Amazon Relational Database Service: User Guide
Copyright © 2014 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is Amazon Relational Database Service?</td>
<td>1</td>
</tr>
<tr>
<td>Amazon RDS Best Practices</td>
<td>6</td>
</tr>
<tr>
<td>Getting Started with Amazon RDS</td>
<td>8</td>
</tr>
<tr>
<td>Amazon RDS Terminology and Concepts</td>
<td>43</td>
</tr>
<tr>
<td>DB Instance</td>
<td>43</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>47</td>
</tr>
<tr>
<td>Amazon RDS Storage</td>
<td>48</td>
</tr>
<tr>
<td>Regions and Availability Zones</td>
<td>50</td>
</tr>
<tr>
<td>High Availability (Multi-AZ)</td>
<td>52</td>
</tr>
<tr>
<td>DB Instance Maintenance</td>
<td>53</td>
</tr>
<tr>
<td>Amazon RDS and the Amazon Virtual Private Cloud Service</td>
<td>54</td>
</tr>
<tr>
<td>DB Instance Backups</td>
<td>55</td>
</tr>
<tr>
<td>DB Instance Replication</td>
<td>57</td>
</tr>
<tr>
<td>AWS Identity and Access Management</td>
<td>59</td>
</tr>
<tr>
<td>Managing Access to Your Amazon RDS Resources and Databases</td>
<td>62</td>
</tr>
<tr>
<td>Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources</td>
<td>63</td>
</tr>
<tr>
<td>Amazon RDS Security Groups</td>
<td>74</td>
</tr>
<tr>
<td>MySQL on Amazon RDS</td>
<td>77</td>
</tr>
<tr>
<td>Creating a DB Instance Running the MySQL Database Engine</td>
<td>82</td>
</tr>
<tr>
<td>Connecting to a DB Instance Running the MySQL Database Engine</td>
<td>91</td>
</tr>
<tr>
<td>Modifying a DB Instance Running the MySQL Database Engine</td>
<td>93</td>
</tr>
<tr>
<td>Importing and Exporting Data From an Amazon RDS MySQL DB Instance</td>
<td>95</td>
</tr>
<tr>
<td>Importing Offline Data From Existing MySQL Databases</td>
<td>98</td>
</tr>
<tr>
<td>Importing Data From a MySQLInstance Running External to Amazon RDS</td>
<td>99</td>
</tr>
<tr>
<td>Importing Data From Any Source to a MySQL DB Instance</td>
<td>102</td>
</tr>
<tr>
<td>Using Replication to Export Amazon RDS MySQL 5.6 Data</td>
<td>105</td>
</tr>
<tr>
<td>Working with Read Replicas</td>
<td>109</td>
</tr>
<tr>
<td>Appendix: Common DBA Tasks for MySQL</td>
<td>118</td>
</tr>
<tr>
<td>Appendix: Options for MySQL DB Engine</td>
<td>123</td>
</tr>
<tr>
<td>Appendix: MySQL Amazon RDS SQL Reference</td>
<td>127</td>
</tr>
<tr>
<td>mysql.rds_set_external_master</td>
<td>127</td>
</tr>
<tr>
<td>mysql.rds_reset_external_master</td>
<td>129</td>
</tr>
<tr>
<td>mysql.rds_start_replication</td>
<td>130</td>
</tr>
<tr>
<td>mysql.rds_stop_replication</td>
<td>130</td>
</tr>
<tr>
<td>mysql_rds_skip_repl_error</td>
<td>131</td>
</tr>
<tr>
<td>mysql.rds_next_master_log</td>
<td>132</td>
</tr>
<tr>
<td>Oracle on Amazon RDS</td>
<td>134</td>
</tr>
<tr>
<td>Appendix: Options for Oracle DB Engine</td>
<td>138</td>
</tr>
<tr>
<td>Appendix: MySQL Amazon RDS SQL Reference</td>
<td>147</td>
</tr>
<tr>
<td>Oracle on Amazon RDS</td>
<td>147</td>
</tr>
<tr>
<td>Appendix: Options for Oracle DB Engine</td>
<td>150</td>
</tr>
<tr>
<td>Appendix: Options for Oracle DB Engine</td>
<td>156</td>
</tr>
<tr>
<td>Appendix: Oracle Character Sets Supported in Amazon RDS</td>
<td>168</td>
</tr>
<tr>
<td>Appendix: Oracle DB Engine Patch Composition</td>
<td>178</td>
</tr>
<tr>
<td>Appendix: Oracle Character Sets Supported in Amazon RDS</td>
<td>192</td>
</tr>
<tr>
<td>Appendix: Oracle DB Engine Patch Composition</td>
<td>194</td>
</tr>
<tr>
<td>Microsoft SQL Server on Amazon RDS</td>
<td>199</td>
</tr>
<tr>
<td>Working with a SQL Server DB Instance</td>
<td>207</td>
</tr>
<tr>
<td>Creating a DB Instance Running the Microsoft SQL Server Database Engine</td>
<td>207</td>
</tr>
<tr>
<td>Connecting to a DB Instance Running the Microsoft SQL Server Database Engine</td>
<td>215</td>
</tr>
<tr>
<td>Modifying a DB Instance Running the Microsoft SQL Server Database Engine</td>
<td>220</td>
</tr>
<tr>
<td>Importing Data into SQL Server on Amazon RDS</td>
<td>222</td>
</tr>
<tr>
<td>Appendix: Common DBA Tasks for Microsoft SQL Server</td>
<td>229</td>
</tr>
<tr>
<td>Appendix: Options for SQL Server DB Engine</td>
<td>236</td>
</tr>
<tr>
<td>PostgreSQL on Amazon RDS</td>
<td>239</td>
</tr>
<tr>
<td>Working with a PostgreSQL DB Instance</td>
<td>243</td>
</tr>
</tbody>
</table>

Amazon Relational Database Service User Guide
What is Amazon Relational Database Service?

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.

Why would 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. So, for example, 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.
- 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.
- You can have automated backups performed when you need them, or create your own backup snapshot. These backups can be used to restore a database, and Amazon RDS's restore process works reliably and efficiently.
- You can get high availability with a primary instance and a synchronous secondary instance that you can failover to when problems occur. You can also use MySQL read replicas to increase read scaling.
- You can use the database products you are already familiar with: MySQL, PostgreSQL, Oracle, and 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 IAM to define users and permissions. You can also help protect your databases by putting them in a virtual private cloud.
To begin learning more:

- If you are new to RDS but you are familiar with other Amazon Web Services, start with an introduction to the Amazon RDS Components (p. 2). This section discusses the key components of Amazon RDS and how they map to those that you currently work with on your local network.
- 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

Amazon RDS Components

**Topics**

- DB Instances (p. 2)
- Regions and Availability Zones (p. 3)
- Security Groups (p. 3)
- DB Parameter Groups (p. 3)
- DB Option Groups (p. 3)

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 Amazon RDS 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, 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 about DB instance classes, see the DB Instance Class section. For pricing information on DB instance classes, go to the Pricing section of the Amazon Relational Database Service (Amazon RDS) product page.

For each DB instance, you can select from 5 GB to 3 TB of associated *storage* capacity. Each DB instance class has minimum and maximum storage requirements for the DB instances that are created from it. 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.

DB instance storage comes in two types, standard and provisioned IOPS. Standard storage is allocated on Amazon EBS volumes and connected to your DB instance. Provisioned IOPS uses optimized EBS volumes and an optimized configuration stack and provides additional, dedicated capacity for EBS I/O. These optimizations enable instances to fully utilize the IOPS provisioned on an EBS volume. For more information on Provisioned IOPS, see Working with Provisioned IOPS Storage (p. 361).

You can run a DB instance on a virtual private cloud using Amazon's 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 Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 353).

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 a list of regions and Availability Zones, see Regions and Availability Zones (p. 50).

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 synchronous standby replica of your DB instance in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to the standby replica to provide data redundancy, failover support, eliminate I/O freezes, and minimize latency spikes during system backups.

Security Groups

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. For more information about security groups, see Amazon RDS Security Groups (p. 74).

DB Parameter Groups

You manage the configuration of a DB engine by using a DB parameter group. A DB parameter group contains engine configuration values that can be applied to one or more DB instances of the same instance type. Amazon RDS applies a default DB parameter group if you don’t specify a DB parameter group when you create a DB instance. The default group contains defaults for the specific database engine and instance class of the DB instance.

DB Option Groups

Some DB engines offer tools that simplify managing your databases and making the best use of your data. Amazon RDS makes such tools available through option groups. Currently, option groups are available only for Oracle DB instances. For more information about individual Oracle options that are available with Amazon RDS, go to Appendix: Options for Oracle DB Engine (p. 156).

Available RDS Interfaces

Topics

- Amazon RDS Console (p. 4)
- Command Line Interface (p. 4)
- Programmatic Interfaces (p. 4)
There are several ways that you can interact with Amazon RDS.

**Amazon RDS Console**

The Amazon RDS console is a simple web-based user interface. From the console, you can perform almost all tasks you need to do from the RDS 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**

Amazon RDS provides a Java-based command line interface that gives you access to much of the functionality that is available in the amazon RDS API. For more information, see the Amazon RDS Command Line Toolkit.

**Programmatic Interfaces**

The following table lists the resources that you can use to access Amazon RDS programmatically.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS SDKs</td>
<td>The AWS SDKs include sample code, libraries, tools, documentation, and templates. To download the AWS SDKs, go to AWS Software Development Kits (SDKs).</td>
</tr>
</tbody>
</table>
| Libraries      | AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of Amazon Relational Database Service's SOAP and Query APIs. These libraries provide basic functions (not included in Amazon Relational Database Service's SOAP and Query APIs), such as request authentication, request retries, and error handling so you can get started more easily. Libraries and resources are available for the following languages:  

  - Java  
  - PHP  
  - Python  
  - Ruby  
  - Windows and .NET  

For libraries and sample code in all languages, go to Sample Code & Libraries. |
| Amazon RDS API | If you prefer, you can code directly to the Amazon RDS API. For more information, see Using the Amazon RDS API (p. 417), and see the Amazon Relational Database Service API Reference. |

**How You Are Charged for Amazon RDS**

When you use Amazon RDS, you pay only for what you use, and there are no minimum or setup fees. You are billed according to the following criteria.

- Instance class – Pricing is based on the class (e.g., micro, small, large, xlarge) of the DB instance consumed.
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 about Amazon CloudWatch, see Viewing DB Instance Metrics (p. 377). For more information on Amazon RDS event notification, see Using Amazon RDS Event Notification (p. 380)

What's Next?

This 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. 8) section.

DB Engine Specific Topics

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

• Oracle on Amazon RDS (p. 134)
• MySQL on Amazon RDS (p. 77)
• Microsoft SQL Server on Amazon RDS (p. 199)
• PostgreSQL on Amazon RDS (p. 239)
Amazon RDS Best Practices

This section summarizes 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. 6)
- Amazon RDS Security Best Practices (p. 7)
- Best Practices for Working with MySQL Storage Engines (p. 7)

Amazon RDS Basic Operational Guidelines

The following are basic operational guidelines 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 setup 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 WriteIOPS.
- On one DB instance, do not create more than 10,000 tables using Provisioned IOPS or 1000 tables using standard storage. Large numbers of tables will significantly increase database recovery time after a failover or database crash. For information about working with MySQL tables, see Working with InnoDB Tablespaces to Improve Crash Recovery Times (p. 119).
- 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 Provisioned IOPS storage, and use a DB instance class that is optimized for Provisioned IOPS.
  - If you are already using Provisioned IOPS storage, provision additional throughput capacity.
- If your client application is caching the DNS data of your DB instances, set a TTL 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.

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, see AWS Identity and Access Management. For information on IAM best practices, see IAM Best Practices.

Best Practices for Working with MySQL Storage Engines

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. While MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for crash recovery and data durability. For example, MyISAM storage engine does not support reliable crash recovery and may prevent Point-In-Time-Restore or snapshot restore from working as intended. This may result in lost or corrupt data when MySQL is restarted after a crash.

However, if you still choose to use MyISAM with Amazon RDS, following these steps outlined in Automated Backups with Unsupported MySQL Storage Engines (p. 57) may be helpful in certain scenarios for snapshot restore functionality.

If you would like 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.
Getting Started with Amazon RDS

This section shows you how to create and connect to a DB instance using Amazon RDS. A newly created DB instance has a firewall that prevents any access to it; therefore, as part of the creation process, you must create a security group that provides access and then associate it with your DB instance. Next, this section shows you how to connect to the DB instance using a client application on your local computer. Finally, you delete the DB instance.

Sign up for Amazon RDS

To use the Amazon Relational Database Service (Amazon RDS), you must first sign up for the service. After you sign up for the service, you can get your user credentials and start using the Amazon RDS service.

**Important**

The DB instance you’re about to launch will be live (and not running in a sandbox). You will incur the standard Amazon RDS usage fees for the instance until you terminate it. The total charges will be minimal if you complete the exercise described here in one sitting and delete your DB instance when you are finished. For more information about Amazon RDS usage rates, go to the Amazon RDS product page.

To use Amazon RDS, you need an AWS account. If you don’t already have one, you’ll be prompted to create one when you sign up for Amazon RDS.

**To sign up for Amazon RDS**

1. Go to [http://aws.amazon.com/rds](http://aws.amazon.com/rds) and click **Sign Up for Amazon RDS Now**.

API Version 2013-09-09

8
2. Follow the on-screen instructions.

Authorize Access: Creating a Security Group

Depending on your AWS account and region, you can create a DB instance on two platforms: EC2-Classic or EC2-VPC. Each platform requires its own security group. The EC2-Classic platform is the original platform used by Amazon RDS and requires a DB security group, and the EC2-VPC platform provides a default VPC (virtual private cloud) where a new DB instance is created, and requires either an EC2 or VPC security group. If you are a new Amazon RDS customer or you are creating DB instances in a region you have not used before, you are likely on the EC2-VPC platform.

When you create an Amazon RDS DB instance on either the EC2-VPC or the EC2-Classic platform, the instance is created with a firewall that does not allow any connections to it. You must create a security group and specify the port, IP addresses, or EC2 instances that can access your DB instance. You then associate that security group with the DB instance when you create the instance. If you do not know the IP address or port you should use when creating the security group and the DB instance, talk with your network administrator to determine the values you should use. For more information about the EC2-VPC and EC2-Classic platforms, and about EC2 security groups, see Amazon EC2 Security Groups. For information on creating a VPC security group, see Security Groups for Your VPC.

Determining Whether You are Using the EC2-VPC or EC2-Classic Platform

You can tell which platform your AWS account in a given region is using by looking at the RDS console home page.

If Supported Platforms indicates VPC, 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 an EC2 or VPC security group. Note that you can create a VPC on the EC2-Classic platform, but one is not created for you by default as it is on the EC2-VPC platform.

Resources
You are using the following Amazon RDS resources in the US West (N. California) region:
- DB Instances (0)
- DB Snapshots (0)
- DB Parameter Groups (2)
- Reserved DB Purchases (0)
- Recent Events (0)
- Supported Platforms: VPC
- Default Network: vpc-7553f1ae1

Create Instance
Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud. You can click the button below to launch a Database (DB) instance.

If Supported Platforms indicates EC2, VPC, 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. Note that you can create a VPC on the EC2-Classic platform, but one is not created for you by default as it is on the EC2-VPC platform.
Creating a Security Group for the **EC2-VPC** and **EC2-Classic** Platforms

Now that you know which platform you are creating your DB instance on, you can create a security group that will allow you access to your DB instance.

**Topics**

- Creating an EC2/VPC Security Group for a DB Instance in a VPC (p. 10)
- Creating a DB Security Group for a DB Instance on the EC2-Classic platform (p. 11)

**Creating an EC2/VPC Security Group for a DB Instance in a VPC**

If you are creating a DB instance on the **EC2-VPC** platform and therefore in a default VPC, you must create either an EC2 or VPC security group that will allow access through your firewall to the DB instance that is in a VPC.

To create an EC2 security group for your DB instance, you first create the security group, then you specify a custom TCP rule that specifies the port you will use for your DB instance and a single IP address or a
range of IP addresses that you will allow to connect to your DB instance. Since these IP addresses will be allowed access, it is important that you grant access only to the correct IP addresses.

**To create an EC2 security group to Use with a DB Instance on the EC2-VPC platform**

1. Sign in to the AWS Management Console and open the Amazon EC2 console.
2. In the navigation pane, click **Security Groups**.
3. Click **Create Security Group**.
4. Specify a name and description for the security group. Select the ID of your VPC for **VPC**, and then click **Yes, Create**.
5. Select the check box for the EC2 security group you just created, then select the **Inbound** tab.

   In the **Create a new rule** drop down list, select **Custom TCP rule**. Enter the port value you will use when you create the DB instance in the **Port range** text box. Enter the IP address you will use to access the DB instance in the **Source** text box.

6. Click **Add Rule**. You can add additional rules by repeating the process of entering a port range and source IP address values.
7. Click **Apply Rule Changes**.

---

**Creating a DB Security Group for a DB Instance on the EC2-Classic platform**

If you are creating a DB instance on the **EC2-Classic** platform, you must create a DB security group that will allow access through your firewall to the DB instance.
To create a DB security group for your DB instance, you first create the DB security group, and then you specify CIDR (Classless Inter-Domain Routing) values for either a single IP address or a range of IP addresses that you will allow to connect to your DB instance. Since these IP addresses will be allowed access, it is important that you grant access only to the correct IP addresses.

To create a DB security group for a DB Instance on the EC2-Classic platform

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click Security Groups in the navigation pane on the left side of the window.
3. Click Create DB Security Group.
4. Type the name and description of the new DB security group in the Name and Description text boxes. Note that the security group name cannot contain spaces and cannot start with a number.

Create DB Security Group

Name: myseogr
Description: Test DB security group

Yes, Create  Cancel
5. Click **Yes, Create**. The DB security group will be created. A newly created DB security group does not provide access to a DB instance by default. You must specify a range of IP addresses or an EC2 security group that can have access to the DB instance.

6. Click the details page icon for the DB security group you just created.

7. In the **Security Group Details** section, select **CIDR/IP** from the **Connection Type** drop-down list, type the CIDR range for the ingress rule you would like to add to this DB security group into the **CIDR** text box, and click **Add**.

   **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, this could be the CIDR IP you would use.

   The status of the ingress rule will be **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 will change to **authorized**. You can then use the DB security group when you create a DB instance in the next section.
Creating a DB Instance and Connecting to a Database on a DB Instance

You can create, or launch, a DB instance using any of the DB engine versions available in the region you've selected. Editions of MySQL, Oracle, PostgreSQL, and Microsoft SQL Server are available in all regions. Creating a DB instance and connecting to a database on a DB instance is slightly different for each of the DB engines; select the DB engine below that you want to create for detailed information on creating the DB instance.

- Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance (p. 14)
- Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance (p. 19)
- Creating and Connecting to a SQL Server DB Instance (p. 25)
- Creating and Connecting to a PostgreSQL DB Instance (p. 34)

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 Amazon RDS 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.

Topics
- Creating a MySQL DB Instance (p. 14)
- Connecting to a Database on a DB Instance Running the MySQL Database Engine (p. 19)

Creating a MySQL DB Instance

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

In this example, you create a DB instance running the MySQL database engine called west2-mysql-instance1, with a db.m1.small DB instance class, 5 GB 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, select the region in which you want to create the DB instance.
3. In the navigation pane, click Instances.
5. On the **Engine Selection** page, click **Select** for the MySQL DB engine.

6. On the **DB Instance 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, click **Continue**.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Model</td>
<td>Select the default, <strong>General-Public-License</strong>, to use the general license agreement for MySQL. MySQL has only one license model.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>Select the default version of MySQL. Note that Amazon RDS supports multiple versions of MySQL in some regions.</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>Select <strong>db.m1.small</strong> to select a configuration that equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.</td>
</tr>
<tr>
<td>Multi-AZ Deployment</td>
<td>Select <strong>No</strong> to create your DB instance in a single availability zone.</td>
</tr>
<tr>
<td>Auto Minor Version Upgrade</td>
<td>Select <strong>Yes</strong> to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.</td>
</tr>
<tr>
<td>Allocated Storage</td>
<td>Type <strong>5</strong> to allocate 5 GB of storage for your database.</td>
</tr>
</tbody>
</table>
For this parameter... | ...Do this:
---|---
Use Provisioned IOPS | Leave the check box unselected. This option turns on Provisioned IOPS (I/O operations per second), a high-performance storage option in Amazon RDS that is optimized for I/O-intensive, transactional (OLTP) database workloads.

DB Instance Identifier | Type a name for the DB instance that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB engine you selected, for example west2-mysql-instance1.

Master User Name | Type a name using alphanumeric characters that you will use as the master user name to log on to your DB instance. This will be the user name you use to logon to your database on the DB instance for the first time.

Master User Password | Type a password that contains from 8 to 16 printable ASCII characters (excluding /,, and @) for your master user password. This will be the password you will use when you use the user name to logon to your database.

---

7. On the **Additional Configuration** 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 click Continue.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Name</td>
<td>Type a name for your database of up to 8 alpha-numeric characters. If you do not provide a name, Amazon RDS will not automatically create a database on the DB instance you are creating.</td>
</tr>
<tr>
<td>Database Port</td>
<td>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.</td>
</tr>
<tr>
<td>Choose a VPC</td>
<td>This setting depends on the platform you are on. If you are creating a DB instance on the EC2-VPC platform, select the default VPC. If you are creating a DB instance on the E2-Classic platform, select Not in VPC. For more information about VPC, see Amazon RDS and the Amazon Virtual Private Cloud Service (p. 54).</td>
</tr>
<tr>
<td>Publicly Accessible</td>
<td>(Only applies if you choose a VPC) Select Yes to give the DB instance a public IP address; otherwise, select No. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.</td>
</tr>
<tr>
<td>Availability Zone</td>
<td>Leave the default of No Preference unless you want to specify a particular Availability Zone.</td>
</tr>
<tr>
<td>Option Group</td>
<td>Select the default value of default:mysql-5-5 since this option group is used with the MySQL version you selected on the previous page.</td>
</tr>
<tr>
<td>Parameter Group</td>
<td>Leave the default value of default.mysql5.5 unless you created your own DB parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 317).</td>
</tr>
<tr>
<td>DB Security Groups</td>
<td>Select the DB security group that you created in a previous step in the Getting Started section.</td>
</tr>
</tbody>
</table>
8. On the **Management Options** page, you can specify backup and maintenance options for your DB instance. Accept the default values, and then click **Continue**.

9. On the **Review** page, review the options for your DB instance. If you need to make any changes, click **Back** to return to the appropriate page, and then make the necessary corrections. When all the settings are as you want them, click **Launch DB Instance**.

10. On the final page of the wizard, click **Close**.
11. On the 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 a database on the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new DB instance to become available.

![RDS Console Screenshot](image)

### 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**

- Type 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 the master password you used for `<password>`.

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

  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: 5.1.32-log MySQL Community Server (GPL)
  Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
  mysql>
  ```

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

The easiest way to create an Oracle DB instance is to use the RDS console. Once you have created the DB instance, you can use standard Oracle client utilities such as SQL Developer to connect to the instance.
In this example, you create a DB instance running the Oracle database engine called west2-oracle1, with a db.m1.small DB instance class, 10 GB of storage, and automated backups enabled with a retention period of one day.

Topics
- Creating a DB Instance Running the Oracle DB Engine (p. 20)
- Connecting to a DB Instance Running the Oracle Database Engine (p. 24)

Creating a DB Instance Running the Oracle DB Engine

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 Amazon RDS console, select the region in which you want to create the DB instance.
3. In the navigation pane, click DB Instances.
4. Click Launch DB Instance to start the Launch DB Instance Wizard.

The wizard opens on the Engine Selection page.

5. In the Launch DB Instance Wizard window, click the Select button for the Oracle DB engine.
6. On the DB Instance Details page, specify your DB instance information. The following table shows settings for an example DB instance. Click Continue when you are finished.

API Version 2013-09-09

20
<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Model</td>
<td>Select <em>Bring Your Own License</em>, to provide your own license for using Oracle. Some regions support additional licensing options for Oracle.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>Select the default version of Oracle.</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>Select <code>db.m1.small</code> to select a configuration that equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.</td>
</tr>
<tr>
<td>Multi-AZ Deployment</td>
<td>Select <em>No</em> to create your DB instance in a single availability zone.</td>
</tr>
<tr>
<td>Auto Minor Version Upgrade</td>
<td>Select <em>Yes</em> to enable your DB instance to receive minor DB Engine version upgrades automatically when they become available.</td>
</tr>
<tr>
<td>Allocated Storage</td>
<td>Type <strong>10</strong> to allocate 10 GB 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.</td>
</tr>
<tr>
<td>Use Provisioned IOPS</td>
<td>Leave the check box unselected. This option turns on Provisioned IOPS (I/O operations per second), a high-performance storage option in RDS that is optimized for I/O-intensive, transactional (OLTP) database workloads.</td>
</tr>
<tr>
<td>DB Instance Identifier</td>
<td>Type a name for the DB Instance that is unique for your account in the region you selected. You may choose to add some intelligence to the name such as including the region and DB engine you selected, for example <code>oracle-unstance1</code>.</td>
</tr>
<tr>
<td>Master User Name</td>
<td>Type a name that you will 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 the &quot;DBA&quot; role.</td>
</tr>
<tr>
<td>Master User Password</td>
<td>Type a password that contains from 8 to 30 printable ASCII characters (excluding /*, and @) for your master user password.</td>
</tr>
</tbody>
</table>
7. On the **Additional Configuration** page, provide additional information that RDS needs to launch the Oracle DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click Continue.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database Name</strong></td>
<td>Type a name for your database that begins with a letter and contains up to 8 alpha-numeric characters. If you do not provide a name, Amazon RDS will not create a database on the DB instance you are creating.</td>
</tr>
<tr>
<td><strong>Database Port</strong></td>
<td>Leave the default value of 1521 unless you have a specific port you want to access the database through. Oracle installations default to port 1521.</td>
</tr>
<tr>
<td><strong>Choose a VPC</strong></td>
<td>This setting depends on the platform you are on. If you are creating a DB instance on the EC2-VPC platform, select the default VPC. If you are creating a DB instance on the E2-Classic platform, select <strong>Not in VPC</strong>. For more information about VPC, see [Amazon RDS and the Amazon Virtual Private Cloud Service](p. 54).</td>
</tr>
<tr>
<td><strong>Publicly Accessible</strong></td>
<td>(Only applies if you choose a VPC) Select <strong>Yes</strong> to give the DB instance a public IP address; otherwise, select <strong>No</strong>. For more information about hiding DB instances from public access, see [Hiding a DB instance in a VPC from the Internet](p. 54).</td>
</tr>
<tr>
<td><strong>Availability Zone</strong></td>
<td>Leave the default of <strong>No Preference</strong>.</td>
</tr>
<tr>
<td><strong>Character Set Name</strong></td>
<td>Select the default value of <strong>AL32UTF8</strong> for the Unicode 5.0 UTF-8 Universal character set. Note that you cannot change the character set after the DB instance is created.</td>
</tr>
</tbody>
</table>
For this parameter... | ...Do this:
--- | ---
Option Group | Select the default value of default:oracle-ee-11-2.
DB Parameter Group | Leave the default value of default.oracle-ee-11.2.
DB Security Groups | Select the DB security group you created in the previous section.

8. On the **Management Options** page, you can specify backup and maintenance options for your DB instance. For this example, accept the default values, and then click **Continue**.
When all the settings are as you want them, click **Continue**.

9. On the **Review** page, review the options for your DB instance. If you need to make any changes, click **Back** to return to the appropriate page, and then make the necessary corrections. When all the settings are as you want them, click **Launch DB Instance**.

10. On the final page of the wizard, click **Close**.

11. On the 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.

---

### Connecting to a DB Instance Running the Oracle Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. In this example, you connect to a DB instance running the Oracle database engine using the Oracle command line tools. For more information on using Oracle, go to the Oracle website.

This example uses the Oracle *sqlplus* command line utility. This utility is part of the Oracle software distribution. To download a stand-alone version of this utility, go to the SQL*Plus User’s Guide and Reference.

1. Open the RDS console, then select **Instances** in the left column to display a list of your DB instances.
2. In the row for your Oracle DB instance, select the arrow to display the summary information for the instance.
3. The **Endpoint** field contains part of the connection information for your DB instance. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, the part following the colon is the port.
4. Type the following command on one line at a command prompt to connect to a DB instance using the `sqlplus` utility. The value for `Host` will be the DNS name for your DB instance, the value for `Port` will be the port you assigned the DB instance, and the value for the Oracle SID will be the name of the DB instance’s database that you specified when you created the DB instance, not the name of the DB instance.

```sql
PROMPT>sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<endpoint>)(PORT=<port number>))(CONNECT_DATA=(SID=<database name>)))'
```

You will see output similar to the following.

```
SQL>
```

**Creating and Connecting to a SQL Server 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 Server utilities to connect to the DB instance such as the Microsoft SQL Server Management Studio utility.

**Topics**
- Creating a SQL Server DB Instance (p. 26)
Creating a SQL Server DB Instance

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 top right corner of the Amazon RDS console, select the region in which you want to create the DB instance.
3. In the navigation pane, click Instances.
4. Click Launch DB Instance to start the Launch DB Instance Wizard.

The wizard opens on the Engine Selection page.

5. In the Launch DB Instance Wizard window, click the Select button for the SQL Server DB engine you want to use.
6. On the DB Instance Details page, specify your DB instance information. The following table shows settings for an example DB instance using SQL Server Standard Edition. Click Continue when you are finished.
<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>License Model</strong></td>
<td>Select, <em>License Included</em>, to use the general license agreement for Microsoft SQL Server.</td>
</tr>
<tr>
<td><strong>DB Engine Version</strong></td>
<td>Select the default version of SQL Server.</td>
</tr>
<tr>
<td><strong>DB Instance Class</strong></td>
<td>Select <code>db.m1.small</code> to select a configuration that equates to 1.7 GB 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 <a href="#">DB Instance Class</a> (p. 47).</td>
</tr>
<tr>
<td><strong>Multi-AZ Deployment</strong></td>
<td>This feature is only available for Oracle and MySQL DB instances.</td>
</tr>
<tr>
<td><strong>Auto Minor Version Upgrade</strong></td>
<td>Select <em>Yes</em> to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.</td>
</tr>
<tr>
<td><strong>Allocated Storage</strong></td>
<td>Type <code>200</code> to allocate 200 GB 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 <a href="#">Amazon Relational Database Service Features</a>.</td>
</tr>
<tr>
<td><strong>Use Provisioned IOPS</strong></td>
<td>Leave the check box unselected. This option turns on Provisioned IOPS (I/O operations per second), a high-performance storage option in RDS that is optimized for I/O-intensive, transactional (OLTP) database workloads. For more information about high performance storage, see <a href="#">Working with Provisioned IOPS Storage</a> (p. 361).</td>
</tr>
<tr>
<td><strong>DB Instance Identifier</strong></td>
<td>Type a name for the DB instance of 15 alphanumeric characters or less that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB Engine you selected, such as <code>sqlsv-instance1</code>.</td>
</tr>
<tr>
<td><strong>Master User Name</strong></td>
<td>Type a name that you will use as the master username to log on to your DB Instance with all database privileges. The master username is a SQL Server Authentication login that is a member of the processadmin, public, and setupadmin fixed server roles.</td>
</tr>
<tr>
<td><strong>Master User Password</strong></td>
<td>Type a password that contains from 8 to 128 printable ASCII characters (excluding /,&quot;, and @) for your master user password.</td>
</tr>
</tbody>
</table>
7. On the **Additional Configuration** page, provide additional information that RDS needs to launch the SQL Server DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click Continue.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database Port</strong></td>
<td>Leave the default value of <strong>1433</strong> unless you have a specific port you want to access the database through. SQL Server installations default to port 1433.</td>
</tr>
<tr>
<td><strong>Availability Zone</strong></td>
<td>Leave the default of <strong>No Preference</strong> unless you want to specify a particular Availability Zone.</td>
</tr>
<tr>
<td><strong>Choose a VPC</strong></td>
<td>This setting depends on the platform you are on. If you are creating a DB instance on the EC2-VPC platform, select the default VPC. If you are creating a DB instance on the E2-Classic platform, select <strong>Not in VPC</strong>. For more information about VPC, see Amazon RDS and the Amazon Virtual Private Cloud Service (p. 54).</td>
</tr>
<tr>
<td><strong>Publicly Accessible</strong></td>
<td>(Only applies if you choose a VPC) Select <strong>Yes</strong> to give the DB instance a public IP address; otherwise, select <strong>No</strong>. For more information about hiding DB instances from public access, see <strong>Hiding a DB instance in a VPC from the Internet</strong>.</td>
</tr>
</tbody>
</table>
8. On the **Management Options** page, you can specify backup and maintenance options for your DB instance. For this example, accept the default values, and then click **Continue**. Note that setting the **Backup Retention Period** to zero disables automatic backups. For more information about the maintenance window, see *Adjusting the Preferred Maintenance Window* (p. 373). For more information on backups and the backup retention period, see *DB Instance Backups* (p. 55).

When all the settings are as you want them, click **Continue**.
9. On the Review page, review the options for your DB instance. If you need to make any changes, click Back to return to the appropriate page, and then make the necessary corrections. When all the settings are as you want them, click Launch DB Instance.

10. On the final page of the wizard, click Close.

11. On the 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.

Connecting to a SQL Server DB Instance Using Microsoft SQL Server Management Studio

This example uses the Microsoft SQL Server Management Studio utility. This utility is part of the Microsoft SQL Server software distribution. To download a stand-alone version of this utility, go to the Microsoft Download Center – Microsoft SQL Server Management Studio Express.

To connect to a DB Instance using Microsoft SQL Server Management Studio
1. Find the DNS name and port for your DB Instance.
   a. Open the RDS console, then select **Instances** in the left column to display a list of your DB instances.
   b. In the row for your Oracle DB instance, select the arrow to display the summary information for the instance.
   c. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, the part following the colon is the port.

2. Run Microsoft SQL Server Management Studio.
3. The **Connect to Server** dialog box appears.

4. In the **Server type**: drop-down list box, select **Database Engine**.
5. In the **Server name**: text field, enter or paste the DNS name of the DB Instance running the Microsoft SQL Server database engine, followed by a comma and then the port number of the DB Instance. For example, the **Server name** could be: `sqlsv-instance1.cg034hpkmmjt.us-east-1.rds.amazonaws.com,1433`.
6. From the **Authentication** drop-down list box, select **SQL Server Authentication**.
7. Enter the master user name for the DB Instance in the **Login**: text box.
8. Enter the password for the master user in the **Password**: text box.
9. Click the Connect button.

After a few moments, Microsoft SQL Server Management Studio should be connected to your DB Instance.

10. Click the New Query button at the top left of the SQL Server Management Studio window.

A new SQL Query window will open.

11. Type the following SQL query:

