data .	December 14, 2016
New feature	Amazon RDS for Oracle now includes the October 2016 Oracle Database Patch Set Update (PSU). This adds support for Oracle database engine versions 12.1.0.2.v6 and 11.2.0.4.v10. For more information, see Oracle Database Engine Release Notes (p. 921) .	December 12, 2016
New feature	Amazon RDS now supports outbound network access on your DB instances running Oracle. You can use <code>utl_http</code> , <code>utl_tcp</code> , and <code>utl_smtp</code> to connect from your DB instance to the network. For more information, see Using utl_http, utl_tcp, and utl_smtp with an Oracle DB Instance (p. 738) .	December 5, 2016
New feature	Amazon RDS has retired support for MySQL version 5.1. However, you can restore existing MySQL 5.1 snapshots to a MySQL 5.5 instance. For more information, see Supported Storage Engines for MySQL on Amazon RDS (p. 591) .	November 15, 2016

Change	Description	Date Changed
New feature	Amazon RDS now supports PostgreSQL version 9.6.1. For more information, see PostgreSQL Version 9.6.1 on Amazon RDS (p. 1036) .	November 11, 2016
New feature	Amazon RDS now supports Microsoft SQL Server 2016 RTM CU2. For more information, see Microsoft SQL Server on Amazon RDS (p. 488) .	November 4, 2016
New feature	Amazon RDS now supports major version upgrades for DB instances running Oracle. You can now upgrade your Oracle DB instances from 11g to 12c. For more information, see Upgrading the Oracle DB Engine (p. 770) .	November 2, 2016
New feature	You can now create DB instances running Microsoft SQL Server 2014 Enterprise Edition. Amazon RDS now supports SQL Server 2014 SP2 for all editions and all regions. For more information, see Microsoft SQL Server on Amazon RDS (p. 488) .	October 25, 2016
New feature	Amazon Aurora MySQL now integrates with other AWS services: You can load text or XML data into a table from an Amazon S3 bucket, or invoke an AWS Lambda function from database code. For more information, see Integrating Aurora MySQL with Other AWS Services .	October 18, 2016
New feature	You can now access the tempdb database on your Amazon RDS DB instances running Microsoft SQL Server. You can access the tempdb database by using Transact-SQL through Microsoft SQL Server Management Studio (SSMS), or any other standard SQL client application. For more information, see Accessing the tempdb Database on Microsoft SQL Server DB Instances on Amazon RDS (p. 565) .	September 29, 2016
New feature	You can now use the UTL_MAIL package with your Amazon RDS DB instances running Oracle. For more information, see Oracle UTL_MAIL (p. 838) .	September 20, 2016
New feature	Amazon RDS for Oracle now includes the July 2016 Oracle Database Patch Set Update (PSU). This adds support for Oracle database engine versions 12.1.0.2.v5, 12.1.0.1.v6, and 11.2.0.4.v9. For more information, see Oracle Database Engine Release Notes (p. 921) .	September 20, 2016
New features	You can now set the time zone of your new Microsoft SQL Server DB instances to a local time zone, to match the time zone of your applications. For more information, see Local Time Zone for Microsoft SQL Server DB Instances (p. 499) .	September 19, 2016

Change	Description	Date Changed
New features	<p>Added support for new PostgreSQL versions 9.5.4, 9.4.9, and 9.3.14. Also added support for PostgreSQL logical replication, PostgreSQL event triggers, and RAM disk for the PostgreSQL stats_temp_directory.</p> <p>For more information, see Supported PostgreSQL Database Versions (p. 1030), Logical Replication for PostgreSQL on Amazon RDS (p. 1064), Event Triggers for PostgreSQL on Amazon RDS (p. 1065), and RAM Disk for the stats_temp_directory (p. 1067).</p>	September 14, 2016
New feature	<p>You can now use the Oracle Label Security option to control access to individual table rows in your Amazon RDS DB instances running Oracle 12c. With Oracle Label Security, you can enforce regulatory compliance with a policy-based administration model, and ensure that an access to sensitive data is restricted to only users with the appropriate clearance level. For more information, see Oracle Label Security (p. 807).</p>	September 8, 2016
New feature	<p>You can now connect to an Amazon Aurora DB cluster using the reader endpoint, which load-balances connections across the Aurora Replicas that are available in the DB cluster. As clients request new connections to the reader endpoint, Aurora distributes the connection requests among the Aurora Replicas in the DB cluster. This functionality can help balance your read workload across multiple Aurora Replicas in your DB cluster. For more information, see Amazon Aurora Endpoints.</p>	September 8, 2016
New feature	<p>You can now support the Oracle Enterprise Manager Cloud Control on your Amazon RDS DB instances running Oracle. You can enable the Management Agent on your DB instances, and share data with your Oracle Management Service (OMS). For more information, see Oracle Management Agent for Enterprise Manager Cloud Control (p. 799).</p>	September 1, 2016
New feature	<p>This release adds support to get an ARN for a resource. For more information, see Getting an Existing ARN (p. 184).</p>	August 23, 2016
New feature	<p>You can now assign up to 50 tags for each Amazon RDS resource, for managing your resources and tracking costs. For more information, see Tagging Amazon RDS Resources (p. 138).</p>	August 19, 2016
New feature	<p>Amazon RDS now supports the License Included model for Oracle Standard Edition Two. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 742).</p> <p>You can now change the license model of your Amazon RDS DB instances running Microsoft SQL Server and Oracle. For more information, see Licensing Microsoft SQL Server on Amazon RDS (p. 503) and Oracle Licensing (p. 719).</p>	August 5, 2016