```
select @@VERSION
```

12. Click the ! Execute button on the SQL Enterprise Manager toolbar to run the query.

You should see a version string returned from your Microsoft SQL Server DB Instance displayed in the output window.
Troubleshooting a Connection to a DB Instance Running SQL Server

There are two common causes for problems if you are having problems connecting to a SQL Server DB instance: the access rules enforced by your firewall and the IP addresses you authorized to access the DB instance in the DB security group (or VPC security group if your DB instance is inside a VPC). If you used Microsoft SQL Server Management Studio and you followed the settings specified in the steps above and you are unable to connect, the problem is most likely the egress or ingress rules on your firewall. If you cannot send out or receive communications over the port you specified when you created the DB instance, you will not be able to connect to the DB instance. Check with your network administrator to determine if the port you specified for your DB instance is allowed to be used for inbound and outbound communication.

Here are a few things to check if you know that you can send and receive communications through your firewall for the port you specified when you created the DB instance.

- **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) could be: `sqlsvr-pdz.c6c8mdfntzgv0.us-west-2.rds.amazonaws.com,1433`.

- **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 often caused by the user name or password being incorrect.
Creating and Connecting to 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 GB of storage, and automated backups enabled with a retention period of one day.

Topics
• Creating a PostgreSQL DB Instance (p. 34)
• Connecting to a PostgreSQL DB Instance (p. 39)

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, select the region in which you want to create the DB instance.
3. In the navigation pane, click DB Instances.
4. Click Launch DB Instance to start the Launch DB Instance Wizard.

   The wizard opens on the Engine Selection page.
5. On the **Engine Selection** page, click the **Select** button for the PostgreSQL DB engine.

6. Next, the **Production?** page asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option will be preselected in the following step. Click **Next Step** when you are finished.

7. On the **DB Instance Details** page, specify your DB instance information. Click **Next Step** when you are finished.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Model</td>
<td>PostgreSQL has only one license model. Select the default, <strong>Postgresql License</strong>, to use the general license agreement for PostgreSQL.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>Select the default version of PostgreSQL.</td>
</tr>
<tr>
<td><strong>For this parameter...</strong></td>
<td><strong>Do this:</strong></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>DB Instance Class</strong></td>
<td>Select <code>db.m1.small</code> to select a configuration that equates to 1.7 GB 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 <a href="#">DB Instance Class</a>.</td>
</tr>
<tr>
<td><strong>Multi-AZ Deployment</strong></td>
<td>Select <strong>No</strong> to create your DB instance in a single availability zone. For more information about multiple Availability Zones, see <a href="#">Regions and Availability Zones</a>.</td>
</tr>
<tr>
<td><strong>Auto Minor Version Upgrade</strong></td>
<td>Select <strong>Yes</strong> to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.</td>
</tr>
<tr>
<td><strong>Allocated Storage</strong></td>
<td>Type <strong>10</strong> to allocate 10 GB 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 <a href="#">Amazon Relational Database Service Features</a>.</td>
</tr>
<tr>
<td><strong>Use Provisioned IOPS</strong></td>
<td>Leave the check box unselected. This option turns on Provisioned IOPS (I/O operations per second), a high-performance storage option in Amazon RDS that is optimized for I/O-intensive, transactional (OLTP) database workloads. For more information about high performance storage, see <a href="#">Working with Provisioned IOPS Storage</a>.</td>
</tr>
<tr>
<td><strong>DB Instance Identifier</strong></td>
<td>Type a name for the DB instance that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB engine you selected, for example <code>west2-postgres1</code>.</td>
</tr>
<tr>
<td><strong>Master Username</strong></td>
<td>Type a name using alphanumeric characters that you will 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 <a href="#">Things You Should Know About PostgreSQL on Amazon RDS</a>.</td>
</tr>
<tr>
<td><strong>Master Password</strong></td>
<td>Type a password that contains from 8 to 128 printable ASCII characters (excluding <code>/</code>, <code>,</code>, and <code>@</code>) for your master user password.</td>
</tr>
</tbody>
</table>
8. On the **Additional Config** 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 click Next Step.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Name</td>
<td>Type a name for your database of up to 63 alpha-numeric characters. If you do not provide a name, no default database on the DB instance is created.</td>
</tr>
<tr>
<td>Database Port</td>
<td>Specify a port you want to use to access the database. PostgreSQL installations default to port 5432.</td>
</tr>
<tr>
<td>Choose a VPC</td>
<td>This setting depends on the platform you are on. If you are creating a DB instance on the EC2-VPC platform, select the default VPC. If you are creating a DB instance on the E2-Classic platform, select <strong>Not in VPC</strong>. For more information about VPC, see Amazon RDS and the Amazon Virtual Private Cloud Service (p. 54).</td>
</tr>
<tr>
<td>Availability Zone</td>
<td>Leave the default of <strong>No Preference</strong>. For more information about Availability Zones, see Regions and Availability Zones (p. 50).</td>
</tr>
<tr>
<td>Option Group</td>
<td>Option groups are currently not used with PostgreSQL DB instances. For more information about option groups, see Working with Option Groups (p. 306).</td>
</tr>
<tr>
<td>Parameter Group</td>
<td>Select the default parameter group. Each PostgreSQL version has a default parameter group you can use, or you can create your own parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 317).</td>
</tr>
</tbody>
</table>
9. On the **Management Options** page, you can specify backup and maintenance options for your DB instance. Click **Next Step** when the settings are as you want them. For more information about the maintenance window, see *Adjusting the Preferred Maintenance Window* (p. 373). For more information on backups and the backup retention period, see *DB Instance Backups* (p. 55).

10. On the **Review** page, review the options for your DB instance. If you need to make any changes, click **Previous** to return to the appropriate page, and then make the necessary corrections. When all the settings are as you want them, click **Launch DB Instance**.

11. On the final page of the wizard, click **Close**.

12. 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.

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. Select 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 text 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 text box. Enter the user name and user password you entered when you created the DB instance into the Username and Password text boxes, respectively.
4. Click **OK**.

5. In the **Object browser**, expand the **Server Groups**. Select the Server (the DB instance) you created, and then select the database name.
6. Click the plugin icon and click **PSQL Console**. The `psql` command window opens for the default database you created.

![PSQL Console](image)

7. Use the command window to enter SQL or `psql` commands. Type `\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.c6c8mwfdgw0.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 will allow you to access the instance. For more information about Amazon RDS security groups, see [Amazon RDS Security Groups](#).

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 DB Instances list, select the check box next to the DB instance you wish to delete.
3. Click Instance Actions, and then select Delete from the dropdown menu.
4. Select No in the Create final Snapshot? drop-down list box.
5. Click Yes, Delete.
This chapter presents Amazon Relational Database Service (RDS) terminology and concepts. Many of the concepts introduced in this chapter are explored in greater depth in later sections.

**DB Instance**

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 stand-alone database instance. DB instances are simple to create and modify with the Amazon RDS command line tools, APIs, or the AWS Management Console.

*Note*

Amazon RDS supports access from 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 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 API Version 2013-09-09.
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 commands. The DB instance identifier must be unique for that customer in an AWS region.

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

When creating a DB instance, some DB engine types require that a database name be specified. An Amazon DB instance can host multiple databases. The database name value depends on the DB engine type:

- For the MySQL database engine, the database name is the name of a database hosted in your Amazon 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 Amazon 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 RDS command line tools, APIs, or the AWS Management Console. You can also change the master user password and manage users using standard SQL commands.

### DB Instance Status

The status of a DB instance indicates the health of the instance. You can view the status of a DB instance by using the RDS console, the CLI command `rds-describe-db-instances`, or the API action `DescribeDBInstances`.

<table>
<thead>
<tr>
<th>DB Instance Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>available</td>
<td>The instance is healthy and available.</td>
</tr>
<tr>
<td>backing-up</td>
<td>The instance is currently being backed up.</td>
</tr>
<tr>
<td>creating</td>
<td>The instance is being created. The instance is inaccessible while it is being created.</td>
</tr>
<tr>
<td>deleting</td>
<td>The instance is being deleted.</td>
</tr>
<tr>
<td>failed</td>
<td>The instance has failed and Amazon RDS was unable to recover it. Perform a point-in-time restore to the latest restorable time of the instance to recover the data.</td>
</tr>
<tr>
<td>incompatible-network</td>
<td>Amazon RDS is attempting to perform a recovery action on an instance but is unable to do so because the VPC is in a state that is preventing the action from being completed. This status can occur if, for example, all available IP addresses in a subnet were in use and Amazon RDS was unable to get an IP address for the DB instance.</td>
</tr>
<tr>
<td>DB Instance Status</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>incompatible-option-group</td>
<td>Amazon RDS attempted to apply an option group change but was unable to do so, and Amazon RDS was unable to roll back to the previous option group state. Consult the Recent Events list for the DB instance for more information. This status can occur if, for example, the option group contains an option such as TDE and the DB instance does not contain encrypted information.</td>
</tr>
<tr>
<td>incompatible-parameters</td>
<td>Amazon RDS was unable to start up the DB instance because the parameters specified in the instance’s DB parameter group were not compatible. Revert the parameter changes or make them compatible with the instance to regain access to your instance. Consult the Recent Events list for the DB instance for more information about the incompatible parameters.</td>
</tr>
<tr>
<td>incompatible-restore</td>
<td>Amazon RDS is unable to do a point-in-time restore. Common causes for this status include using temp tables or using MyISAM tables.</td>
</tr>
<tr>
<td>modifying</td>
<td>The instance is being modified because of a customer request to modify the instance.</td>
</tr>
<tr>
<td>rebooting</td>
<td>The instance is being rebooted because of a customer request or an Amazon RDS process that requires the rebooting of the instance.</td>
</tr>
<tr>
<td>renaming</td>
<td>The instance is being renamed because of a customer request to rename it.</td>
</tr>
<tr>
<td>resetting-master-credentials</td>
<td>The master credentials for the instance are being reset because of a customer request to reset them.</td>
</tr>
<tr>
<td>storage-full</td>
<td>The instance has reached its storage capacity allocation. This is a critical status and should be remedied immediately; you should scale up your storage by modifying the DB instance. Set CloudWatch alarms to warn you when storage space is getting low so you don’t run into this situation.</td>
</tr>
</tbody>
</table>

**Modifying a DB Instance and Using the Apply Immediately Parameter**

Modifications to a DB instance can be applied immediately or during the next maintenance window, and some parameter changes can result in an outage because the instance must be rebooted for the change to take effect. When you modify a DB instance, you have the option of applying the changes immediately by selecting the **Apply Immediately** option in the RDS console or setting the `ApplyImmediately` parameter to `true` using the CLI or RDS API.

The following table shows when a change is applied when you modify a parameter for a DB instance, and the impact of selecting the **Apply Immediately** option in the RDS console or setting the `ApplyImmediately` parameter to `true` has on that change.

**Note**

Some parameter changes cause an outage to occur when the DB instance is rebooted. Review the impact before changing parameters.
<table>
<thead>
<tr>
<th>Parameter for a DB instance</th>
<th>If Apply Immediately is true</th>
<th>If Apply Immediately is false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup retention period</td>
<td>Change is applied immediately. An immediate outage will occur if you change from 0 to a non-zero value or from a non-zero value to 0.</td>
<td>If you change the parameter from one non-zero value to another non-zero value, the change is asynchronously applied as soon as possible. In all other cases, the change is applied during the next maintenance window. An outage will occur if you change this parameter from 0 to a non-zero value or from a non-zero value to 0.</td>
</tr>
<tr>
<td>Automatically upgrade minor versions (applies only if you opted in to auto-upgrades when you created the DB instance)</td>
<td>No difference in when the change is applied. Change is asynchronously applied as soon as possible. An outage will occur if a newer minor version is available, and Amazon RDS has enabled auto patching for that engine version.</td>
<td>Change is asynchronously applied as soon as possible. An outage will occur if this parameter is set to true during the maintenance window, and a newer minor version is available, and RDS has enabled auto patching for that engine version.</td>
</tr>
<tr>
<td>Instance class</td>
<td>Change is applied immediately and an immediate outage will occur.</td>
<td>Change is applied during the next maintenance window. Changing this parameter causes an outage to occur.</td>
</tr>
<tr>
<td>Parameter group name</td>
<td>Change is applied immediately.</td>
<td>Change is applied during the next maintenance window.</td>
</tr>
<tr>
<td>Security group name</td>
<td>No difference in when the change is applied. Change is asynchronously applied as soon as possible.</td>
<td>Change is asynchronously applied as soon as possible.</td>
</tr>
<tr>
<td>Master password</td>
<td>No difference in when the change is applied. Change is asynchronously applied as soon as possible.</td>
<td>Change is asynchronously applied as soon as possible.</td>
</tr>
<tr>
<td>Provisioned IOPS</td>
<td>Change is applied immediately.</td>
<td>Change is applied during the next maintenance window.</td>
</tr>
<tr>
<td>Multi-AZ</td>
<td>Change is applied immediately.</td>
<td>Change is applied during the next maintenance window.</td>
</tr>
<tr>
<td>Option group</td>
<td>Change is applied immediately.</td>
<td>Change is applied during the next maintenance window. If the parameter change results in an option group that enables OEM, this change can cause a brief (sub-second) period during which new connections are rejected but existing connections are not interrupted.</td>
</tr>
<tr>
<td>Allocated storage</td>
<td>Change is made immediately.</td>
<td>Change is applied during the next maintenance window.</td>
</tr>
</tbody>
</table>
DB Instance Class

The computation and memory capacity of a DB instance is determined by its DB instance class. You can change the CPU and memory available to a DB instance by changing its DB instance class. For pricing information on DB instance classes, go to the Amazon RDS details page.

On the smaller end of the DB instance class spectrum, the db.t1.micro DB instance class is sufficient for testing but should not be used for production applications. A db.t1.micro instance using Oracle is a limited test configuration. We recommend you use db.t1.micro instances with Oracle to test setup and connectivity only; the system resources for a db.t1.micro instance do not meet the recommended configuration for Oracle. No Oracle options are supported on a db.t1.micro instance.

The m3 instance classes are second generation instances that provide more computing capacity than the first generation m1 instance classes at a lower price. On the upper end of the DB instance class spectrum, the db.cr1.8xlarge DB instance class (available for MySQL 5.6 and PostgreSQL) can be used with host applications that require a high-memory instance, such as social media or gaming. A db.cr1.8xlarge instance can access 32 Hyperthreaded cores (88 ECUs) and expanded network bandwidth. This DB instance class, when used with MySQL 5.6 or PostgreSQL and Provisioned IOPS, can realize up to 20,000 IOPS for MySQL and 25,000 IOPS for PostgreSQL.

The following table describes the instance classes that are available. One elastic compute unit (ECU) provides CPU capacity equivalent to a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. All DB instances are 64-bit. For more information on how DB instance class and available network bandwidth work together, see Things You Should Know About Provisioned IOPS Storage (p. 361).

<table>
<thead>
<tr>
<th>Instance Class</th>
<th>vCPU</th>
<th>ECU</th>
<th>Memory (GiB)</th>
<th>EBS Optimized</th>
<th>Network Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Micro Instances</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>db.t1.micro</td>
<td>1</td>
<td>1</td>
<td>.615</td>
<td>No</td>
<td>Very Low</td>
</tr>
<tr>
<td><strong>Standard - Second Generation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>db.m3.medium</td>
<td>1</td>
<td>3</td>
<td>3.75</td>
<td>No</td>
<td>Moderate</td>
</tr>
<tr>
<td>Instance Class</td>
<td>vCPU</td>
<td>ECU</td>
<td>Memory (GiB)</td>
<td>EBS Optimized</td>
<td>Network Performance</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
<td>-------</td>
<td>--------------</td>
<td>---------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>db.m3.large</td>
<td>2</td>
<td>6.5</td>
<td>7.5</td>
<td>No</td>
<td>Moderate</td>
</tr>
<tr>
<td>db.m3.xlarge</td>
<td>4</td>
<td>13</td>
<td>15</td>
<td>500 Mbps</td>
<td>High</td>
</tr>
<tr>
<td>db.m3.2xlarge</td>
<td>8</td>
<td>26</td>
<td>30</td>
<td>1000 Mbps</td>
<td>High</td>
</tr>
<tr>
<td><strong>Standard - First</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>db.m1.small</td>
<td>1</td>
<td>1</td>
<td>1.7</td>
<td>No</td>
<td>Low</td>
</tr>
<tr>
<td>db.m1.medium</td>
<td>1</td>
<td>2</td>
<td>3.75</td>
<td>No</td>
<td>Moderate</td>
</tr>
<tr>
<td>db.m1.large</td>
<td>2</td>
<td>4</td>
<td>7.5</td>
<td>500 Mbps</td>
<td>Moderate</td>
</tr>
<tr>
<td>db.m1.xlarge</td>
<td>4</td>
<td>8</td>
<td>15</td>
<td>1000 Mbps</td>
<td>High</td>
</tr>
<tr>
<td><strong>Memory Optimized</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>db.m2.xlarge</td>
<td>2</td>
<td>6.5</td>
<td>17.1</td>
<td>No</td>
<td>Moderate</td>
</tr>
<tr>
<td>db.m2.2xlarge</td>
<td>4</td>
<td>13</td>
<td>34.2</td>
<td>500 Mbps</td>
<td>Moderate</td>
</tr>
<tr>
<td>db.m2.4xlarge</td>
<td>8</td>
<td>26</td>
<td>68.4</td>
<td>1000 Mbps</td>
<td>High</td>
</tr>
<tr>
<td>db.cr1.8xlarge</td>
<td>32</td>
<td>88</td>
<td>244</td>
<td>No</td>
<td>10 GiB</td>
</tr>
</tbody>
</table>

**Note**

The db.cr1.8xlarge instance class is only available for MySQL 5.6 and PostgreSQL DB instances and is available in the US East (Northern Virginia), US West (Oregon), EU (Ireland), and Asia Pacific (Tokyo) regions.

**Amazon RDS Storage**

Data storage in Amazon RDS is specified by providing a storage size (GB) and optionally selecting Provisioned IOPS when you create a new DB instance. Specifying just the storage size allocates standard storage. Standard storage is not reserved for the DB instance, so performance can vary greatly depending on the demands placed on shared resources by other customers. This type of storage has been available since Amazon RDS initially launched and was the only type of storage available for Amazon RDS until Provisioned IOPS storage was launched in September 2012.

When estimating your storage needs, take into consideration that Amazon RDS allocates a minimum amount of storage for file system structures. This reserved space can be up to 1% of the allocated storage for a DB instance, though in most cases the reserved space is far less.

You can allocate Provisioned IOPS storage for a DB instance by specifying both a storage size and selecting Provisioned IOPS. Provisioned IOPS differs from standard storage in that the specified IO capacity is reserved for the DB instance. By reserving IO capacity for your instance, Amazon RDS ensures that disk resources are available when you need them independent of other customer activity.

You can change from standard storage to Provisioned IOPS storage, or from Provisioned IOPS to standard storage, as well as increase storage, with little to no downtime. Note that you cannot reduce the amount of storage, either standard or Provisioned IOPS, once it has been allocated. The only way to reduce the amount of storage allocated to a DB instance is to dump the data out of the DB instance, create a new DB instance with less storage space, and then load the data into the new DB instance.
Performance Metrics

Amazon RDS provides two types of metrics that you can use to determine performance: disk metrics and database metrics.

Disk Metrics

- **IOPS** – the number of IO operations completed per second. This metric is reported as the average IOPS for a given time interval. Amazon RDS reports read and write IOPS separately on one 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 IO 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 one 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 per second 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 one minute intervals using units of megabytes per second (MB/s). Typical values for throughput range from zero to the IO channel's maximum bandwidth.

- **Queue Depth** – the number of IO requests in the queue waiting to be serviced. These are IO requests that have been submitted by the application but have not been sent to the device because the device is busy servicing other IO 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 one minute intervals. Typical values for queue depth range from zero to several hundred.

Database Metrics

- **Commit Latency** – the elapsed time from submitting a commit request to receiving an acknowledgment. This metric is closely associated with disk metric write latency. Lower disk write latency can result in lower commit latency. CloudWatch metrics do not report this value.

- **Transaction Rate** – the number of transactions completed in a given time interval, typically expressed as TPM (Transactions per Minute) or TPS (Transactions per Second). Another commonly used term for transaction rate is *database throughput*, which should not be confused with the disk metric called throughput. The two metrics are not necessarily related; a database can have a high transaction rate and have little to no disk throughput if, for example, the workload consists of cached reads. CloudWatch metrics do not report this value.

Facts About Amazon RDS Storage

The following points are important facts you should know about Amazon RDS storage:

- The current maximum channel bandwidth available is 1000 megabits per second (Mbps) full duplex. In terms of the read and write throughput metrics, this equates to about 105 megabytes per second (MB/s) in each direction. A perfectly balanced workload of 50% reads and 50% writes may attain a maximum combined throughput of 210 MB/s. Note that this is channel throughput, which includes protocol overhead, so the actual data throughput may be less.

- Provisioned IOPS works with an IO request size of 16 KB. An IO request smaller than 16 KB is handled as one IO; for example, 1000 8 KB IO requests are treated the same as 1000 16 KB requests. IO requests larger than 16 KB consume more than one IO request; Provisioned IOPS consumption is a linear function of IO request size above 16 KB. For example, a 24 KB IO request consumes 1.5 IO requests of storage capacity; a 32 KB IO request consumes 2 IO requests, etc. For more information about Provisioned IOPS, see Working with Provisioned IOPS Storage (p. 361)
Note that IO size does not affect the IOPS values reported by the metrics, which are based solely on the number of IOs over time. This means that it is possible to consume all of the IOPS provisioned with fewer IOs than specified if the IO sizes are larger than 16 KB. For example, a system provisioned for 5,000 IOPS can attain a maximum of 2,500 IOPS with 32 KB IO or 1,250 IOPS with 64 KB IO.

Note that standard storage does not provision IO capacity, so all IO sizes are counted as a single IO.

- The first time a DB instance is started and accesses an area of disk for the first time, the process can take longer than all subsequent accesses to the same disk area. This is known as the “first touch penalty.” Once an area of disk has incurred the first touch penalty, that area of disk does not incur the penalty again for the life of the instance, even if the DB instance is rebooted, restarted, or the DB instance class changes. Note that a DB instance created from a snapshot, a point-in-time restore, or a read replica is a new instance and does incur this first touch penalty.
- Provisioned IOPS provides a way to reserve IO capacity by specifying IOPS. Like any other system capacity attribute, maximum throughput under load will be constrained by the resource that is consumed first. That resource could be IOPS, channel bandwidth, CPU, memory, or database internal resources.

Other Factors that Impact Storage Performance

All of the following system related activities consume IO capacity and may reduce database instance performance while in progress:

- DB snapshot creation
- Nightly backups
- Multi-AZ peer creation
- Read replica creation
- Scaling storage

You may have looked at the read and write IO metrics for your DB instance, added them together, and you did not come up with the total you expected. It is possible that your DB instance is not hitting a channel throughput limit, your queue depths are consistently low, your CPU utilization is under 80%, there is free memory available, there is no swap activity, and there is plenty of free disk space. In addition, your application has dozens of threads all submitting transactions as fast as the database will take them, but there’s clearly unused IO capacity.

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.

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, and 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 zones in the same region. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.
It is important to remember that each 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 region. The default region can be changed in the console, by setting the EC2_REGION environment variable, or it can be overridden by using the --url parameter with the command line interface. See Common Options for API Tools for more information.

Amazon RDS supports a special AWS region called GovCloud that is designed to allow US government agencies and customers to move more sensitive workloads into the cloud by addressing their specific regulatory and compliance requirements. For more information on GovCloud, see the AWS GovCloud (US) home page.

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

Amazon RDS supports the endpoints listed in the following table.

<table>
<thead>
<tr>
<th>Region</th>
<th>Name</th>
<th>Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Northern Virginia) Region</td>
<td>us-east-1</td>
<td><a href="https://rds.us-east-1.amazonaws.com">https://rds.us-east-1.amazonaws.com</a></td>
</tr>
<tr>
<td>US West (Northern California) Region</td>
<td>us-west-1</td>
<td><a href="https://rds.us-west-1.amazonaws.com">https://rds.us-west-1.amazonaws.com</a></td>
</tr>
<tr>
<td>US West (Oregon) Region</td>
<td>us-west-2</td>
<td><a href="https://rds.us-west-2.amazonaws.com">https://rds.us-west-2.amazonaws.com</a></td>
</tr>
<tr>
<td>EU (Ireland) Region</td>
<td>eu-west-1</td>
<td><a href="https://rds.eu-west-1.amazonaws.com">https://rds.eu-west-1.amazonaws.com</a></td>
</tr>
<tr>
<td>Asia Pacific (Tokyo) Region</td>
<td>ap-northeast-1</td>
<td><a href="https://rds.ap-northeast-1.amazonaws.com">https://rds.ap-northeast-1.amazonaws.com</a></td>
</tr>
<tr>
<td>Asia Pacific (Singapore) Region</td>
<td>ap-southeast-1</td>
<td><a href="https://rds.ap-southeast-1.amazonaws.com">https://rds.ap-southeast-1.amazonaws.com</a></td>
</tr>
<tr>
<td>Asia Pacific (Sydney) Region</td>
<td>ap-southeast-2</td>
<td><a href="https://rds.ap-southeast-2.amazonaws.com">https://rds.ap-southeast-2.amazonaws.com</a></td>
</tr>
<tr>
<td>South America (Sao Paulo) Region</td>
<td>sa-east-1</td>
<td><a href="https://rds.sa-east-1.amazonaws.com">https://rds.sa-east-1.amazonaws.com</a></td>
</tr>
</tbody>
</table>
If you do not explicitly specify an endpoint, the US West (Oregon) endpoint is the default.

**Related Topics**

- DB Instance (p. 43)

## High Availability (Multi-AZ)

Amazon RDS provides high availability and failover support for DB instances using Multi-AZ deployments. Multi-AZ deployments for Oracle, PostgreSQL, and MySQL DB instances use Amazon technology to automatically provision and maintain 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.

Note that 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.

You can create a Multi-AZ deployment by simply specifying Multi-AZ when creating a DB instance. You can convert existing DB instances to Multi-AZ deployments by modifying the DB instance and specifying the Multi-AZ option. Note that you still need to use best practices when working with the high availability feature.

The RDS console shows the Availability Zone of the standby replica (called the secondary AZ), or you can use the command `rds-describe-db-instances` or the API action `DescribeDBInstances` to find the secondary AZ. When using the BYOL licensing model, you must have a license for both the primary instance and the standby replica.

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 you use Provisioned IOPS and DB instance classes (m1.large and larger) that are optimized for Provisioned IOPS for fast, consistent performance.
The 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 automatic failover mechanism simply changes the DNS record of the DB instance to point to the standby DB instance. As a result, you will need to re-establish any existing connections to your DB instance.

High availability does not mean that Amazon keeps, for example, two databases running in parallel; it is the data on disk that is replicated. If the primary DB instance becomes unavailable, a failover begins and the database software is started on the standby replica. 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. When the failover is complete, it can take additional time for the RDS console UI to reflect the new Availability Zone.

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 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. 380)
- You can view your DB events via the Amazon RDS console or APIs.
- You can view the current state of your Multi-AZ deployment via the Amazon RDS console and APIs.

DB Instance Maintenance

Periodically, the Amazon RDS system performs maintenance on the DB instance during a user-definable maintenance window. You can think of the maintenance window as an opportunity to control when DB instance modifications (such as implementing pending changes to storage or CPU class for the DB instance) and software patching occur, in the event either are requested or required. If a maintenance event is scheduled for a given week, it will be initiated and completed at some point during the 30 minute maintenance window you identify.

The only maintenance events that require Amazon RDS to take your DB instance offline are scale compute operations, which generally take only a few minutes from start-to-finish for Single-AZ instances and typically less than two minutes for Multi-AZ instances, or required software patching. Required patching is automatically scheduled only for patches that are security and durability related. Such patching occurs infrequently (typically once every few months) and seldom requires more than a fraction of your maintenance window. If you do not specify a preferred weekly maintenance window when creating your DB instance, a 30-minute default value is assigned. If you wish to change when maintenance is performed on your behalf, you can do so by modifying your DB instance in the AWS Management Console or by using the ModifyDBInstance API. Each of your DB instances can have different preferred maintenance windows, if you so choose.

Running your DB instance as a Multi-AZ deployment can further reduce the impact of a maintenance event, as Amazon RDS will conduct maintenance via the following steps: 1) Perform maintenance on
standby 2) Promote standby to primary 3) Perform maintenance on old primary, which becomes the new standby. For more information on Multi-AZ deployments, see High Availability (Multi-AZ) (p. 52).

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, Amazon RDS assigns a 30-minute maintenance window on a randomly selected day of the week.

The following table lists the time blocks for each region from which the default maintenance windows are assigned.

<table>
<thead>
<tr>
<th>Region</th>
<th>Time Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Northern Virginia) Region</td>
<td>03:00-11:00 UTC</td>
</tr>
<tr>
<td>US West (Northern California) Region</td>
<td>06:00-14:00 UTC</td>
</tr>
<tr>
<td>US West (Oregon) Region</td>
<td>06:00-14:00 UTC</td>
</tr>
<tr>
<td>EU (Ireland) Region</td>
<td>22:00-06:00 UTC</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo) Region</td>
<td>17:00-03:00 UTC</td>
</tr>
<tr>
<td>Asia Pacific (Sydney) Region</td>
<td>12:00-20:00 UTC</td>
</tr>
<tr>
<td>Asia Pacific (Singapore) Region</td>
<td>14:00-22:00 UTC</td>
</tr>
<tr>
<td>South America (São Paulo) Region</td>
<td>00:00-08:00 UTC</td>
</tr>
<tr>
<td>AWS GovCloud (US) Region</td>
<td>06:00-14:00 UTC</td>
</tr>
</tbody>
</table>

**Related Topics**

- Adjusting the Preferred Maintenance Window (p. 373)

**Amazon RDS and the Amazon Virtual Private Cloud Service**

Amazon RDS lets you use the Amazon Virtual Private Cloud (VPC) service to create a virtual private cloud where you can launch a DB instance. 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.
Amazon RDS supports two VPC platforms in each region: The **EC2-Classic** platform (shown as **EC2,VPC** in the RDS console) requires you to use the Amazon VPC service if you want to create a VPC, and the **EC2-VPC** platform (shown as **VPC** in the RDS console), which provides your AWS account with a default VPC in a region. If you are a new customer to Amazon RDS or if you are creating DB instances in a region you have not worked in before, chances are good you are on the **EC2-VPC** platform and that you have a default VPC. To determine which platform your account supports in a particular region, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 354).

For more information about using a VPC with Amazon RDS, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 353)

### DB Instance Backups

Amazon RDS provides two different methods for backing up and restoring your Amazon DB instances: automated backups and DB snapshots. Automated backups automatically back up your DB instance during a specific, user-definable backup window, and keeps the backups for a limited, user-specified period of time (called the backup retention period); you can later recover your database to any point in time during that retention period. DB snapshots are user-initiated backups that enable you to back up your DB instance to a known state, and restore to that specific state at any time. Amazon RDS keeps all DB snapshots until you delete them.

**Note**

A brief I/O freeze, typically lasting a few seconds, occurs during both automated backups and DB snapshot operations on Single-AZ DB instances.

#### Automated Backup

Automated backup is an Amazon RDS feature that automatically creates a backup of your database. Automated backups are enabled by default for a new DB instance.

An automated backup occurs during a daily user-configurable period of time known as the preferred backup window. Backups created during the backup window are retained for a user-configurable number of days (the backup retention period).

The preferred backup window is the user-defined period of time during which your DB instance is backed up. Amazon RDS uses these periodic data backups in conjunction with your transaction logs to enable you to restore your DB instance to any second during your retention period, up to the LatestRestorableTime (typically up to the last five minutes). During the backup window, storage I/O may be suspended while your data is being backed up and you may experience elevated latency. This I/O suspension typically lasts for the duration of the snapshot. This period of I/O suspension is shorter for Multi-AZ DB deployments, since the backup is taken from the standby, but latency can occur during the backup process.
When the backup retention changes to a non-zero value, the first backup occurs immediately. Changing the backup retention period to 0 turns off automatic backups for the DB instance, and deletes all existing automated backups for the instance.

If you don't specify a preferred backup window when you create the DB instance, Amazon RDS assigns a default 30-minute backup window which is selected at random from an 8-hour block of time per region.

The following table lists the time blocks for each region from which the default backups windows are assigned.

<table>
<thead>
<tr>
<th>Region</th>
<th>Time Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Northern Virginia) Region</td>
<td>03:00-11:00 UTC</td>
</tr>
<tr>
<td>US West (Northern California) Region</td>
<td>06:00-14:00 UTC</td>
</tr>
<tr>
<td>US West (Oregon) Region</td>
<td>06:00-14:00 UTC</td>
</tr>
<tr>
<td>EU (Ireland) Region</td>
<td>22:00-06:00 UTC</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo) Region</td>
<td>17:00-03:00 UTC</td>
</tr>
<tr>
<td>Asia Pacific (Sydney) Region</td>
<td>12:00-20:00 UTC</td>
</tr>
<tr>
<td>Asia Pacific (Singapore) Region</td>
<td>14:00-22:00 UTC</td>
</tr>
<tr>
<td>South America (São Paulo) Region</td>
<td>00:00-08:00 UTC</td>
</tr>
<tr>
<td>AWS GovCloud (US) Region</td>
<td>03:00-11:00 UTC</td>
</tr>
</tbody>
</table>

Changes to the backup window take effect immediately. The backup window cannot overlap with the weekly maintenance window for the DB instance.

When you delete a DB instance, you can create a final DB snapshot upon deletion; if you do, you can use this DB snapshot to restore the deleted DB instance at a later date. Amazon RDS retains this final user-created DB snapshot along with all other manually created DB snapshots after the DB instance is deleted. All automated backups are deleted and cannot be recovered when you delete a DB instance. Refer to the pricing page for information on backup storage costs.

For more information on working with automated backups, go to Working With Automated Backups (p. 293).

**Point-In-Time Recovery**

In addition to the daily automated backup, Amazon RDS archives database change logs. This enables you to recover your database to any point in time during the backup retention period, up to the last five minutes of database usage.

Amazon RDS stores multiple copies of your data, but for Single-AZ DB instances these copies are stored in a single availability zone. If for any reason a Single-AZ DB instance becomes unusable, you can use point-in-time recovery to launch a new DB instance with the latest restorable data. For more information on working with point-in-time recovery, go to Restoring a DB Instance to a Specified Time (p. 304).

**Note**

Multi-AZ deployments store copies of your data in different Availability Zones for greater levels of data durability. For more information on Multi-AZ deployments, see High Availability (Multi-AZ) (p. 52).
Automated Backups with Unsupported MySQL Storage Engines

Amazon RDS automated backups and DB snapshots are currently supported for all DB engines. For the MySQL DB engine, only the InnoDB storage engine is supported; use of these features with other MySQL storage engines, including MyISAM, may lead to unreliable behavior while restoring from backups. Specifically, since storage engines like MyISAM do not 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.

If you choose to use MyISAM, you can attempt to manually repair tables that become damaged after a crash by using the REPAIR command ([see: http://dev.mysql.com/doc/refman/5.5/en/repair-table.html]). However, as noted in the MySQL documentation, there is a good chance that you will not be able to recover all your data.

If you want to take DB snapshots with MyISAM tables, follow these steps:

**Launch Process**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stop all activity to your MyISAM tables (that is, close all sessions)</td>
</tr>
<tr>
<td>2</td>
<td>Lock and flush each of your MyISAM tables</td>
</tr>
<tr>
<td>3</td>
<td>Issue a CreateDBSnapshot API call, or use the Amazon RDS CLI command rds-create-db-snapshot. When the snapshot has completed, release the locks and resume activity on the MyISAM tables. These steps force MyISAM to flush data stored in memory to disk thereby ensuring a clean start when you restore from a DB snapshot.</td>
</tr>
</tbody>
</table>

Finally, if you would like to convert existing MyISAM tables to InnoDB tables, you can use alter table command (for example, alter table `TABLE_NAME` engine=innodb).

DB Snapshots

DB snapshots are user-initiated and enable you to back up your DB instance in a known state as frequently as you wish, and then restore to that specific state at any time. DB snapshots can be created with the Amazon RDS console or the CreateDBSnapshot action in the Amazon RDS API. DB snapshots are kept until you explicitly delete them with the Amazon RDS console or the DeleteDBSnapshot action in the Amazon RDS API. For more information on working with DB snapshots, see Creating a DB Snapshot (p. 296) and Restoring From a DB Snapshot (p. 298).

Related Topics

- Creating a DB Snapshot (p. 296)
- Restoring From a DB Snapshot (p. 298)
- Copying a DB Snapshot (p. 301)
- Working With Automated Backups (p. 293)

DB Instance Replication

Currently, you can create replicas of your DB instances in two ways. DB instances using the Oracle, PostgreSQL, or MySQL DB engines can use a Multi-AZ deployment, where Amazon RDS automatically provisions and manages a standby replica in a different Availability Zone (independent infrastructure in a physically separate location). In the event of planned database maintenance, DB instance failure, or
an Availability Zone failure, Amazon RDS will automatically failover to the standby so that database operations can resume quickly without administrative intervention. For more information on Multi-AZ deployments, see High Availability (Multi-AZ) (p. 52).

Amazon RDS uses MySQL's built-in replication functionality to create a special type of DB instance called a read replica that allows you to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. Once you create a read replica, database updates on the source DB instance are replicated to the read replica using MySQL's native, asynchronous replication. Amazon RDS currently supports replication only between Amazon RDS instances of MySQL, or for importing and exporting data between instances of MySQL running inside or outside of Amazon RDS.

## Read Replicas

You can create multiple read replicas for a given source DB instance and distribute your application's read traffic among them. You can also create a read replica from another read replica. Since read replicas use MySQL's built-in replication, they are subject to its strengths and limitations. In particular, updates are applied to your read replica(s) after they occur on the source DB instance, and replication lag can vary significantly. Read replicas can be created with Multi-AZ deployments to gain read scaling benefits in addition to the enhanced database write availability and data durability provided by Multi-AZ deployments.

You can monitor the status of a read replica using the RDS console or API. For more information about monitoring the status of a read replica, see Working with Read Replicas (p. 109).

There are a variety of scenarios where deploying one or more read replicas for a given source DB instance may make sense. Common reasons for deploying a read replica include:

- Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads. This excess read traffic can be directed to one or more read replicas.
- Serving read traffic while the source DB instance is unavailable. If your source DB instance cannot take I/O requests (e.g. due to I/O suspension for backups or scheduled maintenance), you can direct read traffic to your read replica(s). For this use case, keep in mind that the data on the read replica may be "stale" since the source DB instance is unavailable.
- Business reporting or data warehousing scenarios; you may want business reporting queries to run against a read replica, rather than your primary, production DB instance.

You can use replication to import data from an instance of MySQL running outside of Amazon RDS to an instance of MySQL running in Amazon RDS. You can also export data from an instance in Amazon RDS to an instance outside of Amazon RDS.

Read replicas require a transactional storage engine and are only supported for the InnoDB storage engine. Non-transactional engines such as MyISAM can prevent read replicas from working as intended. However, if you still choose to use MyISAM with read replicas, we advise you to watch the Amazon CloudWatch "Replica Lag" metric (available via the AWS Management Console or Amazon Cloud Watch APIs) carefully and recreate the read replica should it fall behind due to replication errors. The same considerations apply to the use of temporary tables and any other non-transactional engines.

You can promote a MySQL read replica into a standalone, single-AZ DB instance. There are several reasons you might want to promote a read replica:

- **Perform DDL operations**: DDL operations, such as creating or re-building indexes, can take time and impose a significant performance penalty on your DB instance. You can perform these operations on a 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. Common ways to split a database include 1)splitting tables that are not joined in the same query onto different hosts or 2)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.

- **Implement Failure Recovery** - You can use read replica promotion as a data recovery scheme if the source DB instance fails; however, if your use case requires synchronous replication, automatic failure detection, and failover, we recommend that you run your DB instance as a Multi-AZ deployment instead. 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 would first create a read replica and then monitor the source DB instance for failures. In the event of a failure, you would 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.

You can perform all of these operations using the Amazon Relational Database Service API Reference, and you can automate the process by using the Amazon Simple Workflow Service Developer Guide.

## Related Topics

- Working with Read Replicas (p. 109)

## AWS Identity and Access Management

Amazon RDS integrates with AWS Identity and Access Management (IAM) to let you control which users in your AWS Account can create or modify DB instances for your AWS Account. IAM lets your organization do the following:

- Create users and groups under your organization's AWS account
- Easily share your AWS account resources between the users in the account
- Assign unique security credentials to each user
- Granularly control users access to services and resources
- Get a single AWS bill for all users under the AWS account

IAM can be used to control who can create or change DB instances, but it is not used with Amazon RDS to control access to resources, such as databases. For more information about using IAM with Amazon RDS, see Managing Access to Your Amazon RDS Resources and Databases (p. 62)

For general information about IAM, go to:

- AWS Identity and Access Management (IAM)
- IAM Getting Started Guide
- Using IAM

For specific information about how you can control user access to Amazon RDS, go to Integrating with Other AWS Products in Using IAM.

## Failure to Retrieve Account Attributes

Recent changes to Amazon RDS may cause an error for some IAM users that were set up with permissions based on the Amazon RDS Full Access policy template. The error, "Failed to retrieve account attributes, certain console functions may be impaired. Retrying...," shown at the top of the page, is caused by the
console invoking actions that have not explicitly been given permissions in the Amazon RDS Full Access policy. We are actively working to fix this issue.

In order to resolve this issue, your IAM administrator must update the IAM user’s policy document to allow two additional Amazon EC2 actions: ec2:DescribeAccountAttributes and ec2:DescribeSecurityGroups. You must make this change for any IAM user or group that was assigned a policy that was based on the Amazon RDS Full Access policy template.

For example, the following code is the default policy document for the Amazon RDS Full Access policy template.

```json
{
  "Version":"2013-09-09",
  "Statement": [ 
    { 
      "Action": [
        "rds:*",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeVpcs",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:DescribeAlarms",
        "sns:ListTopics",
        "sns:ListSubscriptions"
      ],
      "Effect": "Allow",
      "Resource": "*"
    } 
  ]
}
```

Add the two additional actions stated above to get the following policy document that will give permission to the console to invoke the needed actions.

```json
{
  "Version":"2013-09-09",
  "Statement": [ 
    { 
      "Action": [
        "rds:*",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeVpcs",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeSecurityGroups",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:DescribeAlarms",
        "sns:ListTopics",
        "sns:ListSubscriptions"
      ],
      "Effect": "Allow",
      "Resource": "*"
    } 
  ]
}
```
For information about updating IAM policies, see Managing IAM Policies.
Managing Access to Your Amazon RDS Resources and Databases

Topics

• Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 63)
• Amazon RDS Security Groups (p. 74)

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:

• Use 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 RDS instances, tag resources, or modify DB security groups.

• Use security groups to control what IP addresses or 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 SSL connections with DB instances running the MySQL or SQL Server DB engines; for more information on using SSL with a DB instance, see Using SSL with a MySQL DB Instance (p. 81) or Using SSL with a SQL Server DB Instance (p. 206).

• Use network encryption and transparent data encryption with Oracle DB instances; for more information, see Oracle Native Network Encryption (p. 160) and Oracle Transparent Data Encryption (TDE) (p. 162)

• Run your DB instance in an Amazon Virtual Private Cloud (VPC) for additional 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 the security features of your DB engine to control who can log in to the databases on a DB instance, just as you would if the database was on your local network.

Note
You only have to configure security for your use cases; you do not 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.
You can use AWS IAM to create permissions that specify which Amazon RDS actions a user or group in your AWS account can perform, and on which RDS resources those actions can be performed. You specify permissions using an IAM policy, which is a JSON document.

When you sign up for an AWS account, you receive AWS account credentials. These AWS account credentials let you perform any AWS action, including Amazon RDS API actions. For example, by using your AWS account credentials, you can create and delete DB instances. IAM policies apply to working with the Amazon RDS CLI and API as well.

Note
AWS account credentials do not grant you the ability to login to databases on a DB instance.

You should not share your AWS account credentials with other users. Instead, you should use AWS IAM to establish user identities and create permissions that specify precisely what Amazon RDS actions and resources those users have access to.

You can create permissions that manage access to the following Amazon RDS resources:

- DB instances
- DB snapshots
- Read replicas
- Reserved instances
- DB security groups
- DB option groups
- DB parameter groups
- Event subscriptions
- DB subnet groups

To manage access to your Amazon RDS resources, you should take the following steps:

1. Create IAM users (user identities) under your AWS account for all users who will manage your Amazon RDS resources. Each user can have a separate password (for console access) and security credentials (for programmatic and CLI access). You can organize IAM users into groups, which makes it easier to manage permissions for multiple users at a time.

2. Determine what tasks each user and group will have regarding your Amazon RDS resources. For example, you could have groups for administrators, security personnel, DBAs, and developers.

3. Optionally, you can tag the Amazon RDS resources you want to control access to. You can assign a tag, a key-value pair, to any Amazon RDS resource, and use that tag as a way to specify a particular resource in an IAM policy.

   Note that tagging is not available in the GovCloud region.

4. Create the IAM policies that define the actions a user can take, and specify the Amazon RDS resources required for each task using Amazon Resource Names (ARNs). If you have used tags for your Amazon RDS resources, you can add conditions to the policy to test for those tag values.

5. Attach the policies to the applicable users or groups.
Creating IAM Policies for Amazon RDS

By default, newly created IAM users do not have permission to access any AWS resources. This means that IAM users also can't use the Amazon RDS console or CLI. To allow IAM users to use the features of Amazon RDS, you must create IAM policies that allow users to access the required Amazon RDS API actions and resources, and then attach the policies to the IAM users or groups that require those permissions.

An IAM policy is a JSON document that consists of one or more statements. Each statement in an IAM policy is made up of elements that define what actions can be taken on what resources. The following example shows a simple policy statement that allows a user to only create a DB instance that must have "test" prefixed to the DB instance name, use the MySQL DB engine, and can only use the micro DB instance class.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Condition": {
        "streq": [
          {"rds:DatabaseEngine": "mysql"},
          {"rds:DatabaseClass": "db.t1.micro"}
        ]
      }
    }
  ]
}
```

The `Version` element is required, and the value must be "2012-10-17". The `Effect` element is set to either "Allow" or "Deny". (Actions are denied by default, so you typically specify "Allow".) The `Action` element lists which AWS APIs the policy will allow (or deny). In this case, the `Action` element lists one action from the Amazon RDS API, so it will be the only action allowed by this policy statement. Note that the action is identified by both service name (rds) and action (CreateDBInstance); policies can list actions from any AWS service. You can use wildcards (*) to specify actions—for example, the action rds:Describe* would allow the user to perform any Amazon RDS action that begins with Describe (DescribeDBInstances, DescribeDBLogFiles, DescribeDBParameterGroups, DescribeDBSnapshots, etc.).

The `Resource` element lets you specify which resources the user can perform the actions on or with. In this example, the user can only create DB instances with the prefix "test" in the DB instance name. You specify resources using an Amazon Resources Name (ARN) that includes the name of the service that the resource belongs to (rds), the region (us-east-1), the account number, and the type of resource (a DB instance). For more information on creating ARNs, see Constructing an Amazon RDS Amazon Resource Name (ARN) (p. 285)

Finally, the optional `Condition` element lets you specify additional restrictions on the policy, such as date/time, source IP address, region, or tags. In this example, the `Condition` element indicates that the actions are allowed only on instances with the MySQL DB engine and the micro DB instance class. For more information on creating conditions, see Condition.

This policy might be attached to an individual IAM user, and in that case, that user would be allowed to perform the listed actions. You could instead attach the policy to an IAM group, and then every IAM user in that group would have these permissions.
Permissions Needed to Use the Amazon RDS Console

When users work with the Amazon RDS console, you must grant them permissions not only to perform the specific actions that you want to allow, but also permissions to actions that the console itself needs. For example, simply to list resources, the console runs the API actions such as DescribeSecurityGroups and DescribeSubnets. Users working in the console must have these permissions; if they don't, portions of the console that users need to work with might simply display a message that users don't have permissions for a task.

The following example policy statement shows permissions that users typically need in order to work in the Amazon RDS console. Notice that this includes RDS actions that start with the word "Describe," a number of EC2 and CloudWatch actions that likewise pertain to describing (listing) resources, and all SNS actions.

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "rds:Describe*",
          "rds:ListTagsForResource",
          "ec2:DescribeAvailabilityZones",
          "ec2:DescribeVpcs",
          "ec2:DescribeAccountAttributes",
          "ec2:DescribeSecurityGroups",
          "ec2:DescribeSubnets",
          "cloudwatch:GetMetricStatistics",
          "cloudwatch:DescribeAlarms",
          "sns:*"  
          ],
        "Resource": "*"
      ]
  }
}
```

How Resource Authorization Works in Amazon RDS

When a user requests an Amazon RDS action, an IAM authorization request is generated for every resource identified in the request. Amazon RDS checks the IAM policy for the user who is making the request. If the policy explicitly allows the user to perform the requested action on the specified resources, then the action takes place.

An authorization request that applies to multiple resources can result in multiple resource authorizations. For example, a point-in-time-restore to a new database instance will generate two authorization requests:

1. An authorization request will be generated for the target database instance.
2. An authorization request will be generated for the snapshot that is being restored.

Note that a policy can, for example, limit the storage or compute size to specific values or ranges. For a fuller explanation about how an IAM policy is evaluated, see IAM Policy Evaluation Logic.
Specifying Conditions in an IAM Policy for Amazon RDS

When creating an IAM policy, you can specify conditions using two types of values. You can specifying the tag associated with a resource as a condition, or you can use a predefined key, such as the DB engine type or the DB engine class. The following tables shows the predefined keys you can use when defining IAM policy for Amazon RDS. Note that tag key/value pairs and predefined keys are case sensitive.

AWS Predefined Keys

AWS provides several predefined keys that apply to all AWS resources that support IAM policies. The following table shows the AWS predefined keys that apply to Amazon RDS resources.

<table>
<thead>
<tr>
<th>AWS Predefined Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws:CurrentTime</td>
<td>The current time. Used for date conditions.</td>
</tr>
<tr>
<td>aws:EpochTime</td>
<td>The current time in epoch or UNIX time format. Used for date conditions.</td>
</tr>
<tr>
<td>aws:principaltype</td>
<td>The type of principal (user, account, federated user, etc.) for the current request.</td>
</tr>
<tr>
<td>aws:SourceIp</td>
<td>The requester’s IP address (see IP Address). Note that if you use aws:SourceIp, and the request comes from an Amazon EC2 instance, the instance’s public IP address is evaluated.</td>
</tr>
<tr>
<td>aws:UserAgent</td>
<td>The requester’s client application.</td>
</tr>
<tr>
<td>aws:userid</td>
<td>The requester’s user ID.</td>
</tr>
<tr>
<td>aws:username</td>
<td>The requester’s user name</td>
</tr>
</tbody>
</table>

Amazon RDS Predefined Keys

Amazon RDS also has predefined keys that you can include in Condition elements in an IAM policy. Amazon RDS predefined keys do not apply to all actions; the Amazon RDS predefined keys apply to the following actions:

- CreateDBInstance
- ModifyDBInstance
- DeleteDBInstance
- DescribeDBLogFiles
- AddTagsToResource (when resource = db)
- RemoveTagsFromResource (when resource = db)
- RestoreDBInstanceToPointInTime
- RestoreDBInstanceFromDBSnapshot (only the request parameters are validated)
- DownloadDBLogFilePortion
- DescribeDBInstance (where an instance id is passed)

The following table shows the Amazon RDS predefined keys that apply to Amazon RDS resources.
<table>
<thead>
<tr>
<th>RDS Predefined Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rds:DatabaseClass</td>
<td>The DB instance class of a DB instance</td>
</tr>
<tr>
<td>rds:DatabaseEngine</td>
<td>The DB engine of the DB instance</td>
</tr>
<tr>
<td>rds:DatabaseName</td>
<td>The name of the database on the DB instance</td>
</tr>
<tr>
<td>rds:MultiAz</td>
<td>Indicates if the DB instance is running in multiple availability zones. 1 indicates that the DB instance is using multi-AZ.</td>
</tr>
<tr>
<td>rds:Piops</td>
<td>This key will be present when a request is made for a DB instance with PIOPs enabled. The value will contain the number of provisioned IOs that an instance supports. 0 indicates does not have PIOPs enabled.</td>
</tr>
<tr>
<td>rds:StorageSize</td>
<td>The storage volume size (in GB)</td>
</tr>
<tr>
<td>rds:Vpc</td>
<td>Indicates if the database instance is running in a virtual private cloud</td>
</tr>
</tbody>
</table>

Using Predefined Keys and Tags with a Condition Element

You can add your own tags to an Amazon RDS resource and use those tags in IAM policies using the Condition element. Amazon RDS resource tags are formatted slightly differently than predefined keys in a Condition element. For example, the following Condition element uses a predefined key and specifies that the condition applies to the DB engine MySQL:

```
"Condition":{"streq":{"rds:DatabaseEngine", "mysql" } }
```

The following Condition element uses an Amazon RDS tag and specifies that the condition applies to a tag with a key named environment and a value of production.

```
"Condition":{"streq":{"rds:db-tag/environment": [“production”]}}
```

The format for the tag in a Condition element is “rds:”, followed by the type of resource (in this case, “db”), a hyphen, then the tag key and tag value separated by a forward slash.

_for more information about the IAM policy Condition element, see Condition._

**Example IAM Policies for Amazon RDS**

The following examples show simple IAM policy statements that you can use to manage the access IAM users have to Amazon RDS resources.

**Example 1: Permit a user to perform any Describe action on any RDS resource**

The following statement allows a user to run the describe action, which shows information about an RDS resource such as a DB instance. Note that the "*" in the Resource element indicates that the actions are allowed for all Amazon RDS resources.

```json
{
"Version":"2012-10-17",
"Statement":[
{"Effect":"Allow",
"Action":"rds:Describe*",
"Resource":"*"
]}
```
Example 2: Permit a user to create a DB instance that uses a specified DB engine

The following statement uses a predefined Amazon RDS key and allows a user to create only DB instances that use the MySQL DB engine. The Condition element indicates that the DB engine requirement is MySQL.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": "rds:CreateDBInstance",
    "Resource": "*",
    "Condition": {
      "strneq": {
        "rds:DatabaseEngine": "mysql"
      }
    }
  }]
}
```

Example 3: Permit a user to create a DB instance that uses the specified DB parameter and security groups

The following statement allows 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": [{
    "Effect": "Allow",
    "Action": "rds:CreateDBInstance",
    "Resource": [
      "arn:aws:rds:us-east-1:1234567890:secgrp:db-production"
    ]
  }]
}
```

Example 4: Prevent a user from creating a DB instance that uses specified DB parameter groups

The following statement prevents a user from creating 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.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": "rds:CreateDBInstance",
    "Resource": "*",
    "Condition": {
      "streq": {
        "rds:db-tag/usage": "prod"
      };
    }
  }]
}
```
Example 5: Prevent users from creating DB instances for certain DB instance classes and from creating DB instances that use Provisioned IOPS

The following statement prevents users from creating DB instances that use the DB instance classes m2.2xlarge and m2.4xlarge, which are the largest and most expensive instances. This example also prevents users from creating DB instances that use Provisioned IOPS, which is an additional cost.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "rds:CreateDBInstance",
            "Resource": "*",
            "Condition": {
                "streq": {
                    "rds:DatabaseClass": ["db.m2.4xlarge", "db.m2.2xlarge"]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": "rds:CreateDBInstance",
            "Resource": "*",
            "Condition": {"NumericNotEquals": {"rds:Piops": "0"}}
        }
    ]
}
```

You can add a tag to an Amazon RDS resource, and then use that tag in a policy to specify a particular resource. The following examples use Amazon RDS resource tags as part of the IAM policy to specify a particular resource.

Example 6: Permits a user to perform an action on a resource tagged with two different values

This following statement allows a user to perform the `ModifyDBInstance` and `CreateDBSnapshot` actions on instances with either the "stage" tag set to "development" or "test."

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds:ModifyDBInstance",
                "rds:CreateDBSnapshot"
            ],
            "Resource": "*",
            "Condition": {
                "streq": {
                    "db-tag/stage": ["development", "test"]
                }
            }
        }
    ]
}
```

Example 7: Permits a user to perform actions on a DB instance with a DB instance name prefixed with the user name

This following statement allows a user to perform any action (except to add or remove tags) 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."

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["rds:*"],
            ],
            "Condition": {
                "streq": {
                    "db-tag/name": ["prefix:myuser"]
                }
            }
        }
    ]
}
```
Sample IAM Policy for Amazon RDS

An IAM policy can apply to other AWS services; for example, this sample policy is focused on managing Amazon RDS resources but includes EC2 resources that are required for accessing information through the Amazon RDS console.

The following policy lets a user create and view all resources, modify and delete resources with a tag of environment=development or environment=test, and add a tag of environment=development or environment=test tags to any Amazon RDS resource unless the resource is already tagged with environment=production. For all resources tagged environment=production, the user cannot modify or delete these resources and they cannot grant themselves access.

```json

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource"
      ],
      "Resource": "*",
      "Condition": {
        "streq": {
          "rds:req-tag/environment": [
            "development",
            "test"
        ]
      }
    },
    {
      "strneq": {
        "rds:db-tag/environment": [
          "production"
        ],
        "rds:es-tag/environment": [
          "production"
        ],
        "rds:og-tag/environment": [
          "production"
        ],
        "rds:pg-tag/environment": [
          "production"
        ]
      }
    }
  ]
}
```

For more information about adding tags to an Amazon RDS resource, see [Constructing an Amazon RDS Amazon Resource Name (ARN)](p. 285). For more information about policies, see Permissions and Policies in the IAM documentation.
"rds:ri-tag/environment": [ "production" ],
"rds:recgrp-tag/environment": [ "production" ],
"rds:snapshot-tag/environment": [ "production" ],
"rds:subgrp-tag/environment": [ "production" ]
}

},
{
"Effect": "Allow",
"Action": [ "rds:RemoveTagsFromResource" ],
"Resource": "+",
"Condition": {
"streq": {
"rds:req-tag/environment": [ "" ]
},
"strneq": {
"rds:db-tag/environment": [ "production" ],
"rds:es-tag/environment": [ "production" ],
"rds:og-tag/environment": [ "production" ],
"rds:pg-tag/environment": [ "production" ],
"rds:ri-tag/environment": [ "production" ],
"rds:recgrp-tag/environment": [ "production" ],
"rds:snapshot-tag/environment": [ "production" ],
"rds:subgrp-tag/environment": [ "production" ]
}
}

}
Sample IAM Policy for Amazon RDS

API Version 2013-09-09

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [  
        "rds:DeleteDBInstance",
        "rds:ModifyDBInstance",
        "rds:CopyDBSnapshot",
        "rds:DownloadDBLogFilePortion",
        "rds:PromoteReadReplica",
        "rds:RebootDBInstance",
        "rds:RestoreDBInstanceFromDBSnapshot",
        "rds:RestoreDBInstanceToPointInTime"
      ],
      "Resource": "*",
      "Condition": {
        "streq": {
          "rds:db-tag/environment": [  
            "development",
            "test"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [  
        "rds:DeleteEventSubscription",
        "rds:ModifyEventSubscription",
        "rds:AddSourceIdentifierToSubscription",
        "rds:RemoveSourceIdentifierFromSubscription"
      ],
      "Resource": "*",
      "Condition": {
        "streq": {
          "rds:es-tag/environment": [  
            "development",
            "test"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [  
        "rds:DeleteOptionGroup",
        "rds:ModifyOptionGroup"
      ],
      "Resource": "*",
      "Condition": {
        "streq": {
          "rds:og-tag/environment": [  
            "development",
            "test"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [  
        "rds:DeleteDBParameterGroup",
        "rds:ModifyDBParameterGroup"
      ],
      "Resource": "*",
      "Condition": {
        "streq": {
          "rds:db-param-tag/environment": [  
            "development",
            "test"
          ]
        }
      }
    }
  ]
}
```
Sample IAM Policy for Amazon RDS

```
{
  "Effect": "Allow",
  "Action": ["rds:resetDBParameterGroup"],
  "Resource": "*",
  "Condition": {
    "streq": {
      "rds:pg-tag/environment": [
        "development",
        "test"
      ]
    }
  }
},
{
  "Effect": "Allow",
  "Action": ["rds:DeleteSecurityGroup",
              "rds:AuthorizeDBSecurityGroupIngress",
              "rds:RevokeDBSecurityGroupIngress"],
  "Resource": "*",
  "Condition": {
    "streq": {
      "rds:secgrp-tag/environment": [
        "development",
        "test"
      ]
    }
  }
},
{
  "Effect": "Allow",
  "Action": ["rds:DeleteDBSnapshot",
              "rds:CopyDBSnapshot",
              "rds:RestoreDBInstanceFromDBSnapshot",
              "rds:RestoreDBInstanceToPointInTime"],
  "Resource": "*",
  "Condition": {
    "streq": {
      "rds:snapshot-tag/environment": [
        "development",
        "test"
      ]
    }
  }
},
{
  "Effect": "Allow",
  "Action": ["rds:DeleteDBSubnetGroup",
              "rds:ModifyDBSubnetGroup"],
  "Resource": "*",
  "Condition": {
    "streq": {
      "rds:subgrp-tag/environment": [
        "development",
        "test"
      ]
    }
  }
}
```

API Version 2013-09-09
Amazon RDS 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 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 (or other AWS instances) inside a VPC, and 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

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 (e.g., 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. Note that DB security group rules apply to inbound traffic only; outbound traffic is not currently permitted for DB instances.

You do not 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 APIs 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 (p. 329)
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 (e.g., 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 do not apply to DB instances. Note that you must use the Amazon EC2 API or the Security Group option on the VPC Console to create VPC security groups.

DB instances deployed within a VPC can be configured to be accessible from the Internet or from EC2 instances outside the VPC. If a VPC security group specifies a port access such as TCP port 22, you would not be able to access the DB instance because the firewall for the DB instance provides access only via the IP addresses specified by the DB security groups the instance is a member of and the port defined when the DB instance was created.

You should use TCP as the protocol for any VPC security group created to control access to a DB instance. The port number for the VPC security group should be the same port number as that used to create the DB instance.

DB Security Groups vs. VPC Security Groups

The following table shows the key differences between DB security groups and VPC security groups.

<table>
<thead>
<tr>
<th>DB Security Group</th>
<th>VPC Security Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controls access to DB instances outside a VPC</td>
<td>Controls access to DB instances in VPC.</td>
</tr>
<tr>
<td>Uses Amazon RDS APIs or Amazon RDS page of the AWS Management Console to create and manage group/rules</td>
<td>Uses Amazon EC2 APIs or Amazon VPC page of the AWS Management Console to create and manage group/rules</td>
</tr>
<tr>
<td>When you add a rule to a group, you do not need to specify port number or protocol.</td>
<td>When you add a rule to a group, you should specify the protocol as TCP, and specify the same port number that you used to create the DB instances (or Options) you plan to add as members to the group.</td>
</tr>
<tr>
<td>Groups allow access from EC2 security groups in your AWS account or other accounts.</td>
<td>Groups allow access from other VPC security groups in your VPC only.</td>
</tr>
</tbody>
</table>

Security Group Scenario

A common use of an RDS instance in a VPC is to share data with an application server running in an EC2 instance in the same VPC and that is accessed by a client application outside the VPC. For this scenario, you would do the following to create the necessary instances and security groups. You can use the RDS and VPC pages on the AWS Console or the RDS and EC2 APIs.

1. Create a VPC security group (for example, "sg-appsrv1") and define inbound rules that use as source the IP addresses of the client application. 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") 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 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") 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 you created in step 3.

The following diagram shows this scenario.

For more information on working with DB security groups, go to Working with DB Security Groups (p. 329).

Related Topics

• Working with DB Security Groups (p. 329)
MySQL on Amazon RDS

Amazon RDS supports DB instances running several versions of MySQL. You first use the Amazon RDS management tools or interfaces to create an Amazon RDS MySQL 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 MySQL utilities and applications to store and access the data in the DB instance.

These are the common management tasks you perform with an Amazon RDS MySQL DB instance, with links to information about each task:

- For planning information, such as MySQL versions, storage engines, security, and features supported in Amazon RDS, see Amazon RDS MySQL Planning Information (p. 78).
- You can create an Amazon RDS MySQL DB instance after you have met prerequisites, such as creating security groups, DB parameter groups, or DB option groups. For information, see Creating a DB Instance Running the MySQL Database Engine (p. 82).
- After creating the security group and DB instance, you can connect to the DB instance from MySQL applications and utilities. For information, see Connecting to a DB Instance Running the MySQL Database Engine (p. 91).
- A newly created Amazon RDS DB instance has one empty database with the name you specified when you created the DB instance, and one masteruser account with the name and password you specified. You must use a MySQL tool or utility to log in as the masteruser, and then use MySQL commands and SQL statements to add all of the users and elements required for your applications to store and retrieve data in the DB instance, such as:
  - Create all user IDs and grant them the appropriate permissions. For information, go to MySQL User Account Management in the MySQL documentation.
  - Create any required databases and objects such as tables and views. For information, go to Data Definition Statements in the MySQL documentation.
  - Establish procedures for importing or exporting data. For information on some recommended procedures, see Importing and Exporting Data From an Amazon RDS MySQL DB Instance (p. 95).
- You may need to periodically change your DB instance, such as to resize or reconfigure the DB instance. For information, see Modifying a DB Instance Running the MySQL Database Engine (p. 93). For additional information on specific tasks, see:
  - Renaming a DB Instance (p. 271)
  - Deleting a DB Instance (p. 273)
  - Rebooting a DB Instance (p. 276)
You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots. For information, see Backing Up and Restoring a DB Instance (p. 292).

You can monitor an instance through actions such as viewing the MySQL logs, CloudWatch Amazon RDS metrics, and events. For information, see Monitoring a DB Instance (p. 376).

You can offload read traffic from your primary MySQL DB instance by creating read replicas. For information, see Working with Read Replicas (p. 109).

There are several Amazon RDS features you can use with MySQL DB instances that are common across the Amazon RDS database engines. For information, see:

- Working with Reserved DB Instances (p. 342)
- Working with Provisioned IOPS Storage (p. 361)

There are also several appendices with useful information about working with Amazon RDS MySQL DB instances:

- Appendix: Common DBA Tasks for MySQL (p. 118)
- Appendix: Options for MySQL DB Engine (p. 123)
- Appendix: MySQL Amazon RDS SQL Reference (p. 127)

### Amazon RDS MySQL Planning Information

**Topics**

- Amazon RDS MySQL Versions (p. 78)
- Amazon RDS Supported Storage Engines (p. 79)
- Amazon RDS and MySQL Security (p. 80)
- MySQL Features Not Supported By Amazon RDS (p. 81)

### Amazon RDS MySQL Versions

Amazon RDS currently supports MySQL versions 5.6, 5.5, and 5.1. Over time, we plan to support additional MySQL versions for Amazon RDS. The number of new version releases supported in a given year will vary based on the frequency and content of the MySQL version releases and the outcome of a thorough vetting of the release by our database engineering team. However, as a general guidance, we aim to support new MySQL versions within 3-5 months of their General Availability release.

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 would be considered major if the major version number changes; for example, going from version 5.1.71 to 5.5.33. A version change would be considered minor if only the minor version number changes - for example, going from version 5.5.31 to 5.5.33.

You can specify any currently supported MySQL version when creating a new DB Instance. You can specify the MySQL 5.6, 5.5, or 5.1 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 (e.g. MySQL 5.6) 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.
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 will occur during your scheduled maintenance window, and it will be announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades, set the AutoMinorVersionUpgrade parameter to “false.”

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 (p. 288).

Amazon RDS currently only supports major version upgrades from MySQL 5.1 to MySQL 5.5. Because major version upgrades involve some compatibility risk, they will not occur automatically; you must manually modify the DB instance. You should thoroughly test any upgrade before applying your production instances. For information about upgrading a DB instance, see Upgrading a DB Instance (p. 288).

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.

The Amazon RDS deprecation policy for MySQL includes the following:

- We intend to support major MySQL version releases, including MySQL 5.1, for 3 years after they are initially supported by Amazon RDS.
- We intend to support minor MySQL version releases (e.g. MySQL 5.1.45) for at least 1 year after they are initially supported by Amazon RDS.
- After a MySQL major or minor version has been “deprecated”, we expect to provide a three month grace period for you to initiate an upgrade to a supported version prior to an automatic upgrade being applied during your scheduled maintenance window.

**Using the memcached option with MySQL 5.6**

Most Amazon RDS DB engines support option groups that allow you to select additional features for your DB instance. MySQL 5.6 DB instances support the memcached option, a distributed memory cache service. For more information about the memcached option, see Appendix: Options for MySQL DB Engine (p. 123). For more information about working with option groups, see Working with Option Groups (p. 306).

**Amazon RDS Supported Storage Engines**

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 to use the InnoDB memcached interface. For more information, see MySQL 5.6 memcached Support (p. 123).

The Federated Storage Engine is currently not supported by Amazon RDS for MySQL.

The MyISAM storage engine does not support reliable recovery and may 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 may be
helpful under some conditions. For more information on MyISAM restrictions, see Automated Backups with Unsupported MySQL Storage Engines (p. 57).

If you would like to convert existing MyISAM tables to InnoDB tables, you can use the alter table command (e.g., alter table TABLE_NAME engine=innodb;). Please 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.

Amazon RDS and MySQL Security

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 Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 63).

• 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 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 MySQL DB instance, authentication of the login and permissions are applied the same way as in a stand-alone instance of MySQL. 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. For information, go to MySQL User Account Management in the MySQL documentation.

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 slave
- select
- show databases
- show view
- trigger
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 API or the rds-modify-db-instance 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 will be 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 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 public key is stored at https://rds.amazonaws.com/doc/mysql-ssl-ca-cert.pem.

Important
The SSL support in Amazon RDS is strictly for encrypting the connection between your client and your DB instance; it should not be relied on for authenticating the server.

To encrypt connections using the default mysql client, launch the mysql client using the --ssl_ca parameter to reference the public key, for example:

```sql
mysql -h myinstance.c9akciq32.rds-us-east-1.amazonaws.com --ssl_ca=rds-ssl-ca-cert.pem
```

you can use the GRANT statement to require SSL connections for specific users accounts. For example, you can use the following statement to require SSL connections on the user account encrypted_user:

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL
```

Note
For more information on SSL connections with MySQL, go to the MySQL documentation.

MySQL Features Not Supported By Amazon RDS

Amazon RDS currently does not support the following MySQL features:

- Global Transaction IDs
- InnoDB Cache Warming
- Transportable Table Space
- Authentication Plugin
- Password Strength Plugin
- Semi-synchronous Replication

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. 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 will have all database-level permissions except for those used for backups (Amazon RDS manages backups for you).

**MySQL Parameter Exceptions for Amazon RDS DB Instances**

This section describes any exceptions and/or special considerations for MySQL database engine parameters.

**lower_case_table_names**

Because Amazon RDS runs on 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 (the default) or 1.

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.

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.

**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. Your applications connect to the endpoint exposed by the DB instance whenever they need to access the databases created in that DB instance. The information you specify when you create the instance controls configuration elements such as size, storage, engine version, security, and maintenance periods.

**DB Instance Prerequisites**

You must meet certain prerequisites before you can create an Amazon RDS MySQL instance.

The first set of prerequisites is to integrate the DB instance into the AWS network infrastructure and authorize connections. The set of network prerequisites you need for a DB instance depend on whether you create it in a default VPC, in a user-defined VPC, or outside of a VPC. For information on determining if your account has a default VPC in a region, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 354). The follow list describes the prerequisites for each network infrastructure option:

- **Default VPC** — if your AWS account has a default VPC in the region, that VPC is configured to support DB instances. If you specify the default VPC when you create the DB instance:
• You must specify a VPC security group that authorizes connections from the devices and Amazon EC2 instances running the applications or utilities that will access the databases in the DB instance. For information, see Step 3: Creating a VPC Security Group (p. 359).
• You can specify the default DB subnet group. If this is the first DB instance you have created in the region, Amazon RDS will create the default DB subnet group when it creates the DB instance.

**User-defined VPC** — if you specify a user-defined VPC when you create a DB instance:
• You must specify a VPC security group.
• 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 RDS and the Amazon Virtual Private Cloud Service (p. 54).
• You must 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. 355).

**No VPC** — if your AWS account does not have a default VPC, and you do not specify a user-defined VPC:
• You must specify a DB security group that authorizes connections from the devices and Amazon EC2 instances running the applications or utilities that will access the databases in the DB instance. For more information, see Working with DB Security Groups (p. 329).

Additional prerequisites for all instances include:

• 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 Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 63).
• If you want to tailor the configuration parameters for your DB instance, you must specify a DB parameter group with the required parameter settings. For information about creating or modifying a DB parameter group, see Working with DB Parameter Groups (p. 317).
• If you want to use the MySQL 5.6 memcached interface, you must specify a DB option group. For information about creating or modifying a DB option group, see Appendix: Options for MySQL DB Engine (p. 123).
• You must determine the TCP/IP port number you will specify for the DB instance. The firewalls at some companies block connections to the default MySQL port (3306). If your company firewall blocks the default port, choose another port for the new DB instance.

**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, select the region in which you want to create the DB instance.
3. In the navigation pane, click **DB Instances**.
4. Click **Launch DB Instance** to start the **Launch DB Instance Wizard**.

The wizard opens on the **Engine Selection** page.
5. In the Launch DB Instance Wizard window, click the Select button for the MySQL DB engine.

6. The next step asks if you are planning to use the DB instance you are creating for production. If you are, select Yes. By selecting Yes, the failover option Multi-AZ and the Provisioned IOPS storage option will be preselected in the following step. Click Next Step when you are finished.

7. On the DB Instance Details page, specify your DB instance information. The following table shows settings for an example DB instance. Click Next Step when you are finished.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Model</td>
<td>MySQL has only one license model. Select the default, General-Public-License, to use the general license agreement for MySQL.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>Select the version of MySQL that you want to work with. Note that Amazon RDS supports several versions of MySQL.</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>Select a DB instance class that defines the processing and memory requirements for the DB instance. For more information about all the DB instance class options, see DB Instance Class (p. 47).</td>
</tr>
<tr>
<td>Multi-AZ Deployment</td>
<td>Determine if you want to create a standby replica of your DB instance in another Availability Zone for failover support. This feature is available for Oracle and MySQL DB instances. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 50).</td>
</tr>
<tr>
<td>For this parameter...</td>
<td>...Do this:</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Auto Minor Version Upgrade</td>
<td>Select <strong>Yes</strong> if you want to enable your DB instance to receive minor DB Engine version upgrades automatically when they become available.</td>
</tr>
<tr>
<td>Allocated Storage</td>
<td>Type a value to allocate storage for your database (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 about storage allocation, see <em>Amazon RDS Storage</em> (p. 48).</td>
</tr>
<tr>
<td>Use Provisioned IOPS</td>
<td>This option turns on Provisioned IOPS (I/O operations per second), a high-performance storage option in Amazon RDS that is optimized for I/O-intensive, transactional (OLTP) database workloads. You can then enter the amount of Provisioned IOPS you want. For more information about high performance storage, see <em>Working with Provisioned IOPS Storage</em> (p. 361).</td>
</tr>
<tr>
<td>DB Instance Identifier</td>
<td>Type a name for the DB instance that is unique for your account in the region you selected. You may choose to add some intelligence to the name such as including the region and DB Engine you selected, for example <em>mysql-instance1</em>.</td>
</tr>
<tr>
<td>Master User Name</td>
<td>Type a name using alphanumeric characters that you will use as the master user name to log on to your DB instance. The default privileges granted to the master user name account include: create, drop, references, event, alter, delete, index, insert, select, update, create temporary tables, lock tables, trigger, create view, show view, alter routine, create routine, execute, create user, process, show databases, grant option.</td>
</tr>
<tr>
<td>Master User Password</td>
<td>Type a password that contains from 8 to 16 printable ASCII characters (excluding /,, and @) for your master user password.</td>
</tr>
</tbody>
</table>
8. On the Additional Configuration 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 click Next Step.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Name</td>
<td>Type a name for your database of up to 8 alpha-numeric characters. If you do not provide a name, Amazon RDS will not create a database on the DB instance you are creating.</td>
</tr>
<tr>
<td>Database Port</td>
<td>Specify the port that applications and utilities will use to access the database. MySQL installations default to port 3306. The firewalls at some companies block connections to the default MySQL port. If your company firewall blocks the default port, choose another port for the new DB instance.</td>
</tr>
<tr>
<td>Choose a VPC</td>
<td>Determine if you want the DB instance to be created in a Virtual Private Cloud (VPC). For more information about VPC, see Amazon RDS and the Amazon Virtual Private Cloud Service (p. 54).</td>
</tr>
<tr>
<td>Publicly Accessible</td>
<td>(Only applies if you choose a VPC) Select yes to give the DB instance a public IP address; otherwise, select no. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.</td>
</tr>
</tbody>
</table>
For this parameter... | ...Do this:
---|---
**Availability Zone** | Determine if you want to specify a particular Availability Zone. If you selected Yes for the Multi-AZ Deployment parameter on the previous page, you will not have any options here. For more information about Availability Zones, see Regions and Availability Zones (p. 50).

**Option Group** | Select an option group. Each MySQL version has a default option group you can use, or you can create your own option group. For more information about option groups, see Working with Option Groups (p. 306).

**Parameter Group** | Select a parameter group. Each MySQL version has a default parameter group you can use, or you can create your own parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 317).

**DB Security Groups** | Select the security group you want to use with this DB instance. For more information about security groups, see Working with DB Security Groups (p. 329).

---

9. On the Management Options page, you can specify backup and maintenance options for your DB instance. Click Next Step when the settings are as you want them. For more information about the maintenance window, see Adjusting the Preferred Maintenance Window (p. 373). For more information on backups and the backup retention period, see DB Instance Backups (p. 55).
In addition, Federated Storage Engine is currently not supported by Amazon RDS for MySQL.

Note

The Point-In-Time-Restore and Snapshot Restore features of Amazon RDS for MySQL require a crash recoverable storage engine, and these two features are supported only for the InnoDB storage engine. While 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 may result in lost or corrupt data when MySQL is restarted after a crash, preventing Point-In-Time-Restore or Snapshot restore from working as intended.

If you would like to convert existing MyISAM tables to InnoDB tables, you can use the alter table command (e.g., alter table TABLE_NAME engine=innodb;). Note 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.

10. On the Review page, review the options for your DB instance. If you need to make any changes, click Back to return to the appropriate page, and then make the necessary corrections. When all the settings are as you want them, click Launch DB Instance.

11. On the final page of the wizard, click Close.

12. 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.
CLI

To create a MySQL DB instance

- Use the CLI command `rds-create-db-instance` to create a DB instance. For more information, go to `rds-create-db-instance` in the Amazon Relational Database Service Command Line Reference. For example:

```
PROMPT>rds-create-db-instance mydbinstance -s 20 -c db.m1.small -e MySQL - u <masterawsuser> -p <masteruserpassword> --backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE  mydbinstance  db.m1.small  mysql  20  sa  creating  3 **** n 5.1.57
SECGROUP  default  active
PARAMGRP  default.mysql5.1  in-sync
```

API

To create a MySQL DB instance

- Call the API action `CreateDBInstance` to create a DB instance. For more information, go to `CreateDBInstance` in the Amazon Relational Database Service API Reference. For example:

```
• DBInstanceIdentifier = mydbinstance
• DBInstanceClass = db.m1.small
• AllocatedStorage = 20
• BackupRetentionPeriod = 3
• MasterUsername = <masterawsuser>
• MasterUserPassword = <masteruserpassword>
```

API Version 2013-09-09
Example

https://rds.us-west-2.amazonaws.com/
  ?Action=CreateDBInstance
  &AllocatedStorage=20
  &BackupRetentionPeriod=3
  &DBInstanceClass=db.m1.small
  &DBInstanceIdentifier=mydbinstance
  &DBName=mydatabase
  &DBSecurityGroups.member.1=mysecuritygroup
  &DBSubnetGroup=mydbsubnetgroup
  &Engine=mysql
  &MasterUserPassword=<masteruserpassword>
  &MasterUsername=<masterawsuser>
  &Version=2013-09-09
  &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

Related Topics

- DB Instance (p. 43)
- DB Instance Class (p. 47)
- Deleting a DB Instance (p. 273)
Connecting to a DB Instance Running the MySQL Database Engine

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.

You can use the AWS Management Console, the `rds-describe-db-instances` CLI command, or the DescribeDBInstances API action to list the details of an Amazon RDS DB instance, including its endpoint. If an endpoint value is `myinstance.123456789012.us-east-1.rds.amazonaws.com:3306`, then you would specify the following values in a MySQL connection string:

- For host or host name, specify `myinstance.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](https://dev.mysql.com/doc/refman/8.0/en/) 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](https://dev.mysql.com/downloads/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. 81).

## 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 for your DB instance. For the `-P` parameter, substitute the port for your DB instance. Enter the master user password when prompted.

  ```
  PROMPT> mysql -h myinstance.123456789012.us-east-1.rds.amazonaws.com -P 3306 -u mymasteruser -p
  ```

  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: 5.1.32-log MySQL Community Server (GPL)
Connecting With SSL

To connect to a DB instance with SSL using the MySQL utility

1. Download the public key for the Amazon RDS signing certificate from https://rds.amazonaws.com/doc/rds-ssl-ca-cert.pem. Note that this will download a file named mysql-ssl-ca-cert.pem.

2. Type 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. Enter the master user password when prompted.

   PROMPT> mysql -h myinstance.123456789012.us-east-1.rds.amazonaws.com --ssl_ca=mysql-ssl-ca-cert.pem

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: 5.1.32-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

Related Topics

- DB Instance (p. 43)
- Creating a DB Instance Running the MySQL Database Engine (p. 82)
- Amazon RDS Security Groups (p. 74)
- Deleting a DB Instance (p. 273)
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. For information about additional tasks, such as renaming, rebooting, deleting, tagging, or upgrading an Amazon RDS DB instance, see Tasks Common to All Amazon RDS DB Engines (p. 270). 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 a DB Instance and Using the Apply Immediately Parameter (p. 45).

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, click Instances.
3. Select the check box for the DB instance that you want to change, and then click Modify.
4. In the Modify DB Instance dialog box, change any of the following settings that you want:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Instance Identifier</td>
<td>You can rename the DB instance by typing a new name.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>In the list provided, click the version of the MySQL database engine that you want to use.</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>In the list provided, click the DB instance class that you want to use. For information about instance classes, see the section called “DB Instance Class” (p. 47).</td>
</tr>
<tr>
<td>Multi-AZ Deployment</td>
<td>If you want to deploy your DB instance in multiple Availability Zones, click Yes; otherwise, click No.</td>
</tr>
<tr>
<td>Auto Minor Version Upgrade</td>
<td>If you want your DB instance to receive minor engine version upgrades automatically when they become available, click Yes. Upgrades are installed only during your scheduled maintenance window.</td>
</tr>
<tr>
<td>Allocated Storage</td>
<td>Specify how much storage, in gigabytes, to allocate for your DB instance. The minimum allowable value is 5 GB; the maximum is 3072 GB. Note that you can only increase the amount of storage when modifying a DB instance, you cannot reduce the amount of storage allocated.</td>
</tr>
</tbody>
</table>
### Setting | Description
---|---
Provisioned IOPS | Select the **Use Provisioned IOPS** check box and enter a value in the **Provisioned IOPS** text box if you want your DB instance to use Provisioned IOPS. For more information on Provisioned IOPS, see Working with Provisioned IOPS Storage (p. 361).
Parameter Group | Select the parameter group you want associated with the DB instance. For more information about parameter groups, see Working with DB Parameter Groups (p. 317).
Security Groups | Select the security group you want associated with the DB instance. For more information about security groups, see Working with DB Security Groups (p. 329).
Option Group | Select the option group you want associated with the DB instance. For more information about option groups, see Working with Option Groups (p. 306).
Master User Password | Type a password for your master user. The password must contain from 8 to 128 alphanumeric characters.
Backup Retention Period | Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0.
Backup Window | Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC) and a duration in hours.
Maintenance Window | Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.

5. To apply the changes immediately, select the **Apply Immediately** check box. Selecting this option can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 45).

6. When all the changes are as you want them, click **Yes, Modify**. If instead you want to cancel any changes that you didn’t apply in the previous step, click **Cancel**.

### CLI

**To modify a MySQL DB instance**

- Use the command `rds-modify-db-instance`.

### API

**To modify a MySQL DB instance**

- Use the `ModifyDBInstance` action.
Importing and Exporting Data From an Amazon RDS MySQL DB Instance

This section presents recommended procedures for importing data into or exporting it from an Amazon RDS MySQL DB instance. The procedures can be used to import data from other Amazon RDS MySQL DB instances, MySQL instances running external to Amazon RDS, and other types of data sources. The section contains a recommended procedure for using replication to export data to an instance of MySQL that is running external to Amazon RDS.

Overview

The following are several recommended procedures for importing data into an Amazon RDS MySQL DB instance:

- To import data from an existing database in an Amazon RDS MySQL DB instance, you can create a read replica, and then promote the read replica. For more information, see Working with Read Replicas (p. 109).
- To import data from an existing MySQL database without using replication or an online conversion, you can use the MySQL mysqldump utility to import the data from a point of time. For more information, see Importing Offline Data From Existing MySQL Databases (p. 98).
- For data in an existing MySQL database that is running external to Amazon RDS, you can use MySQL replication to import the data. For more information, see Importing Data From a MySQL Instance Running External to Amazon RDS (p. 99).
- For large amounts of data in an existing Amazon RDS MySQL DB instance, you can use MySQL replication to export the data to an instance of MySQL running external to Amazon RDS. For more information, see Using Replication to Export Amazon RDS MySQL 5.6 Data (p. 105).

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 error and may render the DB instance and your data inaccessible. If this occurs, the DB instance can be restored from a snapshot using rds-restore-db-instance-from-db-snapshot or recovered using rds-restore-db-instance-to-point-in-time.

Importing Data Considerations

This section contains additional technical information related to loading data into MySQL. It is intended for advanced users who are familiar with the MySQL server architecture. Note that all comments related to LOAD DATA INFILE apply to mysqlimport as well.

Binary Log

Data loads incur a performance penalty and require additional free disk space (up to 4X 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. Depending upon the upload rate, other database activity taking place during the load, and the capacity of your Amazon RDS DB instance, this can severely degrade performance for other database sessions and increase the time required to load the data.

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, 10GB of data loaded as a single transaction will consume at least 30GB disk space during the load: 10GB for the table + 10GB for the binary log cache + 10GB 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 may be some time before the extra 20GB 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 prior to the load. During recovery, MySQL extracts the data from the binary log into a flat file and then executes LOAD DATA LOCAL INFILE, just as the original transaction, only this time the input file is local to the database server. Continuing with the example above, recovery will fail unless there is at least 40GB 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, it's recommended that you take a DB Snapshot of the database instance immediately before the load so that you can quickly and easily undo changes made during loading if the need arises.

After the load, set the backup retention period back to an appropriate (no zero) value.

You cannot 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 as they scan through undo that could have been purged if not for the long running load transaction.

Since undo is stored in the system tablespace and since the system tablespace never shrinks in size, large data load transactions can cause the system tablespace to become quite large, consuming disk space that cannot be reclaimed 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 may 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 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 may 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 may save time.
- Throttle the load rate - Data load impacting other processes? Throttle the load by increasing the interval between files.

Be Careful

The advantages of LOAD DATA INFILE diminish rapidly as transaction size increases. If breaking up a large set of data into smaller ones isn't an option, SQL may 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 will require modification or replacement before the load can resume. The alternative is to restore to the point in time prior to the load and replay the file once 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 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 prior to 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% and data file size cut in half.
- Disable foreign key constraints foreign_key_checks=0 For flat files loaded with LOAD DATA INFILE, this is required in many cases. For any load, disabling FK checks will provide 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

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.

Importing Offline Data From Existing MySQL Databases

The simplest way to import data from an existing MySQL database to an Amazon RDS MySQL DB instance without using replication or an online conversion, is to extract the data with mysqldump and pipe it directly into the Amazon RDS MySQL DB instance. mysqldump is a command line utility included with the MySQL client software distribution that's commonly used to make backups and transfer data from one MySQL server to another. Here's an example showing the "acme" database being copied to Amazon RDS.
We use the --host and --user options to specify the hostname and username used to connect to our DB instance. The --password option prompts for the password. There are other options that allow mysqldump to do some very useful things, such as:

- Sort each table's data by its primary key (--order-by-primary). This can dramatically reduce load times.
- Compress the data before sending it to Amazon RDS (--compress). This option can reduce network bandwidth consumption.
- Ensure that all of the data is consistent with a single point in time (--single-transaction). If there are other processes changing the data while mysqldump is reading it, use this option to maintain data integrity.

For information about mysqldump, go to mysqldump - A Database Backup Program in the MySQL documentation.

### Importing Data From a MySQL Instance Running External to Amazon RDS

To import data from a MySQL database that is running external to Amazon RDS, you can configure replication from that database to an Amazon RDS MySQL DB instance. The external MySQL database you are importing from can be running either on-premises in your data center, or in an Amazon EC2 instance. The Amazon RDS MySQL DB instance must be running either version 5.5.33 or 5.6.13. Using replication to import the data reduces downtime.

Replication from a MySQL instance running external to Amazon RDS is only supported during the time it takes to import a database to an Amazon RDS MySQL DB instance. The replication should be terminated when the data has been transferred and all applications and services have been cut over to use the Amazon RDS instance.

The following list shows the steps to take. Each step is discussed in more detail in later sections.

1. Prepare an Amazon RDS MySQL DB instance that is running either version 5.5.33 or 5.6.13; configure it to be a read replica of the MySQL instance running external to Amazon RDS.
2. Configure the MySQL instance running external to Amazon RDS to be the replication source.
3. Use `mysqldump` to import the database from the MySQL instance running external to Amazon RDS into the Amazon RDS instance.
4. Start replication to the Amazon RDS MySQL DB instance to capture updates made since the mysqldump files were created. Because the source MySQL instance is not in Amazon RDS, you do not use the standard Amazon RDS process to create a replica, you instead use stored procedures to establish replication.
5. After the import completes, point the applications to the Amazon RDS MySQL DB instance. Stop replication by running stored procedures that terminate the replication process. After the stored procedures are run, the replica runs as a standard Amazon RDS MySQL DB instance without any need to promote the instance.

### Prepare an Amazon RDS MySQL DB Instance

Determine which DB instance class is required to support both the import and production workloads. For more information see DB Instance Class (p. 47).
Determine what configuration options are required to support the workloads. If no existing Amazon RDS MySQL parameter group has that configuration, create a new parameter group. For more information see Working with DB Parameter Groups (p. 317).

Determine if Amazon RDS Provisioned IOPS is required to support the workloads. For more information, see Working with Provisioned IOPS Storage (p. 361).

Create an Amazon RDS MySQL DB instance by specifying the correct DB instance class, parameter group, security group, Provisioned IOPS settings, and a single Availability Zone. Select either version 5.5.33 or 5.6.13. For more information, see Creating a DB Instance Running the MySQL Database Engine (p. 82). Do not configure multiple Availability Zones until after the import has completed. We recommend that you also do not create read replicas until the import has completed.

Connect to the instance as the master user, and create the users required to support the administrators, applications, and services that will need to access the instance.

By default, an Amazon RDS MySQL DB instance has the egress rules required for it to connect as a read replica to an instance of MySQL running external to Amazon RDS. If you have changed the security group for the Amazon RDS MySQL DB instance, you may need to configure an egress rule for the Amazon RDS instance to operate as the read replica during the import. Specify an egress rule that allows TCP connections to the port and IP address of the external MySQL source instance. If the read replica is running as an Amazon RDS DB instance in an Amazon VPC, specify the egress rule in a VPC security group. If the read replica is running as an Amazon RDS DB instance that is not in a VPC, specify the egress rule in a database security group.

If you configure an egress rule in a VPC security group, you must also configure these VPC ACL rules. For more information about Amazon VPC network ACLs, go to Network ACLs.

- ACL ingress rule allowing TCP traffic to ports 1024-65535 from the IP address of the MySQL instance.
- ACL egress rule: allowing outbound TCP traffic to the port and IP address of the MySQL instance.

**Prepare the Replication Source**

Follow the directions in the MySQL documentation to prepare the instance of MySQL running external to Amazon RDS as a replication master.

Set the configuration by following the directions in Setting the Replication Master Configuration.

Create a replication account by following the directions in Creating a User For Replication.

Use the MySQL `SHOW MASTER STATUS` statement to record the location of the MySQL binary log. For more information, go to Obtaining the Replication Master Log Binary Coordinates in the MySQL documentation.

Configure ingress rules on the system running the external MySQL source instance that will allow the Amazon RDS read replica to connect during replication. Specify an ingress rule that allows TCP connections to the port used by the MySQL instance from the IP address of the Amazon RDS read replica.

To find the IP address of the Amazon RDS MySQL DB instance, use ping and specify the DNS name of the DB instance. The DNS name is that part of the endpoint before the colon (:) and port number. For example, if the endpoint of an Amazon RDS MySQL DB instance is myinstance.d7c91mxfgb0.us-west-2.rds.amazonaws.com:3306, use the following to get the IP address (on Windows, use -n 5 instead of -c 5):

```
ping myinstance.d7c91mxfgb0.us-west-2.rds.amazonaws.com -c 5
```

If the external MySQL instance is running in an Amazon EC2 instance in a VPC, specify the ingress rules in a VPC security group. If the MySQL instance is running in an Amazon EC2 instance that is not in a VPC, specify the ingress rules in a database security group.
VPC, specify the ingress rules in a security group associated with the Amazon EC2 instance. If the MySQL instance is running on-premises, specify the ingress rules in a firewall.

If the external MySQL instance is running on Amazon EC2 in a VPC, configure VPC ACL rules in addition to the security group ingress rules. For more information about Amazon VPC network ACLs, go to Network ACLs.

- ACL ingress rule: allow TCP connections to the port used by the external MySQL instance from the CIDR range of the Amazon RDS read replica subnet.
- ACL egress rule: allow TCP connections from ports 1024-65535 to the Amazon RDS read replica subnet.

Take a snapshot of the databases to be replicated by running the `mysqldump` utility. Do not include the MySQL system database in the snapshot. Follow the directions in Creating a Dump Snapshot Using `mysqldump`, except do not use `--all-databases` as shown in the example, instead use the `--databases` parameter to specify the databases to dump. The source instance will be in read-only mode during this operation. Do not purge these binary logs until they have been applied to the read replica.

`mysqldump` supports multiple ways to dump the data from a database. Since the source system is in read-only mode while `mysqldump` extracts the data, choose a method that will minimize the amount of time that `mysqldump` runs. For more information about choosing a `mysqldump` format and then loading the data, go to Using `mysqldump` For Backups.

### Start Replication

Load the database dumps from the external MySQL instance into the Amazon RDS MySQL DB instance by using the appropriate method for the dump format you chose when using `mysqldump`.

Run the `mysql.rds_set_external_master` procedure on the Amazon RDS MySQL DB instance to configure it as a read replica of the MySQL instance running external to Amazon RDS. Specify the connection information for connecting to the MySQL instance running external to Amazon RDS, the replication user credentials, the binary log location, and whether to use SSL encryption. For more information, see `mysql.rds_set_external_master` (p. 127).

Run the `mysql.rds_start_replication` procedure on the Amazon RDS instance to start the replication process. For more information, see `mysql.rds_start_replication` (p. 130).

The replication process may stop if some event changes the IP address of one of the instances, such as a Multi-AZ instance failing over to the secondary instance. If this happens, you may need to reconfigure your ingress or egress rules to reflect the new IP address. If the instance is in a VPC subnet, you can minimize the need to change the rule by specifying the CIDR range of the subnet.

### Stop Replication

After the databases have been replicated and you have verified the databases are synchronized, change all applications and services to connect to the Amazon RDS MySQL DB instance.

Run the `mysql.rds_stop_replication` procedure to stop the replication process. For more information, see `mysql.rds_stop_replication` (p. 130).

Verify that the applications and services are running correctly.
Run the `mysql.rds_reset_external_master` procedure to remove the replication configuration. For more information, see `mysql.rds_reset_external_master` (p. 129).

Take a snapshot of the Amazon RDS instance.

Modify the security group to remove the entry for the instance of MySQL that was not running on Amazon RDS.

Advise your network administrators they can remove the firewall rule that authorized the replication connection.

Configure the Amazon RDS instance to run in the configuration environment, such as by enabling multiple Availability Zones or creating read replicas from the instance.

Monitor the performance of the instance after it starts running the production load. If necessary, you can resize and reconfigure the instance if it is not supporting the production load. For more information, see Monitoring a DB Instance (p. 376).

### Importing Data From Any Source to a MySQL DB Instance

If you have more than 1GB of data to load, or if your data is coming from somewhere other than a MySQL database, we recommend creating flat files and loading them with `mysqlimport`. `mysqlimport` is another command line utility bundled with the MySQL client software whose purpose is to load flat files into MySQL. For information about `mysqlimport`, go to `mysqlimport - 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 allows you restore the database to its state prior to the load, should the need arise. 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 `mysqlimport`.
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 1GB, 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 1GB, splitting only at line breaks (-C 1024m). The new files will be named sales.part_00, sales.part_01, etc.

```
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 will be loading to (particularly if other sessions will be modifying the tables being loaded or tables they reference). This will reduce the risk of constraint violations occurring during the load, improve load performance, and make it possible to restore the database instance to the point just prior to the load without losing changes made by processes not involved in the load.

Of course, this may not be possible or practical. If you are unable to stop applications from accessing the DB instance prior to 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 will be loading data into a new DB instance that contains no data, you may skip this step. Otherwise, creating a DB Snapshot of your DB instance will allow you to restore the database instance to the point just prior to 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 rds-create-db-snapshot command to create a DB Snapshot of our AcmeRDS instance and give the DB Snapshot the identifier "preload".

```
rds-create-db-snapshot 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.

It is important to keep in mind that restoring a database from a DB Snapshot creates a new DB instance which, 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.
To reuse the existing endpoint, we must first delete the database instance and then give the restored database the same identifier:

```
# Delete the database instance
rds-delete-db-instance AcmeRDS --final-db-snapshot-identifier AcmeRDS-Final

# Restore the database with the same identifier
rds-restore-db-instance-from-db-snapshot AcmeRDS --db-snapshot-identifier=preload
```

Note that 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 TO RETAIN THE ABILITY TO PERFORM POINT-IN-TIME RECOVERY.** Disabling automated backups erases all existing backups, so point-in-time recovery will not be possible after automated backups have been disabled. Disabling automated backups is a performance optimization and is not required for data loads. Note that DB Snapshots are not affected by disabling automated backups. All existing DB Snapshots are still available for restore.

Disabling automated backups will reduce load time by about 25% and reduce the amount of storage space required during the load. If you will be loading 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 will be loading into a DB instance that already contains data, you must 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 logging on or off.

Use the `rds-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.

```
# Disable automated backups
rds-modify-db-instance AcmeRDS --apply-immediately --backup-retention-period=0
```

You can check the status of your DB instance with the `rds-describe-db-instances` command. The example displays the status of the AcmeRDS database instance and includes the `--headers` option to show column headings.

```
# Display DB instance status
rds-describe-db-instances 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, MySQL will look for the data on the database host, so always specify the --local option.

```bash
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**

Once 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 will restart the DB instance, so be prepared for a brief outage.

In the example, we use the rds-modify-db-instance command to enable automated backups for the AcmeRDS DB instance and set the retention period to 1 day.

```bash
rds-modify-db-instance AcmeRDS --apply-immediately --backup-retention-period=1
```

**Using Replication to Export Amazon RDS MySQL 5.6 Data**

You can use replication to export data from an Amazon RDS MySQL 5.6 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 Amazon RDS 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 higher 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 an Amazon RDS 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 Amazon RDS 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 replication slave, or read replica. For more information, go to 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 Amazon RDS 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, go to Network ACLs.

- ACL ingress rule allowing TCP traffic to ports 1024-65535 from the IP address of the source Amazon RDS MySQL DB instance.
- ACL egress rule: allowing outbound TCP traffic to the port and IP address of the source Amazon RDS MySQL DB instance.

**Prepare the Replication Source**

Prepare the Amazon RDS 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 Amazon RDS 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, go to 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 Automated Backups (p. 293).

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 5.6 Binary Logs (p. 399).

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. 113). 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 against the MySQL instance running external to Amazon RDS, and note the master_host, master_port, master_log_file, and read_master_log_pos values. 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.

```
mysqldump -h RDS instance endpoint -u user -p password --port 3306 --single-transaction --routines --triggers --databases database database2 --compress --compact | /rdsdbbin/mysql/bin/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.

```
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 Amazon RDS MySQL source 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, master_log_file, and read_master_log_pos values you got from the Mysql SHOW SLAVE STATUS statement you ran on the RDS read replica. For more information, go to Setting the Master Configuration on the Slave.

Use the MySQL START SLAVE command to initiate replication from the Amazon RDS source 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, go to [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 Amazon RDS source 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 Amazon RDS source instance.
Working with Read Replicas

Topics
- Replicating Across Regions (p. 110)
- Creating a Read Replica (p. 113)
- Promoting a Read Replica to be a DB Instance (p. 114)
- Monitoring Read Replication (p. 115)
- Troubleshooting a Read Replica Problem (p. 116)

Amazon RDS uses MySQL's 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 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. Read replicas allow you to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.

Note
The information in this topic applies to creating Amazon RDS read replicas, either in the same region as the source Amazon RDS DB instance, or in a separate AWS region. The topic does not apply to setting up replication with an instance of MySQL that is running either in an Amazon EC2 instance or on-premises. Amazon RDS currently supports those replication configurations only for the time needed to import or export large databases to or from DB instances of MySQL 5.6. For more information, see Importing and Exporting Data From an Amazon RDS MySQL DB Instance (p. 95).

When you create a read replica, you specify an existing DB instance as the source. 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 that is native to MySQL 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 can connect to a read replica the same way they would any DB instance. Amazon RDS replicates all databases in the source DB instance. For more information about read replicas, see Read Replicas (p. 58).

Before a 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 MySQL 5.6, not 5.1 or 5.5.

You can create up to 5 read replicas from one DB instance. In order for replication to operate effectively, each read replica should have as much 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 MySQL 5.6, 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 cannot have more than three instances involved in a replication chain. For example, you can create ReadReplica1 from MySourceDBInstance, and then create ReadReplica2 from ReadReplica1, but you cannot create a ReadReplica3 from ReadReplica2.

Amazon RDS does not support circular replication. You cannot configure a DB instance to serve as a replication source to 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 cannot configure ReadReplica1 to replicate back to MyDBInstance. From ReadReplica1, you can only create a new read replica, such as ReadReplica2.
Read replicas are designed to support read queries, but there may be a need for occasional updates, such as adding 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.

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 5 read replicas for the source instance.

You can create a read replica from either Single-AZ or Multi-AZ DB instance deployments. You use a Multi-AZ deployment to improve the durability and availability of a critical system, but you cannot use the Multi-AZ secondary to serve read-only queries. You must create read replicas from a high-traffic, Multi-AZ DB instance to offload read queries from the source DB instance. If the source instance of a Multi-AZ deployment fails over to the secondary, any associated read replicas will be switched to use the secondary as their replication source. It is possible that the read replicas cannot be switched to the secondary if some MySQL binlog events are not flushed during the failure. In this case, you must manually delete and recreate the read replicas. You can reduce the chance of this happening in MySQL 5.1 or 5.5 by setting the sync-binlog=1 and innodb-xa-support=1 dynamic variables. These settings may reduce performance, so test their impact before implementing the changes to a production environment. These problems are less likely to occur if you are using MySQL 5.6. For instances running MySQL 5.6, the parameters are set by default to sync-binlog=1 and innodb-xa-support=1.

When you create a read replica, Amazon RDS sets up a secure communications channel using public key encryption between the source DB instance and the read replica. RDS establishes any AWS security configurations, such as adding security group entries, needed to enable the secure channel.

You usually configure replication between Amazon RDS DB instances, but you can configure replication to import databases from instances of MySQL running outside of Amazon RDS, or to export databases to such instances. For more information, see Importing Data From a MySQL Instance Running External to Amazon RDS (p. 99) and Using Replication to Export Amazon RDS MySQL 5.6 Data (p. 105).

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. 130) and mysql.rds_start_replication (p. 130). 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 From a MySQL Instance Running External to Amazon RDS (p. 99) and Using Replication to Export Amazon RDS MySQL 5.6 Data (p. 105).

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, Single-AZ DB instance.

## Replicating Across Regions

Replicating from one region to another includes benefits such as improving your disaster recovery capabilities, scaling read operations into a region closer to end users, or making it easier to migrate from a data center in one region to a data center in another region. Creating a read replica in a different region than the source instance is very similar to creating a replica in the same region. You run the create read replica command in the region where you want the read replica, and specify the Amazon Resource Name (ARN) of the source DB instance.

## Cross Region Replication Considerations

All of the considerations for performing replication within a 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 MySQL 5.6.
- You can only cross one regional boundary in a given replication chain. You can 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.

An Amazon RDS DB instance that is a read replica of an on-premises or Amazon EC2 instance of MySQL that is not in Amazon RDS.

You cannot set up a replication channel into or out of the AWS GovCloud (US) Region.

You should expect to see some higher level of lag time for any read replica that is in a different region than the source instance, due to the longer network channels between regional data centers.

Within a region, all cross-region replicas created from the same source DB instance must either be in the same Amazon VPC or be outside of a VPC. For those 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 an VPC from a source DB instance that is not in an VPC. You can also create a cross-region read replica that is not in an VPC from a source DB instance that is in a VPC.

**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 region:

- When you create the 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 region to the read replica region.

For more information about Amazon RDS data transfer pricing, go to Amazon Relational Database Service Pricing.

You can reduce your data transfer costs by reducing the number of cross region read replicas you create. For example, if you have a source DB instance in one region and want to have three read replicas in another region, only create one of the read replicas from the source DB instance, and then create the other two replicas from the first read replica instead of the source. For example, if you have `source-instance-1` in one region, you can:

- Create `read-replica-1` in the new 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 will only be charged for the data transferred from `source-instance-1` to `read-replica-1`. You will not be charged for the data transferred from `read-replica-1` to the other two replicas because they are all in the same region. If you created all three replicas directly from `source-instance-1`, you would be charged for the data transfers to all three replicas.
Examples

Example Create Cross Region Read Replica Outside of any VPC

This is an example of creating 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:

```
```

Example Create Cross Region Read Replica in a VPC

This is an example of creating 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:

```
```

Cross Region Replication Process

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 could 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 region and sets the status to creating.
3. Amazon RDS creates an automated DB snapshot of the source DB instance in the source 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 this phase, 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 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 will be 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 replication and replicates 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 will be greater than 0. You can monitor this in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric. When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance.

## Creating a Read Replica

You can create a read replica from an existing MySQL DB instance using the AWS Management Console, CLI, or 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 initiate the creation of a read replica, Amazon RDS takes a DB snapshot of your source DB instance and begins replication. As a result, you will experience a brief I/O suspension on your source DB instance as the DB snapshot occurs. The I/O suspension typically lasts about one minute and can be avoided if the source DB instance is a Multi-AZ deployment (in the case of Multi-AZ deployments, DB snapshots are taken from the standby). An active, long-running transaction can slow the process of creating the read replica, so 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. Automatic backups are supported only for read replicas running MySQL 5.6, not 5.1 or 5.5.

Second, if you are using a non-transactional engine such as MyISAM, you will need to perform the following steps to successfully set up your read replica. These steps are required to ensure that the read replica has a consistent copy of your data. Note that these steps are not required if all of your tables use a transactional engine such as InnoDB.

1. Stop all DML and 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.

## AWS Management Console

To create a read replica from a source 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, click DB Instances.
3. In the My DB Instances pane, right click the MySQL DB instance that you want to use as the source for a read replica and select Create Read Replica.
4. In the DB Instance Identifier text box, type a name for the read replica. Adjust other settings as needed.
5. In the Destination Region box, specify the region for the read replica if it is different than the region of the source DB instance.
6. In the Destination DB Subnet Group box, specify a DB subnet group associated with a VPC if you want the read replica to be created in that VPC. Leave the box empty if you want the read replica to
be created outside of any VPC. The VPC and DB subnet group must exist in the destination region. Within a given region, all read replicas created from the same source DB instance must be either:

- All created in the same VPC.
- All created outside of any VPC.

7. Click **Yes, Create Read Replica**.

**CLI**

To create a read replica from a source MySQL DB instance

- Use the `rds-create-db-instance-read-replica` command.

**API**

To create a read replica from a source MySQL DB instance

- Call `CreateDBInstanceReadReplica`.

**Promoting a Read Replica to be a DB Instance**

You can promote a MySQL read replica into a standalone, Single-AZ DB instance. There are several reasons you might want to convert a read replica into a Single-AZ DB instance. It is important to remember that when promoting a read replica, the instance will be rebooted before it becomes available.

Some DDL operations, such as creating indexes or other maintenance tasks, can be a lengthy process, and some operations can affect performance of a DB instance. You can perform these operations on a read replica, instead. When the DDL operations are complete and the read replica is updated by the source DB instance, you can promote the read replica. You can also promote a read replica as part of a disaster recovery plan.

The new DB instance that is created when you promote a read replica retains the backup retention period, backup window period, and 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 into a Single-AZ DB instance, it is just like any other Single-AZ DB instance. For example, you can convert the new DB instance into a Multi-AZ DB instance, and you can create read replicas from it. You can also take DB snapshots and perform point-in-time restore operations. Because the promoted DB instance is no longer a read replica, you cannot 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.

If you promote a read replica that is in turn replicating to other read replicas, those replications stay active. Consider an example where MyDBInstance1 replicates to MyDBInstance2, and MyDBInstance2 replicates to MyDBInstance3. If you promote MyDBInstance2, there will no longer be any replication from MyDBInstance1 to MyDBInstance2, but MyDBInstance2 will still replicate to MyDBInstance3.

The following steps show the general process for promoting a read replica to a Single-AZ 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. To be able to make changes to the read replica, you must set the read_only parameter to 0 in the DB parameter group for the read replica.

3. Perform all needed DDL operations, such as creating indexes, on the read replica. Actions taken on the read replica do not affect the performance of the source DB instance.

4. Promote the read replica by using the Promote Read Replica option on the Amazon RDS console, the CLI command rds-promote-read-replica, or the PromoteReadReplica 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 Single-AZ DB instance.

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, click Read Replicas.
3. In the Read Replicas pane, select the check box beside the read replica that you want to promote.
4. Click Promote Read Replica.
5. In the Promote Read Replica dialog box, enter the backup retention period and the backup window for the new promoted DB instance.
6. When the settings are as you want them, click Continue.
7. On the acknowledgment page, click Yes, Promote.

CLI

To promote a read replica to a DB instance

• Use the rds-promote-read-replica command.

API

To promote a read replica to a DB instance

• Call PromoteReadReplica.

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; you can also see the status of a read replica using the CLI command rds-describe-db-instances or the API action DescribeDBInstances.
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 Read Replica Problem (p. 116).

• **Stopped**—Replication has stopped because of a customer initiated request.

• **Terminated**—The read replica has lagged the source DB instance for more than 30 days due to replication errors and is terminated. The read replica is still accessible for read operations but cannot synchronize with the source instance.

If replication errors occur in a read replica for 30 days, replication is terminated to prevent increased storage requirements and long failover times. Broken replication can effect storage because the binlogs can grow in size and number due to the high volume of errors messages being written to the binlog. Broken replication can also effect failure recovery due to the time Amazon RDS requires to maintain and process the large number of binlogs during recovery.

You can monitor how far a replica is lagging the source DB instance by viewing the Seconds_Behind_Master data returned by the MySQL Show Slave Status command, or the CloudWatch Replica Lag statistic. If a replica lags too far behind for your environment, consider deleting and recreating the read replica. Also consider increasing the scale of the read replica to speed replication.

## Troubleshooting a Read Replica Problem

MySQL’s read replication technology is asynchronous. Because it is 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 writes to the source DB instance can occur in parallel, while writes 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 in the MySQL documentation, see Replication Implementation Details.

There are several things you can do to reduce the lag between updates to a source DB instance and the subsequent updates to the read replica, such as:

- 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 \texttt{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, such as DML queries being run on your read replica that 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 \texttt{RDS-EVENT-0045} (p. 383), \texttt{RDS-EVENT-0046} (p. 383), and \texttt{RDS-EVENT-0047} (p. 383). For more information about events and subscribing to events, see Using Amazon RDS Event Notification (p. 380). If the 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 \texttt{max\_allowed\_packet} parameter for a read replica is less than the \texttt{max\_allowed\_packet} parameter for the source DB instance. The \texttt{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 that can be executed on the database. If the \texttt{max\_allowed\_packet} parameter value in the DB parameter group associated with a source DB instance is smaller than the \texttt{max\_allowed\_packet} parameter value in the DB parameter group associated with the source's read replica, 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 \texttt{max\_allowed\_packet} parameter values.

Other common situations that can cause replication errors include:

- Writing to tables on a read replica. If you are creating indexes on a read replica, you need to have the \texttt{read\_only} parameter set to 0 to create the indexes. If you are writing to tables on the read replica, it may 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.
- Using unsafe non-deterministic queries such as \texttt{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. 118). Otherwise, you can delete the read replica and create a 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 \texttt{replicating}.
Appendix: Common DBA Tasks for MySQL

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 Working with MySQL Database Log Files (p. 397)

Topics
- Killing a Session or Query (p. 118)
- Skipping the Current Replication Error (p. 118)
- Working with InnoDB Tablespaces to Improve Crash Recovery Times (p. 119)
- Managing the Global Status History (p. 121)

Killing a Session or Query

To terminate user sessions or queries on DB instances, Amazon RDS provides the following commands:

```
PROMPT> CALL mysql.rds_kill(thread-ID)
PROMPT> CALL mysql.rds_kill_query(thread-ID)
```

For example, to kill the session that is running on thread 99, you would type the following:

```
PROMPT> CALL mysql.rds_kill(99);
```

To kill the query that is running on thread 99, you would type the following:

```
PROMPT> 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.