Change	Description	Date Changed
New feature	<p>You can now use the AWS Management Console to easily move your DB instance to a different VPC, or to a different subnet group in the same VPC. For more information, see Updating the VPC for a DB Instance (p. 425).</p> <p>If your DB instance is not in a VPC, you can now use the AWS Management Console to easily move your DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 426).</p>	August 4, 2016
New feature	Amazon RDS now supports native backup and restore for Microsoft SQL Server databases using full backup files (.bak files). You can now easily migrate SQL Server databases to Amazon RDS, and import and export databases in a single, easily-portable file, using Amazon S3 for storage, and AWS KMS for encryption. For more information, see Importing and Exporting SQL Server Databases (p. 533) .	July 27, 2016
New feature	You can now copy the source files from a MySQL database to an Amazon Simple Storage Service (Amazon S3) bucket, and then restore an Amazon Aurora DB cluster from those files. This option can be considerably faster than migrating data using <code>mysqldump</code> . For more information, see Migrating Data from an External MySQL Database to an Aurora MySQL DB Cluster .	July 20, 2016
New feature	You can now restore an unencrypted Amazon Aurora DB cluster snapshot to create an encrypted Amazon Aurora DB cluster by including an AWS Key Management Service (AWS KMS) encryption key during the restore operation. For more information, see Encrypting Amazon RDS Resources .	June 30, 2016
New feature	Amazon RDS for Oracle now includes the April 2016 Oracle Database Patch Set Update (PSU). This PSU adds support for Oracle database engine versions 12.1.0.2.v4, 12.1.0.1.v5, and 11.2.0.4.v8. For more information, see Oracle Database Engine Release Notes (p. 921) .	June 17, 2016
New feature	You can use the Oracle Repository Creation Utility (RCU) to create a repository on Amazon RDS for Oracle. For more information, see Using the Oracle Repository Creation Utility on Amazon RDS for Oracle (p. 913) .	June 17, 2016
New feature	Adds support for PostgreSQL cross-region Read Replicas. For more information, see Creating a Read Replica in a Different AWS Region (p. 148) .	June 16, 2016
New feature	You can now use the AWS Management Console to easily add Multi-AZ with Mirroring to a Microsoft SQL Server DB instance. For more information, see Adding Multi-AZ to a Microsoft SQL Server DB Instance (p. 551) .	June 9, 2016