**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. 131)`.

## 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 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.
You can move an InnoDB table's metadata from its own tablespace to the shared tablespace by using the following command, which will rebuild the table metadata according to the `innodb_file_per_table` parameter setting.

```sql
PROMPT>ALTER TABLE name ENGINE = InnoDB
```

For example, the following query returns an `ALTER TABLE` statement for every InnoDB table.

```sql
SELECT CONCAT('ALTER TABLE `', REPLACE(TABLE_SCHEMA, '`', '``'), '`.`', REPLACE(TABLE_NAME, '`', '``'), '` ENGINE=InnoDB;')
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE'
AND ENGINE = 'InnoDB' AND TABLE_SCHEMA <> 'mysql';
```

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 RDS 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 RDS 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 5.1.62 and 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. 317).

You can then use the procedures in the following table to enable and configure GoSH. For each procedure, at the command prompt, type the following:

```
PROMPT> CALL procedure-name;
```

Where `procedure-name` is one of the procedures in the table.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rds_enable_gsh_collector</td>
<td>Enables GoSH to take default snapshots at intervals specified by <code>rds_set_gsh_collector</code>.</td>
</tr>
<tr>
<td>rds_set_gsh_collector</td>
<td>Specifies the interval, in minutes, between snapshots. Default value is 5.</td>
</tr>
<tr>
<td>rds_disable_gsh_collector</td>
<td>Disables snapshots.</td>
</tr>
<tr>
<td>rds_collect_global_status_history</td>
<td>Takes a snapshot on demand.</td>
</tr>
<tr>
<td>rds_enable_gsh_rotation</td>
<td>Enables rotation of the contents of the <code>mysql.global_status_history</code> table to <code>mysql.global_status_history_old</code> at intervals specified by <code>rds_set_gsh_rotation</code>.</td>
</tr>
<tr>
<td>rds_set_gsh_rotation</td>
<td>Specifies the interval, in days, between table rotations. Default value is 7.</td>
</tr>
<tr>
<td>rds_disable_gsh_rotation</td>
<td>Disables table rotation.</td>
</tr>
<tr>
<td>rds_rotate_global_status_history</td>
<td>Rotates the contents of the <code>mysql.global_status_history</code> table to <code>mysql.global_status_history_old</code> on demand.</td>
</tr>
</tbody>
</table>

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 rds_global_status_history as a join 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'
Appendix: Options for MySQL DB Engine

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 options, see Things You Should Know About Option Groups (p. 306).

The following option is currently supported for MySQL 5.6:

- **MEMCACHED**

MySQL 5.6 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.

Memcached is a distributed memory cache service, where a web service implements a key-value cache. Applications use a memcached network protocol to insert, manipulate, and retrieve key-value data pairs from the cache. MySQL 5.6 includes 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, go to InnoDB Integration with memcached.

You enable memcached support for an Amazon RDS MySQL 5.6 instance by:

1. Determining 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 Amazon RDS Security Groups (p. 74).

2. Creating a custom DB option group, selecting MySQL as the engine type and a 5.6 version. For more information about creating an option group, see Creating an Option Group (p. 308).

3. Adding 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. 309).

4. Modifying 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. 313).

5. Applying 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. 82).
   - 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. 93).

6. Specifying which columns in your MySQL tables can be accessed through the memcached interface. The memcached plugin 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, go to Internals of the InnoDB memcached Plugin. For an example of mapping an InnoDB table and accessing it through memcached, go to 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 Amazon RDS Security Groups (p. 74).

You turn off the memcached support for an instance by modifying the instance and specifying the MySQL 5.6 default option group. For more information about modifying a MySQL instance, see Modifying a DB Instance Running the MySQL Database Engine (p. 93).

### MySQL memcached Security Considerations

The memcached protocol does not support user authentication. For more information about MySQL memcached security considerations, go to [memcached Deployment](https://memcached.org/) and [Using memcached as a MySQL Caching Layer](https://dev.mysql.com/doc/mysql-index.html).

You can take the following actions to help increase the security of the memcache 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 Amazon RDS Security Groups (p. 74).

### 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.cg034hpkmmjt.region.rds.amazonaws.com` and the memcached interface is using port 11212, the connection information specified in PHP would be:

```php
<?php
$cache = new Memcache;
$cache->connect('my-cache-instance.cg034hpkmmjt.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/](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, click **Instances**.
4. Select the arrow to the left of name of the DB Instance running the MySQL database engine. In the description display, note the value of the `endpoint` field. The DNS name is the part of the endpoint up to the semicolon (:). Ignore the semicolon and the port number after the semicolon, that port is not used to access the memcached interface.
5. Note the name listed in the **Option Group(s)** field.
6. In the navigation pane, click **Option Groups**.
7. Select the arrow to the left of the name of the option group used by the MySQL DB instance. In the description display, note the value of the `port` setting in 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, 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, when set to 1 (true) it specifies that memcached will return an error rather than evicting items when there is no more memory to store items. If set to 0
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** - an 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_MEMCAHCED_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.
Appendix: MySQL 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 to manage replication.

- mysql.rds_set_external_master (p. 127)
- mysql.rds_reset_external_master (p. 129)
- mysql.rds_start_replication (p. 130)
- mysql.rds_stop_replication (p. 130)
- mysql_rds_skip_repl_error (p. 131)
- mysql.rds_next_master_log (p. 132)

SQL Reference Conventions

This section explains the conventions that are used to describe the syntax of the system stored procedures and tables described in the SQL reference section.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPERCASE</td>
<td>Words in uppercase are key words.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Square brackets indicate optional arguments.</td>
</tr>
<tr>
<td>{ }</td>
<td>Braces indicate that you are required to choose one of the arguments inside the braces.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>italics</td>
<td>Words in italics indicate placeholders. You must insert the appropriate value in place of the word in italics.</td>
</tr>
<tr>
<td>...</td>
<td>An ellipsis indicates that you can repeat the preceding element.</td>
</tr>
<tr>
<td>'</td>
<td>Words in single quotes indicate that you must type the quotes.</td>
</tr>
</tbody>
</table>

mysql.rds_set_external_master

Configures an Amazon RDS MySQL DB instance to be a read replica of an instance of MySQL running external to Amazon RDS.

Syntax

CALL mysql.rds_set_external_master (host_name, host_port, replication_user_name, replication_user_password)
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 SLAVE permissions in the Amazon RDS MySQL DB instance to be configured as the read replica.

**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**
Specifies whether SSL encryption is used on the replication connection. 1 specifies to use SSL encryption, 0 specifies to not use encryption. Using SSL encryption is recommended. The default is 0.

Usage Notes

`mysql.rds_set_external_master` must be run by the master user. It must be run on the Amazon RDS MySQL DB instance to be configured as the read replica of a MySQL instance running external to Amazon RDS. Before running `mysql.rds_set_external_master`, you must have configured the instance of MySQL running external to Amazon RDS as a replication master. For more information, see Importing and Exporting Data From an Amazon RDS MySQL DB Instance (p. 95).

**Warning**
Do not use `mysql.rds_set_external_master` to manage replication between two Amazon RDS DB instances. Use it only when replicating with an instance of MySQL running external to RDS. For information about managing replication between Amazon RDS DB instances, see Working with Read Replicas (p. 109).

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. 130) to start the replication process. You can call `mysql.rds_reset_external_master` (p. 129) 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.

`mysql.rds_set_external_master` is available in these versions of Amazon RDS MySQL:

- 5.5.33
- 5.6.13
Examples

When run on an Amazon RDS 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('Sourcedb.some.com',3306,'ReplicationUser','SomePassW0rd','mysql-bin-changelog.0777',120,1);
```

Related Topics

- `mysql.rds_set_external_master` (p. 127)
- `mysql.rds_start_replication` (p. 130)
- `mysql.rds_stop_replication` (p. 130)

**mysql.rds_reset_external_master**

Reconfigures an Amazon RDS MySQL DB instance to no longer be a read replica of an instance of MySQL running external to Amazon RDS.

**Syntax**

```sql
CALL mysql.rds_reset_external_master;
```

**Usage notes**

`mysql.rds_reset_external_master` must be run by the master user. It must be run on the Amazon RDS MySQL DB instance to be removed as a read replica of a MySQL instance running external to Amazon RDS.

**Warning**

Do not use `mysql.rds_reset_external_master` to manage replication between two Amazon RDS DB instances. Use it only when replicating with an instance of MySQL running external to Amazon RDS. For information about managing replication between Amazon RDS DB instances, see *Working with Read Replicas* (p. 109).

For more information about using replication to import data from an instance of MySQL running external to Amazon RDS, see *Importing and Exporting Data From an Amazon RDS MySQL DB Instance* (p. 95).

`mysql.rds_reset_external_master` is available in these versions of Amazon RDS MySQL:

- 5.5.33
- 5.6.13

Related Topics

- `mysql.rds_set_external_master` (p. 127)
- `mysql.rds_start_replication` (p. 130)
- `mysql.rds_stop_replication` (p. 130)
mysql.rds_start_replication

Initiates replication from an Amazon RDS MySQL DB instance.

Syntax

CALL mysql.rds_start_replication;

Usage notes

`mysql.rds_start_replication` must be run by the master user.

If you are configuring replication to import data from an instance of MySQL running external to Amazon RDS, you call `mysql.rds_start_replication` to start the replication process after you have called `mysql.rds_set_external_master` to build the replication configuration. For more information, see Importing and Exporting Data From an Amazon RDS MySQL DB Instance (p. 95).

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` to control some replication actions, such as purging binary logs. For more information, see Using Replication to Export Amazon RDS MySQL 5.6 Data (p. 105).

You can also use `mysql.rds_start_replication` to restart any replication process that you previously stopped by calling `mysql.rds_stop_replication` (p. 130). For more information, see Working with Read Replicas (p. 109).

`mysql.rds_start_replication` is available in these versions of Amazon RDS MySQL:

- 5.5.33
- 5.6.13

Related Topics

- `mysql.rds_set_external_master` (p. 127)
- `mysql.rds_reset_external_master` (p. 129)
- `mysql.rds_stop_replication` (p. 130)

mysql.rds_stop_replication

Terminates replication from an Amazon RDS MySQL DB instance.

Syntax

CALL mysql.rds_stop_replication;

Usage notes

`mysql.rds_stop_replication` must be run by the master user.

If you are configuring replication to import data from an instance of MySQL running external to Amazon RDS, you call `mysql.rds_stop_replication` to stop the replication process after the import has completed.
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` to control some replication actions, such as purging binary logs. For more information, see Using Replication to Export Amazon RDS MySQL 5.6 Data (p. 105).

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 replica, such as creating a large index on the replica. You can restart any replication process that you stopped by calling `mysql.rds_start_replication` (p. 130). For more information, see Working with Read Replicas (p. 109).

`mysql.rds_stop_replication` is available in these versions of Amazon RDS MySQL:

- 5.5.33
- 5.6.13

### Related Topics

- `mysql.rds_set_external_master` (p. 127)
- `mysql.rds_reset_external_master` (p. 129)
- `mysql.rds_start_replication` (p. 130)

### `mysql_rds_skip_repl_error`

Skips and deletes a replication error on an Amazon RDS MySQL DB instance.

**Syntax**

```
CALL mysql.rds_skip_repl_error;
```

**Usage notes**

`mysql.rds_skip_repl_error` must be run by the master user.

Run the MySQL `show slave status\G` command to determine if there are errors. If a replication error is not critical, you can elect to use `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, go to SHOW SLAVE STATUS Syntax in the MySQL documentation.

For more information about addressing replication errors with Amazon RDS, see Troubleshooting a Read Replica Problem (p. 116).

`mysql.rds_skip_repl_error` is available in these versions of Amazon RDS MySQL:

- For MySQL 5.1, version 5.1.62 or later.
- For MySQL 5.5, version 5.5.23 or later.
- For MySQL 5.6, version 5.6.12 or later.
**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 an Amazon RDS read replica.

**Syntax**

```sql
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**

`mysql.rds_next_master_log` must be run by the master user.

**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](#).

`mysql.rds_next_master_log` is available in these versions of Amazon RDS MySQL:

- 5.1.71
- 5.5.33
- 5.6.13

**Examples**

Assume replication fails on an Amazon RDS read replica. Running `SHOW SLAVE STATUS\G` on the replica returns the following result:

```
*************************** 1. row ***************************
Slave_IO_State: Master_Host: myhost.XXXXXXXXXXXXXX.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
```

---

API Version 2013-09-09

132
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relay_Log_File</td>
<td>relaylog.012340</td>
</tr>
<tr>
<td>Relay_Log_Pos</td>
<td>30223388</td>
</tr>
<tr>
<td>Relay_Master_Log_File</td>
<td>mysql-bin-changelog.012345</td>
</tr>
<tr>
<td>Slave_IO_Running</td>
<td>No</td>
</tr>
<tr>
<td>Slave_SQL_Running</td>
<td>Yes</td>
</tr>
<tr>
<td>Replicate_Do_DB</td>
<td></td>
</tr>
<tr>
<td>Replicate_Ignore_DB</td>
<td></td>
</tr>
<tr>
<td>Replicate_Do_Table</td>
<td></td>
</tr>
<tr>
<td>Replicate_Ignore_Table</td>
<td></td>
</tr>
<tr>
<td>Replicate_Wild_Do_Table</td>
<td></td>
</tr>
<tr>
<td>Replicate_Wild_Ignore_Table</td>
<td></td>
</tr>
<tr>
<td>Last_Errno</td>
<td>0</td>
</tr>
<tr>
<td>Last_Error</td>
<td></td>
</tr>
<tr>
<td>Skip.Counter</td>
<td>0</td>
</tr>
<tr>
<td>Exec_Master_Log_Pos</td>
<td>30223232</td>
</tr>
<tr>
<td>Relay_Log_Space</td>
<td>5248928866</td>
</tr>
<tr>
<td>UntilCondition</td>
<td>None</td>
</tr>
<tr>
<td>Until_Log_File</td>
<td></td>
</tr>
<tr>
<td>Until_Log_Pos</td>
<td>0</td>
</tr>
<tr>
<td>Master_SSL_Allowed</td>
<td>No</td>
</tr>
<tr>
<td>Master_SSL_CA_File</td>
<td></td>
</tr>
<tr>
<td>Master_SSL_CA_Path</td>
<td></td>
</tr>
<tr>
<td>Master_SSL_Cert</td>
<td></td>
</tr>
<tr>
<td>Master_SSL_Cipher</td>
<td></td>
</tr>
<tr>
<td>Master_SSL_Verify_Server_Cert</td>
<td>No</td>
</tr>
<tr>
<td>Last_IO_Errno</td>
<td>1236</td>
</tr>
<tr>
<td>Last_IO_Error</td>
<td>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.'</td>
</tr>
<tr>
<td>Last_SQL_Errno</td>
<td>0</td>
</tr>
<tr>
<td>Last_SQL_Error</td>
<td></td>
</tr>
<tr>
<td>Replicate_Ignore_Server_Ids</td>
<td></td>
</tr>
<tr>
<td>Master_Server_Id</td>
<td>67285976</td>
</tr>
</tbody>
</table>

The last IO error 1236 shows that the instance is receiving I/O error 1236. The Relay_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:

```sql
CALL mysql.rds_next_master_log(12345);
```
Oracle on Amazon RDS

Topics
- Things You Should Know About Oracle on Amazon RDS (p. 134)
- Working with an Oracle DB Instance (p. 138)
- Appendix: Options for Oracle DB Engine (p. 156)
- Appendix: Common DBA Tasks for Oracle (p. 168)
- Appendix: Using Oracle GoldenGate with Amazon RDS (p. 178)
- Appendix: Oracle Character Sets Supported in Amazon RDS (p. 192)
- Appendix: Oracle DB Engine Patch Composition (p. 194)

Things You Should Know About Oracle on Amazon RDS

Topics
- Engine Features (p. 135)
- Security (p. 136)
- Oracle Version Management (p. 136)
- Licensing (p. 136)
- Using OEM, APEX, TDE, and other options (p. 137)

Amazon RDS supports DB instances running several editions of Oracle Database. 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.

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 Developer. Amazon RDS does not allow direct host access to a DB instance via Telnet or Secure Shell (SSH). When you create a DB instance, you create a master account that gets DBA privileges (with some limitations) and the SYS password or SYSDBA privileges are not provided.
For information about importing Oracle data into a DB instance, see Importing Data Into Oracle on Amazon RDS (p. 150).

Engine Features

The following list shows a subset of the key Oracle database engine features that are currently supported by Amazon RDS. The availability of the Oracle feature is dependent on the edition of Oracle that you choose. For example, OEM optional packs such as the Database Diagnostic Pack and the Database Tuning Pack are only available with Oracle Enterprise Edition.

The following list shows the Oracle features supported by Amazon RDS; for a complete list of features supported by each Oracle edition, go to Oracle Database 11g Editions.

- 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
- Distributed Queries/Transactions
- Text
- Materialized Views
- Import/Export and sqlldr 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. 176)
- Datapump (network only)
- Native network encryption (part of the Oracle Advanced Security feature)
- Transparent data encryption (Oracle TDE, part of the Oracle Advanced Security feature)

Oracle database engine features that are not currently supported include the following:

- Real Application Clusters (RAC)
- Real Application Testing
- Data Guard / Active Data Guard
- Oracle Enterprise Manager Grid Control
- Automated Storage Management
- Database Vault
- Streams
- Java Support
- Locator
- Spatial
• Oracle XML DB Protocol Server
• Network access utilities such as utl_http, utl_tcp, utl_smtp, and utl_mail, are not supported at this time.

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

Oracle 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 Appendix: Oracle DB Engine Patch Composition (p. 194).

Taking advantage of the DB Engine Version Management feature of Amazon RDS is easily accomplished using the ModifyDBInstance API call or the rds-modify-db-instance command line utility. Your DB instances are upgraded to minor patches by default (you can override this setting). You can also explicitly specify that an upgrade be applied to your database. For more information, see Upgrading a DB Instance (p. 288).

Licensing

There are two types of licensing options available for using Amazon RDS for Oracle.

Bring Your Own License (BYOL)

In this licensing model, you can use your existing Oracle Database licenses to run Oracle deployments on Amazon RDS. To run a DB instance under the BYOL model, 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, go to Licensing Oracle Software in the Cloud Computing Environment.
License Included

In the License Included service model, you do not need separately purchased Oracle licenses; AWS holds the license for the Oracle Database software.

Oracle Licensing and Amazon RDS

Amazon RDS currently supports the following Oracle Database Editions under each of the licensing models below:

- **BYOL:** Standard Edition One (SE1), Standard Edition (SE) and Enterprise Edition (EE)

  To run a DB instance under the BYOL model, you must have the appropriate Oracle Database license (with Software Update License & Support) for the DB instance class and Oracle Database edition you wish to run. You must follow Oracle’s policies for licensing Oracle Database software in the cloud computing environment. DB instances reside in the Amazon EC2 environment, and Oracle’s licensing policy for Amazon EC2 is located here.

  Under this model, you will continue to use your active Oracle support account and contact Oracle directly for Oracle Database specific service requests. If you have an active AWS Premium Support account, you can contact AWS Premium Support for Amazon RDS specific issues. Amazon Web Services and Oracle have multi-vendor support process for cases which require assistance from both organizations.

- **License Included:** Standard Edition One (SE1)

  In the "License Included" service model, you do not need separately purchased Oracle licenses; the Oracle Database software has been licensed by AWS.

  In this model, if you have an active AWS Premium Support account, you should contact AWS Premium Support for both Amazon RDS and Oracle Database specific service requests.

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 OEM, TDE, APEX, and Native Network Encryption. For a complete list of supported Oracle options, see Appendix: Options for Oracle DB Engine (p. 156). For more information about working with option groups, see Working with Option Groups (p. 306).
Working with an Oracle DB Instance

Topics
- Creating a DB Instance Running the Oracle Database Engine (p. 138)
- Connecting to a DB Instance Running the Oracle Database Engine (p. 145)
- Modifying a DB Instance Running the Oracle Database Engine (p. 147)
- Importing Data Into Oracle on Amazon RDS (p. 150)

Most tasks you need to perform on a DB instance are performed the same way for all DB engines. Creating a DB instance, connecting to that DB instance, and importing data into that DB instance are all tasks that are specific for each DB engine. In addition, the appendixes in this section contain important information on working with Oracle DB instances.

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 will use to run your Oracle databases.

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, select the region in which you want to create the DB instance.
3. In the navigation pane, click DB Instances.
4. Click Launch DB Instance to start the Launch DB Instance Wizard.

   The wizard opens on the Engine Selection page. The Oracle editions available will vary by region.
5. In the **Engine Selection** window, click the **Select** button for the Oracle DB engine you want to use.

6. The next step asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option will be preselected in the following step. Click **Next Step** when you are finished.

7. On the **DB Instance Details** page, specify your DB instance information. The following table shows the parameters you need to set to create a DB instance. Click **Next Step** when you are finished.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Model</td>
<td>Select the license option you want to use. Some regions support additional licensing options for Oracle.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>Select the Oracle version you want to use. Currently <strong>11.2.0.2.v7</strong> is the default version of Oracle.</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>Select the DB instance class you want to use. For more information about all the DB instance class options, see <strong>DB Instance Class (p. 47)</strong>.</td>
</tr>
<tr>
<td>Multi-AZ Deployment</td>
<td>Determine if you want to create a standby replica of your DB instance in another availability zone for failover support. This feature is available for Oracle and MySQL DB instances. For more information about multiple availability zones, see <strong>Regions and Availability Zones (p. 50)</strong>.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Auto Minor Version Upgrade</td>
<td>Select <strong>Yes</strong> if you want to enable your DB instance to receive minor DB Engine version upgrades automatically when they become available.</td>
</tr>
<tr>
<td>Allocated Storage</td>
<td>Type a value to allocate of storage for your database (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 about storage allocation, see Amazon RDS Storage (p. 48).</td>
</tr>
<tr>
<td>Use Provisioned IOPS</td>
<td>This option turns on Provisioned IOPS (I/O operations per second), a high-performance storage option in RDS that is optimized for I/O-intensive, transactional (OLTP) database workloads. For more information about high performance storage, see Working with Provisioned IOPS Storage (p. 361).</td>
</tr>
<tr>
<td>DB Instance Identifier</td>
<td>Type a name for the DB instance that is unique for your account in the region you selected. You may choose to add some intelligence to the name such as including the region and DB engine you selected, for example oracle-instance1.</td>
</tr>
<tr>
<td>Master User Name</td>
<td>Type a name that you will 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.</td>
</tr>
<tr>
<td>Master User Password</td>
<td>Type a password that contains from 8 to 30 printable ASCII characters (excluding /,, and @) for your master user password.</td>
</tr>
</tbody>
</table>

---

### DB Instance Details

To get started, choose a DB engine below and click **Continue**.

**DB Engine:** oracle-ee

- **License Model:** Bring Your Own License
- **DB Engine Version:** 11.2.0.2.0
- **DB Instance Class:** db.m1.medium
- **Multi-AZ Deployment:** No
- **Auto Minor Version Upgrade:** Yes

**Allocated Storage:** 80 GB (Minimum: 10 GB, Maximum: 3072 GB) Higher allocated storage may improve IOPS performance.

**DB Instance Identifier:** oracle-instance1 (e.g., mydbinstance)

**Master Username:** myMasterUser (e.g., myuser)

**Master Password:** ********** (e.g., mypassword)
8. On the **Additional Configuration** page, you provide additional information that RDS needs to launch the Oracle DB instance. The following table shows the additional parameters you provide for a DB instance. Specify your DB instance information, then click Next Step.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database Name</strong></td>
<td>Type a name for your database that begins with a letter and contains up to 8 alpha-numeric characters. If you do not provide a name, Amazon RDS will not create a database on the DB instance you are creating.</td>
</tr>
<tr>
<td><strong>Database Port</strong></td>
<td>Specify the port you want to access the database through. Oracle installations default to port 1521.</td>
</tr>
<tr>
<td><strong>Choose a VPC</strong></td>
<td>Determine if you want to create the DB instance in a Virtual Private Cloud. For more information about VPC, see Amazon RDS and the Amazon Virtual Private Cloud Service (p. 54).</td>
</tr>
<tr>
<td><strong>Publicly Accessible</strong></td>
<td>(Only applies if you choose a VPC) Select <strong>Yes</strong> to give the DB instance a public IP address; otherwise, select <strong>No</strong>. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.</td>
</tr>
<tr>
<td><strong>Availability Zone</strong></td>
<td>Determine if you want to specify a particular Availability Zone. If you selected <strong>Yes</strong> for the <strong>Multi-AZ Deployment</strong> parameter on the previous page, you will not have any options here. For more information about availability zones, see Regions and Availability Zones (p. 50).</td>
</tr>
<tr>
<td><strong>Character Set Name</strong></td>
<td>Select a character set for your DB instance. The default value of <strong>AL32UTF8</strong> is for the Unicode 5.0 UTF-8 Universal character set. Note that you cannot change the character set after the DB instance is created.</td>
</tr>
<tr>
<td><strong>Option Group</strong></td>
<td>Select an option group. You can choose the default option group or you can create an option group and select that option group. For more information about option groups, see Working with Option Groups (p. 306).</td>
</tr>
<tr>
<td><strong>Parameter Group</strong></td>
<td>Select a parameter group. You can choose the default parameter group or you can create a parameter group and select that parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 317).</td>
</tr>
<tr>
<td><strong>DB Security Groups</strong></td>
<td>Select the security group you want to use with this DB instance. For more information about security groups, see Working with DB Security Groups (p. 329).</td>
</tr>
</tbody>
</table>
9. On the Management Options page, you can specify backup and maintenance options for your DB instance. Note that setting the Backup Retention Period to zero or setting Enabled Automatic Backups to No disables automatic backups. For more information about the maintenance window, see DB Instance Maintenance (p. 53) and Adjusting the Preferred Maintenance Window (p. 373). For more information on backups and the backup retention period, see DB Instance Backups (p. 55).

When all the settings are as you want them, click Next Step.

10. On the Review page, review the options for your DB instance. If you need to make any changes, click Previous to return to the appropriate page, and then make the necessary corrections. When all the settings are as you want them, click Launch DB Instance.

11. On the final page of the wizard, click Close.

12. On the 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 storage allocated, it could take several minutes for the new instance to be available.

### CLI

**To create an Oracle DB instance**

- Use the command `rds-create-db-instance` to create a DB instance. The following command will launch the example DB instance.

```bash
PROMPT>rds-create-db-instance mydbinstance -s 20 -c db.m1.small -e oracle-se1
   - u <masterawsuser> -p <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  11.2.0.3.v1
SECGROUP  default  active
PARAMGRP  default.oracle-se1-11.2  in-sync
```

### API

**To create an Oracle DB instance**

- Call the `CreateDBInstance` action. For example, you could use the following parameters:

  - `DBInstanceIdentifier = mydbinstance`
  - `Engine = oracle-se1`
  - `DBInstanceClass = db.m1.small`
  - `AllocatedStorage = 20`
  - `BackupRetentionPeriod = 3`
  - `MasterUsername = <masterawsuser>`
• MasterUserPassword = <masteruserpassword>

Example

https://rds.amazonaws.com/
?Action=CreateDBInstance
&AllocatedStorage=20
&BackupRetentionPeriod=3
&DBInstanceClass=db.m1.small
&DBInstanceIdentifier=mydbinstance
&DBName=mydatabase
&DBSecurityGroups.member.1=mysecuritygroup
&DBSubnetGroup=mydbsubnetgroup
&Engine=oracle-se1
&M asterUserPassword=<masteruserpassword>
&M asterUsername=<masterawsuser>
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2013-09-09
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140202/us-west-2/rds/aws4_request
&X-Amz-Date=20140202T190545Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=60e907d8d43fdc978941c1566f7b3c5054e0328622a871fb59b61782ee1f30d8

Related Topics

• DB Instance (p. 43)
• Amazon RDS Security Groups (p. 74)
• DB Instance Class (p. 47)
• Deleting a DB Instance (p. 273)
Connecting to a DB Instance Running the Oracle Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. In this example, you connect to a DB instance running the Oracle database engine using the Oracle command line tools. For more information on using Oracle, go to the Oracle website.

**Note**
This example uses the Oracle sqlplus command line utility. This utility is part of the Oracle software distribution. To download a stand-alone version of this utility, go to the SQL*Plus User's Guide and Reference.

### CLI

#### To connect to a DB Instance using sqlplus

1. Find the DNS name for your DB instance using the `rds-describe-db-instances` command

   ```
PROMPT>rds-describe-db-instances --headers
   ```

   You will see output similar to the following:

<table>
<thead>
<tr>
<th>DBINSTANCE</th>
<th>DBInstanceId</th>
<th>Created</th>
<th>Class</th>
<th>Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracledb</td>
<td>2011-05-14T01:11:01.727Z</td>
<td>db.m1.small</td>
<td>oracle-ee</td>
<td>20</td>
</tr>
<tr>
<td>mydbusr</td>
<td>available</td>
<td>oracledb.mydnsnameexample.rds.amazonaws.com</td>
<td>1521</td>
<td>us-east-1a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11.2.0.2.v3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bring-your-own-license</td>
</tr>
</tbody>
</table>

   **Note**
   You can also use the AWS Management Console to find this information.

2. Type the following command on one line at a command prompt to connect to a DB instance using the sqlplus utility; substitute the DNS name for your DB instance, the port, and the Oracle SID as appropriate. The SID value is the name of the 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=oracledb.mydnsnameexample.rds.amazonaws.com)(PORT=1521))(CONNECT_DATA=(SID=mydbname)))'
   ```

   You will see output similar to the following.
Related Topics

- DB Instance (p. 43)
- Creating a DB Instance Running the MySQL Database Engine (p. 82)
- Amazon RDS Security Groups (p. 74)
- Deleting a DB Instance (p. 273)
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. This topic guides you through modifying an Amazon RDS Oracle DB instance, and describes the settings for Oracle instances. For information about additional tasks, such as renaming, rebooting, deleting, tagging, or upgrading an Amazon RDS DB instance, see Tasks Common to All Amazon RDS DB Engines (p. 270).

Before you upgrade your production DB instances to a new Oracle Database version, we recommend you test the upgrade process on a test instance to verify its duration and to validate your applications. 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 GB) would be to copy your data using Data Pump, where we also recommend testing before migrating your production instances.

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 a DB Instance and Using the Apply Immediately Parameter (p. 45).

**AWS Management 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, click **DB Instances**.
3. Select the check box for the DB instance that you want to change, and then click **Modify**.
4. In the **Modify DB Instance** dialog box, change any of the following settings that you want:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Instance Identifier</td>
<td>You can rename the DB instance by typing a new name.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>In the list provided, click the version of the Oracle database engine that you want to use. Before you upgrade your production database instances, we recommend you test the upgrade process on a test instance to verify its duration and to validate your applications. 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 GB) would be to copy your data using Data Pump, where we also recommend testing before migrating your production instances.</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>In the list provided, click the DB instance class that you want to use. For information about instance classes, see the section called “DB Instance Class” (p. 47).</td>
</tr>
<tr>
<td>Multi-AZ Deployment</td>
<td>If you want to deploy your DB instance in multiple Availability Zones, click Yes; otherwise, click No.</td>
</tr>
</tbody>
</table>
If you want your DB instance to receive minor engine version upgrades automatically when they become available, click **Yes**. Upgrades are installed only during your scheduled maintenance window.

Specify how much storage, in gigabytes, will be initially allocated for your DB instance. The minimum allowable value is 10 GB; the maximum is 3072 GB.

Select the **Use Provisioned IOPS** check box and enter a value in the **Provisioned IOPS** text box if you want your DB instance to use Provisioned IOPS. For more information on Provisioned IOPS, see [Working with Provisioned IOPS Storage](p. 361).

Select the parameter group you want associated with the DB instance. For more information about parameter groups, see [Working with DB Parameter Groups](p. 317).

Select the security group you want associated with the DB instance. For more information about security groups, see [Working with DB Security Groups](p. 329).

Select the option group you want associated with the DB instance. For more information about option groups, see [Working with Option Groups](p. 306).

Type a password for your master user. The password must contain from 8 to 128 alphanumeric characters.

Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0.

Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC) and a duration in hours.

Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.

To apply the changes immediately, select the **Apply Immediately** check box. Selecting this option can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option, see [Modifying a DB Instance and Using the Apply Immediately Parameter](p. 45).

When all the changes are as you want them, click **Yes, Modify**. If instead you want to cancel any changes that you didn't apply in the previous step, click **Cancel**.

**CLI**

To modify an Oracle DB instance

- Use the command `rds-modify-db-instance`.

---

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Minor Version Upgrade</td>
<td>If you want your DB instance to receive minor engine version upgrades automatically when they become available, click <strong>Yes</strong>. Upgrades are installed only during your scheduled maintenance window.</td>
</tr>
<tr>
<td>Allocated Storage</td>
<td>Specify how much storage, in gigabytes, will be initially allocated for your DB instance. The minimum allowable value is 10 GB; the maximum is 3072 GB.</td>
</tr>
<tr>
<td>Provisioned IOPS</td>
<td>Select the <strong>Use Provisioned IOPS</strong> check box and enter a value in the <strong>Provisioned IOPS</strong> text box if you want your DB instance to use Provisioned IOPS. For more information on Provisioned IOPS, see [Working with Provisioned IOPS Storage](p. 361).</td>
</tr>
<tr>
<td>Parameter Group</td>
<td>Select the parameter group you want associated with the DB instance. For more information about parameter groups, see [Working with DB Parameter Groups](p. 317).</td>
</tr>
<tr>
<td>Security Group</td>
<td>Select the security group you want associated with the DB instance. For more information about security groups, see [Working with DB Security Groups](p. 329).</td>
</tr>
<tr>
<td>Option Group</td>
<td>Select the option group you want associated with the DB instance. For more information about option groups, see [Working with Option Groups](p. 306).</td>
</tr>
<tr>
<td>Master User Password</td>
<td>Type a password for your master user. The password must contain from 8 to 128 alphanumeric characters.</td>
</tr>
<tr>
<td>Backup Retention Period</td>
<td>Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0.</td>
</tr>
<tr>
<td>Backup Window</td>
<td>Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC) and a duration in hours.</td>
</tr>
<tr>
<td>Maintenance Window</td>
<td>Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.</td>
</tr>
</tbody>
</table>
API

To modify an Oracle DB instance

• Use the `ModifyDBInstance` action.
Importing Data Into Oracle on Amazon RDS

Platform migration for Oracle databases varies in complexity based on the size of the data and the number and variety of database objects. Depending on the complexity of your database and size of your data, you might be able to use one or more of these techniques.

Before you use any of these migration techniques, we recommend the best practice of taking a backup of your database. You can back up your Amazon RDS instances by creating snapshots. Later, you can restore the database from the snapshots using the Restore from DB Snapshot or Restore to Point In Time options on the RDS tab of the AWS Management Console. You can also use the Amazon RDS command line methods `rds-restore-db-instance-from-db-snapshot` or `rds-restore-db-instance-to-point-in-time`. These and other best practices are addressed in this section.

Oracle SQL Developer

Oracle SQL Developer is 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, although it could migrate a large number of objects. 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.


Oracle also has documentation on how to migrate from other databases, including MySQL and SQL Server. To learn more, go to http://www.oracle.com/technetwork/database/migration.

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 import data to a DB instance from an Amazon EC2 instance with an Oracle database, a local Oracle database, or from another Oracle DB instance.

The following process uses Oracle Data Pump and the `DBMS_FILE_TRANSFER` package. The process connects to an Oracle instance and exports data using Oracle Data Pump. It then uses the `DBMS_FILE_TRANSFER.PUT_FILE` method to copy the dump file from the Oracle instance to the `DATA_PUMP_DIR` on the target DB instance that is connected via a database link. The final step imports the data from the copied dump file into the RDS instance.

The process has the following requirements:

- You must have execute privileges on the `DBMS_FILE_TRANSFER` package
- The target DB instance must be version 11.2.0.2.v6 or later
- 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. Given that the current storage size limit of an Oracle DB instance is 3 TB, you can't use this process to upload any dump file larger than about 1.3 TB. Note that 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 source database
• Step 2: Use DBMS_DATAPUMP to create a dump file
• Step 3: Create a database link to the target DB instance
• Step 4: Use DBMS_FILE_TRANSFER to copy the exported dump file to the Amazon RDS instance
• Step 5: Import the dump file into a database on the Amazon RDS instance

**Step 1: 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.

The following commands create a new user and grant the necessary permissions:

```sql
SQL> create user user1 identified by test123;
SQL> grant create session, create table to USER1;
SQL> alter user USER1 quota 100M on users;
SQL> grant read, write on directory data_pump_dir to USER1;
SQL> grant execute on dbms_datapump to USER1;
```

You can use your own table, or you can create one to test the process. The following commands create a sample table for importing into a DB instance:

```sql
SQL> create table USER1.tab1
tablespace users
as select 'USER1_'||object_name str_col, sysdate dt_col from all_objects;
```

**Step 2: Use DBMS_DATAPUMP to create a dump file**

Use SQL Plus or Oracle SQL Developer to connect to the Oracle instance and use the Oracle Data Pump utility to create a dump file. The following script creates a dump file named *tab1.dmp* in the DATA_PUMP_DIR directory.

```sql
DECLARE    
    hdnl NUMBER;
BEGIN
    hdnl := DBMS_DATAPUMP.open( operation => 'EXPORT', job_mode => 'SCHEMA',
job_name => null);
    DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'tab1.dmp', directory =>
```
Step 3: Create a database link to the target DB instance

Next, create a database link between your source instance and your target DB instance. The following command creates a database link named to_rds to another user at the target DB instance database:

```sql
create database link to_rds connect to USER2 identified by user2pwd using '(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<remotehost>.<instance_name>.amazonaws.com)(PORT=4080))(CONNECT_DATA=(SID=<remoteSID>)))';
```

Note

> If you are creating a database link between two DB instances inside a VPC, the two DB instances must be either in the same VPC or you must create an EC2 or VPC security group that both DB instances are a member of.

Step 4: Use DBMS_FILE_TRANSFER to copy the exported dump file to an Amazon RDS DB instance

Next, 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 tab1.dmp from the source instance to a target database link named to_rds (created in the previous step):

```sql
BEGIN
    DBMS_FILE_TRANSFER.PUT_FILE(
        source_directory_object       => 'DATA_PUMP_DIR',
        source_file_name              => 'tab1.dmp',
        destination_directory_object  => 'DATA_PUMP_DIR',
        destination_file_name         => 'tab1_copied.dmp',
        destination_database          => 'to_rds'
    );
END;
/
```

Step 5: Use Data Pump to import the data file on the DB instance

On the DB instance, use Oracle Data Pump to import the schema. The first part of the listing shows the format for the data import statement, and the second part shows importing a data file called tab1_copied.dmp.
impdp <username>@<TNS_ENTRY> DUMPFILE=user1copied.dmp DIRECTORY=DATA_PUMP_DIR full=y
impdp copy1@copy1 DUMPFILE=tab1_copied.dmp DIRECTORY=DATA_PUMP_DIR full=y

You can verify the data import by viewing the table on the DB instance.

SQL> select count(*) from user1.tab1;

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.

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.

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 will be 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 will be 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.

```sql
cat << EOF > sqlldr_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"
)
EOF
```

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.

```sql
sqlldr cust_dba@targetdb control=sqlldr_1.ctl BINDSIZE=10485760 READ SIZE=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= sourcedb.rds.amazonaws.com)
        (port=1521))
        (connect_data=(sid=sourcedb)))' /
```

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;
```
Appendix: Options for Oracle DB Engine

This appendix describes options, or additional features, that are available for Amazon RDS instances running the Oracle database engine. To enable these options, you can add them to an option group, and then associate the option group with your DB instance. Note that some options are permanent and persistent; permanent means that an option cannot be removed from an option group and persistent means that once an option group with this option is assigned to a DB instance, the option group cannot be removed from the DB instance. For more information about working with options, see Things You Should Know About Option Groups (p. 306).

The following options are currently supported for Oracle:

- Oracle Enterprise Manager Database Control (p. 156)
- Oracle XML DB (p. 157)
- Oracle Application Express (APEX) (p. 157)
- Oracle Transparent Data Encryption (TDE) (p. 162) (a feature of the Oracle Advanced Security option available in Oracle Enterprise Edition)
- Oracle Statspack (p. 163)
- Oracle Time Zone (p. 166)

Note
Some of these options may require additional memory in order 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. Before you enable these options, please consider whether your DB instance has enough available memory. You can adjust the Oracle parameters so that the database requires less RAM; alternatively, you can scale up to a larger DB instance.

Oracle Enterprise Manager Database Control

Oracle Enterprise Manager (OEM) Database Control is a web-based interface for Oracle database administration. Note that OEM cannot be run on DB instances that use the db.t1.micro or db.t1.small instance classes.

The default port number for OEM Database Control is 1158; you can accept this port number, or choose a different one, when you enable the OEM Database Control option for your DB instance. You can then go to your web browser and begin using OEM Database Control.

The following example shows how to access OEM Database Control from your web browser. Suppose that the endpoint for your Amazon RDS instance is mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com, and that you specified port 1158. The URL to access OEM Database Control would be:

https://mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com:1158/em

The OEM Database Control login window appears, prompting you for a username and password. Enter the master username and master password for your DB instance. You can now manage your database using the OEM Database Control console.
Oracle XML DB

Oracle XML DB adds native XML support to your DB instance. With the Amazon RDS XMLDB option, DB instances running the Oracle engine can store and retrieve structured or unstructured XML, in addition to relational data.

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.

Oracle Application Express (APEX)

Oracle Application Express (APEX) is a development and runtime environment for web-based applications. Using APEX, developers can build applications entirely within the web browser, and customers can run these applications without installing any additional software.

Note
Amazon RDS supports Oracle APEX version 4.1.1.

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 APEX clients. The listener accepts incoming connections from web browsers and forwards them to the Amazon RDS instance for processing, and then sends results from the repository back to the browsers.

When you add the APEX option for your Oracle DB instance, Amazon RDS installs the APEX repository only. You must install the listener on a separate host — an Amazon EC2 instance, an on-premises server at your company, or your desktop computer.

The following sections explain how to configure the Oracle APEX repository and listener for use with Amazon RDS.

Repository Configuration

To configure the APEX repository

1. Create a new Amazon RDS instance running the Oracle engine, or choose an existing instance. The version number for the Oracle engine must be 11.2.0.2.v4 or newer.
2. Create a new option group, or select an existing option group. Apply the following options to this option group:

   - XMLDB
   - APEX
   - APEX_DEV

   (If you only want to deploy the APEX runtime environment, you can remove the APEX_DEV option at a later time. This option must be present during this configuration procedure, however.)
3. Apply the option group to your DB instance. Amazon RDS will install the repository components in your DB instance; this process takes a few minutes to complete.
4. After the option group is successfully applied, you will need to change the password for the APEX_PUBLIC_USER database account and unlock it. You can do this using the Oracle SQL*Plus command line utility: Connect to your DB instance as the master user and issue the following commands:

```
alter user APEX_PUBLIC_USER identified by newpass;
alter user APEX_PUBLIC_USER account unlock;
```

Replace `newpass` with a password of your choice.

**Listener Configuration**

You are now ready to configure a listener for use with Oracle APEX. You can use either of these products for this purpose:

- Oracle Application Express Listener
- Oracle HTTP Server and `mod_plsql`

**Note**

Amazon RDS does not support the Oracle XML DB HTTP server with the embedded PL/SQL gateway; you cannot use this as an APEX listener. This restriction is in line with Oracle's recommendation against using the embedded PL/SQL gateway for applications that run on the Internet.

The listener must be installed on a separate host, such as an Amazon EC2 instance or a server that you own. You also must have the following prerequisite software installed on the separate host acting as the listener:

- Java Runtime Environment (JRE) — Oracle APEX Listener is a Java application.
- Oracle Net Services, to enable the APEX listener to connect to your Amazon RDS instance.
- SQL*Plus, to perform administrative tasks from the command line.

The following procedure shows how to configure the Oracle Application Express Listener product. We will assume that the name of your APEX host is `myapexhost.example.com`, and that this host is running Linux.

**To configure an APEX listener**

1. Log in to `myapexhost.example.com` as `root`.
2. We recommend that you create a nonprivileged OS user to own the APEX listener installation. The following command will create a new user named `apexuser`:

   ```
   useradd -d /home/apexuser apexuser
   ```

   Now assign a password to `apexuser`:

   ```
   passwd apexuser
   ```
3. Log in to `myapexhost.example.com` as `apexuser`, and download the APEX and APEX Listener installation files from Oracle:


4. Open the APEX file:

   ```
   unzip apex_4.1.1.zip
   ```

5. Create a new directory and open the APEX Listener file:

   ```
   mkdir /home/apexuser/apexlistener
   cd /home/apexuser/apexlistener
   unzip ../apex_listener.1.1.3.243.11.40.zip
   ```

6. While you are still in the `apexlistener` directory, run the APEX Listener program:

   ```
   java -Dapex.home=./apex -Dapex.images=/home/apexuser/apex/images -Dapex.erase -jar ./apex.war
   ```

   The program will prompt you for the following:
   - The APEX Listener Administrator username — the default is `adminlistener`
   - A password for the APEX Listener Administrator.
   - The APEX Listener Manager username — the default is `managerlistener`
   - A password for the APEX Listener Administrator.

   The program will print a URL that you will need in order to complete the configuration:

   ```
   INFO: Please complete configuration at: http://localhost:8080/apex/listener
   Configure
   Database is not yet configured
   ```

   Leave the APEX Listener running. It needs to continue running in order for you to use Oracle Application Express. (When you have finished this configuration procedure, you can run the listener in the background.)

7. From your web browser, go to the URL provided by the APEX Listener program. The Oracle Application Express Listener administration window appears. Enter the following information:

   - **Username** — `APEX_PUBLIC_USER`
   - **Password** — the password for `APEX_PUBLIC_USER`. (This is the password that you specified earlier, when you configured the APEX repository.)
   - **Connection Type** — Basic
   - **Hostname** — the endpoint of your Amazon RDS instance, such as `mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com`
   - **Port** — 1521
• **SID**— the name of the database on your Amazon RDS instance, such as *mydb*

Click **Apply** button. The APEX administration window appears.

8. You will need to 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 issue the following commands:

   ```sql
   grant APEX_ADMINISTRATOR_ROLE to master;
   /home/apexuser/apex/apxchpwd.sql
   ``

   Replace `master` with your master user name. Enter a new `admin` password when the `apxchpwd.sql` script prompts you.

9. Return to the APEX administration window in your browser and click **Administration**. Next, click **Application Express Internal Administration**. You will be prompted for APEX internal administration credentials. Enter the following information:

   - **Username**— `admin`
   - **Password**— the password you set using the `apxchpwd.sql` script.

   Click **Login**. You will be required to set a new password for the **admin** user.

Oracle Application Express is now ready for use.

**Oracle Native Network Encryption**

Amazon RDS supports Oracle native network encryption, a feature of the Oracle Advanced Security option available in Oracle Enterprise Edition. With native network encryption, you can encrypt data as it moves to and from a DB instance.

To use Oracle native network encryption with a DB instance, you add the `NATIVE_NETWORK_ENCRYPTION` option to an option group and associate that option group with the DB instance. You should first determine if the DB instance is associated with an option group that has the `NATIVE_NETWORK_ENCRYPTION` option. To view the option group that a DB instance is associated, you can use the RDS console, the `rds-describe-db-instance` CLI command, or the API action `DescribeDBInstances`. Amazon RDS supports Oracle native network encryption for any DB instance class larger than `db.t1.micro`.

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. Note that non-default TDE encryption algorithms only work with Oracle version 11.202.v7 and later. For information about the algorithms and keys that are available through Oracle Advanced Security, see Oracle Advanced Security in the Oracle documentation. For more information about AWS security, see the AWS Security Center.

The process for using Oracle native network encryption with Amazon RDS is as follows:

1. If the DB instance is not associated with an option group that has the network encryption option (`NATIVE_NETWORK_ENCRYPTION`), you must either modify an existing option group to add the `NATIVE_NETWORK_ENCRYPTION` option or create a new option group and add the `NATIVE_NETWORK_ENCRYPTION` option to it. For information about creating or modifying an option group, see Working with Option Groups (p. 306). For information about adding an option to an option group, see Adding an Option to an Option Group (p. 309).

2. Specify the `NATIVE_NETWORK_ENCRYPTION` option settings for the option group. For information about modifying option settings, see Modifying an Option Setting (p. 313).
These settings include:

- **SQLNET.ENCRYPTION_SERVER**—Specifies the encryption behavior when a client, or a server acting as a client, connects to the DB instance. Allowable values are Accepted, Rejected, Requested (the default), and Required. Requested indicates that the DB instance does not require traffic from the client to be encrypted.