Change	Description	Date Changed
New feature	You can now use Multi-AZ Deployments Using SQL Server Mirroring in the following additional regions: Asia Pacific (Sydney), Asia Pacific (Tokyo), and South America (Sao Paulo). For more information, see Multi-AZ Deployments for Microsoft SQL Server (p. 551) .	June 9, 2016
New feature	Updated to support MariaDB version 10.1. For more information, see MariaDB on Amazon RDS (p. 432) .	June 1, 2016
New feature	Updated to support Amazon Aurora cross-region DB clusters that are Read Replicas. For more information, see Replicating Aurora MySQL DB Clusters Across AWS Regions .	June 1, 2016
New feature	Enhanced Monitoring is now available for Oracle DB instances. For more information, see Enhanced Monitoring (p. 256) and Modifying a DB Instance Running the Oracle Database Engine (p. 758) .	May 27, 2016
New feature	Updated to support manual snapshot sharing for Amazon Aurora DB cluster snapshots. For more information, see Sharing a DB Cluster Snapshot .	May 18, 2016
New feature	You can now use the MariaDB Audit Plugin to log database activity on MariaDB and MySQL database instances. For more information, see Options for MariaDB Database Engine (p. 477) and Options for MySQL DB Instances (p. 677) .	April 27, 2016
New feature	In-place, major version upgrades are now available for upgrading from MySQL version 5.6 to version 5.7. For more information, see Upgrading the MySQL DB Engine (p. 618) .	April 26, 2016
New feature	Enhanced Monitoring is now available for Microsoft SQL Server DB instances. For more information, see Enhanced Monitoring (p. 256) .	April 22, 2016
New feature	Added support for PostgreSQL versions 9.5.2, 9.4.7, and 9.3.12. For more information, see Supported PostgreSQL Database Versions (p. 1030) .	April 8, 2016
New feature	Updated to support Oracle database versions 11.2.0.4.v7, 12.1.0.1.v4, and 12.1.0.2.v3 with the January 2016 Oracle Patch Set Updates (PSU). For more information, see Oracle Database Engine Release Notes (p. 921) .	April 1, 2016
New feature	Updated to provide an Amazon Aurora Clusters view in the Amazon RDS console. For more information, see Viewing an Aurora DB Cluster .	April 1, 2016
New feature	Updated to support SQL Server Multi-AZ with mirroring in the Asia Pacific (Seoul) region. For more information, see Multi-AZ Deployments for Microsoft SQL Server (p. 551) .	March 31, 2016

Change	Description	Date Changed
New feature	Updated to support Amazon Aurora Multi-AZ with mirroring in the Asia Pacific (Seoul) region. For more information, see Availability for Amazon Aurora MySQL .	March 31, 2016
New feature	PostgreSQL DB instances have the ability to require connections to use SSL. For more information, see Using SSL with a PostgreSQL DB Instance (p. 1068) .	March 25, 2016
New feature	Enhanced Monitoring is now available for PostgreSQL DB instances. For more information, see Enhanced Monitoring (p. 256) .	March 25, 2016
New feature	Microsoft SQL Server DB instances can now use Windows Authentication for user authentication. For more information, see Using Windows Authentication with a Microsoft SQL Server DB Instance (p. 578) .	March 23, 2016
New feature	Enhanced Monitoring is now available in the Asia Pacific (Seoul) region. For more information, see Enhanced Monitoring (p. 256) .	March 16, 2016
New feature	You can now customize the order in which Aurora Replicas are promoted to primary instance during a failover. For more information, see Fault Tolerance for an Aurora DB Cluster .	March 14, 2016
New feature	Updated to support encryption when migrating to an Aurora DB cluster. For more information, see Migrating Data to an Aurora DB Cluster .	March 2, 2016
New feature	Updated to support local time zone for Aurora DB clusters. For more information, see Local Time Zone for Aurora DB Clusters .	March 1, 2016
New feature	Updated to add support for MySQL version 5.7 for current generation Amazon RDS DB instance classes.	February 22, 2016
New feature	Updated to support db.r3 and db.t2 DB instance classes in the AWS GovCloud (US-West) region.	February 11, 2016
New feature	Updated to support encrypting copies of DB snapshots and sharing encrypted DB snapshots. For more information, see Copying a Snapshot (p. 221) and Sharing a DB Snapshot (p. 230) .	February 11, 2016
New feature	Updated to support Amazon Aurora in the Asia Pacific (Sydney) region. For more information, see Availability for Amazon Aurora MySQL .	February 11, 2016
New feature	Updated to support SSL for Oracle DB Instances. For more information, see Using SSL with an Oracle DB Instance (p. 723) .	February 9, 2016
New feature	Updated to support local time zone for MySQL and MariaDB DB instances. For more information, see Local Time Zone for MySQL DB Instances (p. 595) and Local Time Zone for MariaDB DB Instances (p. 441) .	December 21, 2015