- **SQLNET.CRYPTO_CHECKSUM_SERVER**—Specifies the data integrity behavior when a client, or a server acting as a client, connects to the DB instance. Allowable values are Accepted, Rejected, Requested (the default), and Required. Requested indicates that the DB instance does not require the client to perform a checksum.

- **SQLNET.ENCRYPTION_TYPES_SERVER**—Specifies 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.
  a. RC4_256: RSA RC4 (256-bit key size)
  b. AES256: AES (256-bit key size)
  c. AES192: AES (192-bit key size)
  d. 3DES168: 3-key Triple-DES (188-bit effective key size)
  e. RC4_128: RSA RC4 (128-bit key size)
  f. AES128: AES (128-bit key size)
  g. 3DES112: 2-key Triple-DES (112-bit effective key size)
  h. RC4_56: RSA RC4 (56-bit key size)
  i. DES: Standard DES (56-bit key size)
  j. RC4_40: RSA RC4 (40-bit key size)
  k. DES40: DES40 (40-bit key size)

- **SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER**—Specifies the checksum algorithm. The default is sha-1, but md5 is also supported.

3. List the options in the option group to ensure that you have added the `NATIVE_NETWORK_ENCRYPTION` option and specified the correct settings. You can view the options in an option group using the RDS console, the CLI command `rds-describe-option-group-options`, or the Amazon RDS API action `DescribeOptionGroupOptions`.

4. Associate the DB instance with the option group that has the `NATIVE_NETWORK_ENCRYPTION` option. For information about associating a DB instance with an option group, see Modifying a DB Instance Running the Oracle Database Engine (p. 147).

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:

```plaintext
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.

**Oracle Transparent Data Encryption (TDE)**

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.

**Note**
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.

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.

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.

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. 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 (for versions 11.2.0.2.v7 or later), 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 rds-describe-db-instance CLI command, or the API action DescribeDBInstances.

Amazon RDS manages the Oracle Wallet and TDE master key for the DB instance. To comply with several security standards, Amazon RDS is working to implement automatic periodic master key rotation.

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. 306). For information about adding an option to an option group, see Adding an Option to an Option Group (p. 309).

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. 147).

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; however, 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. 150).

### Oracle Statspack

The Oracle Statspack option (STATSPACK) 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. 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 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 will get an error and the RDS event RDS-Event-0058 will be generated.

You can drop the PERFSTAT account by running the following command:

```
DROP USER perfstat CASCADE;
```

2. 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;
```

3. 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(SYS
DATE+1/24,'''HH24'''));
commit;
```

4. 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;
```

5. 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 52:

```
exec RDSADMIN.RDS_RUN_SPREPORT(1,52);
```

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 52 would be named ORCL_spreport_1_52.lst. You can download the Statspack report by selecting the report in the Log section of the RDS console and clicking Download or you can use the trace file procedures explained in Working with Oracle Trace Files (p. 401).
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 52, the report file would be named ORCL_spreport_1_52.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. 401).

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 will be 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

The **Timezone** option lets you change the system time zone used by Oracle databases in a DB instance. You might need to change the time zone for a DB instance if you need to have time compatibility with an on-premises environment or a legacy application. This option changes the time zone at the host level and impacts all date columns and values including `SYSDATE` and `SYSTIMESTAMP`. This option can only be applied once to a DB instance. You should take a DB snapshot of your DB instance before applying this option to a DB instance so that you can recover the instance if the time zone option is set incorrectly.

**Note**

Applying the **Timezone** option to option groups used by existing DB instances could cause problems with tables that use system date to add dates or time, so you should analyze your data to determine what impact a time zone change will have. We strongly urge you to test setting this option on a test DB instance before setting it on your production instances.

The **Timezone** option is a permanent and persistent option that cannot be removed from an option group once it is added and the option group cannot be disassociated from a DB instance. This option can be applied immediately by selecting **Apply Immediately** or it can be applied at the next maintenance window.

There are three ways that you can add the **Timezone** option to an option group. You can use the Amazon RDS console, the `rds-add-option-to-option-group` Amazon RDS CLI command, or the `ModifyOptionGroup` API action.

The following example uses the Amazon RDS CLI command `rds-add-option-to-option-group` to add the **Timezone** option and the **TIME_ZONE** option setting to an option group called `myoptiongroup`. The time zone is set to **Asia/Japan**.

```
rds-add-option-to-option-group myoptiongroup --option-name Timezone --settings "TIME_ZONE=Asia/Tokyo"
```

The **Timezone** option differs from the `rdsadmin_util.alter_db_time_zone` command. The `rdsadmin_util.alter_db_time_zone` command only changes the time zone for certain data types, while the **Timezone** option changes the time zone at the host level and impacts all date columns and values such as `SYSDATE`.

The following values can be used for the **TIME_ZONE** option setting:

Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek, America/Araguaina, America/Asuncion, America/Bogota, America/Caracas, America/Chihuahua, America/Cuiba, America/Denver, America/Fortaleza, America/Guatemala, America/Halifax, America/Manaus, America/Matamoros, America/Monterrey, America/Montevideo, America/Phoenix, America/Santiago, America/Tijuana, Asia/Amman, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Irkutsk, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Krasnoyarsk, Asia/Magadan, Asia/Muscat, Asia/Novosibirsk, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran,
Appendix: Common DBA Tasks for Oracle

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the Oracle database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

For information about working with Oracle log files on Amazon RDS, see Working with Oracle Database Log Files (p. 400)

Topics
- Enabling and disabling Restricted Session (p. 168)
- Flushing the Shared Pool (p. 169)
- Flushing the Buffer Cache (p. 169)
- Checkpointing the Database (p. 169)
- Killing a Session (p. 169)
- Switching Online Log files (p. 400)
- Adding, dropping and resizing online redo logs (p. 170)
- Setting Default Tablespace (p. 173)
- Retaining Archived Redo Logs (for version 11.2.0.2.v7 and later) (p. 173)
- Setting Supplemental Logging (for version 11.2.0.3.v1 and later) (p. 173)
- Renaming the Global Name (for version 11.2.0.3.v1 and later) (p. 174)
- Setting Force Logging (for version 11.2.0.3.v1 and later) (p. 174)
- Disconnecting a Session (for version 11.2.0.3.v1 and later) (p. 174)
- Setting Distributed Recovery (for version 11.2.0.3.v1 and later) (p. 175)
- Granting SELECT or EXECUTE privileges to SYS Objects (for version 11.2.0.3.v1 and later) (p. 175)
- Granting Privileges to Non-Master Users (p. 175)
- Setting the Database Time Zone (p. 176)
- Working with Automatic Workload Repository (AWR) (p. 176)
- Creating and Resizing Tablespaces and Data Files (p. 176)
- Adjusting Database Links for Use with DB Instances in a VPC (p. 177)

## Enabling and disabling Restricted Session

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter system enable restricted session;</td>
<td>exec rdsadmin.rdsadmin_util.restricted_session(true);</td>
</tr>
<tr>
<td>alter system disable restricted session;</td>
<td>exec rdsadmin.rdsadmin_util.restricted_session(false);</td>
</tr>
</tbody>
</table>

The following example shows how to enable and disable restricted sessions.

```sql
select logins from v$instance;
LOGINS
```
ALLOWED
 exec rdsadmin.rdsadmin_util.restricted_session(true);

select logins from v$instance;

LOGINS
RESTRICTED
exec rdsadmin.rdsadmin_util.restricted_session(false);

select logins from v$instance;

LOGINS
ALLOWED

Flushing the Shared Pool

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter system flush shared_pool;</td>
<td>exec rdsadmin.rdsadmin_util.flush_shared_pool;</td>
</tr>
</tbody>
</table>

Flushing the Buffer Cache

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter system flush buffer_cache;</td>
<td>exec rdsadmin.rdsadmin_util.flush_buffer_cache;</td>
</tr>
</tbody>
</table>

Checkpointing the Database

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter system checkpoint;</td>
<td>exec rdsadmin.rdsadmin_util.checkpoint;</td>
</tr>
</tbody>
</table>

Killing a Session

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter system kill session 'sid, serial#' IMMEDIATE;</td>
<td>exec rdsadmin.rdsadmin_util.kill(sid, serial#);</td>
</tr>
<tr>
<td></td>
<td>For use with version 11.2.0.3.v1 or higher:</td>
</tr>
<tr>
<td></td>
<td>exec rdsadmin_util.kill(sid_number, serial_number, method varchar default null);</td>
</tr>
</tbody>
</table>
If you are using version 11.2.0.3.v1 or higher, you can specify PROCESS as a value for method. This enables you to kill processes associated with a session. You should only do this if killing the session was unsuccessful.

### Switching Online Log files

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter system switch logfile;</td>
<td>exec rdsadmin.rdsadmin_util.switch logfile;</td>
</tr>
</tbody>
</table>

### Adding, dropping and resizing online redo logs

A newly created Amazon RDS instance using the Oracle database engine will have four 128MB online redo logs. Note that in cases where you want to add more logs, the same restrictions apply to naming physical files as they do for naming online redo logs.

Use the following procedures to add or drop redo logs:

- exec rdsadmin.rdsadmin_util.add_logfile(size bytes);
- exec rdsadmin.rdsadmin_util.drop_logfile(group#);

If you are using version 11.2.0.3.v1 or later, you can specify the size modifier as well. For example, the following command would add a 100 Mb log file:

```sql
exec rdsadmin.rdsadmin_util.add_logfile('100M');
```

### Example

The following example shows how you can use the Amazon RDS-provided procedures to resize your online redo logs from their default size to 512M.

```sql
# Start with four 128m logs.
SQL>SELECT GROUP#, BYTES, STATUS FROM v$log;

GROUP#   BYTES   STATUS
---------- ---------- -------------
1 134217728  INACTIVE
2 134217728  CURRENT
3 134217728  INACTIVE
4 134217728  INACTIVE

4 rows selected.

# Add four new logs with that are each 512m.
SQL>exec rdsadmin.rdsadmin_util.add_logfile(536870912);

PL/SQL procedure successfully completed.

SQL>exec rdsadmin.rdsadmin_util.add_logfile(536870912);

PL/SQL procedure successfully completed.
```
SQL> exec rdsadmin.rdsadmin_util.add_logfile(536870912);
PL/SQL procedure successfully completed.

SQL> exec rdsadmin.rdsadmin_util.add_logfile(536870912);
PL/SQL procedure successfully completed.

# Now query v4log to show that there are 8 logs:

SQL> 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

8 rows selected.

# Now, drop each INACTIVE log using the group#.

SQL> exec rdsadmin.rdsadmin_util.drop_logfile(1);
PL/SQL procedure successfully completed.

SQL> exec rdsadmin.rdsadmin_util.drop_logfile(3);
PL/SQL procedure successfully completed.

SQL> exec rdsadmin.rdsadmin_util.drop_logfile(4);
PL/SQL procedure successfully completed.

#

SQL> 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

8 rows selected.

# Switch logs so that group 2 is no longer current:

SQL> exec rdsadmin.rdsadmin_util.switch_logfile;
PL/SQL procedure successfully completed.
# 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

5 rows selected.

# Issue a checkpoint to clear log 2
SQL> exec rdsadmin.rdsadmin_util.checkpoint;

PL/SQL procedure successfully completed.

# SQL> 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

5 rows selected.

# Checkpointing clears log group 2 so that its status is now INACTIVE allowing
us to drop the final log group 2:
SQL> exec rdsadmin.rdsadmin_util.drop_logfile(2);

PL/SQL procedure successfully completed.

# Now, there are four 512m logs. Oracle using Oracle Managed Files (OMF) will
automatically remove the old logfiles from the file system.

SQL> select GROUP#, BYTES, STATUS from v$log;

GROUP#      BYTES STATUS
---------- ---------- ----------------
5  536870912 CURRENT
6  536870912 UNUSED
7  536870912 UNUSED
8  536870912 UNUSED

4 rows selected.
Setting Default Tablespace

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter database default tablespace users2;</td>
<td>exec rdsadmin.rdsadmin_util.alter_default_tablespace('users2');</td>
</tr>
</tbody>
</table>

Retaining Archived Redo Logs (for version 11.2.0.2.v7 and later)

You can retain archived redo logs on your DB instance for use with products like Oracle LogMiner (DBMS_LOGMNR). Once you have retained the redo logs, you can use LogMiner to analyze the logs as explained in the Oracle documentation. Note that you need to ensure that the DB instance has enough allocated storage to store the retained logs.

Use the Amazon RDS method `rdsadmin.rdsadmin_util.set_configuration` to retain archived redo logs. The following example shows how to retain 24 hours of redo logs:

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

If you need 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 start_time>=sysdate-X/24 and dest_id=1;
```

Setting Supplemental Logging (for version 11.2.0.3.v1 and later)

The following Amazon RDS method enables supplemental logging, including minimal supplemental logging. Oracle Database does not enable supplemental logging by default. Supplemental logging ensures that LogMiner and products that use LogMiner technology will have sufficient information to support chained rows and various storage arrangements such as cluster tables. For more information on supplemental logging, see the Oracle documentation.

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter database [add</td>
<td>drop] supplemental log;</td>
</tr>
<tr>
<td>alter database add supplemental log data (PRIMARY KEY) columns;</td>
<td>exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD','PRIMARY KEY');</td>
</tr>
</tbody>
</table>
Renaming the Global Name (for version 11.2.0.3.v1 and later)

The following Amazon RDS method changes the global name of the database. Note that the database must be open for the name change to take effect. For more information about changing the global name of a database, see the Oracle documentation.

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter database add supplemental log data (ALL) columns;</td>
<td>exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD','ALL');</td>
</tr>
<tr>
<td>alter database add supplemental log data (UNIQUE) columns;</td>
<td>exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD','UNIQUE');</td>
</tr>
</tbody>
</table>

Setting Force Logging (for version 11.2.0.3.v1 and later)

The following Amazon RDS method puts the database in or removes the database from FORCE LOGGING mode. In FORCE LOGGING mode, Oracle logs all changes to the database except changes in temporary tablespaces and temporary segments. For more information about forcing logging, see the Oracle documentation.

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter database [no] force logging;</td>
<td>exec rdsadmin_util.force_logging(p_enable in boolean := true);</td>
</tr>
</tbody>
</table>

Disconnecting a Session (for version 11.2.0.3.v1 and later)

The following Amazon RDS method disconnects the current session by ending the dedicated server process. Note that the database must be open to use this method. For more information about disconnecting a session, see the Oracle documentation.

You must specify both the SID and serial number of the session. To obtain these values, query the V$SESSION view. For example, the following query shows all sessions for the user AWSUSER:

```sql
SELECT SID, SERIAL#, STATUS
FROM V$SESSION
WHERE USERNAME = 'AWSUSER';
```
Setting Distributed Recovery (for version 11.2.0.3.v1 and later)

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter system disconnect</td>
<td>exec rdsadmin_util.disconnect(sid number, serial number, method varchar default</td>
</tr>
<tr>
<td>session;</td>
<td>'IMMEDIATE');</td>
</tr>
<tr>
<td>alter system disable</td>
<td>rdsadmin_util.enable_distr_recovery and</td>
</tr>
<tr>
<td>distributed recovery;</td>
<td>rdsadmin_util.disable_distr_recovery (mydatabase);</td>
</tr>
</tbody>
</table>

Granting SELECT or EXECUTE privileges to SYS Objects (for version 11.2.0.3.v1 and later)

The following procedure transfers existing privileges such as SELECT and EXECUTE via a role to another account. Note that it only grants privileges that the master account already has via a role or direct grant.

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant select on V_$SESSION to myuser;</td>
<td>exec rdsadmin.rdsadmin_util.grant_sys_object('V_$SESSION','MYUSER');</td>
</tr>
</tbody>
</table>

Granting Privileges to Non-Master Users

The following example creates a non-master user named user1 and grants the CREATE SESSION privilege and the SELECT privilege for a database named sh.sales:

```
CREATE USER user1 IDENTIFIED BY password;
GRANT CREATE SESSION TO user1;
GRANT SELECT ON sh.sales TO user1;
```

You can grant explicit object privileges for objects in the SYS schema using the SELECT_CATALOG_ROLE and the EXECUTE_CATALOG_ROLE roles. The SELECT_CATALOG_ROLE role allows users SELECT privileges on data dictionary views and the EXECUTE_CATALOG_ROLE role allows users EXECUTE privileges for packages and procedures in the data dictionary.

The following example grants the SELECT_CATALOG_ROLE role to a user named user1:

```
GRANT SELECT_CATALOG_ROLE TO user1;
```
The following example grants the EXECUTE_CATALOG_ROLE role to a user named user1:

```sql
GRANT EXECUTE_CATALOG_ROLE TO user1;
```

To view the permissions that the SELECT_CATALOG_ROLE and the EXECUTE_CATALOG_ROLE roles allow, use the following query:

```sql
SELECT * FROM ROLE_TAB_PRIVS
WHERE ROLE IN ('SELECT_CATALOG_ROLE', 'EXECUTE_CATALOG_ROLE')
ORDER BY ROLE, TABLE_NAME ASC;
```

## Setting the Database Time Zone

You can alter the time zone of a database in two ways, by either using the `rdsadmin_util.alter_db_time_zone` command or by setting the Oracle Time Zone (p. 166) option. The `rdsadmin_util.alter_db_time_zone` command changes the time zone for only certain data types and does not change SYSDATE, and must be used with versions 11.2.0.2.v4 or later. The `Timezone` option changes the time zone at the host level and impacts all date columns and values such as SYSDATE.

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter database set time_zone = '+3:00'</td>
<td>exec rdsadmin.rdsadmin_util.alter_db_time_zone('+3:00')</td>
</tr>
</tbody>
</table>

After you alter the time zone, you must reboot the DB instance for the change to take effect.

There are additional restrictions on setting time zones listed in the Oracle documentation.

### Working with Automatic Workload Repository (AWR)

If you user Oracle 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. Since you do not have access to host directories, you can download the scripts from Oracle or by generating them using Oracle Enterprise Manager (OEM).

### Creating and Resizing Tablespaces and Data Files

Amazon RDS only supports Oracle Managed Files (OMF) for data files, log files and control files. When creating data files and log files you cannot specify physical file names.

The following example creates a tablespace:

```sql
create tablespace users2;
```

The following example creates temporary tablespace:

```sql
create temporary tablespace users2;
```
create temporary tablespace temp01;

Because the Oracle `ALTER DATABASE` system privilege is not available on Amazon RDS, you must use `ALTER TABLESPACE` to resize a tablespace. The following example resizes a tablespace named `users2` to 200 MB:

```sql
alter tablespace users2 resize 200M;
```

### Adjusting Database Links for Use with DB Instances in a VPC

To use Oracle database links with DB instances inside a VPC, the two instances must be either in the same VPC or you must create an EC2 or VPC security group that both DB instances are a member of. For example, when using Oracle Data Pump and Oracle DBLinks to move data between DB instances, the instances must be members of the same VPC or EC2 security group or they must be in the same VPC. For more information about using database links with Oracle Data Pump, see Oracle Data Pump (p. 150)
Appendix: Using Oracle GoldenGate with Amazon RDS

Topics

• Setting Up an Oracle GoldenGate Hub on EC2 (p. 181)
• Setting Up a Source Database for Use with GoldenGate on Amazon RDS (p. 183)
• Setting Up a Target Database for Use with GoldenGate on Amazon RDS (p. 186)
• Working with Oracle GoldenGate’s Extract and Replicat Utilities (p. 187)
• Troubleshooting Issues When Using Oracle GoldenGate with Amazon RDS (p. 190)

Oracle 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 Oracle 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 Oracle GoldenGate on Amazon RDS:

• Oracle GoldenGate with Amazon RDS is available under the “Bring-your-own-license” model in all AWS regions. You are responsible for the set up and management of GoldenGate on Amazon RDS.
• You can use GoldenGate on Amazon RDS with Oracle Database Standard Edition One (SE1), Standard Edition (SE), and Enterprise Edition (EE).
• The Oracle database version must be 11.2.0.3 (with patch 13328193) or 11.2.0.4, and you must use Oracle GoldenGate version 11.2.1.
• Amazon RDS supports migration and replication across Oracle databases using Oracle 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.
• Oracle GoldenGate DDL is not currently supported.

The Oracle 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 running Oracle Database 11.2.0.3 and with GoldenGate 11.2.1, 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.

Oracle GoldenGate on Amazon RDS supports the following common scenarios:

Scenario 1: An on-premises Oracle source database and on-premises Oracle 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.
Any issues that impact running Oracle 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 11.2.0.3 on an Amazon RDS DB instance.

To set up Oracle 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 an Oracle GoldenGate Hub on EC2 (p. 181)
• Setting Up a Source Database for Use with GoldenGate on Amazon RDS (p. 183)
• Setting Up a Target Database for Use with GoldenGate on Amazon RDS (p. 186)
• Working with Oracle GoldenGate's Extract and Replicat Utilities (p. 187)

Setting Up an Oracle GoldenGate Hub on EC2

There are several steps to creating an Oracle GoldenGate hub on an Amazon EC2 instance. First, you create an EC2 instance with a full installation of Oracle DBMS 11g version 11.2.0.3. The EC2 instance must also have Oracle 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
- Create a GLOBALS parameter file
- Configure 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:

```
prompt$ cd $GGHOME
prompt$ 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 GLOBALS file, then start the manager using `ggsci`:

```
PORT 8199
PurgeOldExtracts ./dirdat/*, UseCheckpoints, MINKEEPDAYS 5
```
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**

There are several differences in the set up steps between a source database running Oracle version 11.2.0.3 and version 11.2.0.4. See the appropriate version for the correct set up steps.

**Topics**

- For Source Databases Running Oracle 11.2.0.3 (p. 183)
- For Source Databases Running Oracle 11.2.0.4 or Later (p. 185)

**For Source Databases Running Oracle 11.2.0.3**

The following tasks set up a source database running version 11.2.0.3 for use with GoldenGate; each task is explained in detail in this section. The tasks include:

- Set the `compatible` parameter to 11.2.0.3.
- Enable supplemental logging.
- 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 11.2.0.3. 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 11.2.0.3 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:

```bash
rds-create-db-parameter-group example-goldengate -d "Parameters to allow GoldenGate" -f oracle-ee-11.2
rds-modify-db-parameter-group example-goldengate -p "name=compatible, value=11.2.0.3, method=pending-reboot"
rds-reboot-db-instance 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 source database must have the supplemental logging parameter enabled. If you are using an Oracle database on an Amazon RDS DB instance as the source database, you can use the following Amazon RDS procedures to enable supplemental logging:

```
exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD');
exec rdsadmin.rdsadmin_util.force_logging(true);
exec rdsadmin.rdsadmin_util.switch_logfile;
```

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 Oracle 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 start_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);
```

### For Source Databases Running Oracle 11.2.0.4 or Later

If your source database is running version 11.2.0.4 or later, there are only 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.

The source database must have the `compatible` parameter set to 11.2.0.3 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 11.2.0.3 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:

```
rds-create-db-parameter-group example-goldengate -d "Parameters to allow GoldenGate" -f oracle-ee-11.2
rds-modify-db-parameter-group example-goldengate -p "name=compatible, value=11.2.0.3, method=pending-reboot"
rds-modify-db-instance example-test -g example-goldengate --apply-immediately
rds-reboot-db-instance 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:

```sql
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 Oracle 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.

```sql
select sum(blocks * block_size) bytes from v$archived_log where start_time>=sysdate-X/24 and dest_id=1;
```

**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; each task is explained in detail in this section. The tasks include:

- 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`:

```sql
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;

Working with Oracle GoldenGate's Extract and Replicat Utilities

The Oracle 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 Oracle 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 (NLS_LANG=AL32UTF8)

USERID oggadm1@TEST, PASSWORD XXXXXX
EXTRAIL /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:

```sql
dblogin userid <user>@<db tnsname>
```

- Add a checkpoint table for the database:

```bash
add checkpointtable
```

- Add transdata to turn on supplemental logging for the database table:

```bash
add trandata <user>.<table>
```

Alternatively, you can add transdata to turn on supplemental logging for all tables in the database:

```bash
add trandata <user>.*
```

- Using the *ggsci* command line, enable the *Extract* utility using the following commands:

```bash
add extract <extract name> tranlog, INTEGRATED tranlog, begin now
add extrail <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 Oracle 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 (NLS_LANG=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 Oracle GoldenGate with Amazon RDS

This section explains the most common issues when using GoldenGate with Amazon RDS.

Topics
  • Using GoldenGate with Amazon EC2 Instances (p. 190)
  • Log Retention (p. 191)

Using GoldenGate with Amazon EC2 Instances

If you are using GoldenGate with an EC2 instance, the EC2 instance must have a full installation of Oracle DBMS 11g version 11.2.0.3. The EC2 instance must also have Oracle GoldenGate 11.2.1 installed, and you must have Oracle patch 13328193 installed. If you do not have these items correctly installed, you will see this error message:

2014-03-06 07:09:21 ERROR  OGG-02021 This database lacks the required libraries to support integrated capture.

To determine what patches you currently have installed, run the command opatch lsinventory on your EC2 instance.
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_9k4bpln6_.log for sequence 1306Not able to establish initial position for begin time 2014-03-06 06:16:55.
```
Appendix: 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 page with the `--character-set` parameter of the `rds-create-db-instance` command or with the `CharacterSetName` parameter of the `CreateDBInstance` API action.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL32UTF8</td>
<td>Unicode 5.0 UTF-8 Universal character set (default)</td>
</tr>
<tr>
<td>AR8ISO8859P6</td>
<td>ISO 8859-6 Latin/Arabic</td>
</tr>
<tr>
<td>AR8MSWIN1256</td>
<td>Microsoft Windows Code Page 1256 8-bit Latin/Arabic</td>
</tr>
<tr>
<td>BLT8ISO8859P13</td>
<td>ISO 8859-13 Baltic</td>
</tr>
<tr>
<td>BLT8MSWIN1257</td>
<td>Microsoft Windows Code Page 1257 8-bit Baltic</td>
</tr>
<tr>
<td>CL8ISO8859P5</td>
<td>ISO 8859-5 Latin/Cyrillic</td>
</tr>
<tr>
<td>CL8MSWIN1251</td>
<td>Microsoft Windows Code Page 1251 8-bit Latin/Cyrillic</td>
</tr>
<tr>
<td>EE8ISO8859P2</td>
<td>ISO 8859-2 East European</td>
</tr>
<tr>
<td>EL8ISO8859P7</td>
<td>ISO 8859-7 Latin/Greek</td>
</tr>
<tr>
<td>EE8MSWIN1250</td>
<td>Microsoft Windows Code Page 1250 8-bit East European</td>
</tr>
<tr>
<td>EL8MSWIN1253</td>
<td>Microsoft Windows Code Page 1253 8-bit Latin/Greek</td>
</tr>
<tr>
<td>IW8ISO8859P8</td>
<td>ISO 8859-8 Latin/Hebrew</td>
</tr>
<tr>
<td>IW8MSWIN1255</td>
<td>Microsoft Windows Code Page 1255 8-bit Latin/Hebrew</td>
</tr>
<tr>
<td>JA16EUC</td>
<td>EUC 24-bit Japanese</td>
</tr>
<tr>
<td>JA16EUCTILDE</td>
<td>Same as JA16EUC except for mapping of wave dash and tilde to and from Unicode</td>
</tr>
<tr>
<td>JA16SJIS</td>
<td>Shift-JIS 16-bit Japanese</td>
</tr>
<tr>
<td>JA16SJISTILDE</td>
<td>Same as JA16SJIS except for mapping of wave dash and tilde to and from Unicode</td>
</tr>
<tr>
<td>KO16MSWIN949</td>
<td>Microsoft Windows Code Page 949 Korean</td>
</tr>
<tr>
<td>NE8ISO8859P10</td>
<td>ISO 8859-10 North European</td>
</tr>
<tr>
<td>NEE8ISO8859P4</td>
<td>ISO 8859-4 North and Northeast European</td>
</tr>
<tr>
<td>TH8TISASCII</td>
<td>Thai Industrial Standard 620-2533-ASCII 8-bit</td>
</tr>
<tr>
<td>TR8MSWIN1254</td>
<td>Microsoft Windows Code Page 1254 8-bit Turkish</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>US7ASCII</td>
<td>ASCII 7-bit American</td>
</tr>
<tr>
<td>UTF8</td>
<td>Unicode 3.0 UTF-8 Universal character set, CESU-8 compliant</td>
</tr>
<tr>
<td>VN8MSWIN1258</td>
<td>Microsoft Windows Code Page 1258 8-bit Vietnamese</td>
</tr>
<tr>
<td>WE8ISO8859P1</td>
<td>Western European 8-bit ISO 8859 Part 1</td>
</tr>
<tr>
<td>WE8ISO8859P15</td>
<td>ISO 8859-15 West European</td>
</tr>
<tr>
<td>WE8ISO8859P9</td>
<td>ISO 8859-9 West European and Turkish</td>
</tr>
<tr>
<td>WE8MSWIN1252</td>
<td>Microsoft Windows Code Page 1252 8-bit West European</td>
</tr>
<tr>
<td>ZHS16GBK</td>
<td>GBK 16-bit Simplified Chinese</td>
</tr>
<tr>
<td>ZHT16HKSCS</td>
<td>Microsoft Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001. Character set conversion is based on Unicode 3.0.</td>
</tr>
<tr>
<td>ZHT16MSWIN950</td>
<td>Microsoft Windows Code Page 950 Traditional Chinese</td>
</tr>
<tr>
<td>ZHT32EUC</td>
<td>EUC 32-bit Traditional Chinese</td>
</tr>
</tbody>
</table>
Appendix: Oracle DB Engine Patch Composition

This section provides information about Amazon RDS patch sets for the Oracle DB Engine.

Topics
- DB Engine Version: 11.2.0.2.v2 (p. 194)
- DB Engine Version: 11.2.0.2.v3 (p. 194)
- DB Engine Version: 11.2.0.2.v4 or 11.2.0.2.v5 (p. 194)
- DB Engine Version: 11.2.0.2.v6 (p. 195)
- DB Engine Version: 11.2.0.2.v7 (p. 196)
- DB Engine Version: 11.2.0.3.v1 (p. 197)
- DB Engine Version: 11.2.0.4.v1 (p. 198)

DB Engine Version: 11.2.0.2.v2

Base line: Oracle Database Patch Set Update (PSU) 11.2.0.2.1

Bugs fixed: 10151017, 10158965, 10080579, 9788588, 10073683, 10077191, 9744252, 9735237, 10248523, 9956713, 10019218, 9715581, 9770451, 9539440, 10022980, 10209232, 10079168, 10013431, 9881076, 10238786, 10040531

DB Engine Version: 11.2.0.2.v3

Base line: Oracle Database Patch Set Update (PSU) 11.2.0.2.3

Bugs fixed: 10151017, 10158965, 11724916, 10190642, 12586486, 12586487, 10129643, 12586488, 12586489, 10018799, 9744252, 10248523, 9956713, 10356513, 9715581, 9770451, 10378005, 10170431, 10425676, 10222719, 10126094, 9591812, 10127360, 10132870, 10094201, 9443361, 9719846, 11664046, 11069199, 10324294, 10245086, 12586490, 10205230, 12586491, 10052141, 12586492, 12586493, 12586494, 10142788, 11818335, 11830776, 12586495, 9905049, 11830777, 12586496, 11830778, 6892311, 10040921, 10077191, 10358019, 12431716, 10219576, 10258337, 11707699, 10264680, 10209232, 11651810, 10102506, 10078765, 9881076, 10278372, 10040531, 10621169, 10155605, 10082277, 10356782, 10218814, 9078442, 9788588, 10157249, 10073683, 10719847, 12326246, 11707302, 10310299, 10636231, 10230571, 11065646, 12419321, 10368698, 10079168, 10013431, 10228151, 10233732, 10324526, 8223165, 10238786, 10217802, 10661015, 9953542, 9572778, 10052956, 10080579, 11699057, 12620422, 10323111, 1022788, 10329146, 10332589, 10110863, 10073683, 9869401, 10019218, 10229719, 11664719, 9539440, 10373381, 9735282, 9748749, 11724984, 10022980, 10411618, 11800854, 12419331, 11674485, 10187168, 6523037, 10648873, 9724970, 10053725, 10084145, 10367188, 11800170, 11695285, 10157402, 9651350, 10299224

DB Engine Version: 11.2.0.2.v4 or 11.2.0.2.v5

Base line: Oracle Database Patch Set Update (PSU) 11.2.0.2.7

Bugs fixed: 10249791, 11877623, 12569737, 14038791, 10026601, 12378147, 10115630, 11814891, 11417510, 10412247, 13923804, 12656535, 9709292, 10220033, 10092858, 12391602, 12321860, 10142857, 10620808, 12573949, 12337012, 12879027, 11811073, 11064851, 13003179, 9903826, 11738259, 14107384, 10207092, 14107385, 11882425, 9858539, 14107386, 14107387, 10633840, 14107388, 10419629, 14107389, 11708510, 10131867, 14040433, 11063191, 13916709, 12880299, 11872103, 12595730, 11056082, 12596444, 13099577, 13632725, 10031806, 13769501, 13769502, 13769503, 13769504, 9744252, 13769505, 9956713, 13769506, 13769507, 9972680, 13769508, 13769509, 11853815, 10635701, 9591812, 10127360, 11723722, 9443361, 12846268, 12846269, 9707965, 10245086, 9401552, 10039731, 11689702, 13769510, 12366627, 10077191, 9829397,
Amazon Relational Database Service User Guide
DB Engine Version: 11.2.0.2.v6

11785938, 10258337, 10264680, 10094823, 10209232, 10284570, 8672862, 9672816, 12830339,
9881076, 10621169, 10048701, 12569482, 9078442, 11057263, 10322959, 12780098, 12976376,
12340939, 11788856, 8223165, 10264696, 10142909, 11800959, 13476583, 10052956, 10285022,
10329146, 10332589, 9895207, 9869401, 12828071, 9285259, 10229719, 11724984, 10411618,
11670161, 9724970, 10113990, 10312847, 11893621, 10200390, 10084145, 10367188, 10285394,
10190642, 12586486, 12586487, 10129643, 12586488, 12917230, 12586489, 11866952, 10232083,
9715581, 10302581, 11690639, 12423475, 11889177, 10126094, 10396041, 10269503, 9970255,
9436324, 12400751, 12589039, 11785390, 12586490, 12586491, 12586492, 9795214, 12586493,
10142788, 12586494, 12586495, 9905049, 12586496, 11674898, 10419984, 6892311, 11815753,
10358019, 12431716, 9906422, 10422126, 13343244, 11937253, 9965655, 11890804, 11651810,
9382956, 11067567, 11716621, 10126822, 9869287, 9375300, 10155605, 10356782, 10326338,
10165083, 10051315, 13696224, 10218814, 13554409, 11076894, 10278773, 11707302, 10230571,
12419321, 9966609, 12633340, 12546006, 10137324, 11894889, 10061015, 9572787, 10284838,
10073683, 12639234, 9578670, 9748749, 10022980, 10237773, 10089333, 12419331, 11674485,
12685431, 10187168, 10648873, 10158965, 11061775, 12635537, 9746210, 10204358, 10356513,
10378005, 10170431, 12639177, 10222719, 10384285, 10035737, 12345717, 9873405, 11069199,
12670165, 10159846, 13257247, 10205230, 10052141, 11818335, 12371955, 12655433, 10040921,
11827088, 10219576, 12408350, 13343424, 11707699, 12370722, 11695333, 11841309, 11924400,
12737666, 12797765, 10281887, 10278372, 10013177, 13503598, 12543639, 10157249, 12531263,
9735237, 10317487, 10219583, 9727147, 10310299, 10636231, 11065646, 10055063, 10368698,
10079168, 11695416, 10233732, 10314582, 9953542, 10080579, 11699057, 12620422, 10427260,
11666137, 10110863, 10363186, 10417716, 10019218, 10388660, 12748240, 9539440, 10373381,
10239480, 10158493, 11842991, 10399808, 10417216, 11695285, 11800170, 10157402, 9651350,
10299224, 10151017, 11724916, 9564886, 9847634, 10018789, 10248523, 11694127, 10630870,
9770451, 10425676, 9683047, 10180307, 9835264, 10132870, 10094201, 10193846, 11664046,
10324294, 9414040, 9819805, 11830776, 11830777, 11830778, 11683713, 10200404, 10102506,
12827726, 11733179, 10229886, 10040531, 10082277, 9788588, 12326246, 12397410, 10622001,
13468884, 13386082, 10040035, 12539000, 11867127, 9842573, 9771278, 10013431, 10228151,
10324526, 12417369, 10238786, 10217802, 10332111, 10227288, 10623249, 9943960, 10021022,
9824435, 11664719, 12950644, 9735282, 11800854, 10097711, 11858315, 6523037, 10053725, 8685446

DB Engine Version: 11.2.0.2.v6
Base line: Oracle Database Patch Set Update (PSU) 11.2.0.2.8
Bugs fixed: 13250244, 13737746, 11063821, 12409916, 14461356, 14461357, 11878443, 14461358,
14683459, 14275621, 14467061, 10114837, 12649442, 10207551, 12794305, 14473913, 10171273,
10373013, 10210507, 11883472, 13080778, 10172453, 14624146, 14613900, 10213073, 9373370,
9478199, 9877980, 10021111, 10228393, 12899768, 12713993, 9470768, 14390377, 10140809,
12894807, 11686968, 12374212, 12764337, 12326708, 9956835, 11734067, 7312717, 11775474,
12834027, 13326736, 9952554, 10249791, 11877623, 12569737, 14038791, 10026601, 12378147,
10115630, 11814891, 14127510, 10412247, 13923804, 12656535, 9709292, 10220033, 10092858,
12391602, 12323180, 10142857, 10620808, 12579349, 12337012, 12879027, 11811073, 11064851,
13001379, 9903826, 11738259, 14107384, 10207092, 14107385, 11882425, 9858539, 14107386,
14107387, 10633840, 14107388, 10419629, 14107389, 11708510, 10131867, 14040433, 11063191,
13916709, 12880299, 11872103, 12595730, 11056082, 12596444, 13099577, 13632725, 10031806,
13769501, 13769502, 13769503, 13769504, 9744252, 13769505, 9956713, 13769506, 13769507,
9972680, 13769508, 13769509, 11853815, 10635701, 9591812, 10127360, 11723722, 9443361,
12846268, 12846269, 9707965, 10245086, 9401552, 10039731, 11689702, 13769510, 12366627,
10077191, 9829397, 11785938, 10258337, 10264680, 10094823, 10209232, 10284570, 8672862,
9672816, 12830339, 9881076, 10621169, 10048701, 12569482, 9078442, 11057263, 10322959,
12780098, 12976376, 12340939, 11788856, 8223165, 10264696, 10142909, 11800959, 13476583,
10052956, 10285022, 10329146, 10332589, 9895207, 9869401, 12828071, 9285259, 10229719,
11724984, 10411618, 11670161, 9724970, 10113990, 10312847, 11893621, 10200390, 10084145,
10367188, 10285394, 10190642, 12586486, 12586487, 10129643, 12586488, 12917230, 12586489,
11866952, 10232083, 9715581, 10302581, 11690639, 12423475, 11889177, 10126094, 10396041,
10269503, 9970255, 9436324, 12400751, 12589039, 11785390, 12586490, 12586491, 12586492,

API Version 2013-09-09
195


DB Engine Version: 11.2.0.2.v7

Base line: Oracle Database Patch Set Update (PSU) 11.2.0.2.10 (April2013)

Bugs fixed: 16344871, 9671271, 16294412, 14841558, 12579464, 16056267, 10435074, 14273397, 12487971, 12314102, 10138589, 14841812, 12842402, 16303117, 10732924, 12539487, 12594032, 13377816, 16033114, 16175381, 14220720, 13561951, 9868876, 9913542, 16303114, 10362871, 9801919, 12755116, 15324899, 16303115, 10350832, 16303118, 12582664, 13596521, 14459552, 13810393, 13147164, 15896341, 10247152, 14076523, 10395345, 14236366, 13467683, 11706168, 15896247, 14263073, 9926929, 10190172, 11715084, 15896432, 9896536, 15896428, 15896429, 14841437, 14240002, 14262913, 13399435, 10369874, 8547978, 14727315, 15896434, 14546575, 9860769, 14258925, 15896433, 14546638, 18344448, 14741727, 14546673, 12854115, 15896430, 12595561, 13550185, 14263036, 9912965, 14205448, 15896435, 10350825, 12848798, 11856395, 10175192, 14469090, 12381357, 9233544, 9811335, 13250244, 13737746, 11063821, 12409916, 14461356, 14461357, 11878443, 14461358, 14683459, 14275621, 14467061, 11014837, 12649442, 12027551, 12794930, 14479913, 10172373, 10301057, 11883472, 13087077, 10172453, 14624146, 14613900, 10213073, 9373370, 9478199, 9877980, 10021111, 10228393, 12899768, 12713993, 9470768, 14390377, 10140809, 12894807, 11686968, 12374326, 12326708, 9956835, 11734067, 7312717, 11775474, 12834027, 13236736, 9952554, 10249791, 11877623, 12596977, 14038791, 10026601, 12371847, 10115630, 11841491, 14271250, 10412247, 13923804, 12656535, 9709292, 10220033, 10092858, 12391602, 12323180, 10142857, 10620808, 12579349, 13373012, 12879027, 11811073, 11684851, 13001379, 9903826, 11738259, 14107384, 10207092, 14107385, 11882425, 9858539, 14107386, 14107387, 10633840, 14107388, 10419629, 14107389, 11708510, 10131867, 14044043, 11063191, 13916709, 12880299, 11872103, 12559730, 11056082, 12596444, 13099577, 13632572, 10031806, 13769501, 13769502, 13769503, 13769504, 9744252, 13769505, 9956713, 13769506, 13769507, 9972680, 13769508, 13769509, 11853815, 10632871, 12892393, 10045152, 10039731, 11667980, 13769510, 12366627, 10077191, 9829397, 11785938, 10258337, 10264680, 1004823, 10209232, 10284570, 8672862, 9672816, 12303039, 9881076, 10261199, 10048701, 12569482, 9078442, 11057263, 10322559, 12780098, 12976376, 12340393, 11788856, 8223165, 10264936, 10142909, 11809059, 13476583, 10258205, 10329146, 10332589, 9885937, 9869001, 12885071, 9282559, 10229719, 11724984, 10411618, 11670161, 9724970, 10131990, 10312847, 11893621, 10200390, 10084145, 10367188, 10285394, 10190642, 12586846, 12586848, 12586848,
DB Engine Version: 11.2.0.3.v1

Base line: Oracle Database Patch Set Update (PSU) 11.2.0.3.7 (July2013)

Bugs fixed: 13593999, 13566938, 10350832, 14138130, 12919564, 13561951, 13624984, 13588248, 13080579, 12873183, 13645875, 14472647, 12880299, 14646355, 12998795, 14409183, 13719081, 14469088, 13492735, 14263036, 12857027, 13496884, 13015379, 12463073, 13742433, 13732266, 16314469, 16368108, 12905058, 6690853, 13742434, 12849688, 12950644, 13742435, 13640024, 11664046, 10342924, 94141040, 9819005, 11830776, 11830778, 11683713, 10020040, 10102506, 12827726, 11733179, 10229886, 10040531, 7985858, 12326246, 12397410, 10622001, 13468884, 13386082, 10040035, 11724916, 9842573, 11858315, 6523037, 10053725, 8685446

API Version 2013-09-09

197
DB Engine Version: 11.2.0.4.v1

Base line: Oracle Database Patch Set Update (PSU) 11.2.0.4.1 (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
Microsoft SQL Server on Amazon RDS

Amazon RDS supports DB instances running several editions of Microsoft SQL Server 2008 R2 and SQL Server 2012. 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.

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 will have all database-level permissions except for those that are used for backups (Amazon RDS manages backups for you).

Common Management Tasks for SQL Server on Amazon RDS

These are the common management tasks you perform with an Amazon RDS SQL Server DB instance, with links to information about each task:

- For planning information, such as SQL Server versions, storage engines, security, and features supported in Amazon RDS, see Amazon RDS SQL Server Planning Information (p. 201).
- If you are creating a DB instance for production purposes, you should understand how instance classes, storage, and Provisioned IOPS work in Amazon RDS. For more information about DB instance classes, see DB Instance Class (p. 47) For more information about Amazon RDS storage, see Amazon RDS Storage (p. 48). For more information about Provisioned IOPS, see Working with Provisioned IOPS Storage (p. 361).
- 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 ports you will use to access the DB instance. The security group you need to create will depend on what platform you DB instance is on, and whether you will be accessing your DB instance from an EC2 instance. For more information about the two
platforms supported by Amazon RDS, EC2-VPC and EC2-Classic, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 354). In general, if your DB instance is on the EC2-Classic 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. For more information about security groups, see Amazon RDS Security Groups (p. 74).

- If your AWS account has a default VPC (a default virtual private network), then your DB instance will automatically be created inside the default VPC. If your account does not have a default VPC and you want the DB instance to be inside a VPC, you must create the VPC before you create the DB instance. For more information about determining if your account has a default VPC, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 354). For more information about using VPCs with Amazon RDS, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 353).

- If your DB instance is going to require specific database parameters or options, you should create the parameter or option groups before you create the DB instance. For more information on parameter groups, see Working with DB Parameter Groups (p. 317). For more information on options for SQL Server, see Appendix: Options for SQL Server DB Engine (p. 236). For more information about creating a DB instance, see Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 207).

- After creating a security group and assigning 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. For more information on connecting to a DB instance, see Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 215).

- A newly created SQL Server DB instance does not contain a database. The instance has one master user account with the name and password you specified when you created the DB instance that you can use to create users and grant permissions. You must use a SQL Server tool such as SQL Server Management Studio or a utility to log in as the masteruser, and then use SQL Server commands and SQL statements to add the users and elements required by your applications to store and retrieve data in the DB instance.

- To import SQL Server data into a DB instance, follow the information in the Importing Data into SQL Server on Amazon RDS (p. 222) section. You cannot use the BACKUP and RESTORE commands to import data into a DB instance because Amazon RDS does not allow OS-level access that would enable you to place files in a physical location that the database engine could access. You also cannot import data using the Copy Database Wizard in SQL Server Management Studio because the tool requires sysadmin privilege on the source and destination servers and this permission is not available to the master user account for a DB instance.

- You may need to periodically modify your DB instance, such as to resize, reconfigure, or add more storage. For information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 220). For additional information on specific tasks, see:
  - Renaming a DB Instance (p. 271)
  - Deleting a DB Instance (p. 273)
  - Rebooting a DB Instance (p. 276)
  - Tagging Amazon RDS Resources (p. 278)
  - Upgrading a DB Instance (p. 288)
  - Adjusting the Preferred Maintenance Window (p. 373)

- You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots. For information, see Backing Up and Restoring a DB Instance (p. 292).

- You can monitor an instance through actions such as viewing the MySQL logs, CloudWatch Amazon RDS metrics, and events. For information, see Monitoring a DB Instance (p. 376).

- There are several Amazon RDS features you can use with SQL Server DB instances that are common across all the Amazon RDS database engines. For more information, see:
  - Working with Reserved DB Instances (p. 342)
  - Working with Provisioned IOPS Storage (p. 361)
There are also several appendices with useful information about working with Amazon RDS SQL Server DB instances:

- For information on common DBA tasks for SQL Server on Amazon RDS, see Appendix: Common DBA Tasks for Microsoft SQL Server (p. 229).
- For information on the options that you can use with SQL Server on Amazon RDS, see Appendix: Options for SQL Server DB Engine (p. 236).

## Amazon RDS SQL Server Planning Information

You can choose the version of SQL Server you want to have on your DB instance. Amazon RDS supports DB instances running several editions of Microsoft SQL Server 2008 R2 and SQL Server 2012. You should also be aware of the limits for SQL Server DB instances.

### Topics

- General Limits for SQL Server DB Instances (p. 201)
- SQL Server DB Engine Features (p. 240)
- SQL Server Licensing (p. 204)
- DB Engine Version Management (p. 205)
- Supported SQL Server Roles and Permissions (p. 205)
- Using SSL with a SQL Server DB Instance (p. 206)
- Using the TDE option to encrypt data at rest (p. 207)

### General Limits for SQL Server DB Instances

The Amazon RDS implementation of SQL Server on a DB instance has some limitations you should be aware of:

- The maximum number of databases on a single Microsoft SQL Server DB Instance is 30.
- Databases cannot be renamed.
- The maximum storage size for a Microsoft SQL Server DB Instance is 1024 GB for all instances except the SQL Server Express edition, which limits storage to 10 GB per database.
- Amazon RDS reserves up to 40 connections for system maintenance. If you specify a value for the user connections parameter, you will need to add 40 to the number of connections that you expect to use.
- Because of the extensibility limitations of striped storage attached to Windows Server, Amazon RDS does not currently support increasing storage on a SQL Server DB Instance. We recommend that you provision storage according to anticipated future storage growth. If you need to increase the storage of a SQL Server DB Instance, you will need to export the data, create a new DB Instance with increased storage, and then import the data into the new DB Instance. For more information, go to the RDS SQL Server Data Migration Guide.
- The minimum storage size for a Microsoft SQL Server DB Instance is 20 GB for the Microsoft SQL Server Express and Web Editions and 200 GB for the Standard and Enterprise Editions.
- DROP DATABASE is supported, but because of limitations in Microsoft SQL Server, restoring to a point in time after successful execution of a DROP DATABASE command might not reflect the dropping of a database. For example, the dropped database will typically be restored to its state up to 5 minutes before the DROP DATABASE command was issued. To work around this, you can reissue the DROP DATABASE command after the restore operation is completed.
- In addition, restoring to a point in time before successful execution of a DROP DATABASE may not reflect the state of that database at that point in time. For example, the dropped database will typically be restored to its state up to 5 minutes before the DROP DATABASE command was issued, which
means that you will not be able to restore the transactions made during those few minutes on your dropped database.

- You can run other SQL Server components that are not supported with Amazon RDS in an Amazon EC2 instances with EBS storage, pursuant to Microsoft licensing policies. This includes components such as SQL Server Analysis Services, SQL Server Integration Services, SQL Server Reporting Services, Data Quality Services, and Master Data Services.
- Note that the following ports are reserved for Amazon RDS use and cannot be used when creating a DB instance for SQL Server: 1434, 3389, 47001, and 49152 through 49156.

# SQL Server DB Engine Features

Currently, Amazon RDS supports database engine components for the SQL Server 2008 R2 and SQL Server 2012 editions. For a list of supported features, click the edition link below.

**Topics**
- SQL Server 2008 R2 Supported Features on Amazon RDS (p. 202)
- SQL Server 2012 Supported Features on Amazon RDS (p. 203)

# SQL Server 2008 R2 Supported Features on Amazon RDS

The following list shows a subset of the key database engine features that are currently supported by the 2008 R2 version of SQL Server. For a complete list of features supported by the 2008 R2 SQL Server database engine, go to [Features Supported by the Editions of SQL Server](#).

- 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
- SSL
- Transparent Data Encryption (Enterprise Edition only)
- Spatial and location features

Microsoft SQL Server database features that are not currently supported include the following:

- Maintenance Plans
- Database Mail
- Distributed Queries (i.e., Linked Servers) - only supported for DB instances not in a VPC
- Service Broker
- Database Log Shipping
- Database Mirroring
- Windows Authentication
- Replication
- The ability to run Reporting, Analysis, Integration, or Master Data Services on the same server as the DB Instance
- Performance Data Collector
- Additional T-SQL endpoints
- Distribution Transaction Coordinator (MSDTC)
- WCF Data Services
- FILESTREAM support
- Policy-Based Management
- SQL Server Audit
- BULK INSERT and OPENROWSET(BULK...) features

SQL Server 2012 Supported Features on Amazon RDS

Amazon RDS supports the initial release (RTM) version of SQL Server 2012, version 11.00.2100.60. For more information on SQL Server 2012, see What’s New in SQL Server 2012.

The following list shows a subset of the key database engine features that are currently supported in SQL Server 2012:

- 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
- 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

Some features in SQL Server 2012 are not currently supported, including some features not supported for version 2008 R2. These include the following:

- Data Quality Services
- Master Database Services
- Always On (available in 2012 Enterprise Edition)
- File Tables
- SQL Server Audit (now available in all editions in 2012)
- Database Log Shipping
- Database Mirroring
- WCF Data Services
- FILESTREAM support
- Policy-Based Management
- SQL Server Audit
- Maintenance Plans
- Database Mail
• Service Broker
• Windows Authentication
• Replication
• The ability to run Reporting, Analysis, Integration, or Master Data Services on the same server as the DB instance
• Performance Data Collector
• Additional T-SQL endpoints
• Distribution Transaction Coordinator (MSDTC)
• BULK INSERT and OPENROWSET(BULK...) features

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, go to Contained Databases.

Since the initial release of SQL Server 2012, there have been several cumulative updates and one service pack. Amazon RDS plans to release a minor version upgrade of SQL Server 2012 in the coming months that will include new cumulative update patches and any released service packs.

The following table shows the license models that are supported for each SQL Server 2012 engine type.

<table>
<thead>
<tr>
<th>SQL Server 2012 Engine Type</th>
<th>license-included</th>
<th>bring-your-own-license</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Edition</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Standard Edition</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Web Edition</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Express Edition</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**SQL Server Licensing**

Currently, Amazon RDS offers two licensing options for SQL Server, License Included and License Mobility (Bring Your Own License). This section explains each.

**Note**

In accordance with Microsoft’s usage rights, SQL Server Web Edition can be used only to support public and Internet-accessible web pages, websites, web applications, and Web services. For more information, go to AWS Service Terms.

**License Included**

Amazon RDS uses the License Included service model for DB Instances running the Microsoft SQL Server Express Edition, Microsoft SQL Server Web Edition, and Microsoft SQL Server Standard Edition (SE). In this model, the license is held by AWS and is included in the price of the DB Instance.
License Mobility (Bring Your Own License)

Microsoft’s License Mobility program allows Microsoft customers to easily move current on-premises Microsoft Server application workloads to Amazon Web Services (AWS), without any additional Microsoft software license fees. This benefit is available to Microsoft Volume Licensing (VL) customers with eligible server applications covered by active Microsoft Software Assurance (SA) contracts. Currently, Microsoft SQL Server Standard Edition and Microsoft SQL Server Enterprise Edition are the eligible Database editions for this program. Refer to Microsoft’s Product Use Rights for the latest licensing terms.

DB Engine Version Management

With Amazon RDS, you can control when to upgrade your SQL Server 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.

Unless you specify otherwise, your DB Instance will automatically be upgraded to new SQL Server minor versions as they are supported by Amazon RDS. This patching will occur during your scheduled maintenance window, and it will be announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades, set the AutoMinorVersionUpgrade parameter for your DB instance to false.

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 (p. 288).

Note
Amazon RDS periodically aggregates official Microsoft SQL Server database patches and assigns an Amazon RDS-specific DB Engine version. The current supported versions are SQL Server 2008 R2 Service Pack 1 and SQL Server 2012.

Major Version Change: Upgrading from 2008 R2 to 2012

Amazon RDS supports major version upgrades from Microsoft SQL Server 2008 R2 to SQL Server 2012. You perform the upgrade by using the Amazon RDS modify DB instance operation. You should thoroughly test any major version upgrade before upgrading your production instances. For information about upgrading a DB instance, see Upgrading a DB Instance (p. 288)

Supported SQL Server Roles and Permissions

The SQL Server database engine uses role-based security. A role is a collection of privileges that can be granted to or revoked from a user.

Any user who creates a database will be assigned to the db_owner role for that database and will have 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 a SQL Server DB Instance:

- ADMINISTER BULK OPERATIONS
- ALTER ANY CREDENTIAL
- ALTER ANY EVENT NOTIFICATION
- 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. 317).
- AUTHENTICATE 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)

Using SSL with a SQL Server DB Instance

Amazon RDS supports SSL encryption for SQL Server DB Instances. Using SSL, you can encrypt a SQL Server connection between your applications and your SQL Server DB Instances. SSL support is available in all AWS regions for all SQL Server editions including Express, Web, Standard and Enterprise.

To use a SQL Server DB Instance over SSL, follow these general steps:

2. Import the certificate into your Windows operating system. For more information on importing a certificate, see How to Import a Trusted Root Certification Authority in Windows 7/Vista/XP.
3. Do one of the following:
   a. Connect to your SQL Server DB Instance over SSL by appending "encrypt=true" to your connection string. If you are connecting to a SQL Server DB Instance in a VPC, you must also append the connection string with "TrustServerCertificate=true".
   b. If you use SQL Server Management Studio to connect to your SQL Server DB Instance, do the following. For more information on using the SQL Server Management Studio, see Use SQL Server Management Studio.
      i. Launch SQL Server Management Studio.
      ii. Enter the server information, login user name, and password.
      iii. Click Option>>.
      iv. Click the Encrypt connection check box.
      v. If you are connecting to a SQL Server DB Instance in a VPC, click the Additional Connection Parameters tab and enter "TrustServerCertificate=true" in the text box.
      vi. Click Connect.
To display the encrypted status of your connection, run the following SQL query:

```
SELECT encrypt_option FROM sys.dm_exec_connections WHERE session_id = @@SPID
```

### Using the TDE option to encrypt data at rest

Most Amazon RDS DB engines support option groups that allow you to select additional features for your DB instance. SQL Server support includes the TDE option, which automatically encrypts stored data for SQL Server 2008 R2 Enterprise Edition and SQL Server 2012 Enterprise Edition. For more information about SQL Server TDE, see SQL Server Transparent Data Encryption (p. 236). For more information about working with option groups, see Working with Option Groups (p. 306).

### Working with a SQL Server DB Instance

#### Topics
- Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 207)
- Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 215)
- Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 220)
- Importing Data into SQL Server on Amazon RDS (p. 222)

Most tasks you need to perform on a DB instance are performed the same way for all DB engines. Creating a DB instance, connecting to that DB instance, and importing data into that DB instance are all tasks that are specific for each DB engine. In addition, the appendix in this section contains important information on working with SQL Server DB instances.

### Creating a DB Instance Running the Microsoft SQL Server Database Engine

The basic building block of Amazon RDS is the DB instance. This is the environment in which you will use to run your SQL Server databases.

In this example, you create a DB instance running the Microsoft SQL Server database engine called `sqlsv-instance1`, with a `db.m1.small` DB Instance class, 200 GB of storage, and automated backups enabled with a retention period of one day.

### AWS Management Console

#### To create a DB Instance Running the Microsoft SQL Server Database Engine

1. Sign in to the AWS Management Console and open the Amazon RDS console at [https://console.aws.amazon.com/rds/](https://console.aws.amazon.com/rds/).
2. In the top right corner of the Amazon RDS console, select the region in which you want to create the DB instance.
3. In the navigation pane, click DB Instances.
4. Click **Launch DB Instance** to start the Launch DB Instance Wizard.

The wizard opens on the **Engine Selection** page. The SQL Server editions available will vary by region.
5. In the Engine Selection window, click the Select button for the SQL Server DB engine you want to use.

6. The next step asks if you are planning to use the DB instance you are creating for production. If you are, select Yes. By selecting Yes, the failover option Multi-AZ and the Provisioned IOPS storage option will be preselected in the following step. Click Next Step when you are finished.

7. On the DB Instance Details page, specify your DB instance information. Click Next Step when you are finished.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Model</td>
<td>Select the licensing model you want to use. Select License Included to use the general license agreement for Microsoft SQL Server that is included with your DB instance, or select Bring Your Own License to use your existing license. Each licensing model may not be available for all versions or in all regions.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>Select the version of Microsoft SQL Server you want to use. DB engine version SQL Server 2012 11.00.210... (default) is the default version of SQL Server for Amazon RDS.</td>
</tr>
<tr>
<td>For this parameter...</td>
<td>...Do this:</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>DB Instance Class</strong></td>
<td>Select a configuration for your DB instance. For example, a <code>db.m1.small</code> instance class equates to 1.7 GB 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. 47).</td>
</tr>
<tr>
<td><strong>Multi-AZ Deployment</strong></td>
<td>This parameter determines if you want to create a standby replica of your DB instance in another availability zone for failover support. This feature is only available for Oracle, PostgreSQL, and MySQL DB instances. For more information about multiple availability zones, see [Regions and Availability Zones](p. 50).</td>
</tr>
<tr>
<td><strong>Auto Minor Version Upgrade</strong></td>
<td>Select <strong>Yes</strong> if you want to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.</td>
</tr>
<tr>
<td><strong>Allocated Storage</strong></td>
<td>Type a value to allocate storage 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 about storage allocation, see [Amazon RDS Storage](p. 48).</td>
</tr>
<tr>
<td><strong>Use Provisioned IOPS</strong></td>
<td>Determine if you want to use Provisioned IOPS (I/O operations per second), a high-performance storage option in RDS that is optimized for I/O-intensive, transactional (OLTP) database workloads. For more information about high performance storage, see [Working with Provisioned IOPS Storage](p. 361).</td>
</tr>
<tr>
<td><strong>DB Instance Identifier</strong></td>
<td>Type a name for the DB instance of 15 alphanumeric characters or less that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB Engine you selected, such as <code>sqlsv-instance1</code>.</td>
</tr>
<tr>
<td><strong>Master User Name</strong></td>
<td>Type a name that you will use as the master username to log on to your DB Instance with all database privileges. The master username 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.</td>
</tr>
<tr>
<td><strong>Master User Password</strong></td>
<td>Type a password that contains from 8 to 128 printable ASCII characters (excluding /,, and @) for your master user password.</td>
</tr>
</tbody>
</table>
8. On the **Additional Configuration** page, provide additional information that Amazon RDS needs to launch the SQL Server DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click Next Step.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Port</td>
<td>Specify a port you want to access the database through. SQL Server installations default to port 1433. This must be the same port value you provided when creating the security group that will provide access the DB instance.</td>
</tr>
<tr>
<td>Choose a VPC</td>
<td>Determine if you want to create the DB instance in a Virtual Private Cloud. For more information about VPC, see Amazon RDS and the Amazon Virtual Private Cloud Service (p. 54).</td>
</tr>
<tr>
<td>Publicly Accessible</td>
<td>(Only applies if you choose a VPC) Select Yes to give the DB instance a public IP address; otherwise, select No. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.</td>
</tr>
<tr>
<td>Availability Zone</td>
<td>Determine if you want to specify a particular Availability Zone.</td>
</tr>
</tbody>
</table>
For this parameter... | ...Do this:
--- | ---
**Option Group** | Select an option group. You can choose the default option group or you can create an option group and select that option group. For more information about option groups, see Working with Option Groups (p. 306).

**Parameter Group** | Select a DB parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 317).

**Security Groups** | Select a security group. For more information about security groups, see Working with DB Security Groups (p. 329).

---

### Launch DB Instance Wizard

Provide the optional additional configuration details below.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Port</td>
<td>1433</td>
</tr>
<tr>
<td>Choose a VPC</td>
<td>Not in VPC (Only VPCs with a DB Subnet Group(s) are allowed)</td>
</tr>
<tr>
<td>Availability Zone</td>
<td>No Preference</td>
</tr>
<tr>
<td>Option Group</td>
<td>default.sqlserver-se-11-00</td>
</tr>
<tr>
<td>Parameter Group</td>
<td><a href="#">default</a></td>
</tr>
<tr>
<td>DB Security Group(s)</td>
<td><a href="#">default</a></td>
</tr>
</tbody>
</table>

If you have custom DB Parameter Groups or DB Security Groups you would like to associate with this DB Instance, select them below, otherwise proceed with default settings.

When all the settings are as you want them, click **Next Step**.

---

9. On the **Management Options** page, you can specify backup and maintenance options for your DB instance. For more information about the maintenance window, see Adjusting the Preferred Maintenance Window (p. 373). For more information on backups and the backup retention period, see DB Instance Backups (p. 55).
10. On the Review page, review the options for your DB instance. If you need to make any changes, click Previous to return to the appropriate page, and then make the necessary corrections. When all the settings are as you want them, click Launch DB Instance.

11. On the final page of the wizard, click Close.

12. 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.

**CLI**

To create a DB Instance Running the Microsoft SQL Server Database Engine

- Use the command `rds-create-db-instance` to create a DB Instance.
This command should produce output similar to the following:

```
PROMPT>rds-create-db-instance mymsftsqlserver -s 250 -c db.m1.large -e sqlserver-se
   -u <masterawsuser> -p <masteruserpassword> --backup-retention-period 3
```

```
DBINSTANCE  mymsftsqlserver  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 DB Instance

- Call the CreateDBInstance action. For example, you could use the following parameters:

  * **DBInstanceIdentifier** = mymsftsqlserver
  * **Engine** = sqlserver-se
  * **DBInstanceClass** = db.m1.large
  * **AllocatedStorage** = 250
  * **BackupRetentionPeriod** = 3
  * **MasterUsername** = <masterawsuser>
  * **MasterUserPassword** = <masteruserpassword>
Example

https://rds.amazonaws.com/
?Action=CreateDBInstance
&AllocatedStorage=250
&BackupRetentionPeriod=3
,DBInstanceClass=db.m1.large
,DBInstanceIdentifier=mymsftsqlserver
,DBName=mydatabase
,DBSecurityGroups.member.1=mysecuritygroup
,DBSubnetGroup=mydbsubnetgroup
,Engine=sqlserver-se
,MasterUserPassword=<masteruserpassword>
,MasterUsername=<masterawsuser>
,SignatureMethod=HmacSHA256
,SignatureVersion=4
,Version=2013-09-09
,X-Amz-Algorithm=AWS4-HMAC-SHA256
,X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-2/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

Related Topics

- DB Instance (p. 43)
- Amazon RDS Security Groups (p. 74)
- Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 215)
- DB Instance Class (p. 47)
- Deleting a DB Instance (p. 273)
Connecting to a DB Instance Running the Microsoft SQL Server Database Engine

Once Amazon RDS provisions your DB Instance, you can use any standard SQL client application to connect to the instance. In this example, you connect to a DB Instance running the Microsoft SQL Server database engine using the Microsoft SQL Server command line tools. For information on connecting to a Microsoft SQL Server DB Instance using SSL, see Using SSL with a SQL Server DB Instance (p. 206). For more information on using Microsoft SQL Server, go to the Microsoft SQL Server website.

Note
This example uses the Microsoft SQL Server Management Studio utility. This utility is part of the Microsoft SQL Server software distribution. To download a stand-alone version of this utility, go to the Microsoft Download Center - Microsoft SQL Server Management Studio Express.

Microsoft SQL Server Management Studio

To connect to a DB Instance using Microsoft SQL Server Management Studio

1. Find the DNS name for your DB Instance
   a. On the My DB Instances page of the AWS Management Console, select the check box next to the DB instance running the Microsoft SQL Server database engine. In the **Description** tab of the lower panel, note the endpoint and port of the DB instance.

   ![My DB Instances](image)

   **Port:** 143

   **Endpoint:** mymsftsqlserver.215dowmc.mysrds.amazonaws.com

   b. To find the DNS name using the “new look” RDS console UI.
On the DB instances page of the AWS Management Console, select the arrow in the line for the DB instance running the Microsoft SQL Server database engine.

c. In the Summary information, note the endpoint of the DB instance to use in the next step.

2. Run Microsoft SQL Server Management Studio.
3. The Connect to Server dialog box appears.

4. In the Server type: drop-down list box, select Database Engine.
5. In the Server name: text field, enter or paste the endpoint of the DB instance running the Microsoft SQL Server database engine. Replace the colon separating the port number with a comma. For example, the Server name could be: sqlsvr-pdz.c6c8mdfntzgv0.region.rds.amazonaws.com,1433.
6. From the Authentication drop-down list box, select SQL Server Authentication.
7. Enter the master user name for the DB instance in the Login: text box.
8. Enter the password for the master user in the Password: text box.
9. Click the Connect button.
After a few moments, Microsoft SQL Server Management Studio should be connected to your DB instance.

10. Click the **New Query** button at the top left of the SQL Server Management Studio window. A new SQL Query window will open.

11. Type the following SQL query:

```
select @@VERSION
```

12. Click the ! Execute button on the SQL Enterprise Manager toolbar to run the query.

You should see a version string returned from your Microsoft SQL Server DB instance displayed in the output window.
Troubleshooting a Connection to a DB Instance Running SQL Server

There are two common causes for problems if you are having problems connecting to a SQL Server DB instance: the access rules enforced by your firewall and the IP addresses you authorized to access the DB instance in the DB security group (or VPC security group if your DB instance is inside a VPC). If you used Microsoft SQL Server Management Studio and you followed the settings specified in the steps above and you are unable to connect, the problem is most likely the egress or ingress rules on your firewall. If you cannot send out or receive communications over the port you specified when you created the DB instance, you will not be able to connect to the DB instance. Check with your network administrator to determine if the port you specified for your DB instance is allowed to be used for inbound and outbound communication.

Here are a few things to check if you know that you can send and receive communications through your firewall for the port you specified when you created the DB instance.

- **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) could 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** - You were able to reach the DB instance but the connection was refused. This is often caused by the user name or password being incorrect.

**Related Topics**

- DB Instance (p. 43)
- Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 207)
- Amazon RDS Security Groups (p. 74)
- Deleting a DB Instance (p. 273)
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 adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS SQL Server DB instance, and describes the settings for SQL Server instances. For information about additional tasks, such as renaming, rebooting, deleting, tagging, or upgrading an Amazon RDS DB instance, see Tasks Common to All Amazon RDS DB Engines (p. 270). 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 a DB Instance and Using the Apply Immediately Parameter (p. 45).

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, click DB Instances.
3. Select the check box for the DB instance that you want to change, and then click Modify.
4. In the Modify DB Instance dialog box, change any of the following settings that you want:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Instance Identifier</td>
<td>You can rename the DB instance by typing a new name.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>In the list provided, click the version of the SQL Server database engine that you want to use.</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>In the list provided, click the DB instance class that you want to use. For information about instance classes, see the section called “DB Instance Class” (p. 47).</td>
</tr>
<tr>
<td>Multi-AZ Deployment</td>
<td>Multi-AZ deployment is not available with SQL Server DB instances.</td>
</tr>
<tr>
<td>Auto Minor Version Upgrade</td>
<td>If you want your DB instance to receive minor engine version upgrades automatically when they become available, click Yes. Upgrades are installed only during your scheduled maintenance window.</td>
</tr>
<tr>
<td>Allocated Storage</td>
<td>You cannot change the allocated storage for a SQL Server DB instance.</td>
</tr>
<tr>
<td>Provisioned IOPS</td>
<td>You cannot change the Provisioned IOPS setting for a SQL Server DB instance. For more information on Provisioned IOPS, see Working with Provisioned IOPS Storage (p. 361).</td>
</tr>
<tr>
<td>Parameter Group</td>
<td>Select the parameter group you want associated with the DB instance. For more information about parameter groups, see Working with DB Parameter Groups (p. 317).</td>
</tr>
</tbody>
</table>
Select the security group you want associated with the DB instance. For more information about security groups, see Working with DB Security Groups (p. 329).

Select the option group you want associated with the DB instance. For more information about option groups, see Working with Option Groups (p. 306).

Type a password for your master user. The password must contain from 8 to 30 alphanumeric characters.

Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0.

Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC) and a duration in hours.

Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.

5. To apply the changes immediately, select the **Apply Immediately** check box. Selecting this option can cause an outage in some cases; for more information on the impact of the Apply Immediately option, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 45).

6. When all the changes are as you want them, click **Yes, Modify**. If instead you want to cancel any changes that you didn’t apply in the previous step, click **Cancel**.

### CLI

To modify an SQL Server DB instance

- Use the command `rds-modify-db-instance`.

### API

To modify an SQL Server DB instance

- Use the `ModifyDBInstance` action.
Importing Data into SQL Server on Amazon RDS

If you have an existing Microsoft SQL Server 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 are transferring. For example, a database that contains data sets on the order of gigabytes, along with stored procedures and triggers, is going to be more complicated than a simple database with only a few megabytes of test data and no triggers or stored procedures.

RDS for SQL Server service does not currently support RESTORE DATABASE ... FROM FILE, because the database and log file backups must be local to the SQL Server instance. Similarly, FILESTREAM is also not supported at this time.

The BULK INSERT and OPENROWSET(BULK...) statements are not supported import procedures due to their dependency on the ADMINISTER BULK OPERATIONS permission which is not granted for SQL Server DB instances. Please use the process outlined below to import data to a SQL Server DB instance.

The process that we recommend to import data into a SQL Server DB instance is as follows:

1. Create a DB Instance. (p. 207)
2. Before you load data into the destination DB Instance, you should do some preparation (p. 222), such as disabling foreign key constraints and database triggers. You should also disable automated backups.
3. Query the source SQL Server instance for any logins that you want to import (p. 224) to the destination DB Instance.
4. In your existing SQL Server deployment, generate scripts that obtain data from the source SQL Server instance, and then apply the scripts to the destination DB Instance (p. 225). If you have existing scripts, you can apply those scripts to the destination DB Instance. If you are importing a large dataset, your script can define only the database schema; otherwise, it can also include the data and all other database objects.
5. After your data is imported, reverse any preparations that you made earlier (p. 227), re-enable foreign key constraints and database triggers, switch the recovery model to its original state, and then re-enable automated backups.

Note
Amazon RDS for SQL Server does not currently support importing data into the msdb database. SQL Server features that use msdb, such as Database Mail and Replication, are not currently supported in Amazon RDS.

Preparing to Import Data into Your SQL Server DB Instance

Before you import data into your SQL Server DB Instance, we recommend the following best practices:

- Stop applications from accessing the destination DB Instance.
- Create a snapshot of the target database.
- Disable automated backups on the target database.
- Disable foreign key constraints, if applicable.
- Disable database triggers, if applicable.

Stop Applications from Accessing the Target DB Instance

If you prevent access to your DB Instance while you are importing data, data transfer will be 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 will lose will be 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 (p. 329).

Create a Database Snapshot

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. 296).

Note
When you take a database snapshot, I/O operations to the database are suspended for a few minutes while the backup is in progress.

Disable Automated Backups

Disabling automated backups on the target DB Instance will improve performance while you are importing your data. There are, however, 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 Automated Backups (p. 293).

Disable Foreign Key Constraints

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
```

Disable Database Triggers

If you need to disable database triggers, you can do so with the following script.

```
```
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.'+QUOTE
    NAME(@table)+' ';
    EXEC (@cmd);
    FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
END

CLOSE trigger_cursor;
DEALLOCATE trigger_cursor;
GO

---Disable triggers on all tables

Import Logins to Your SQL Server DB Instance

SQL Server stores logins and passwords in the master database. Because Amazon RDS does not 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 DDL file that includes all logins and passwords that you want to add to the destination DB Instance, as well as role memberships and permissions that you want to transfer.

For information about querying the master database, go to How to Transfer the Logins and the Passwords Between Instances of SQL Server 2005 and SQL Server 2008 on the Microsoft Knowledge Base.

The output of the script is another script that you can run on the destination DB Instance. Amazon RDS currently supports only SQL Server Authentication. Attempts to log in by using Windows Authentication will fail. You can ignore these failures, or you can edit the Microsoft script to include only logins that use SQL Server Authentication. Where the script in the Knowledge Base article has the following:

```sql
SELECT p.type
```

Use the following instead:

```sql
p.type IN
```
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.

**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 feature

**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 GB 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 may not be the fastest way to remotely export data from your local deployment. For an even faster way, you may want to consider the SQL Server bulk copy feature.

For detailed information about the Import and Export Wizard, go to the Microsoft SQL Server documentation.

In the wizard, on the **Choose a Destination** page, do the following:

- In the **Server Name** box, enter the name of the endpoint for your DB Instance.
• For the server authentication mode, click **Use SQL Server Authentication**.

• Under **User name** and **Password**, enter 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, go to 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.

```
PROMPT> bcp dbname.schema_name.table_name out C:\table_name.txt -n -S localhost -U username -P password -b 10000
```

Where:

• `-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.

For a full description of the command line syntax for the **bcp** utility, go to 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`.

```
PROMPT> 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 will be those of the master user and password for your DB Instance. The syntax is as follows:

```
PROMPT> bcp store.dbo.customers in C:\customers.txt -n -S endpoint -U admin -P insecure -b 10000
```
PROMPT> bcp dbname.schema_name.table_name in C:\table_name.txt -n -S end point, 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 will use port 4080. The command would be as follows:

PROMPT> bcp store.dbo.customers in C:\customers.txt -n -S rds.ckz2kqd4qsn1.us-east-1.rds.amazon.com,4080 -U admin -P insecure -b 10000

Cleaning Up

If you followed the best practices outlined earlier in this topic for preparing to import data to your DB Instance, you will need to perform the following tasks now:

- Grant applications access to the target DB Instance.
- Enable automated backups on the target DB Instance.
- Enable foreign key constraints.
- Enable database triggers.

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 (p. 329).

Enable Automated Backups on the Target DB Instance

For information about automated backups, see Working With Automated Backups (p. 293).

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
```
Enable Database Triggers

If you disabled database triggers earlier, you can now enable them with the following script:

```sql
-- 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.'+QUOTE_NAME(@table)+' ';
  EXEC (@cmd);
  FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
END
CLOSE trigger_cursor;
DEALLOCATE trigger_cursor;
```
Appendix: 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.

For information on working with SQL Server log files on Amazon RDS, see Working with SQL Server Database Log Files (p. 403)

**Topics**
- Determining a Recovery Model (p. 229)
- Collations and Character Sets for SQL Server (p. 229)
- Transitioning a Database from OFFLINE to ONLINE (p. 230)
- Analyzing your Database Workload on a DB Instance Using SQL Server Tuning Advisor (p. 230)
- Using SQL Server Agent (p. 233)

### Determining a Recovery Model

In RDS, the recovery model, retention period, and database status are linked. Changes to one can impact the other settings. For example:

- Changing a database’s recovery model to “Simple” while backup retention is enabled will result in RDS setting the recovery model to “Full” about five minutes after the setting was changed.
- Setting the backup retention to “0” days results in 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 RDS setting the recovery model back to “Simple.”
- Setting a database to “offline” will cause the database to remain “offline.”

### Collations and Character Sets for SQL Server

Amazon RDS creates a default server collation for character sets when a SQL Server DB instance is created. This default server collation is currently English (United States), or more precisely, SQL_Latin1_General_CP1_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_CP1_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 newly created database to Japanese_CI_AS:

```sql
CREATE TABLE [dbo].[Account] 
```
The 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

### Transitioning a Database from OFFLINE to ONLINE

<table>
<thead>
<tr>
<th>SQL Server method</th>
<th>Amazon RDS method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER DATABASE <em>name</em> SET ONLINE;</td>
<td>EXEC rdsadmin.dbo.rds_set_database_online <em>name</em></td>
</tr>
</tbody>
</table>

### Analyzing your Database Workload on a DB Instance Using 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 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 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 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 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 RDS, a workload can be a file on a client computer or a database table on an 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 RDS DB Instance, storage space is consumed by the trace. It is important to track of how much space is consumed to prevent it from becoming a bottleneck.
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 an 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 you 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.

```sql
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
```
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.

```sql
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:

```sql
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`:

```sql
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 RDS instance. Microsoft includes documentation on using the Database Engine Tuning Advisor in MSDN. Using Tuning Advisor with 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 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 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 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**.

### Using SQL Server Agent

With Amazon RDS, you can use SQL Server Agent on a DB Instance running 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](https://docs.microsoft.com/en-us/sql-server/relational-databases/server-agents/sql-server-agent-overview) 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 cannot use SQL Server Agent to create backups.

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];

You cannot use the UI in SQL Server Management Console to delete a SQL Server Agent job. To delete a SQL Server Agent job, run the following T-SQL statement.

EXEC msdb..sp_delete_job @job_name = '<job-name>';

### Viewing the SQL Server Agent Log

To view the SQL Server Agent log, you can use the RDS console. For information on viewing log files, see [Viewing and Listing Database Log Files](p. 406). You can also use the stored procedure `rdsadmin.dbo.rds_read_error_log` to view the agent log as described below.

```sql
CREATE PROCEDURE [dbo].[rds_read_error_log] @index INT = 0, @type INT = 1,
@search_str1 VARCHAR(255) = NULL, @search_str2 VARCHAR(255) = NULL,
@start_time DATETIME = NULL, @end_time DATETIME = NULL,
@sort_order NVARCHAR(4) = N'asc'
```

Two parameters are important when you call the `rdsadmin.dbo.rds_read_error_log` stored procedure:

- The `@index` parameter indicates the log that Amazon RDS will read from. The default value of 0 indicates the current log is used. A value of 1 indicates that the previously rotated log is used.
- The `@type` parameter indicates which type of log is read. The default value of 1 indicates that the SQL Server Error Log is used. A value of 2 indicates that the SQL Server Agent Log is used. All the other parameters are related to searching and sorting results and they can be kept at their default values.

For example, to read the current SQL Server Agent Log, you execute the following statement, where 0 indicates the current log and 2 indicates the SQL Server Agent Log.
EXEC rdsadmin..rds_read_error_log 0, 2;

You can specify the dbo schema name and list the parameters, but it is not necessary. The following three statements are equivalent to the statement in the previous example.

EXEC rdsadmin.dbo.rds_read_error_log 0, 2;
EXEC rdsadmin..rds_read_error_log @index = 0, @type = 2;
EXEC rdsadmin.dbo.rds_read_error_log @index = 0, @type = 2;

To read the last rotated log relative to the current SQL Server Agent Log, you execute the following statement, where 1 indicates the previous log and 2 indicates the SQL Server Agent Log.

EXEC rdsadmin..rds_read_error_log 1, 2;

If a rotated log does not exist, for example if the SQL Server Agent Log has never been rotated, then the statement returns the following error message.

Msg 22004, Level 16, State 1, Line 0
xp_readerrorlog() returned error 2, 'The system cannot find the file specified.'

Note
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.
Appendix: Options for SQL Server DB Engine

This appendix describes options, or additional features, that are available for Amazon RDS instances running the Microsoft SQL Server DB engine. To enable these options, you can add them to an option group, and then associate the option group with your DB instance. For more information about working with options, see Things You Should Know About Option Groups (p. 306).

The following option is currently supported for SQL Server:

• Transparent Data Encryption (TDE) for SQL Server

SQL Server Transparent Data Encryption

Amazon RDS supports using transparent data encryption (TDE) to encrypt stored data for SQL Server 2008 R2 Enterprise Edition and SQL Server 2012 Enterprise Edition. TDE automatically encrypts data before it is written to storage and automatically decrypts data when the data is read from storage. 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 rds-describe-db-instance 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_SQLServer 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. 306). For information about adding an option to an option group, see Adding an Option to an Option Group (p. 309).

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. 220).

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 RDSTDECertificate to encrypt a database called customerDatabase.
--- 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 Things You Should Know About Option Groups (p. 306).

Because the TDE option is a persistent option, you can also inadvertently 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 a DB instance that no longer uses TDE from a point-in-time DB snapshot that was taken when the DB instance was using TDE. The option group for the DB instance that no longer uses TDE will conflict with the restored DB instance that uses TDE.

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 using the RDS Console to remove the
TDE option from an option group, the console will indicate it is processing and an event will be 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.

```
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 will be 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_SQLServer option or you can remove the TDE_SQLServer 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 will be 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 M1.Large DB instance class.
Amazon RDS supports DB instances running PostgreSQL versions 9.3.1 and 9.3.2. You can create DB instances and DB snapshots, point-in-time restores and backups. DB instances running PostgreSQL support Multi-AZ deployments, Provisioned IOPS, and can be created inside a VPC. You can also use SSL to connect to a DB instance running PostgreSQL.

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. In order to deliver a managed service experience, Amazon RDS does not 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 does not allow direct host access to a DB instance via Telnet or Secure Shell (SSH).

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 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 on your DB instance and 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
- Manage tablespaces, including creating and deleting them
- 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 replication attribute onto all roles that are not the `rds-superuser` role

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.

You can modify parameters for your PostgreSQL DB instance by editing values in the instance's associated DB parameter group. For more information about setting parameters and other DBA tasks for PostgreSQL, see Appendix: Common DBA Tasks for PostgreSQL (p. 258)

To import PostgreSQL data into a DB instance, follow the information in the Importing Data into PostgreSQL on Amazon RDS (p. 256) section.

## DB Engine Features

PostgreSQL uses extensions that allow related pieces of functionality, such as datatypes and functions, to be bundled together and installed in a database with a single command. Note that the XML data type is currently supported only in version 9.3.2.

The following list shows a subset of the key PostgreSQL extensions that are currently supported by PostgreSQL on Amazon RDS. For more information on PostgreSQL extensions, see Packaging Related Objects into an Extension.

- **Data Type Extensions:**
  - `hstore` - Provides a key/value pair store.
  - `citext` - Provides a case-insensitive character string type.
  - `ltree` - Provides a data type for representing labels of data stored in a hierarchical tree-like structure.
  - `isn` - Provides data types for international product numbering standards such as EAN13, UPC, ISSN, and ISBN.
  - `cube` - Provides a data type for representing multidimensional cubes.
- **Full Text Search Dictionaries:**
  - `dict_int` - An add-on dictionary template for full-text search often used to control the indexing of integers.
  - `unaccent` - A text search dictionary that removes accents (diacritic signs) from lexemes.
  - `PostGIS`, `postgis_tiger_geocoder`, and `postgis_topology` - Spatial and geographic objects for PostgreSQL.
  - `dblink` - Supports connections to other PostgreSQL databases from within a database session.
- **Misc Extensions**
  - `earthdistance` - Calculates great circle distances on the surface of the Earth.
  - `fuzzystrmatch` - Determines similarities and distance between strings.
  - `intarray` - Provides functions and operators for manipulating null-free arrays of integers.
  - `pgcrypto` - Provides cryptographic functions.
  - `pg_trgm` - Functions that determine the similarity of alphanumeric text based on trigram matching.
  - `tablefunc` - Provides various functions that return tables.
  - `uuid-ossp` - Generates UUID's (does requires the OSSP UUID library, which can be found at http://www.ossp.org/pkg/lib/uuid/ - MIT License).
  - `btree_gin` - Provides a sample GIN operator that uses B-tree-like behavior for certain data types.
chkpass - Provides a data type designed for storing encrypted passwords.

intagg - Provides an integer aggregator and enumerator. This module is now obsolete but still provides a compatible wrapper around the built-in functions that superseded it.

tsearch2 - Provides backwards-compatible text search functionality.

pgrowlocks - Provides row locking information for a specified table.

sslinfo - Provides information about the SSL certificate provided by the current client when it connected to PostgreSQL.

Index Types

btree_gist - Provides GiST index operator classes that implement B-tree.

Supported PL languages include:

- PL/pgSQL
- PL/Tcl
- PL/Perl

The current list of extensions supported by Amazon RDS can be found in the default DB parameter group for PostgreSQL, called "default.postgres9.3." You can see the current extensions list using psql by showing the rds.extensions parameter.

```
SHOW rds.extensions;
```

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 minimum storage size for a PostgreSQL DB instance is 5 GB.
- The maximum storage size for a PostgreSQL DB instance is 3072 GB for all instances.
- Amazon RDS reserves up to 3 connections for system maintenance. If you specify a value for the user connections parameter, you will need to add 3 to the number of connections that you expect to use.

Minor Version Upgrades

With Amazon RDS, you can control when to upgrade your PostgreSQL instance to new versions supported by Amazon RDS. You can maintain compatibility with specific PostgreSQL versions, test new versions with your application before deploying in production, and perform version upgrades on your own terms and timelines.

Unless you specify otherwise, your DB Instance will automatically be upgraded to new PostgreSQL minor versions as they are supported by Amazon RDS. This patching will occur during your scheduled maintenance window, and it will be announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades, set the `AutoMinorVersionUpgrade` parameter for your DB instance to `false`.

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 (p. 288).
Using SSL with a PostgreSQL DB Instance

Amazon RDS supports SSL encryption for PostgreSQL DB instances. Using SSL, you can encrypt a PostgreSQL connection between your applications and your PostgreSQL DB instances. SSL support is available in all AWS regions for PostgreSQL.

To use a PostgreSQL DB instance over SSL, follow these general steps:

2. Import the certificate into your operating system.
3. Connect to your PostgreSQL DB instance over SSL by appending "encrypt=true" to your connection string. If you are connecting to a PostgreSQL DB instance in a VPC, you must also append the connection string with "TrustServerCertificate=true".

The encrypted status of your connection is shown when you connect to the DB instance in the logon banner:

Password for user master:
psql (9.3.1)
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 true (t) if the connection is using SSL, otherwise it returns false (f).

postgres=> create extension sslinfo;
CREATE EXTENSION

postgres=> select ssl_is_used();
ssl_is_used
--------
t
(1 row)

If the SSL parameter is set to true (the default) in the associated parameter group, you can also show the parameter value using the following command:

postgres=> show ssl;
ssl
-----
on
(1 row)
Working with a PostgreSQL DB Instance

Topics

• Creating a DB Instance Running the PostgreSQL Database Engine (p. 243)
• Connecting to a DB Instance Running the PostgreSQL Database Engine (p. 250)
• Modifying a DB Instance Running the PostgreSQL Database Engine (p. 254)

Most tasks you need to perform on a DB instance are performed the same way for all DB engines. Creating a DB instance, connecting to that DB instance, and importing data into a DB instance are all tasks that are specific for each DB engine. In addition, the appendix in this section contains important information on working with PostgreSQL DB instances.

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 use to run your PostgreSQL databases. It is important that you understand what each value you select means so you create a DB instance that meets your needs; refer to the links for each parameter for more information.

AWS Management Console

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 region in which you want to create the DB instance.
3. In the navigation pane, click DB Instances.
4. Click Launch DB Instance to start the Launch DB Instance Wizard.

The wizard opens on the Engine Selection page.
5. On the **Engine Selection** page, click the **Select** button for the PostgreSQL DB engine.

6. Next, the **Production?** page asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option **Multi-AZ** and the ** Provisioned IOPS** storage option will be preselected in the following step. Click **Next Step** when you are finished.

7. On the **DB Instance Details** page, specify your DB instance information. Click **Next Step** when you are finished.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Model</td>
<td>PostgreSQL has only one license model. Select the default, <strong>Postgresql License</strong>, to use the general license agreement for PostgreSQL.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>Select the version of PostgreSQL that you want to work with.</td>
</tr>
<tr>
<td>For this parameter...</td>
<td>...Do this:</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>DB Instance Class</strong></td>
<td>Select a DB instance class that defines the processing and memory requirements for the DB instance. For more information about all the DB instance class options, see DB Instance Class (p. 47).</td>
</tr>
<tr>
<td><strong>Multi-AZ Deployment</strong></td>
<td>Determine if you want to create a standby replica of your DB instance in another Availability Zone for failover support. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 50).</td>
</tr>
<tr>
<td><strong>Auto Minor Version Upgrade</strong></td>
<td>Select Yes to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.</td>
</tr>
<tr>
<td><strong>Allocated Storage</strong></td>
<td>Type a value to allocate storage for your database (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. The minimum allocated storage for a PostgreSQL instance is 5 GB. For more information about storage allocation, see Amazon Relational Database Service Features.</td>
</tr>
<tr>
<td><strong>Use Provisioned IOPS</strong></td>
<td>This option turns on Provisioned IOPS (I/O operations per second), a high-performance storage option in Amazon RDS that is optimized for I/O-intensive, transactional (OLTP) database workloads. For more information about high performance storage, see Working with Provisioned IOPS Storage (p. 361).</td>
</tr>
<tr>
<td><strong>DB Instance Identifier</strong></td>
<td>Type a name for the DB instance that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB engine you selected, for example <code>postgresql-instance1</code>.</td>
</tr>
<tr>
<td><strong>Master Username</strong></td>
<td>Type a name using alphanumeric characters that you will 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 Things You Should Know About PostgreSQL on Amazon RDS (p. 239)</td>
</tr>
<tr>
<td><strong>Master Password</strong></td>
<td>Type a password that contains from 8 to 128 printable ASCII characters (excluding /,, and @) for your master user password.</td>
</tr>
</tbody>
</table>
8. On the **Additional Config** 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 click Next Step.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>...Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database Name</strong></td>
<td>If you want to specify a database name for the default database, type a name for your database of up to 63 alpha-numeric characters. If you do not provide a name, no default database on the DB instance is created.</td>
</tr>
<tr>
<td><strong>Database Port</strong></td>
<td>Specify a port you want to use to access the database. PostgreSQL installations default to port 5432.</td>
</tr>
<tr>
<td><strong>Choose a VPC</strong></td>
<td>Determine if you want the DB instance to be created in a Virtual Private Cloud (VPC). For more information about VPC, see [Amazon RDS and the Amazon Virtual Private Cloud Service](p. 54).</td>
</tr>
<tr>
<td><strong>Publicly Accessible</strong></td>
<td>(Only applies if you choose a VPC) Select <strong>Yes</strong> to give the DB instance a public IP address; otherwise, select <strong>No</strong>. For more information about hiding DB instances from public access, see [Hiding a DB instance in a VPC from the Internet](p. 54).</td>
</tr>
<tr>
<td><strong>Availability Zone</strong></td>
<td>Determine if you want to specify a particular Availability Zone. If you selected <strong>Yes</strong> for the Multi-AZ Deployment parameter on the previous page, you will not have any options here. For more information about Availability Zones, see [Regions and Availability Zones](p. 50).</td>
</tr>
<tr>
<td><strong>Option Group</strong></td>
<td>Option groups are currently not used with PostgreSQL DB instances. For more information about option groups, see [Working with Option Groups](p. 306).</td>
</tr>
</tbody>
</table>
For this parameter... | ...Do this:
--- | ---
Parameter Group | Select a parameter group. Each PostgreSQL version has a default parameter group you can use, or you can create your own parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 317).

Security Groups | Select the security group you want to use with this DB instance. The security group you select must specify the IP addresses or EC2 instances that can access the DB instance. For more information about security groups, see Working with DB Security Groups (p. 329).

9. On the Management Options page, you can specify backup and maintenance options for your DB instance. Click Next Step when the settings are as you want them. For more information about the maintenance window, see Adjusting the Preferred Maintenance Window (p. 373). For more information on backups and the backup retention period, see DB Instance Backups (p. 55).
10. On the **Review** page, review the options for your DB instance. If you need to make any changes, click **Previous** to return to the appropriate page, and then make the necessary corrections. When all the settings are as you want them, click **Launch DB Instance**.

11. On the final page of the wizard, click **Close**.

12. 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.

---

**CLI**

To create a PostgreSQL DB instance

- Use the command `rds-create-db-instance` to create a DB instance.

```bash
PROMPT>rds-create-db-instance pgdbinstance -s 20 -c db.m1.small -e postgresql - u <masterawsuser> -p <masteruserpassword>
```

This command should produce output similar to the following:

```
DBINSTANCE  pgdbinstance  db.m1.small  postgresql  20  sa  creating  3  ****  n  9.3
SECGROUP  default  active
PARAMGRP  default.PostgreSQL9.3  in-sync
```

**API**

To create a PostgreSQL DB instance

- Call the `CreateDBInstance` action. For example, you could use the following parameters:

```bash
DBEngine = postgresql
DBInstanceIdentifier = pgdbinstance
DBInstanceClass = db.m1.small
AllocatedStorage = 20
BackupRetentionPeriod = 3
```

---

Amazon Relational Database Service User Guide
Creating a DB Instance Running the PostgreSQL
Database Engine

---

API Version 2013-09-09
248
• MasterUsername = <masterawsuser>
• MasterUserPassword = <masteruserpassword>

Example

https://rds.amazonaws.com/
?Action=CreateDBInstance
&AllocatedStorage=20
&BackupRetentionPeriod=3
&DBInstanceClass=db.m1.small
&DBInstanceIdentifier=pgdbinstance
&DBName=mydatabase
&DBSecurityGroups.member.1=mysecuritygroup
&DBSubnetGroup=mydbsubnetgroup
&Engine=postgresql
&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

Related Topics

• DB Instance (p. 43)
• DB Instance Class (p. 47)
• Deleting a DB Instance (p. 273)
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. 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. Select 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 text 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 text box. Enter the user name and user password you entered when you created the DB instance into the Username and Password text boxes, respectively.
4. Click OK.

5. In the **Object browser**, expand the **Server Groups**. Select the Server (the DB instance) you created, and then select the database name.
6. Click the plugin icon and click **PSQL Console**. The PSQL command window opens for the default database you created.

![PSQL Console](image.png)

7. Use the command window to enter SQL or PSQL commands. Type `\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 will allow you to access the instance. For more information about Amazon RDS security groups, see [Amazon RDS Security Groups](#) (p. 74)
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 DB security group assigned to the DB instance has the necessary rules to allow access through any firewall your connection may be going through.

**Related Topics**

- DB Instance (p. 43)
- Creating a DB Instance Running the PostgreSQL Database Engine (p. 243)
- Amazon RDS Security Groups (p. 74)
- Deleting a DB Instance (p. 273)
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 Tasks Common to All Amazon RDS DB Engines (p. 270). 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 a DB Instance and Using the Apply Immediately Parameter (p. 45).

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, click Instances.
3. Select the check box for the DB instance that you want to change, and then click Modify.
4. In the Modify DB Instance dialog box, change any of the following settings that you want:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Instance Identifier</td>
<td>You can rename the DB instance by typing a new name.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>In the list provided, click the version of the PostgreSQL database engine that you want to use.</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>In the list provided, click the DB instance class that you want to use. For information about instance classes, see the section called “DB Instance Class” (p. 47).</td>
</tr>
<tr>
<td>Multi-AZ Deployment</td>
<td>If you want to create a standby replica or your DB instance in another Availability Zones, click Yes; otherwise, click No. For more information on Multi-AZ deployments, see High Availability (Multi-AZ) (p. 52).</td>
</tr>
<tr>
<td>Auto Minor Version Upgrade</td>
<td>If you want your DB instance to receive minor engine version upgrades automatically when they become available, click Yes. Upgrades are installed only during your scheduled maintenance window.</td>
</tr>
<tr>
<td>Allocated Storage</td>
<td>Specify how much storage, in gigabytes, to allocate for your DB instance. The minimum allowable value is 5 GB; the maximum is 3072 GB. Note that you can only increase the amount of storage when modifying a DB instance, you cannot reduce the amount of storage allocated. For more information on allocated storage, see Amazon RDS Storage (p. 48)</td>
</tr>
</tbody>
</table>
### Setting | Description
--- | ---
Provisioned IOPS | Select the Use Provisioned IOPS check box and enter a value in the Provisioned IOPS text box if you want your DB instance to use Provisioned IOPS. For more information on Provisioned IOPS, see Working with Provisioned IOPS Storage (p. 361).
Parameter Group | Select the parameter group you want associated with the DB instance. For more information about parameter groups, see Working with DB Parameter Groups (p. 317).
Security Groups | Select the security group you want associated with the DB instance. For more information about security groups, see Working with DB Security Groups (p. 329).
Option Group | No options are available for PostgreSQL DB instances. For more information about option groups, see Working with Option Groups (p. 306).
Master User Password | Type a password for your master user. The password must contain from 8 to 128 alphanumeric characters.
Backup Retention Period | Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0.
Backup Window | Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC) and a duration in hours.
Maintenance Window | Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.

5. To apply the changes immediately, select the **Apply Immediately** check box. Selecting this option can cause an outage in some cases; for more information on the impact of the Apply Immediately option, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 45).

6. When all the changes are as you want them, click **Continue**. If you want to cancel any changes, click the **X** in the upper right corner of the page.

7. Confirm that the changes you want are listed in the summary screen, and then click **Modify DB Instance**.

### CLI

**To modify a PostgreSQL DB instance**

- Use the command `rds-modify-db-instance`.

### API

**To modify a PostgreSQL DB instance**

- Use the `ModifyDBInstance` action.
Importing Data into PostgreSQL on Amazon RDS

If 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 are transferring. For example, a database that contains data sets on the order of gigabytes, along with stored procedures and triggers, is going to be more complicated than a simple database with only a few megabytes of test data and no triggers or stored procedures.

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 be loaded

pg_dump 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.

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, CLI, or API. Create the instance with the backup retention setting set to 0 and disable Multi-AZ. This will allow faster data import. You must create a database on the instance that has the same name as the database that contained the dumped data; the pg_dump command requires that the database exist on the target PostgreSQL DB instance.

Step 3: Use psql to create the database on the DB instance and load the 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`:

```sql
psql -f mydb2dump.sql --host=mypginstance.c6c8mntzhgv0.us-west-2.rds.amazonaws.com
--port=8199 --username=myawsuser --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 and can also be used to seed new database instances. For information about creating a DB snapshot, see Creating a DB Snapshot (p. 296).

### 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.

**Note**
The `\copy` command does not 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:

```bash
$psql target-db -U <admin user> -p <port> -h <DB instance name> -c "\copy source-table from 'source-table.csv' with DELIMITER ','"
```
Appendix: Common DBA Tasks for PostgreSQL

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the PostgreSQL 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 PostgreSQL log files on Amazon RDS, see Working with PostgreSQL Database Log Files (p. 404)

Topics
• Creating Roles (p. 258)
• Managing PostgreSQL Database Access (p. 258)
• Working with PostgreSQL Parameters (p. 259)
• Setting up PostGIS (p. 267)
• Using pgBadger for Log Analysis with Amazon RDS PostgreSQL (p. 269)

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 pre-defined 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 on your DB instance and you should not assign this role to users unless they need the most access to the DB instance.

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.

```
postgres=> create role testuser with password 'testuser' login;
CREATE ROLE
postgres=> grant rds_superuser to testuser;
GRANT ROLE
postgres=>
```

Managing PostgreSQL Database Access

By default, when PostgreSQL database objects are created, they receive "public" access privileges. You can revoke all privileges to a database and then explicitly add privileges back as you need them.

As the master user, you can remove all privileges from a database using the following command format:

```
postgres=> revoke all on database <database name> from public;
REVOKE
```

You can then add privileges back to a user. For example, the following command grants connect access to a user named mytestuser to a database named test.

```
test=> grant connect on database test to mytestuser;
GRANT
```
Note that on a local instance, you could specify database privileges in the pg_hba.conf file, but when using PostgreSQL with Amazon RDS it is better to restrict privileges at the Postgres level. Changes to the pg_hba.conf file require a server restart so you cannot edit the pg_hba.conf in Amazon RDS, but privilege changes at the Postgres level occur immediately.

# Working with PostgreSQL Parameters

PostgreSQL parameters that you would 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.postgres9.3`.

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 the SET commands. Note that you cannot use the command line `postgres` command nor the `env PGOPTIONS` command because you will 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. Note that 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, fixed and dynamic. Fixed parameters require that the DB instance be rebooted before they are applied. Dynamic parameters can be applied immediately. The following table shows parameters you can modify for a PostgreSQL DB instance and the parameter's type:
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Apply_Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>application_name</td>
<td>Dynamic</td>
<td>Sets the application name to be reported in statistics and logs.</td>
</tr>
<tr>
<td>array_nulls</td>
<td>Dynamic</td>
<td>Enables input of NULL elements in arrays.</td>
</tr>
<tr>
<td>authentication_timeout</td>
<td>Dynamic</td>
<td>Sets the maximum allowed time to complete client authentication.</td>
</tr>
<tr>
<td>autovacuum</td>
<td>Dynamic</td>
<td>Starts the autovacuum subprocess.</td>
</tr>
<tr>
<td>autovacuum_analyze_scale_factor</td>
<td>Dynamic</td>
<td>Number of tuple inserts, updates, or deletes prior to analyze as a fraction of reltuples.</td>
</tr>
<tr>
<td>autovacuum_analyze_threshold</td>
<td>Dynamic</td>
<td>Minimum number of tuple inserts, updates, or deletes prior to analyze.</td>
</tr>
<tr>
<td>autovacuum_naptime</td>
<td>Dynamic</td>
<td>Time to sleep between autovacuum runs.</td>
</tr>
<tr>
<td>autovacuum_vacuum_cost_delay</td>
<td>Dynamic</td>
<td>Vacuum cost delay, in milliseconds, for autovacuum.</td>
</tr>
<tr>
<td>autovacuum_vacuum_cost_limit</td>
<td>Dynamic</td>
<td>Vacuum cost amount available before napping, for autovacuum.</td>
</tr>
<tr>
<td>autovacuum_vacuum_scale_factor</td>
<td>Dynamic</td>
<td>Number of tuple updates or deletes prior to vacuum as a fraction of reltuples.</td>
</tr>
<tr>
<td>autovacuum_vacuum_threshold</td>
<td>Dynamic</td>
<td>Minimum number of tuple updates or deletes prior to vacuum.</td>
</tr>
<tr>
<td>backslash_quote</td>
<td>Dynamic</td>
<td>Sets whether a backslash () is allowed in string literals.</td>
</tr>
<tr>
<td>bgwriter_delay</td>
<td>Dynamic</td>
<td>Background writer sleep time between rounds.</td>
</tr>
<tr>
<td>bgwriter_lru_maxpages</td>
<td>Dynamic</td>
<td>Background writer maximum number of LRU pages to flush per round.</td>
</tr>
<tr>
<td>bgwriter_lru_multiplier</td>
<td>Dynamic</td>
<td>Multiple of the average buffer usage to free per round.</td>
</tr>
<tr>
<td>bytea_output</td>
<td>Dynamic</td>
<td>Sets the output format for bytea.</td>
</tr>
<tr>
<td>check_function_bodies</td>
<td>Dynamic</td>
<td>Checks function bodies during CREATE FUNCTION.</td>
</tr>
<tr>
<td>checkpoint_completion_target</td>
<td>Dynamic</td>
<td>Time spent flushing dirty buffers during checkpoint, as fraction of checkpoint interval.</td>
</tr>
<tr>
<td>checkpoint_segments</td>
<td>Dynamic</td>
<td>Sets the maximum distance in log segments between automatic WAL checkpoints.</td>
</tr>
<tr>
<td>checkpoint_timeout</td>
<td>Dynamic</td>
<td>Sets the maximum time between automatic WAL checkpoints.</td>
</tr>
<tr>
<td>checkpoint_warning</td>
<td>Dynamic</td>
<td>Enables warnings if checkpoint segments are filled more frequently than this.</td>
</tr>
<tr>
<td>client_encoding</td>
<td>Dynamic</td>
<td>Sets the client's character set encoding.</td>
</tr>
<tr>
<td>client_min_messages</td>
<td>Dynamic</td>
<td>Sets the message levels that are sent to the client.</td>
</tr>
<tr>
<td>commit_delay</td>
<td>Dynamic</td>
<td>Sets the delay in microseconds between transaction commit and flushing WAL to disk.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Apply_Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>commit_siblings</td>
<td>Dynamic</td>
<td>Sets the minimum concurrent open transactions before performing commit_delay.</td>
</tr>
<tr>
<td>constraint_exclusion</td>
<td>Dynamic</td>
<td>Enables the planner to use constraints to optimize queries.</td>
</tr>
<tr>
<td>cpu_index_tuple_cost</td>
<td>Dynamic</td>
<td>Sets the planner's estimate of the cost of processing each index entry during an index scan.</td>
</tr>
<tr>
<td>cpu_operator_cost</td>
<td>Dynamic</td>
<td>Sets the planner's estimate of the cost of processing each operator or function call.</td>
</tr>
<tr>
<td>cpu_tuple_cost</td>
<td>Dynamic</td>
<td>Sets the planner's estimate of the cost of processing each tuple (row).</td>
</tr>
<tr>
<td>cursor_tuple_fraction</td>
<td>Dynamic</td>
<td>Sets the planner's estimate of the fraction of a cursor's rows that will be retrieved.</td>
</tr>
<tr>
<td>datestyle</td>
<td>Dynamic</td>
<td>Sets the display format for date and time values.</td>
</tr>
<tr>
<td>deadlock_timeout</td>
<td>Dynamic</td>
<td>Sets the time to wait on a lock before checking for deadlock.</td>
</tr>
<tr>
<td>debug_pretty_print</td>
<td>Dynamic</td>
<td>Indents parse and plan tree displays.</td>
</tr>
<tr>
<td>debug_print_parse</td>
<td>Dynamic</td>
<td>Logs each query's parse tree.</td>
</tr>
<tr>
<td>debug_print_plan</td>
<td>Dynamic</td>
<td>Logs each query's execution plan.</td>
</tr>
<tr>
<td>debug_print_rewritten</td>
<td>Dynamic</td>
<td>Logs each query's rewritten parse tree.</td>
</tr>
<tr>
<td>default_statistics_target</td>
<td>Dynamic</td>
<td>Sets the default statistics target.</td>
</tr>
<tr>
<td>default_tablespace</td>
<td>Dynamic</td>
<td>Sets the default tablespace to create tables and indexes in.</td>
</tr>
<tr>
<td>default_transaction_deferrable</td>
<td>Dynamic</td>
<td>Sets the default deferrable status of new transactions.</td>
</tr>
<tr>
<td>default_transaction_isolation</td>
<td>Dynamic</td>
<td>Sets the transaction isolation level of each new transaction.</td>
</tr>
<tr>
<td>default_transaction_read_only</td>
<td>Dynamic</td>
<td>Sets the default read-only status of new transactions.</td>
</tr>
<tr>
<td>default_with_oids</td>
<td>Dynamic</td>
<td>Creates new tables with OIDs by default.</td>
</tr>
<tr>
<td>effective_cache_size</td>
<td>Dynamic</td>
<td>Sets the planner's assumption about the size of the disk cache.</td>
</tr>
<tr>
<td>effective_io_concurrency</td>
<td>Dynamic</td>
<td>Number of simultaneous requests that can be handled efficiently by the disk subsystem.</td>
</tr>
<tr>
<td>enable_bitmapscan</td>
<td>Dynamic</td>
<td>Enables the planner's use of bitmap-scan plans.</td>
</tr>
<tr>
<td>enable_hashagg</td>
<td>Dynamic</td>
<td>Enables the planner's use of hashed aggregation plans.</td>
</tr>
<tr>
<td>enable_hashjoin</td>
<td>Dynamic</td>
<td>Enables the planner's use of hash join plans.</td>
</tr>
<tr>
<td>enable_indexscan</td>
<td>Dynamic</td>
<td>Enables the planner's use of index-scan plans.</td>
</tr>
<tr>
<td>enable_material</td>
<td>Dynamic</td>
<td>Enables the planner's use of materialization.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Apply_Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>enable_mergejoin</td>
<td>Dynamic</td>
<td>Enables the planner's use of merge join plans.</td>
</tr>
<tr>
<td>enable_nestloop</td>
<td>Dynamic</td>
<td>Enables the planner's use of nested-loop join plans.</td>
</tr>
<tr>
<td>enable_seqscan</td>
<td>Dynamic</td>
<td>Enables the planner's use of sequential-scan plans.</td>
</tr>
<tr>
<td>enable_sort</td>
<td>Dynamic</td>
<td>Enables the planner's use of explicit sort steps.</td>
</tr>
<tr>
<td>enable_tidscan</td>
<td>Dynamic</td>
<td>Enables the planner's use of TID scan plans.</td>
</tr>
<tr>
<td>escape_string_warning</td>
<td>Dynamic</td>
<td>Warns about backslash () escapes in ordinary string literals.</td>
</tr>
<tr>
<td>extra_float_digits</td>
<td>Dynamic</td>
<td>Sets the number of digits displayed for floating-point values.</td>
</tr>
<tr>
<td>fromCollapseLimit</td>
<td>Dynamic</td>
<td>Sets the FROM-list size beyond which subqueries are not collapsed.</td>
</tr>
<tr>
<td>fsync</td>
<td>Dynamic</td>
<td>Forces synchronization of updates to disk.</td>
</tr>
<tr>
<td>fullPageWrites</td>
<td>Dynamic</td>
<td>Writes full pages to WAL when first modified after a checkpoint.</td>
</tr>
<tr>
<td>geqo</td>
<td>Dynamic</td>
<td>Enables genetic query optimization.</td>
</tr>
<tr>
<td>geqo_effort</td>
<td>Dynamic</td>
<td>GEQO: effort is used to set the default for other GEQO parameters.</td>
</tr>
<tr>
<td>geqo_generations</td>
<td>Dynamic</td>
<td>GEQO: number of iterations of the algorithm.</td>
</tr>
<tr>
<td>geqo_pool_size</td>
<td>Dynamic</td>
<td>GEQO: number of individuals in the population.</td>
</tr>
<tr>
<td>geqo_seed</td>
<td>Dynamic</td>
<td>GEQO: seed for random path selection.</td>
</tr>
<tr>
<td>geqo_selection_bias</td>
<td>Dynamic</td>
<td>GEQO: selective pressure within the population.</td>
</tr>
<tr>
<td>geqo_threshold</td>
<td>Dynamic</td>
<td>Sets the threshold of FROM items beyond which GEQO is used.</td>
</tr>
<tr>
<td>gin_fuzzy_search_limit</td>
<td>Dynamic</td>
<td>Sets the maximum allowed result for exact search by GIN.</td>
</tr>
<tr>
<td>intervalstyle</td>
<td>Dynamic</td>
<td>Sets the display format for interval values.</td>
</tr>
<tr>
<td>joinCollapseLimit</td>
<td>Dynamic</td>
<td>Sets the FROM-list size beyond which JOIN constructs are not flattened.</td>
</tr>
<tr>
<td>lc_messages</td>
<td>Dynamic</td>
<td>Sets the language in which messages are displayed.</td>
</tr>
<tr>
<td>lc_monetary</td>
<td>Dynamic</td>
<td>Sets the locale for formatting monetary amounts.</td>
</tr>
<tr>
<td>lc_numeric</td>
<td>Dynamic</td>
<td>Sets the locale for formatting numbers.</td>
</tr>
<tr>
<td>lc_time</td>
<td>Dynamic</td>
<td>Sets the locale for formatting date and time values.</td>
</tr>
<tr>
<td>log_autovacuum_min_duration</td>
<td>Dynamic</td>
<td>Sets the minimum execution time above which autovacuum actions will be logged.</td>
</tr>
<tr>
<td>log_checkpoints</td>
<td>Dynamic</td>
<td>Logs each checkpoint.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Apply_Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>log_connections</td>
<td>Dynamic</td>
<td>Logs each successful connection.</td>
</tr>
<tr>
<td>log_disconnections</td>
<td>Dynamic</td>
<td>Logs end of a session, including duration.</td>
</tr>
<tr>
<td>log_duration</td>
<td>Dynamic</td>
<td>Logs the duration of each completed SQL statement.</td>
</tr>
<tr>
<td>log_error_verbosity</td>
<td>Dynamic</td>
<td>Sets the verbosity of logged messages.</td>
</tr>
<tr>
<td>log_executor_stats</td>
<td>Dynamic</td>
<td>Writes executor performance statistics to the server log.</td>
</tr>
<tr>
<td>log_filename</td>
<td>Dynamic</td>
<td>Sets the file name pattern for log files.</td>
</tr>
<tr>
<td>log_hostname</td>
<td>Dynamic</td>
<td>Logs the host name in the connection logs.</td>
</tr>
<tr>
<td>log_lock_waits</td>
<td>Dynamic</td>
<td>Logs long lock waits.</td>
</tr>
<tr>
<td>log_min_duration_statement</td>
<td>Dynamic</td>
<td>Sets the minimum execution time above which statements will be logged.</td>
</tr>
<tr>
<td>log_min_error_statement</td>
<td>Dynamic</td>
<td>Causes all statements generating an error at or above this level to be logged.</td>
</tr>
<tr>
<td>log_min_messages</td>
<td>Dynamic</td>
<td>Sets the message levels that are logged.</td>
</tr>
<tr>
<td>log_parser_stats</td>
<td>Dynamic</td>
<td>Writes parser performance statistics to the server log.</td>
</tr>
<tr>
<td>log_planner_stats</td>
<td>Dynamic</td>
<td>Writes planner performance statistics to the server log.</td>
</tr>
<tr>
<td>log_rotation_age</td>
<td>Dynamic</td>
<td>Automatic log file rotation will occur after N minutes.</td>
</tr>
<tr>
<td>log_rotation_size</td>
<td>Dynamic</td>
<td>Automatic log file rotation will occur after N kilobytes.</td>
</tr>
<tr>
<td>log_statement</td>
<td>Dynamic</td>
<td>Sets the type of statements logged.</td>
</tr>
<tr>
<td>log_statement_stats</td>
<td>Dynamic</td>
<td>Writes cumulative performance statistics to the server log.</td>
</tr>
<tr>
<td>log_temp_files</td>
<td>Dynamic</td>
<td>Logs the use of temporary files larger than this number of kilobytes.</td>
</tr>
<tr>
<td>maintenance_work_mem</td>
<td>Dynamic</td>
<td>Sets the maximum memory to be used for maintenance operations.</td>
</tr>
<tr>
<td>max_stack_depth</td>
<td>Dynamic</td>
<td>Sets the maximum stack depth, in kilobytes.</td>
</tr>
<tr>
<td>max_standby_archive_delay</td>
<td>Dynamic</td>
<td>Sets the maximum delay before canceling queries when a hot standby server is processing archived WAL data.</td>
</tr>
<tr>
<td>max_standby_streaming_delay</td>
<td>Dynamic</td>
<td>Sets the maximum delay before canceling queries when a hot standby server is processing streamed WAL data.</td>
</tr>
<tr>
<td>quote_all_identifiers</td>
<td>Dynamic</td>
<td>Adds quotes (&quot;) to all identifiers when generating SQL fragments.</td>
</tr>
<tr>
<td>random_page_cost</td>
<td>Dynamic</td>
<td>Sets the planner's estimate of the cost of a non-sequentially fetched disk page.</td>
</tr>
<tr>
<td>rds.log_retention_period</td>
<td>Dynamic</td>
<td>Amazon RDS will delete PostgreSQL logs that are older than N minutes.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Apply_Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>search_path</td>
<td>Dynamic</td>
<td>Sets the schema search order for names that are not schema-qualified.</td>
</tr>
<tr>
<td>seq_page_cost</td>
<td>Dynamic</td>
<td>Sets the planner’s estimate of the cost of a sequentially fetched disk page.</td>
</tr>
<tr>
<td>session_replication_role</td>
<td>Dynamic</td>
<td>Sets the sessions behavior for triggers and rewrite rules.</td>
</tr>
<tr>
<td>sql_inheritance</td>
<td>Dynamic</td>
<td>Causes subtables to be included by default in various commands.</td>
</tr>
<tr>
<td>ssl_renegotiation_limit</td>
<td>Dynamic</td>
<td>Sets the amount of traffic to send and receive before renegotiating the encryption keys.</td>
</tr>
<tr>
<td>standard_conforming_strings</td>
<td>Dynamic</td>
<td>Causes ... strings to treat backslashes literally.</td>
</tr>
<tr>
<td>statement_timeout</td>
<td>Dynamic</td>
<td>Sets the maximum allowed duration of any statement.</td>
</tr>
<tr>
<td>synchronize_seqscans</td>
<td>Dynamic</td>
<td>Enables synchronized sequential scans.</td>
</tr>
<tr>
<td>synchronous_commit</td>
<td>Dynamic</td>
<td>Sets the current transactions synchronization level.</td>
</tr>
<tr>
<td>tcp_keepalives_count</td>
<td>Dynamic</td>
<td>Maximum number of TCP keepalive retransmits.</td>
</tr>
<tr>
<td>tcp_keepalives_idle</td>
<td>Dynamic</td>
<td>Time between issuing TCP keepalives.</td>
</tr>
<tr>
<td>tcp_keepalives_interval</td>
<td>Dynamic</td>
<td>Time between TCP keepalive retransmits.</td>
</tr>
<tr>
<td>temp_buffers</td>
<td>Dynamic</td>
<td>Sets the maximum number of temporary buffers used by each session.</td>
</tr>
<tr>
<td>temp_tablespaces</td>
<td>Dynamic</td>
<td>Sets the tablespaces to use for temporary tables and sort files.</td>
</tr>
<tr>
<td>timezone</td>
<td>Dynamic</td>
<td>Sets the time zone for displaying and interpreting time stamps.</td>
</tr>
<tr>
<td>track_activities</td>
<td>Dynamic</td>
<td>Collects information about executing commands.</td>
</tr>
<tr>
<td>track_counts</td>
<td>Dynamic</td>
<td>Collects statistics on database activity.</td>
</tr>
<tr>
<td>track_functions</td>
<td>Dynamic</td>
<td>Collects function-level statistics on database activity.</td>
</tr>
<tr>
<td>track_io_timing</td>
<td>Dynamic</td>
<td>Collects timing statistics on database I/O activity.</td>
</tr>
<tr>
<td>transaction_deferrable</td>
<td>Dynamic</td>
<td>Indicates whether to defer a read-only serializable transaction until it can be executed with no possible serialization failures.</td>
</tr>
<tr>
<td>transaction_isolation</td>
<td>Dynamic</td>
<td>Sets the current transactions isolation level.</td>
</tr>
<tr>
<td>transaction_read_only</td>
<td>Dynamic</td>
<td>Sets the current transactions read-only status.</td>
</tr>
<tr>
<td>transform_null_equals</td>
<td>Dynamic</td>
<td>Treats expr=NULL as expr IS NULL.</td>
</tr>
<tr>
<td>update_process_title</td>
<td>Dynamic</td>
<td>Updates the process title to show the active SQL command.</td>
</tr>
<tr>
<td>vacuum_cost_delay</td>
<td>Dynamic</td>
<td>Vacuum cost delay in milliseconds.</td>
</tr>
<tr>
<td>vacuum_cost_limit</td>
<td>Dynamic</td>
<td>Vacuum cost amount available before napping.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Apply_Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>vacuum_cost_page_dirty</td>
<td>Dynamic</td>
<td>Vacuum cost for a page dirtied by vacuum.</td>
</tr>
<tr>
<td>vacuum_cost_page_hit</td>
<td>Dynamic</td>
<td>Vacuum cost for a page found in the buffer cache.</td>
</tr>
<tr>
<td>vacuum_cost_page_miss</td>
<td>Dynamic</td>
<td>Vacuum cost for a page not found in the buffer cache.</td>
</tr>
<tr>
<td>vacuum_defer_cleanup_age</td>
<td>Dynamic</td>
<td>Number of transactions by which vacuum and hot cleanup should be deferred, if any.</td>
</tr>
<tr>
<td>vacuum_freeze_min_age</td>
<td>Dynamic</td>
<td>Minimum age at which vacuum should freeze a table row.</td>
</tr>
<tr>
<td>vacuum_freeze_table_age</td>
<td>Dynamic</td>
<td>Age at which vacuum should scan a whole table to freeze tuples.</td>
</tr>
<tr>
<td>wal_writer_delay</td>
<td>Dynamic</td>
<td>WAL writer sleep time between WAL flushes.</td>
</tr>
<tr>
<td>work_mem</td>
<td>Dynamic</td>
<td>Sets the maximum memory to be used for query workspaces.</td>
</tr>
<tr>
<td>xmlbinary</td>
<td>Dynamic</td>
<td>Sets how binary values are to be encoded in XML.</td>
</tr>
<tr>
<td>xmloption</td>
<td>Dynamic</td>
<td>Sets whether XML data in implicit parsing and serialization operations is to be considered as documents or content fragments.</td>
</tr>
<tr>
<td>autovacuum_freeze_max_age</td>
<td>Static</td>
<td>Age at which to autovacuum a table to prevent transaction ID wraparound.</td>
</tr>
<tr>
<td>autovacuum_max_workers</td>
<td>Static</td>
<td>Sets the maximum number of simultaneously running autovacuum worker processes.</td>
</tr>
<tr>
<td>max_connections</td>
<td>Static</td>
<td>Sets the maximum number of concurrent connections.</td>
</tr>
<tr>
<td>max_files_per_process</td>
<td>Static</td>
<td>Sets the maximum number of simultaneously open files for each server process.</td>
</tr>
<tr>
<td>max_locks_per_transaction</td>
<td>Static</td>
<td>Sets the maximum number of locks per transaction.</td>
</tr>
<tr>
<td>max_pred_locks_per_transaction</td>
<td>Static</td>
<td>Sets the maximum number of predicate locks per transaction.</td>
</tr>
<tr>
<td>max_prepared_transactions</td>
<td>Static</td>
<td>Sets the maximum number of simultaneously prepared transactions.</td>
</tr>
<tr>
<td>shared_buffers</td>
<td>Static</td>
<td>Sets the number of shared memory buffers used by the server.</td>
</tr>
<tr>
<td>ssl</td>
<td>Static</td>
<td>Enables SSL connections.</td>
</tr>
<tr>
<td>track_activity_query_size</td>
<td>Static</td>
<td>Sets the size reserved for pg_stat_activity.current_query, in bytes.</td>
</tr>
<tr>
<td>wal_buffers</td>
<td>Static</td>
<td>Sets the number of disk-page buffers in shared memory for WAL.</td>
</tr>
</tbody>
</table>

Amazon RDS uses the default PostgreSQL units for all parameters. The following table shows the PostgreSQL unit value for each parameter.
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>effective_cache_size</td>
<td>8 kB</td>
</tr>
<tr>
<td>segment_size</td>
<td>8 kB</td>
</tr>
<tr>
<td>shared_buffers</td>
<td>8 kB</td>
</tr>
<tr>
<td>temp_buffers</td>
<td>8 kB</td>
</tr>
<tr>
<td>wal_buffers</td>
<td>8 kB</td>
</tr>
<tr>
<td>wal_segment_size</td>
<td>8 kB</td>
</tr>
<tr>
<td>log_rotation_size</td>
<td>kB</td>
</tr>
<tr>
<td>log_temp_files</td>
<td>kB</td>
</tr>
<tr>
<td>maintenance_work_mem</td>
<td>kB</td>
</tr>
<tr>
<td>max_stack_depth</td>
<td>kB</td>
</tr>
<tr>
<td>ssl_renegotiation_limit</td>
<td>kB</td>
</tr>
<tr>
<td>temp_file_limit</td>
<td>kB</td>
</tr>
<tr>
<td>work_mem</td>
<td>kB</td>
</tr>
<tr>
<td>log_rotation_age</td>
<td>min</td>
</tr>
<tr>
<td>autovacuum_vacuum_cost_delay</td>
<td>ms</td>
</tr>
<tr>
<td>bgwriter_delay</td>
<td>ms</td>
</tr>
<tr>
<td>deadlock_timeout</td>
<td>ms</td>
</tr>
<tr>
<td>lock_timeout</td>
<td>ms</td>
</tr>
<tr>
<td>log_autovacuum_min_duration</td>
<td>ms</td>
</tr>
<tr>
<td>log_min_duration_statement</td>
<td>ms</td>
</tr>
<tr>
<td>max_standby_archive_delay</td>
<td>ms</td>
</tr>
<tr>
<td>max_standby_streaming_delay</td>
<td>ms</td>
</tr>
<tr>
<td>statement_timeout</td>
<td>ms</td>
</tr>
<tr>
<td>vacuum_cost_delay</td>
<td>ms</td>
</tr>
<tr>
<td>wal_receiver_timeout</td>
<td>ms</td>
</tr>
<tr>
<td>wal_sender_timeout</td>
<td>ms</td>
</tr>
<tr>
<td>wal_writer_delay</td>
<td>ms</td>
</tr>
<tr>
<td>archive_timeout</td>
<td>s</td>
</tr>
<tr>
<td>authentication_timeout</td>
<td>s</td>
</tr>
<tr>
<td>autovacuum_naptime</td>
<td>s</td>
</tr>
<tr>
<td>checkpoint_timeout</td>
<td>s</td>
</tr>
<tr>
<td>checkpoint_warning</td>
<td>s</td>
</tr>
</tbody>
</table>
There is a bit of setup you need to do before you can use the PostGIS extension. The following list shows what you need to do. Each step is described in greater detail in this section.

- Connect to the DB instance using the master username 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 username used to create the DB instance

The master username you used to create the DB instance is automatically assigned the rds_superuser role. When you connect to the DB instance, you will be in 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:

```
mydb1=> select current_user;
 current_user
  --------
   myawsuser
(1 row)
```

### Step 2: Load the PostGIS extensions

Use the CREATE EXTENSION statements to load the PostGIS extensions. Note that you must also load the fuzzystrmatch extension. You can then use the \dn psql command to list the owners of the PostGIS schemas.

```
mydb1=> create extension postgis;
CREATE EXTENSION
mydb1=> create extension fuzzystrmatch;
CREATE EXTENSION
```
mydb1=> create extension postgis_tiger_geocoder;
CREATE EXTENSION
mydb1=> create extension postgis_topology;
CREATE EXTENSION
mydb1=> \dn
List of schemas
| Name    | Owner   |
|---------+---------|
| public  | myawsuser |
| tiger   | 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.

mydb1=> alter schema tiger owner to rds_superuser;
ALTER SCHEMA
mydb1=> alter schema topology owner to rds_superuser;
ALTER SCHEMA
mydb1=> \dn
List of schemas
| Name    | Owner     |
|---------+-----------|
| public  | myawsuser |
| tiger   | 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. You can run the function from the psql prompt:

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$
BEGIN
EXECUTE $1;
RETURN $1;
END;
$f$;
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

Test tiger by using the following SELECT statement:

```
mydb1=> 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:

```
mydb1=> select topology.createtopology('my_new_topo',26986,0.5);

createtopology
----------------
1
(1 row)
```

Using pgBadger for Log Analysis with Amazon RDS PostgreSQL

You can use a log analyzer such as pgbadger to analyze PostgreSQL logs. Although the pgbadger documentation states that the %l pattern (log line for session/process) should be a part of the prefix, 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 would correctly format 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
```
Tasks Common to All Amazon RDS DB Engines

Topics
- Renaming a DB Instance (p. 271)
- Deleting a DB Instance (p. 273)
- Rebooting a DB Instance (p. 276)
- Tagging Amazon RDS Resources (p. 278)
- Upgrading a DB Instance (p. 288)
- Backing Up and Restoring a DB Instance (p. 292)
- Working with Option Groups (p. 306)
- Working with DB Parameter Groups (p. 317)
- Working with DB Security Groups (p. 329)
- Working with Reserved DB Instances (p. 342)
- Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 353)
- Working with Provisioned IOPS Storage (p. 361)
- Adjusting the Preferred Maintenance Window (p. 373)

Most tasks you need to perform on a DB instance are performed the same way for all DB engines. Creating a DB instance, connecting to that DB instance, and importing data into that DB instance are all tasks that are specific for each DB engine. For information on performing these tasks on a DB instance for a specific DB engine, see the following topics.

This section covers the Amazon RDS operations you are most likely to use, and provides procedural instruction and examples.
Renaming a DB Instance

You can rename a DB instance by using the AWS Management Console, the `rds-modify-db-instance` command, or the `ModifyDBInstance` API 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 will be 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 will be 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.

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 Working with Read Replicas (p. 109).

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, select **DB Instances**.
3. Select the check box next to the DB instance you want to rename.
4. From the **Instance Actions** dropdown menu, select **Modify**.
5. Enter a new name in the **DB Instance Identifier** text box. Select the **Apply Immediately** check box, and then click **Continue**.

6. Click **Modify DB Instance** to complete the change.

### CLI

**To rename a DB instance**

- Use the command `rds-modify-db-instance` and provide the old `DBInstanceIdentifier` value and use the `-n` switch with the name of the new DB instance. The syntax is as follows:

  ```
  PROMPT>rds-modify-db-instance DBInstanceIdentifier -n NewDBInstanceIdentifier
  ```

### API

**To rename a DB instance**

- Call `ModifyDBInstance` with the following parameters:
  - `NewDBInstanceIdentifier = new name for the instance`

### Related Topics

- **Promoting a Read Replica to be a DB Instance (p. 114)**
- **DB Instance (p. 43)**
Deleting a DB Instance

You can delete a DB instance in any state and at any time. To delete a DB instance, you must specify
the name of the instance and specify if you want to have a final DB snapshot taken of the instance. If the
DB instance you are deleting has a status of "Creating," you will not be able to have a final DB snapshot
taken.

Important
If you choose not to create a final DB snapshot, you will not be able to later restore the DB
instance to its final state. When you delete a DB instance, all automated backups are deleted
and cannot be recovered. Manual DB snapshots of the instance are not deleted.

If the DB instance you want to delete has a read replica, you should either promote the read replica or
delete it. For more information on promoting a read replica, see Promoting a Read Replica to be a DB
Instance (p. 114)

In the following examples, you delete a DB instance both with and without a final DB snapshot.

Deleting a DB Instance with No Final Snapshot

You can skip creating a final DB snapshot if you want to quickly delete a DB instance. Note that when
you delete a DB instance, all automated backups are deleted and cannot be recovered. Manual snapshots
are not deleted.

AWS Management Console

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
2. In the DB Instances list, select the check box next to the DB instance you wish to delete.
3. Click Instance Actions, and then select Delete from the context menu.
4. Select No in the Create final Snapshot? drop-down list box.
5. Click Yes, Delete.

CLI

To delete a DB instance with no final DB snapshot

• Use the command rds-delete-db-instance to delete an instance.

```
PROMPT>rds-delete-db-instance mydbinstance mydbinstance --skip-final-snapshot
```

API

To delete a DB instance with no final DB snapshot

• Call DeleteDBInstance with the following parameters:

  • DBInstanceIdentifier = mydbinstance
  • SkipFinalSnapshot = true
Deleting a DB Instance with a Final Snapshot

You can create a final DB snapshot if you want to be able to restore a deleted DB instance at a later time. All automated backups will also be deleted and cannot be recovered. Manual snapshots are not deleted.

AWS Management Console

To delete a DB instance with a 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 DB Instances list, select the check box next to the DB Instance you wish to delete.
3. Click Instance Actions, and then select Delete from the context menu.
4. Select Yes in the Create final Snapshot? drop-down list box.
5. Type the name of your final DB snapshot into the Final Snapshot name text box.
6. Click Yes, Delete.

CLI

To delete a DB instance with a final DB snapshot

• Use the command rds-delete-db-instance to delete an instance.

PROMPT>rds-delete-db-instance mydbinstance mydbinstance --final-snapshot-identifier myfinaldbsnapshot

Once you begin deleting this database, it will no longer be able to accept connections.
Are you sure you want to delete this database? [Ny]y
DBINSTANCE  mydbinstance  2009-10-21T01:54:49.521Z  db.m1.large  MySQL  50
sa  deleting  us-east-1a  3
API

To delete a DB instance with a final DB snapshot

- Call `DeleteDBInstance` with the following parameters:
  - `DBInstanceIdentifier = mydbinstance`
  - `FinalDBSnapshotIdentifier = myfinaldbsnapshot`

Example

https://rds.amazonaws.com/
?Action=DeleteDBInstance
&DBInstanceIdentifier=mydbinstance
&FinalDBSnapshotIdentifier=myfinaldbsnapshot
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-14T22%3A20%3A46.297Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>

Related Topics

- Creating a DB Instance Running the MySQL Database Engine (p. 82)
- DB Instance (p. 43)
Rebooting a DB Instance

In some cases, if you modify a DB instance, the DB parameter group associated with the instance, or a static DB parameter in a parameter group the instances uses, you must reboot the instance for the changes to take effect.

Rebooting a DB instance restarts the database engine service. A reboot also applies to the DB instance any modifications to the associated DB parameter group that were pending. Rebooting a DB instance results in a momentary outage of the instance, during which the DB instance status is set to rebooting. If the Amazon RDS instance is configured for MultiAZ, it is possible that the reboot will be conducted through a failover. An Amazon RDS event is created when the reboot is completed.

If your DB instance is deployed in multiple Availability Zones, you can force a failover from one AZ to the other when you select the Reboot option. 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 will need to re-establish any existing connections to your DB instance. Reboot 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).

The time required to reboot is a function of the specific database engine's crash recovery process. To improve the reboot time, we recommend that you reduce database activities as much as possible during the reboot process to reduce 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, click Instances.
3. Select the check box of the DB instance that you want to reboot.
4. Select Instance Actions and then select Reboot from the drop down menu.
5. To force a failover from one AZ to another, select the Reboot with failover? check box in the Reboot DB Instance dialog box.
6. Click Yes, Reboot. To cancel the reboot instead, click Cancel.

CLI

To reboot a DB instance

- Use the rds-reboot-db-instance command. To force a failover from one AZ to the other, use the force-failover parameter.

PROMPT>rds-reboot-db-instance dbInstanceID --force-failover true
To reboot a DB instance

- Call RebootDBInstance. To force a failover from one AZ to the other, add the following parameter:
  - ForceFailover = true
Tagging Amazon RDS Resources

What You Should Know About Amazon RDS Resource Tags

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
- Read replicas
- DB snapshots
- Reserved DB instances
- Event subscriptions
- DB option groups
- DB parameter groups
- DB security groups
- DB subnet groups

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. A tag key could be used, for example, to define a category, and the tag value could be an item in that category. For example, you could 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 could 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 Account Billing.

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 ten 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. AWS does not automatically set any tags on Amazon RDS resources.

The following list describes the characteristics of a DB instance tag.

- 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 may contain only the set of Unicode letters, digits, white-space, ",", ",", ",", ",", ",", "+", "+", "+", "+" (Java regex: "^[\p{L}\p{Z}\p{N}_.:/=+\-]*$").
• The tag value is an optional string value of the tag. The string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with “aws:” or “rds:”. The string may contain only the set of Unicode letters, digits, white-space, ‘.', ',', ';', '=', '+', '-' (Java regex: “^[\p{L}\p{Z}\p{N}_.:/=+\-]+$”). 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.

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 Amazon RDS Amazon Resource Name (ARN) (p. 285).

Note that tags are cached for authorization purposes. Because of this, additions and updates to tags on Amazon RDS resources may take several minutes before they are available.

**AWS Management Console**

The process to tag an Amazon RDS resource is similar for all resources. The following example 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, click **Instances**.
   
   **Note**
   
   To filter the list of DB instances in the **DB Instances** pane, in the box beside the **Viewing** box, type a text string. Only DB instances that contain the string will appear.
3. Select the check box for the DB instance that you want to tag.
4. Click the details icon.
5. In the details pane, click **Tags**.
6. Click **Add/Edit Tags**.

7. Type a name and value for the tag. Click **Save Tags**.

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, click **Instances**.

   **Note**
   To filter the list of DB instances in the **DB Instances** pane, in the box beside the **Viewing** box, type a text string. Only DB instances that contain the string will appear.

3. Select the check box for the DB instance from which you want to remove a tag.

4. Click the details icon.

5. In the details pane, click **Tags**.
6. Click the red "X" in the Remove column next to the tag you want to delete.

7. Click the Save Tags button.

**CLI**

To add, list, or remove tags for a DB instance

- To add a tag to an Amazon RDS resource, use the `rds-add-tag-to-resource` command.
- To list tags that are assigned to an Amazon RDS resource, use the `rds-list-tags-for-resource` command.
- To remove tags from an Amazon RDS resource, use the `rds-remove-tags-from-resource` command.

To learn more about how to construct the required ARN, see Constructing an Amazon RDS Amazon Resource Name (ARN) (p. 285)
API

To add, list, or remove tags for a DB instance

- 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 Amazon RDS Amazon Resource Name (ARN) (p. 285)

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. Note that values for Key and Value are case dependent. For example, project=Trinity and PROJECT=Trinity are two distinct tags.

<table>
<thead>
<tr>
<th>Tagging element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagSet</td>
<td>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.</td>
</tr>
<tr>
<td>Tag</td>
<td>A tag is a user-defined key-value pair. There can be from 1 to 10 tags in a tag set.</td>
</tr>
<tr>
<td>Key</td>
<td>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 &quot;rds:&quot; or &quot;aws:&quot;. The string may only contain only the set of Unicode letters, digits, white-space, <em>, ., , , =, +, - (Java regex: &quot;^[\p{L}\p{Z}\p{N}</em>.:/=+-]*$&quot;. 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.</td>
</tr>
</tbody>
</table>

API Version 2013-09-09
284
Constructing an Amazon RDS Amazon Resource Name (ARN)

Resources that are created in Amazon Web Services are identified by a unique identifier call an Amazon Resource Name (ARN). If you use the CLI or Amazon RDS API to add, modify, or delete tags, you must supply the ARN of the resource you want to work with.

An ARN for an Amazon RDS resource uses the following syntax:

```
arn:aws:rds:<region>[:<account number>:]<resourcetype>:<name>
```

- `<region>` is the AWS region ID where the Amazon RDS resource was created, such as us-west-2.

The following table shows AWS region names and the value you should use when constructing an ARN.

<table>
<thead>
<tr>
<th>Region</th>
<th>Name</th>
<th>Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Northern Virginia) Region</td>
<td>us-east-1</td>
<td><a href="https://rds.us-east-1.amazonaws.com">https://rds.us-east-1.amazonaws.com</a></td>
</tr>
<tr>
<td>US West (Northern California) Region</td>
<td>us-west-1</td>
<td><a href="https://rds.us-west-1.amazonaws.com">https://rds.us-west-1.amazonaws.com</a></td>
</tr>
<tr>
<td>US West (Oregon) Region</td>
<td>us-west-2</td>
<td><a href="https://rds.us-west-2.amazonaws.com">https://rds.us-west-2.amazonaws.com</a></td>
</tr>
<tr>
<td>EU (Ireland) Region</td>
<td>eu-west-1</td>
<td><a href="https://rds.eu-west-1.amazonaws.com">https://rds.eu-west-1.amazonaws.com</a></td>
</tr>
<tr>
<td>Asia Pacific (Tokyo) Region</td>
<td>ap-northeast-1</td>
<td><a href="https://rds.ap-northeast-1.amazonaws.com">https://rds.ap-northeast-1.amazonaws.com</a></td>
</tr>
<tr>
<td>Asia Pacific (Singapore) Region</td>
<td>ap-southeast-1</td>
<td><a href="https://rds.ap-southeast-1.amazonaws.com">https://rds.ap-southeast-1.amazonaws.com</a></td>
</tr>
<tr>
<td>Asia Pacific (Sydney) Region</td>
<td>ap-southeast-2</td>
<td><a href="https://rds.ap-southeast-2.amazonaws.com">https://rds.ap-southeast-2.amazonaws.com</a></td>
</tr>
<tr>
<td>South America (Sao Paulo) Region</td>
<td>sa-east-1</td>
<td><a href="https://rds.sa-east-1.amazonaws.com">https://rds.sa-east-1.amazonaws.com</a></td>
</tr>
</tbody>
</table>
• `<account number>` is your account number with dashes omitted. To find your account number, log into your AWS account at http://aws.amazon.com, click My Account/Console, and then click My Account.

• `<resourcetype>` is the type of Amazon RDS resource.

The following table shows the resource type you should use when constructing an ARN for a particular Amazon RDS resource.

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>ARN Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB instance</td>
<td>arn:aws:rds:&lt;region&gt;:&lt;account&gt;:db:&lt;dbinstance name&gt;</td>
</tr>
<tr>
<td>DB option group</td>
<td>arn:aws:rds:&lt;region&gt;:&lt;account&gt;:og:&lt;option group name&gt;</td>
</tr>
<tr>
<td>DB parameter group</td>
<td>arn:aws:rds:&lt;region&gt;:&lt;account&gt;:pg:&lt;parameter group name&gt;</td>
</tr>
<tr>
<td>Reserved DB instance</td>
<td>arn:aws:rds:&lt;region&gt;:&lt;account&gt;:ri:&lt;reserve instance name&gt;</td>
</tr>
<tr>
<td>DB security group</td>
<td>arn:aws:rds:&lt;region&gt;:&lt;account&gt;:secgrp:&lt;security group name&gt;</td>
</tr>
<tr>
<td>DB snapshot</td>
<td>arn:aws:rds:&lt;region&gt;:&lt;account&gt;:snapshot:&lt;snapshot name&gt;</td>
</tr>
<tr>
<td>DB subnet group</td>
<td>arn:aws:rds:&lt;region&gt;:&lt;account&gt;:subgrp:&lt;subnet group name&gt;</td>
</tr>
</tbody>
</table>

• `<name>` is the resource identifier for the Amazon RDS resource.

The following table shows examples of ARNs for RDS resources with an AWS account of 123456789012, that were created in the US East (Northern Virginia) region, and that have a resource name that begins with “my-”:

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Sample ARN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB instance</td>
<td>arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance</td>
</tr>
<tr>
<td>DB option group</td>
<td>arn:aws:rds:us-east-1:123456789012:og:my-option-group-oracle-tde</td>
</tr>
<tr>
<td>DB parameter group</td>
<td>arn:aws:rds:us-east-1:123456789012:pg:my-param-enable-logs</td>
</tr>
<tr>
<td>Reserved DB instance</td>
<td>arn:aws:rds:us-east-1:123456789012:ri:my-reserved-multiaz</td>
</tr>
<tr>
<td>DB security group</td>
<td>arn:aws:rds:us-east-1:123456789012:secgrp:my-public</td>
</tr>
<tr>
<td>DB subnet group</td>
<td>arn:aws:rds:us-east-1:123456789012:subgrp:my-subnet-10</td>
</tr>
</tbody>
</table>
Related Topics

- What You Should Know About Amazon RDS Resource Tags (p. 278)
Upgrading a DB Instance

Amazon RDS supports the following major version upgrades:

- MySQL 5.1 to MySQL 5.5
- MySQL 5.5 to MySQL 5.6
- Microsoft SQL Server 2008 to SQL Server 2012

You use the Amazon RDS modify operation to perform a major version upgrade of a DB instance. You can also use the modify operation to manually apply a minor version upgrade.

Upgrade Overview

You control when to upgrade your DB 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.

By default, your DB Instance will automatically be upgraded to new minor versions as they are supported by Amazon RDS. This patching will occur during your scheduled maintenance window, and it will be announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades, set the AutoMinorVersionUpgrade parameter of a DB instance to "false".

If you opt out of automatically scheduled upgrades, you can manually upgrade a DB instance to a supported minor version release by following the procedure documented in this topic to modify the DB instance, specifying the new minor version number.

Because major version upgrades involve some compatibility risk, they will not occur automatically; you must manually modify the DB instance. You should thoroughly test any upgrade before applying your production instances.

Testing an Upgrade

Before you perform a major version upgrade on your DB instance, you should thoroughly test both your database and the host application for compatibility. We suggest you do the following:

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:
   - MySQL 5.5 Upgrade Documentation
   - MySQL 5.6 Upgrade Documentation
   - Upgrade to SQL Server 2012

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. 317)

3. Create a DB snapshot of the DB instance to be upgraded. For more information, see Creating a DB Snapshot (p. 296).

4. Restore the DB snapshot to create a new test DB instance. For more information, see Restoring From a DB Snapshot (p. 298)

5. Modify this new test DB instance to upgrade it to the new version, using one of the methods detailed below. 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 host applications to the upgraded DB instance.

8. If all tests pass, then perform the upgrade on your production DB instance. We suggest you do not allow writes to the DB instance until you can confirm that everything is working correctly.

## DB Instance Upgrade Considerations

If the 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 may not be available until the upgrade is complete.

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.

If Amazon RDS encounters any issues when upgrading SQL Server and needs to rollback, Amazon RDS restores your DB instance from the first DB snapshot.

For MySQL upgrades, the first snapshot is a user snapshot that is retained until you delete it. For SQL Server upgrades, the first snapshot is a system snapshot that is retained only until the end of the backup retention period.

MySQL major version upgrades typically complete in about 10 minutes, and SQL Server upgrades typically complete in about 30 minutes. Upgrades may take longer, depending on the class of DB instance, or whether the instance follows the operational guidelines in the Amazon RDS Best Practices (p. 6). If you are upgrading a DB instance from the Amazon RDS console, the status of the DB instance will indicate when the upgrade is complete. If you are using the CLI, use the `rds-describe-db-instance` command and check the `Status` value.

After the upgrade is complete, you cannot 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.

In addition to applying major version upgrades to instances of MySQL and SQL Server, you can manually apply minor version upgrades to instances of MySQL or Oracle. You can also specify that minor version upgrades be applied automatically to instances of MySQL and Oracle. For more information about managing minor version upgrades, see Amazon RDS MySQL Versions (p. 78) or Oracle Version Management (p. 136)

**Note**
During a 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.

**Note**
During a major version upgrade of MySQL, Amazon RDS runs the MySQL binary `mysql_upgrade` to upgrade tables, if required. Also, Amazon RDS may empty the slow_log and general_log tables.