Change	Description	Date Changed
New feature	Updated to support Enhanced Monitoring of OS metrics for MySQL and MariaDB instances and Aurora DB clusters. For more information, see Viewing DB Instance Metrics (p. 254) .	December 18, 2015
New feature	Updated to support Oracle Standard Edition Two with Bring-Your- Own-License licensing. Also added support for Oracle versions 11.2.0.4.v5, 12.1.0.1.v3, and 12.1.0.2.v2. For more information, see Oracle Database Engine Release Notes (p. 921) .	December 14, 2015
New feature	Updated to support db.t2, db.r3, and db.m4 DB instance classes for MySQL version 5.5. For more information, see DB Instance Class (p. 82) .	December 4, 2015
New feature	Updated to support modifying the database port for an existing DB instance.	December 3, 2015
New feature	Updated to support three new extensions for PostgreSQL versions 9.3.10 and 9.4.5 DB instances. For more information, see Supported PostgreSQL Database Versions (p. 1030) .	December 1, 2015
New feature	Updated to support PostgreSQL versions 9.3.10 and 9.4.5 DB instances. For more information, see Supported PostgreSQL Database Versions (p. 1030) .	November 27, 2015
New feature	Updated to support major version upgrades of the database engine for PostgreSQL instances. For more information, see Upgrading the PostgreSQL DB Engine (p. 988) .	November 19, 2015
New feature	Updated to support modifying the public accessibility of an existing DB instance. Updated to support db.m4 standard DB instance classes.	November 11, 2015
New feature	Updated to support manual DB snapshot sharing. For more information, see Sharing a DB Snapshot (p. 230) .	October 28, 2015
New feature	Updated to support Microsoft SQL Server 2014 for the Web, Express, and Standard editions.	October 26, 2015
New feature	Updated to support the MySQL-based MariaDB database engine. For more information, see MariaDB on Amazon RDS (p. 432) .	October 7, 2015
New feature	Updated to support Amazon Aurora in the Asia Pacific (Tokyo) region. For more information, see Availability for Amazon Aurora MySQL .	October 7, 2015
New feature	Updated to support db.t2 burst-capable DB instance classes for all DB engines and the addition of the db.t2.large DB instance class. For more information, see DB Instance Class (p. 82) .	September 25, 2015

Change	Description	Date Changed
New feature	Updated to support Oracle DB instances on R3 and T2 DB instance classes. For more information, see DB Instance Class (p. 82) .	August 5, 2015
New feature	Updated to support PostgreSQL versions 9.4.4 and 9.3.9. For more information, see Supported PostgreSQL Database Versions (p. 1030) .	July 30, 2015
New feature	Microsoft SQL Server Enterprise Edition is now available with the License Included service model. For more information, see Licensing Microsoft SQL Server on Amazon RDS (p. 503) .	July 29, 2015
New feature	Amazon Aurora has officially released. The Amazon Aurora DB engine supports multiple DB instances in a DB cluster. For detailed information, see What Is Amazon Aurora? .	July 27, 2015
New feature	Updated to support copying tags to DB snapshots.	July 20, 2015
New feature	Updated to support Oracle 12c database version "12.1.0.2", including the In-Memory option, Oracle 11g April PSU patches, and improved integration with AWS CloudHSM.	July 20, 2015
New feature	Updated to support increases in storage size for all DB engines and an increase in Provisioned IOPS for SQL Server.	June 18, 2015
New feature	Updated options for reserved DB instances.	June 15, 2015
New feature	Updated to support Oracle version 12c.	April 2, 2015
New feature	Updated to support PostgreSQL versions 9.3.6 and 9.4.1.	March 18, 2015
New feature	Updated to support using Amazon CloudHSM with Oracle DB instances using TDE.	January 8, 2015
New feature	Updated to support encrypting data at rest and new API version 2014-10-31.	January 6, 2015
New feature	Updated to support Oracle version 11.2.0.4.v3 that includes the PSU released in October 2014.	November 20, 2014
New feature	Updated to include the new Amazon DB engine: Aurora. The Amazon Aurora DB engine supports multiple DB instances in a DB cluster. Amazon Aurora is currently in preview release and is subject to change. For detailed information, see What Is Amazon Aurora? .	November 12, 2014
New feature	Updated to support PostgreSQL Read Replicas.	November 10, 2014
New features	Updated to support Oracle 11.2.0.4v2.	October 16, 2014
New API and features	Updated to support the GP2 storage type and new API version 2014-09-01. Updated to support the ability to copy an existing option or parameter group to create a new option or parameter group.	October 7, 2014

Change	Description	Date Changed
New feature	Updated to support InnoDB Cache Warming for DB instances running MySQL version 5.6.19 and later.	September 3, 2014
New feature	Updated to support SSL certificate verification when connecting to MySQL version 5.6, SQL Server, and PostgreSQL database engines.	August 5, 2014
New feature	Updated to support the db.t2 burst-capable DB instance classes.	August 4, 2014
New feature	Updated to support the db.r3 memory-optimized DB instance classes for use with the MySQL (version 5.6), SQL Server, and PostgreSQL database engines.	May 28, 2014
New feature	Updated to support SQL Server Multi-AZ deployments using SQL Server Mirroring.	May 19, 2014
New feature	Updated to support upgrades from MySQL version 5.5 to version 5.6.	April 23, 2014
New feature	Updated to support Oracle 11.2.0.4.	April 23, 2014
New feature	Updated to support Oracle GoldenGate.	April 3, 2014
New feature	Updated to support the M3 DB instance classes.	February 20, 2014
New feature	Updated to support the Oracle Timezone option.	January 13, 2014
New feature	Updated to support replication between Amazon RDS MySQL DB instances in different regions.	November 26, 2013
New feature	Updated to support the PostgreSQL DB engine.	November 14, 2013
New feature	Updated to support SQL Server transparent data encryption (TDE).	November 7, 2013
New API and new feature	Updated to support cross region DB snapshot copies; new API version, 2013-09-09.	October 31, 2013
New features	Updated to support Oracle Statspack.	September 26, 2013
New features	Updated to support using replication to import or export data between instances of MySQL running in Amazon RDS and instances of MySQL running on-premises or on Amazon EC2.	September 5, 2013
New features	Updated to support the db.cr1.8xlarge DB instance class for MySQL 5.6.	September 4, 2013
New feature	Updated to support replication of Read Replicas.	August 28, 2013
New feature	Updated to support parallel Read Replica creation.	July 22, 2013
New feature	Updated to support fine-grained permissions and tagging for all Amazon RDS resources.	July 8, 2013
New feature	Updated to support MySQL 5.6 for new instances, including support for the MySQL 5.6 memcached interface and binary log access.	July 1, 2013

Change	Description	Date Changed
New feature	Updated to support major version upgrades from MySQL 5.1 to MySQL 5.5.	June 20, 2013
New feature	Updated DB parameter groups to allow expressions for parameter values.	June 20, 2013
New API and new feature	Updated to support Read Replica status; new API version, 2013-05-15.	May 23, 2013
New features	Updated to support Oracle Advanced Security features for native network encryption and Oracle Transparent Data Encryption.	April 18, 2013
New features	Updated to support major version upgrades for SQL Server and additional functionality for Provisioned IOPS.	March 13, 2013
New feature	Updated to support VPC By Default for RDS.	March 11, 2013
New API and feature	Updated to support log access; new API version 2013-02-12	March 4, 2013
New feature	Updated to support RDS event notification subscriptions.	February 4, 2013
New API and feature	Updated to support DB instance renaming and the migration of DB security group members in a VPC to a VPC security group.	January 14, 2013
New feature	Updated for AWS GovCloud (US-West) support.	December 17, 2012
New feature	Updated to support m1.medium and m1.xlarge DB Instance classes.	November 6, 2012
New feature	Updated to support Read Replica promotion.	October 11, 2012
New feature	Updated to support SSL in Microsoft SQL Server DB Instances.	October 10, 2012
New feature	Updated to support Oracle micro DB Instances.	September 27, 2012
New feature	Updated to support SQL Server 2012.	September 26, 2012
New API and feature	Updated to support provisioned IOPS. API version 2012-09-17.	September 25, 2012
New features	Updated for SQL Server support for DB Instances in VPC and Oracle support for Data Pump.	September 13, 2012
New feature	Updated for support for SQL Server Agent.	August 22, 2012
New feature	Updated for support for tagging of DB Instances.	August 21, 2012
New features	Updated for support for Oracle APEX and XML DB, Oracle time zones, and Oracle DB Instances in a VPC.	August 16, 2012
New features	Updated for support for SQL Server Database Engine Tuning Advisor and Oracle DB Instances in VPC.	July 18, 2012
New feature	Updated for support for option groups and first option, Oracle Enterprise Manager Database Control.	May 29, 2012

Change	Description	Date Changed
New feature	Updated for support for Read Replicas in Amazon Virtual Private Cloud.	May 17, 2012
New feature	Updated for Microsoft SQL Server support.	May 8, 2012
New features	Updated for support for forced failover, Multi-AZ deployment of Oracle DB Instances, and nondefault character sets for Oracle DB Instances.	May 2, 2012
New feature	Updated for Amazon Virtual Private Cloud (VPC) Support.	February 13, 2012
Updated content	Updated for new Reserved Instance types.	December 19, 2011
New feature	Updated for Oracle engine support.	May 23, 2011
Updated content	Console updates.	May 13, 2011
Updated content	Edited content for shortened backup and maintenance windows.	February 28, 2011
New feature	Added support for MySQL 5.5.	January 31, 2011
New feature	Added support for Read Replicas.	October 4, 2010
New feature	Added support for AWS Identity and Access Management (IAM).	September 2, 2010
New feature	Added DB Engine Version Management.	August 16, 2010
New feature	Added Reserved DB Instances.	August 16, 2010
New Feature	Amazon RDS now supports SSL connections to your DB Instances.	June 28, 2010
New Guide	This is the first release of the Amazon RDS User Guide.	June 7, 2010