## AWS Management Console

### To upgrade 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, click **Instances**.
3. Select the check box for the DB instance that you want to upgrade.
4. Click **Instance Actions** and click **Modify**.
5. In the **DB Engine Version** box, click the new version.
6. To upgrade immediately, select the **Apply Immediately** check box. To delay the upgrade to the next maintenance window, clear the check box.
7. Click **Continue**.
8. Review the modification summary information. To proceed with the upgrade, click **Modify DB Instance**. To cancel the upgrade instead, click the X in the upper right corner.

## CLI

**To upgrade a DB instance**

- Use the CLI command `rds-modify-db-instance` specifying the DB instance identifier and using the following parameters:
  
  - `--engine-version =` the new DB engine version
  - `--allow-major-version-upgrade = true`
  - `--apply-immediately = true` to upgrade immediately, or `false` to delay the upgrade until the next maintenance window

**Example**

```
PROMPT>rds-modify-db-instance SQLServer1 --engine-version 11.00.2100.60.v1 --allow-major-version-upgrade true --apply-immediately true
```

## API

**To upgrade a DB instance**

- Call **ModifyDBInstance** with the following parameters:
  
  - `DBInstanceIdentifier =` the identifier of the instance to be upgraded
  - `EngineVersion =` the new DB engine version
  - `AllowMajorVersionUpgrade = true`
  - `ApplyImmediately = true` to upgrade immediately, or `false` to delay the upgrade until the next maintenance window

**Example**

```
https://rds.amazonaws.com/ ?Action=ModifyDBInstance &DBInstanceIdentifier=MySQL-Instance1 &EngineVersion=5.5.31 &AllowMajorVersionUpgrade=true &ApplyImmediately=true
```
Upgrading from MySQL 5.1 to MySQL 5.6

To upgrade a MySQL version 5.1 DB instance on Amazon RDS to MySQL version 5.6, you must first upgrade your DB instance to MySQL version 5.5, as described in previous sections. Once you have upgraded to MySQL version 5.5, follow the instructions in this topic to upgrade your DB instance to version 5.6.

Upgrading from MySQL 5.5 to MySQL 5.6

To upgrade your MySQL DB version 5.5 instance on Amazon RDS to MySQL version 5.6, follow the instructions in the previous sections. However, MySQL 5.5 DB instances created before April 23, 2014 cannot be automatically upgraded using the console. If you attempt to upgrade a MySQL 5.5 DB instance created before April 23, 2014 using the Amazon RDS CLI or API, you will receive the following error:

```
The requested modify operation is currently unavailable for this
DB instance as it is running an older version of the
system software. You can however perform this modification by fol-
lowing a few steps. You can create a new Read Replica,
perform the modify operation on it, wait for the Read Replica to
catch up and then promote it. Refer to the Working with
Read Replicas section of the Amazon RDS User Guide to learn more.
Alternatively, you can take a snapshot and restore it
to a new DB instance and become eligible to perform this operation.
```

You can still upgrade your MySQL 5.5 DB instance created before April 23, 2014 to a MySQL 5.6 using the following procedure.

1. Using the Amazon RDS console, create a read replica of your MySQL 5.5 DB instance. This will create an upgradable copy of your database.
   a. In the console, click Instances and select the DB instance that you want to upgrade.
   b. Click Instance Actions and select Create Read Replica.
   c. Provide a 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. Click Yes, Create Read Replica.

2. When the read replica has been created and the Status shows as available, upgrade the read replica to MySQL 5.6.
   a. In the console, click Instances and select the read replica that you just created.
   b. Click Instance Actions and click Modify.
   c. In the DB Engine Version box, select the MySQL 5.6 version to upgrade to and select the Apply Immediately check box. Click Continue.
   d. Click Modify DB Instance to start the upgrade.

3. When the upgrade is complete and the Status shows as 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.
4. 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 will no longer be a replication slave to your MySQL 5.5 DB instance. It is recommended 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 transactions are suspended. When the promotion is completed, you can direct your write transactions to the upgraded MySQL 5.6 DB instance to ensure that no write transactions are lost.

   **Important**
   Prior to promoting your MySQL 5.6 read replica, it is recommended that you perform all necessary DDL operations, such as creating indexes, on the MySQL 5.6 read replica. This will avoid any effects on the performance of the MySQL 5.6 read replica after it has been promoted.

   a. In the console, click Instances and select the read replica that you just upgraded.
   b. Click Instance Actions and select Promote Read Replica.
   c. Set your desired backup behavior for the DB instance and click Continue.
   d. Click Yes, Promote Read Replica.

5. 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.

   **Note**
   You can also upgrade your MySQL 5.5 DB instance on Amazon RDS to MySQL 5.6 by creating a snapshot of your MySQL 5.5 DB instance, restore that snapshot to a new MySQL 5.5 DB instance, and then upgrade that new DB instance to MySQL version 5.6. However, updates made to your original MySQL 5.5 DB instance during the time it takes to create the snapshot and restore it to a new DB instance will not be present in your upgraded DB instance. As a result, it is recommended that you use read replicas as described in this section to upgrade your DB instance.

---

**Backing Up and Restoring a DB Instance**

This section shows how to backup and restore a DB instance.

**Topics**
- Working With Automated Backups (p. 293)
- Creating a DB Snapshot (p. 296)
- Restoring From a DB Snapshot (p. 298)
- Copying a DB Snapshot (p. 301)
- Restoring a DB Instance to a Specified Time (p. 304)
Working With Automated Backups

Amazon RDS can automatically back up all of your DB instances. You can set the backup retention period when you create a DB instance. If you don't set the backup retention period, Amazon RDS uses a default period retention period of one day. You can modify the backup retention period; valid values are 0 (for no backup retention) to a maximum of 35 days.

All automated backups are deleted and cannot be recovered when you delete a DB instance. Manual snapshots are not deleted. For information on pricing for storing manual snapshots long-term, see Amazon RDS Pricing.

In this example, you will enable and then disable backups for an existing DB instance called mydbinstance.

Disabling Automated Backups

You may 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. 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 these examples, you disable automated backups for a DB instance by setting the backup retention parameter to 0.

AWS Management 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, click DB Instances, and then select the check box next to the DB instance you want to modify.
3. Click the Modify button.
   
   The Modify DB Instance window appears.
4. Select 0 in the Backup Retention Period drop-down list box.
5. Check the Apply Immediately check box.
6. Click the OK button.

CLI

To disable automated backups immediately

1. Set the backup retention period to 0.

   PROMPT>rds-modify-db-instance mydbinstance --backup-retention-period 0 --apply-immediately

2. Call rds-describe-db-instances for the DB instance until the value for backup retention period is 0 and mydbinstance status is available.

   PROMPT>rds-describe-db-instances mydbinstance --headers
API

To disable automated backups immediately

- Call ModifyDBInstance 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. The same request used to disable automated backups can be used to enable them by using a non-zero value for the backup retention period. When automated backups are enabled, a backup is immediately created.

All automated backups are deleted and cannot be recovered when you delete a DB instance. Manual snapshots are not deleted.

In this example, you enable automated backups for a DB instance by setting the backup retention period parameter for the DB instance to a non-zero value (in this case, 3).

AWS Management 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, click DB Instances, and then select the check box next to the DB instance you want to modify.
3. Click the Modify button or right-click the DB instance and select Modify from the context menu. The Modify DB Instance window appears.
4. Select 3 in the Backup Retention Period drop-down list box.
5. Check the Apply Immediately check box.
6. Click the OK button.

CLI

To enable automated backups immediately
In this example, we will enable automated backups by setting the backup retention period to 3.

- Set the backup retention period to 3.

```
PROMPT>rds-modify-db-instance mydbinstance --backup-retention-period 3 --apply-immediately
```

**API**

To enable automated backups immediately

- Call `ModifyDBInstance` with the following parameters:
  - `DBInstanceIdentifier = mydbinstance`
  - `BackupRetentionPeriod = 3`
  - `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>
```

**Related Topics**

- Restoring a DB Instance to a Specified Time (p. 304)
- DB Instance Backups (p. 55)
Creating a DB Snapshot

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.

**Note**
Creating a DB snapshot creates a backup of your DB instance. Creating this backup on a Single-AZ DB instance results in a brief I/O suspension that typically lasting no more than a few minutes. Multi-AZ DB instances are not effected by this I/O suspension since the backup is taken on the standby.

In this example, you create a DB snapshot called `mydbsnapshot` for a DB instance called `mydbinstance`.

### 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/](https://console.aws.amazon.com/rds/).
2. In the navigation pane, click **DB Instances**.
3. Click **Instance Actions**, and then click **Take DB Snapshot**.

   The **Take DB Snapshot** window appears.

4. Type the name of the snapshot in the **Snapshot Name** text box.

   5. Click **Yes, Take Snapshot**.

### CLI

**To create a DB snapshot**

- Use the command `rds-create-db-snapshot` to create a database snapshot.

```bash
PROMPT>rds-create-db-snapshot -i mydbinstance -s mydbsnapshot
```

The output from this command should look similar to the following:

```
DBSNAPSHOT  mydbsnapshot  mydbinstance  2009-10-21T01:54:49.521Z  MySQL
```
To create a DB snapshot

- Call CreateDBSnapshot with the following parameters:
  - `DBSnapshotIdentifier = mydbsnapshot`
  - `DBInstanceIdentifier = mydbinstance`

Example

```text
https://rds.amazon.com/
?Action=CreateDBSnapshot
&DBSnapshotIdentifier=mydbsnapshot
&DBInstanceIdentifier=mydbinstance
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-14T17%3A48%3A21.746Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Related Topics

- Restoring From a DB Snapshot (p. 298)
- Copying a DB Snapshot (p. 301)
- DB Instance Backups (p. 55)
Restoring From a DB Snapshot

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. You cannot restore from a DB snapshot to an existing
DB instance; a new DB instance is created when you restore.

When you restore a DB instance, only the default DB parameter and security groups are associated with
the restored instance. As soon as the restore is complete, you should associate the custom DB parameter
or security group you used by the instance you restored from. You must apply these changes explicitly
using the RDS console’s Modify command, the ModifyDBInstance API, or the
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.

Note
If you use Oracle GoldenGate, 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 option group associated with the DB snapshot is associated with the restored DB instance once 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, which had the TDE option. When an option
group is assigned to a DB instance, it is also 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 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 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.

You can change to a different edition of the DB engine when restoring from a DB snapshot only if the DB
snapshot has the required storage allocated for the new edition. For example, to change from SQL Server
Web Edition to SQL Server Standard Edition, the DB snapshot must have been created from a SQL
Server DB instance that had at least 200 GB of allocated storage, which is the minimum allocated storage

In this example, you restore from a previously created DB snapshot called mydbsnapshot and create a
new DB instance called mynewdbinstance.

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
2. In the navigation pane, click Snapshots.
3. Click on the DB snapshot that you want to restore from.
4. Click **Restore Snapshot**.

   The **Restore DB Instance** window appears.

5. Type the name of the restored DB instance in the **DB Instance Identifier** text box.

6. Click the **Launch DB Instance** button.

**CLI**

**To restore a DB instance from a DB snapshot**

- Use the command `rds-restore-db-instance-from-db-snapshot` to restore a DB snapshot to a new DB instance.

```
PROMPT>rds-restore-db-instance-from-db-snapshot mynewdbinstance -s mydbsnapshot
```

This command returns output similar to the following:

```
DBINSTANCE  mynewdbinstance  db.m1.large  MySQL     50       sa creating  3  n  5.1.57  general-public-license
```

**API**

**To restore a DB instance from a DB snapshot**

- Call `RestoreDBInstanceFromDBSnapshot` with the following parameters:
  
  - `DBSnapshotIdentifier = mydbsnapshot`
  - `DBInstanceIdentifier = mynewdbinstance`
Example

https://rds.amazon.com/
?Action=RestoreDBInstanceFromDBSnapshot
&DBSnapshotIdentifier=mydbsnapshot
&DBInstanceIdentifier=mynewdbinstance
&DBInstanceClass=db.m1.xlarge
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-15T17%3A48%3A21.746Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>

Related Topics

- Creating a DB Snapshot (p. 296)
- Copying a DB Snapshot (p. 301)
- DB Snapshots (p. 57)
Copying a DB Snapshot

Amazon RDS supports two types of DB snapshot copies. First, you can copy an automated DB snapshot to create a manual snapshot in the same AWS region that will be retained after the automated DB snapshot is deleted. Second, you can copy either an automated or manual DB snapshot from one region to create a manual DB snapshot in another region.

Amazon RDS deletes an automated DB snapshot at the end of its retention period, when you disable automated DB snapshots for the DB instance, or when you delete the DB instance. If you want to keep an automated DB snapshot for a longer period, you can copy it, which creates a manual DB snapshot. Manual DB snapshots are retained until you delete them.

You can also copy a DB snapshot from one region to another. You can copy either an automated or a manual DB snapshot from the source region. You perform the DB snapshot copy in the destination region, and use an Amazon RDS ARN to specify the location of the DB snapshot in the source region. For information about Amazon RDS ARN formats, see Constructing an Amazon RDS Amazon Resource Name (ARN) (p. 285).

**Note**

You cannot copy a DB snapshot to or from the AWS GovCloud (US) region. You also cannot copy a DB snapshot across regions if it was created from a DB instance that is using Oracle TDE.

You can have one active cross region DB snapshot copy per destination region per AWS customer account. Copying a snapshot out of the source region incurs Amazon RDS data transfer charges. For more information about Amazon RDS data transfer pricing, go to Amazon Relational Database Service Pricing.

Depending on the regions involved and the amount of data to be copied, a cross region snapshot could take hours to complete. If there are large numbers of cross region DB snapshot copy requests from a given source region, Amazon RDS may queue new cross region copy requests for that source region until some of the in-progress copies have completed. No progress information is displayed about the copy requests while they are in the queue. Progress information is displayed when the copy starts.

After the DB snapshot copy has been created in the new region, the copy behaves the same as all other DB snapshots in that region. For example, the following CLI copy command results in a DB snapshot in the us-west-2 region with the identifier `mysql-instance1-snapshot-20130805-copy`.

```bash
```

When the copy is finished, the AWS Management Console will show the DB snapshot with the name `mysql-instance1-snapshot-20130805-copy` in your list of DB snapshots in us-west-2. You can perform all DB snapshot actions on the DB snapshot identifier. For example, running the following CLI command in the us-west-2 region will create a new DB instance with data from the DB snapshot copy:

```bash
rds-restore-db-instance-from-db-snapshot mysql-instance1-west --region us-west-2 --db-snapshot-identifier mysql-instance1-snapshot-20130805-copy
```

A snapshot copied across regions does not include either the parameter group or option group that was used by the DB instance the snapshot was created from. When you restore a snapshot to create a new DB instance, that DB instance is assigned the default parameter group and default option group for the region it is created in. To give the new DB instance the same parameters and options as the source, you must do the following:
1. In the destination region, create a parameter group with the same settings as the parameter group used by the source DB instance, or note the name of an existing parameter group that has those settings.

2. In the destination region, create an option group with the same settings as the option group used by the source DB instance, or note the name of an existing option group that has those settings.

3. After restoring the snapshot in the destination region, modify the new DB instance to add the parameter group and option group available in the destination region.

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, click Snapshots.


   Select the check box for the automated DB snapshot you want to copy.

   Click Copy Snapshot

   The Copy DB Snapshot window appears.

4. Verify that the name of the automated DB snapshot you want to copy appears in the Source DB Snapshot: field.

To copy the DB snapshot to a different region, select that region in the Destination Region: list box.

Type the name of the DB snapshot copy in the New DB Snapshot Identifier: text box.

5. Click Yes, Copy Snapshot.

CLI

To copy a DB snapshot

- Use the rds-copy-db-snapshot command to copy a DB snapshot.
Example

```
rds-copy-db-snapshot -s rds:mydbinstance-2013-09-04-22-50 -t mydbsnapshotcopy
```

The output from this command should look similar to the following:

```
default:mysql-5-6  5.6.12  general-public-license  manual
```

API

To copy a DB snapshot

- Call `CopyDBSnapshot` with the following parameters:
  - `SourceDBSnapshotIdentifier = rds:mydbinstance-2013-09-04-22-50`
  - `TargetDBSnapshotIdentifier = mydbsnapshotcopy`

Example

```
https://rds.amazon.com/
?Action=CopyDBSnapshot
&SourceDBSnapshotIdentifier=rds:mydbinstance-2013-09-04-22-50
&DBInstanceIdentifier=mydbsnapshotcopy
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2011-12-12T06%3A27%3A42.551Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Related Topics

- Creating a DB Snapshot (p. 296)
- Restoring From a DB Snapshot (p. 298)
- DB Instance Backups (p. 55)
Restoring a DB Instance to a Specified Time

The Amazon RDS automated backup feature automatically creates a backup of your database. This backup occurs during a daily user-configurable 30 minute period known as the backup window. Automated backups are kept for a configurable number of days (called the backup retention period). You can restore your DB instance to any specific time during this retention period, 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, ModifyDBInstance API, or the rds-modify-db-instance command line tool once the DB instance is available.

You can restore to any point in time during your backup retention period. To determine the latest restorable time for a DB instance, use the rds-describe-db-instance command with the --show-long and --headers parameters and look at the value returned in the Latest Restorable Time column. The latest restorable time for a DB instance is typically within 5 minutes of the current time.

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.

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.

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.

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, click DB Instances.
3. Click Instance Actions, and then click Restore To Point In Time.
   
   The Restore DB Instance window appears.
4. Click on the Use Custom Restore Time radio button.
5. Enter the date and time that you wish to restore to in the Use Custom Restore Time text boxes.
6. Type the name of the restored DB instance in the DB Instance Identifier text box.
7. Click the Launch DB Instance button.

CLI

To restore a DB instance to a specified time
Use the command `rds-restore-db-instance-to-point-in-time` to create a new database instance.

```
PROMPT>rds-restore-db-instance-to-point-in-time mytargetdbinstance -s mysourcedbinstance -r 2009-10-14T23:45:00.000Z
```

**API**

To restore a DB instance to a specified time

- Call `RestoreDBInstanceToPointInTime` with the following parameters:
  - `SourceDBInstanceIdentifier = mysourcedbinstance`
  - `TargetDBInstanceIdentifier = mytargetdbinstance`
  - `RestoreTime = 2009-10-14T23:45:00.000Z`

**Example**

```
https://rds.amazonaws.com/?Action=RestoreDBInstanceToPointInTime
&SourceDBInstanceIdentifier=mysourcedbinstance
&TargetDBInstanceIdentifier=mytargetdbinstance
&RestoreTime=2009-10-14T23:45:00.000Z
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-15T17%3A48%3A21.746Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

**Related Topics**

- Creating a DB Snapshot (p. 296)
- Restoring From a DB Snapshot (p. 298)
- Copying a DB Snapshot (p. 301)
- DB Instance Backups (p. 55)
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.

Note
Currently, option groups are available for Oracle, Microsoft SQL Server, and MySQL 5.6 DB instances. For more information about individual Oracle options, see Appendix: Options for Oracle DB Engine (p. 156). For more information about SQL Server options, see Appendix: Options for SQL Server DB Engine (p. 236). For more information about MySQL 5.6 options, see Appendix: Options for MySQL DB Engine (p. 123).

Topics
• Things You Should Know About Option Groups (p. 306)
• Creating an Option Group (p. 308)
• Adding an Option to an Option Group (p. 309)
• Listing the Options and Option Settings for an Option Group (p. 312)
• Modifying an Option Setting (p. 313)
• Removing an Option from an Option Group (p. 315)

Things You Should Know About Option Groups

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 you create derives its settings from the default option group. To apply an option to a DB instance, you must create an option group (or use an existing option group) and then add one or more options to the option group. You can then associate the DB instance with that option group. To remove all options from a DB instance at once, you associate the DB instance with the default (empty) option group. 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.

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 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 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.

The following options are currently supported on Amazon RDS:
### Things You Should Know About Option Groups

<table>
<thead>
<tr>
<th>DB Engine</th>
<th>Option ID</th>
<th>Description</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>OEM</td>
<td>Oracle Database Manager Database Control. Default port: 1158.</td>
<td>port value setting</td>
</tr>
<tr>
<td>Oracle</td>
<td>XMLDB</td>
<td>Oracle XML DB support</td>
<td>No settings</td>
</tr>
<tr>
<td>Oracle</td>
<td>APEX, APEX-DEV</td>
<td>Oracle Application Express (APEX)</td>
<td>No settings</td>
</tr>
<tr>
<td>Oracle</td>
<td>NATIVE_NETWORK_ENCRYPTION</td>
<td>Oracle native network encryption, a feature of the Oracle Advanced Security option available in Oracle Enterprise Edition.</td>
<td>For a description of all option settings for NATIVE_NETWORK_ENCRYPTION, see Oracle Native Network Encryption.</td>
</tr>
<tr>
<td>Oracle</td>
<td>STATSPACK</td>
<td>Oracle Statspack performance statistics</td>
<td>No settings</td>
</tr>
<tr>
<td>Oracle</td>
<td>TDE</td>
<td>Oracle Transparent Data Encryption (TDE)</td>
<td>No settings</td>
</tr>
<tr>
<td>MySQL</td>
<td>MEMCACHED</td>
<td>MySQL 5.6 memcached interface to InnoDB tables</td>
<td>For a list and description of all the supported memcached parameters, see MySQL 5.6 memcached Support.</td>
</tr>
<tr>
<td>SQL Server</td>
<td>TDE</td>
<td>SQL Server Transparent Data Encryption</td>
<td>No settings</td>
</tr>
</tbody>
</table>

Both DB instances and DB snapshots can be associated with an option group. When you restore from a DB snapshot or when you 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, but the new option group must contain any persistent or permanent options that were included in the original option group.

Option group changes must be applied immediately in two cases: 1) When you add an option that adds or updates a port value, such as the OEM option, or 2) when you add or remove an option group with an option that includes a port value. In these cases the Apply-Immediately parameter must be enabled. Options that don't include port values can be applied immediately or can be applied during the next maintenance window for the DB instance.

Options require additional memory in order to run on a DB instance, so you may need to launch a larger instance, depending on your current use of the 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.

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, cannot be removed from an option group while DB instances are associated with the option group. All DB instances must be disassociated from the option group before a persistent option can be removed. When restoring from a snapshot or when performing a point-in-time-restore, if the option group associated with the DB snapshot contains a persistent option, the restored DB instance can only be associated 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 restoring from a DB snapshot or when performing a point-in-time-restore, if the option group associated with the snapshot contains a permanent option, the restored DB instance can only be associated with an option group with the permanent option.

Each DB instance indicates the status of its association an option group. For example, a status of **Active** indicates the DB instance is associated with the option group while a status of **Invalid** indicates that the option group associated with the DB instance does not contain the options the DB instance requires. If you query a DB instance for the status of its associated option group, Amazon RDS may also return a **ChangeState** value such as **Pending** or **Applying** when it is attempting to change the association from one state to another. For example, the status of the association of a DB instance in an option group could be **Creating/Pending**.

### Creating an Option Group

**AWS Management Console**

1. Sign in to the AWS Management Console and open the Amazon RDS console at [https://console.aws.amazon.com/rds/](https://console.aws.amazon.com/rds/).
2. In the navigation pane, click **Option Groups**.
3. Click **Create Group**.
4. In the **Create Option Group** dialog box, do the following:
   - In the **Name** box, type a name for the option group that is unique within your AWS account. The name can contain only letters, digits, and hyphens.
   - In the **Description** box, type a brief description of the option group. The description will be used for display purposes.
   - In the **Engine** box, click the DB engine that you want.
   - In the **Major Engine Version** box, click the major version of the DB engine that you want.

5. To continue, click **Yes, Create**. To cancel the operation instead, click **Cancel**.

You have now created a new option group with no options. See the next section, **Adding an Option to an Option Group (p. 309)** to add an option to the option group. Once you have added the options you want, you can then associate a DB instance with the option group so the options become available on the DB instance. For information about associating a DB instance with an option group, see **Modifying a DB Instance Running the Oracle Database Engine (p. 147)**.

**CLI**

To create an option group

- Call the `rds-create-option-group` command.

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.

```
PROMPT> rds-create-option-group TestOptionGroup --engine-name oracle-ee
```
You have now created a new option group with no options. See the next section, Adding an Option to an Option Group (p. 309) to add an option to the option group. Once you have added the options you want, you can then associate a DB instance with the option group so the options become available to the DB instance. For information about associating an Oracle DB instance with an option group, see Modifying a DB Instance Running the Oracle Database Engine (p. 147). For information about associating a MySQL DB instance with an option group, see Modifying a DB Instance Running the MySQL Database Engine (p. 93).

API

To create an option group

• Call the CreateOptionGroup action.

Adding an Option to an Option Group

You can add an option to an existing option group.

AWS Management Console

To add an option to 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, click Option Groups.
3. Select the check box for the option group that you want to modify, and then click Add Option.
In the Add Option dialog box, do the following:

- Click the option that you want to add. You may need to provide additional values, depending on the option that you select. For example, when you select the OEM option, you must also enter a port value and specify a DB security group.

- To enable the option on all associated DB instances as soon as you add it, under Apply Immediately, click Yes. If you click No (the default), the option will be enabled for each associated DB instance during its next maintenance window.

4. When the settings are as you want them, click Add Option.
CLI

To add an option to an option group

- Run the `rds-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` argument. By default, the option will be enabled for each associated DB instance during its next maintenance window.

The following example adds the Oracle Enterprise Manager Database Control (OEM) to an option group named `TestOptionGroup` and immediately enables it. Note that even if you use the default security group, you must specify it.

```
PROMPT> rds-add-option-to-option-group TestOptionGroup --option-name OEM --security-groups default --apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP testoptiongroup oracle-ee 11.2 Test option group
OPTION OEM 1158 Oracle Enterprise Manager
SECGROUP default authorized
```

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.

```
PROMPT> rds-add-option-to-option-group my-option-group --option-name OEM --port 5432 --security-groups default
```

Command output is similar to the following:

```
OPTIONGROUP my-option-group oracle-se 11.2 My option group
OPTION OEM 1158 Oracle Enterprise Manager
SECGROUP default authorized
```

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.

```
PROMPT> rds-add-option-to-option-group my-option-group --option-name NATIVE_NETWORK_ENCRYPTION --settings "SQLNET.ENCRYPTION_SERVER=REQUIRED; SQLNET.ENCRYPTION_TYPES_SERVER=AES256,AES192,DES"
```

Command output is similar to the following:
## 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

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, click **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

- Run the `rds-describe-option-groups` command. Specify the name of the option group whose options and settings you want to view. If you do not specify an option group name, all option groups are described.
The following example lists the options and option settings for an option group named TestOptionGroup.

```bash
PROMPT>rds-describe-option-groups TestOptionGroup
```

**API**

To list the options and option settings for an option group

- Call the `DescribeOptionGroups` action.

**Modifying an Option Setting**

After you have added an option that has modifiable option settings, you can modify the settings at any time.

**AWS Management Console**

To modify an option setting

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, click **Option Groups**.
3. Click the check box for the option group whose option that you want to modify, and then click **Modify Option**.
4. In the **Modify Option** dialog box, in the **Installed Options** box, click 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, under **Apply Immediately**, click **Yes**. If you click **No** (the default), the option will be enabled for each associated DB instance during its next maintenance window.
6. When the settings are as you want them, click **Modify Option**.

**CLI**

To modify an option setting

- Run the `rds-add-option-to-option-group` command with the option group and option that you want to modify. To apply the change immediately to all associated DB instances, include the `--apply-immediately` argument. By default, the option will be enabled for each associated DB instance during its next maintenance window.

To set an option setting, use the `--settings` argument. For more information on what settings are available for the various options, see Things You Should Know About Option Groups (p. 306)

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.
PROMPT> rds-add-option-to-option-group TestOptionGroup --option-name OEM -p 5432 --apply-immediately

Command output is similar to the following:

```
OPTIONGROUP testoptiongroup oracle-ee 11.2 Test Option Group
  OPTION OEM 5432 Oracle Enterprise Manager
  SECGROUP default authorized
```

The following example modifies the Oracle option NATIVE_NETWORK_ENCRYPTION and changes the option settings.

PROMPT> rds-add-option-to-option-group my-option-group -n NATIVE_NETWORK_ENCRYPTION --settings "SQLNET.ENCRYPTION_SERVER=REQUIRED; SQLNET.ENCRYPTION_TYPES_SERVER=AES256,AES192,DES"

Command output is similar to the following:

```
OPTIONGROUP Group Name Engine Major Engine Version Description
  VpcSpecific
OPTIONGROUP my-option-group oracle-ee 11.2 My option group
  OPTION Name Persistent Permanent Description
    NATIVE_NETWORK_ENCRYPTION n n Oracle Advanced Security - Native Network Encryption

OPTIONSETTING Name Description Value
  SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER Specifies list of checksumming algorithms in order of intended use SHA1,MD5 true
  SQLNET.ENCRYPTION_TYPES_SERVER Specifies list of encryption algorithms in order of intended use AES256,AES192,DES true
  SQLNET.ENCRYPTION_SERVER Specifies the desired encryption behavior REQUIRED true
  SQLNET.CRYPTO_CHECKSUM_SERVER Specifies the desired data integrity behavior REQUESTED true
```

**API**

To modify an option setting

- Call **ModificarOptionGroup** action.
Removing an Option from an Option Group

You can remove an option from an option group. Even if you remove all options from an option group, Amazon RDS will not delete it. DB instances that are associated with the empty option group will continue to be associated with it; they just won’t have access to any options.

You can remove an option from an option group as long as the option is not persistent or permanent. 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.

Even if you remove all options from an option group, Amazon RDS will not delete it. DB instances that are associated with the empty option group will continue to be associated with it; they just won’t have access to any options.

AWS Management Console

To remove an option from 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, click Option Groups.
3. Select the check box for the option group whose option you want to remove, and then click Delete Option.

4. In the Delete Option dialog box, do the following:
   • Select the check box that corresponds to the option that you want to delete.
   • For the deletion to take effect as soon as you make it, under Apply Immediately, click Yes. If you select No (the default), the option will be deleted for each associated DB instance during its next maintenance window.
5. When the settings are as you want them, click **Yes, Delete**.

**CLI**

To remove an option from an option group

- Run the `rds-remove-option-from-option-group` command with the option that you want to delete. To apply the change immediately, include the `--apply-immediately` argument. By default, the option is deleted on each associated DB instance during its next maintenance window.

The following example removes the Oracle Enterprise Manager Database Control (OEM) option from an option group named `TestOptionGroup` and immediately applies the change.

```
PROMPT> rds-remove-option-from-option-group TestOptionGroup -options OEM -apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP testoptiongroup oracle-ee 11.2 Test option group
```

**API**

To remove an option from an option group

- Call the `ModifyOptionGroup` action.
Working with DB Parameter Groups

What You Should Know About DB Parameter Groups

You manage the DB engine configuration through the use of DB parameter groups. 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 used if you create a DB instance without specifying a DB parameter group. This default group contains database engine defaults and Amazon RDS system defaults based on the engine, compute class, and allocated storage of the instance. Note that you cannot modify the parameter settings of a default DB parameter group—you must create your own parameter group to change the settings—and not all DB engine parameters are available for modification in a DB parameter group.

If you want your DB instance to run a user-modified DB parameter group, you simply create a new DB parameter group, modify the desired parameters, and modify the DB instance to use the new DB parameter group. All DB instances that are members of a particular DB parameter group get all parameter updates to that DB parameter group.

Once you create a DB parameter group, you need to add your DB instance as a member. When you add a running DB instance to a DB parameter group, you must reboot the DB instance for the new DB parameter group settings to take effect.

The value for a DB parameter can be specified as an integer, or an integer expression built from formulas, variables, functions, and operators. For more information, see DB Parameter Values (p. 326)

Caution
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.

Note
Some MySQL parameters are constrained or disabled for a DB instance. For more information, please see MySQL Parameter Exceptions for Amazon RDS DB Instances (p. 82).

Creating a DB Parameter Group

In this example, you create a new DB parameter group.

Important
After you create 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 create 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 `rds-describe-db-parameters` command to verify that your DB parameter group has been created or modified.

**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. Click **DB Parameter Groups** in the **Navigation** list on the left side of the window.
3. Click the **Create DB Parameter Group** button.

   The **Create DB Parameter Group** window appears.
4. Select a DB parameter group family in the **DB Parameter Group Family** drop-down list box.
5. Type the name of the new DB parameter group in the **DB Parameter Group** text box.
6. Type a description for the new DB parameter group in the **Description** text box.
7. Click the **Yes, Create** button.

**CLI**

To create a DB parameter group

- Use the `rds-create-db-parameter-group` command. The following example creates a DB parameter group named `mydbparametergroup` for MySQL version 5.1 with a description of "My new parameter group."

```bash
PROMPT>rds-create-db-parameter-group mydbparametergroup -f MySQL5.1 -d "My new parameter group"
```

This command produces output similar to the following:

```bash
DBPARAMETERGROUP  mydbparametergroup  mysql5.1  My new parameter group
```

**API**

To create a DB parameter group

- Call `CreateDBParameterGroup`. The following examples creates a new parameter group named `mydbparametergroup` for MySQL version 5.1 and that has the description "My new parameter group."
Example

https://rds.amazonaws.com/
?Action=CreateDBParameterGroup
&DBParameterGroupName=mydbparametergroup
&Description=My%20new%20parameter%20group
&DBParameterGroupFamily=MySQL5.1
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2012-01-15T22%3A06%3A23.624Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>

The command returns a response like the following:

```xml
<CreateDBParameterGroupResponse xmlns="http://rds.amazonaws.com/admin/2012-01-15/">
  <CreateDBParameterGroupResult>
    <DBParameterGroupFamily>mysql5.1</DBParameterGroupFamily>
    <Description>My new parameter group</Description>
    <DBParameterGroupName>mydbparametergroup</DBParameterGroupName>
  </DBParameterGroup>
</CreateDBParameterGroupResult>
<ResponseMetadata>
  <RequestId>700a8afe-0b81-11df-85f9-eb5c71b54ddc</RequestId>
</ResponseMetadata>
</CreateDBParameterGroupResponse>
```

Modifying a DB Parameter Group

You can modify parameter values in a user-created DB parameter group, but you cannot change the parameter values in a default DB parameter group. Changes to parameters in a DB parameter group are applied to DB instances that are members of the DB parameter group either immediately or on the next RebootDbInstance API call of the DB instance, depending on the type of the parameter (dynamic or static) and the apply method chosen for the parameter update.

<table>
<thead>
<tr>
<th>DB Engine</th>
<th>Apply Immediately</th>
<th>Pending Reboot</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>Dynamic</td>
<td>Dynamic and Static</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Dynamic</td>
<td>Dynamic and Static</td>
</tr>
<tr>
<td>Oracle</td>
<td>Dynamic</td>
<td>Dynamic and Static</td>
</tr>
<tr>
<td>SQL Server</td>
<td>Dynamic</td>
<td>Static only</td>
</tr>
</tbody>
</table>
Important
After you modify 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 modify 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 `rds-describe-db-parameters` command to verify that your DB parameter group has been created or modified.

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. Click **DB Parameter Groups** in the **Navigation** list on the left side of the window.
   
   The available DB parameter groups appear in the **My DB Parameter Groups** list.
3. In the **My DB Parameter Groups** list, select the parameter group you want to modify.
4. In the information panel at the bottom of the window, select **Edit Parameters**.
5. In the **Edit Parameters** dialog box, change the values of the parameters you want to modify. You can scroll through the parameters using the arrow keys at the top right of the dialog box.
   
   Note that you cannot change values in a default parameter group.
6. Click **Save Changes**, then click **Close** in the confirmation dialog box.

CLI

To modify a DB parameter group

- Use the `rds-modify-db-parameter-group` command. The following example modifies the `max_connections` and `max_allowed_packet` values in the DB parameter group named `mydbparametergroup`.

  ```bash
  Note
  Amazon RDS does not support passing multiple comma-delimited parameter values for a single parameter.
  
  PROMPT>rds-modify-db-parameter-group mydbparametergroup --parameters "name=max_connections,value=250,method=immediate" --parameters "name=max_allowed_packet,value=1024,method=immediate"
  
  The command produces output like the following:
  
  DBPARAMETERGROUP  mydbparametergroup
  ```
API

To modify a DB parameter group

Note
Amazon RDS does not support passing multiple comma-delimited parameter values for a single parameter.

- Call the `ModifyDBParameterGroup` action. The following example modifies the `max_connections` and `max_allowed_packet` values in the DB parameter group named `mydbparametergroup`.

**Example**

```xml
https://rds.amazonaws.com/
?Action=ModifyDBParameterGroup
&DBParameterGroupName=mydbparametergroup
&Parameters.member.1.ParameterName=max_allowed_packet
&Parameters.member.1.ParameterValue=1024
&Parameters.member.1.ApplyMethod=immediate
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2012-01-15T22%3A29%3A47.865Z
```

The command returns a response like the following:

```xml
<ModifyDBParameterGroupResponse xmlns="http://rds.amazonaws.com/admin/2012-01-15/">
  <ModifyDBParameterGroupResult>
    <DBParameterGroupName>mydbparametergroup</DBParameterGroupName>
  </ModifyDBParameterGroupResult>
  <ResponseMetadata>
    <RequestId>3b824e10-0b87-11df-972f-21e99bc6881d</RequestId>
  </ResponseMetadata>
</ModifyDBParameterGroupResponse>
```

**Listing DB Parameter Groups**

You can list the DB parameter groups you've created for your AWS account.

Note
The `default.mysql5.5` parameter group family is automatically created the first time you describe engine default parameters for the MySQL5.5 parameter group family or the first time you create a MySQL 5.5-based DB instance without specifying a custom MySQL 5.5 parameter group.

**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. Click DB Parameter Groups in the Navigation list on the left side of the window.

The DB parameter groups appear in the My DB Parameter Groups list.

**CLI**

**To list all DB parameter groups for an AWS account**

- Use the command `rds-describe-db-parameter-groups` command. The following example lists all available DB parameter groups for an AWS account.

```
PROMPT>rds-describe-db-parameter-groups
```

The command returns a response like the following:

```
DBPARAMETERGROUP  default.mysql5.1     mysql5.1  Default parameter group
for MySQL5.1
DBPARAMETERGROUP  default.mysql5.5     mysql5.5  Default parameter group
for MySQL5.5
DBPARAMETERGROUP  mydbparametergroup   mysql5.5  My new parameter group
```

**API**

**To list all DB parameter groups for an AWS account**

- Call `DescribeDBParameterGroups` action. The following example returns a list of DB parameter groups.

```
```
Example

https://rds.amazonaws.com/
?Action=DescribeDBParameterGroups
&MaxRecords=100
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-22T19%3A31%3A42.262Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>

The command returns a response like the following:

```
<DescribeDBParameterGroupsResponse xmlns="http://rds.amazonaws.com/admin/2012-01-15/">
  <DescribeDBParameterGroupsResult>
    <DBParameterGroups>
      <DBParameterGroup>
        <Engine>mysql5.1</Engine>
        <Description>Default parameter group for MySQL5.1</Description>
        <DBParameterGroupName>default.mysql5.1</DBParameterGroupName>
      </DBParameterGroup>
      <DBParameterGroup>
        <Engine>mysql5.1</Engine>
        <Description>My new parameter group</Description>
        <DBParameterGroupName>mydbparametergroup</DBParameterGroupName>
      </DBParameterGroup>
    </DBParameterGroups>
  </DescribeDBParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>41731881-0b82-11df-9a9b-clbd5894571c</RequestId>
  </ResponseMetadata>
</DescribeDBParameterGroupsResponse>
```

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. Click **DB Parameter Groups** in the **Navigation** list on the left side of the window.

   The DB parameter groups appear in the **My DB Parameter Groups** list.

3. Select a DB parameter group from the **My DB Parameter Groups** list.

   The list of parameters defined for the DB parameter group appears in the information panel. You can scroll through the parameters using the arrow keys at the top right of the information panel.

**CLI**

**To view the parameter values for a DB parameter group**

- Use the `rds-describe-db-parameters` command. The following example lists the parameters and parameter values for a DB parameter group named `mydbparametergroup`.

```
PROMPT>rds-describe-db-parameters mydbparametergroup
```

The command returns a response like the following:

<table>
<thead>
<tr>
<th>DBPARAMETER</th>
<th>Parameter Name</th>
<th>Parameter Value</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Type</td>
<td>Apply Type</td>
<td>Is Modifiable</td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td>static</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>DBPARAMETER</td>
<td>allow-suspicious-udfs</td>
<td>engine-default</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>dynamic</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>DBPARAMETER</td>
<td>auto_increment_increment</td>
<td>engine-default</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>dynamic</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>DBPARAMETER</td>
<td>auto_increment_offset</td>
<td>engine-default</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>dynamic</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>DBPARAMETER</td>
<td>binlog_cache_size</td>
<td>32768</td>
<td>system</td>
</tr>
<tr>
<td>integer</td>
<td>dynamic</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>DBPARAMETER</td>
<td>socket</td>
<td>/tmp/mysql.sock</td>
<td>system</td>
</tr>
<tr>
<td>string</td>
<td>static</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

**API**

**To view the parameter values for a DB Parameter Group**

- Call `DescribeDBParameters` action. The following example lists the parameters and parameter values for a DB parameter group named `mydbparametergroup`.

```
```
The command returns a response like the following:

```xml
<DescribeDBParametersResponse xmlns="http://rds.amazonaws.com/admin/2012-01-15/">
  <DescribeDBParametersResult>
    <Marker>bWF4X3RtcF90YWJsZXM=</Marker>
    <Parameters>
      <Parameter>
        <DataType>boolean</DataType>
        <Source>engine-default</Source>
        <IsModifiable>false</IsModifiable>
        <Description>Controls whether user-defined functions that have only an xxx symbol for the main function can be loaded</Description>
        <ApplyType>static</ApplyType>
        <AllowedValues>0,1</AllowedValues>
        <ParameterName>allow-suspicious-udfs</ParameterName>
      </Parameter>
      <Parameter>
        <DataType>integer</DataType>
        <Source>engine-default</Source>
        <IsModifiable>true</IsModifiable>
        <Description>Intended for use with master-to-master replication, and can be used to control the operation of AUTO_INCREMENT columns</Description>
        <ApplyType>dynamic</ApplyType>
        <AllowedValues>1-65535</AllowedValues>
        <ParameterName>auto_increment_increment</ParameterName>
      </Parameter>
      <Parameter>
        <DataType>integer</DataType>
        <Source>engine-default</Source>
        <IsModifiable>true</IsModifiable>
        <Description>Determines the starting point for the AUTO_INCREMENT column value</Description>
        <ApplyType>dynamic</ApplyType>
        <AllowedValues>1-65535</AllowedValues>
        <ParameterName>auto_increment_offset</ParameterName>
      </Parameter>
    </Parameters>
  </DescribeDBParametersResult>
</DescribeDBParametersResponse>
```
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.

DB Parameter Formulas

A DB parameter formula is an expression that resolves to an integer value, and is enclosed in braces: {}. Formulas can be specified for either a DB parameter value or as an argument to a DB parameter function.

Syntax

\{FormulaVariable\}

\{FormulaVariable*Integer\}

\{FormulaVariable/Integer\}

DB Parameter Formula Variables

Formula variables return integers. 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.

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: *  
Divides the dividend by the divisor, returning an integer quotient. Decimals in the quotient 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.

**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.

**Caution**
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 9868951:

```
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 1MB or DBInstanceClass/256:

```
LEAST(DBInstanceClassMemory/256, 10485760)
```
Working with DB Security Groups

A DB security group controls network access to a DB instance that is not inside a VPC. 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.

If you are a new customer to Amazon RDS or if you are an existing customer who is using a new region, your DB instance is most likely in a default VPC. You cannot use a DB security group for a DB instance inside a VPC; you must create a VPC security group. For information on creating a VPC security group, see Security Groups for Your VPC. To determine if you have a default VPC, see step 2 in the following procedure.

Topics
- Creating a DB Security Group (p. 329)
- Listing Available DB Security Groups (p. 332)
- Viewing a DB security group (p. 333)
- Authorizing Network Access to a DB Security Group from an IP Range (p. 335)
- Authorizing Network Access to a DB Instance from an Amazon EC2 Instance (p. 337)
- Revoking Network Access to a DB Instance from an IP Range (p. 339)
- Related Topics (p. 341)

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. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Determine what platforms are supported for your AWS account in your current region.

   If Supported Platforms indicates EC2, VPC, your AWS account in the current region does not use a default VPC. You can continue following the steps below to create a DB security group that will enable access to your DB instance.
If Supported Platforms indicates VPC, your AWS account in the current region uses a default VPC. This means that you must create a VPC security group to enable access to a DB instance instead of a DB security group. For information on creating a VPC security group, see Security Groups for Your VPC.

<table>
<thead>
<tr>
<th>Resources</th>
<th>Create Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>You are using the following Amazon RDS resources in the US West (Oregon) region:</strong></td>
<td></td>
</tr>
<tr>
<td>DB Instances (6)</td>
<td>Reserved DB Purchases (0)</td>
</tr>
<tr>
<td>DB Snapshots (15)</td>
<td>Recent Events (8)</td>
</tr>
<tr>
<td>DB Parameter Groups (15)</td>
<td>Supported Platforms EC2, VPC</td>
</tr>
<tr>
<td>DB Security Groups (8)</td>
<td>Default Network none</td>
</tr>
</tbody>
</table>

Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud. You can click the button below to launch a Database (DB) Instance in...
3. Click **Security Groups** in the navigation pane on the left side of the window.
4. Click **Create DB Security Group**.

5. Type the name and description of the new DB security group in the **Name** and **Description** text boxes. Note that the security group name cannot contain spaces and cannot start with a number.

6. Click **Yes, Create**. The DB security group will be created. Note that a newly created DB security group does not 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. 335).

**CLI**

To create a DB security group

- Use the command `rds-create-db-security-group` with the following parameters:

  ```
  PROMPT>rds-create-db-security-group mydbsecuritygroup -d "My new security group"
  ```

Note that a newly created DB security group does not 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. 335).
API

To create a DB security group

- Call `CreateDBSecurityGroup` with the following parameters:
  - `DBSecurityGroupName = mydbsecuritygroup`
  - `Description = "My new security group"`

Example

```
https://rds.amazonaws.com/
?Action=CreateDBSecurityGroup
&DBParameterGroupName=mydbsecuritygroup
&Description=My%20new%20db%20security%20group
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2012-01-22T23%3A06%3A23.624Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Note that a newly created DB security group does not 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. 335).

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. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click 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 command `rds-describe-db-security-groups` to list all available DB security groups for your AWS account.

```
PROMPT>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. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click **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.
4. The detailed information for the DB security group is displayed.
To view properties of a specific DB security group

- Use the `rds-describe-db-security-groups` to view a DB security group. Specify the DB security group you want to view.

```
PROMPT>rds-describe-db-security-groups <mydbsecuritygroup>
```

**API**

To view properties of a specific DB security group

- Call `DescribeDBSecurityGroups` with the following parameters:

  - `DBSecurityGroupName` = `<mydbsecuritygroup>`

**Example**

```plaintext
https://rds.amazonaws.com/?Action=DescribeDBSecurityGroups
&DBParameterGroupName=mydbsecuritygroup
&Version=2009-10-16
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-16T22:23:3A07.107Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```
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.

Caution
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

Configure a DB security group with an ingress rule for a CIDR IP range

1. Sign in to the AWS Management Console and 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 Security Group Details section, select CIDR/IP from the Connection Type drop-down list, type the CIDR range for the ingress rule you would like to add to this DB security group into the CIDR text box, and click Add.

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, this could be the CIDR IP you could use.
5. The status of the ingress rule will be **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 will change to **authorized**.

### CLI

**To configure a DB security group with an ingress rule for a CIDR IP range**

- Use the command `rds-authorize-db-security-group-ingress` to modify a DB security group.

```
PROMPT>rds-authorize-db-security-group-ingress mydbsecuritygroup --cidr-ip 192.168.1.10/27
```

The command should produce output similar to the following:

```
SECGROUP  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 `AuthorizeDBSecurityGroupIngress` with the following parameters:

  - `DBSecurityGroupName` = `mydbsecuritygroup`
  - `CIDRIP` = `192.168.1.10/27`
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 will access.

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 would 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 region that refers to IP address in another 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 does not 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. Sign in to the AWS Management Console and 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 grant access.
4. In the **Security Group Details** section, select *EC2 Security Group* from the **Connection Type** drop-down list, and then select the Amazon EC2 security group you want to use. Then click **Add**.

5. The status of the ingress rule will be **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 will change to **authorized**.

**CLI**

To grant access to an Amazon EC2 security group

- Use the command `rds-authorize-db-security-group-ingress` to grant access to an Amazon EC2 security group

```
PROMPT>rds-authorize-db-security-group-ingress default --ec2-security-group-name myec2group --ec2-security-group-owner-id 987654321021
```

The command should produce output similar to the following:
API

To authorize network access to an Amazon EC2 security group

- Call `AuthorizeDBSecurityGroupIngress` with the following parameters:
  - `EC2SecurityGroupName` = myec2group
  - `EC2SecurityGroupOwnerId` = 987654321021

Example

```plaintext
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 on a DB Security Group.

AWS Management Console

To revoke an ingress rule for a CIDR IP range on a DB Security Group.

1. Sign in to the AWS Management Console and open the Amazon RDS console at
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 that has the ingress rule you want to revoke.
4. In the **Security Group Details** section, click **Remove** next to the ingress rule you would like to revoke.

5. The status of the ingress rule will be **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 will be removed from the DB security group.

**CLI**

To revoke an ingress rule for a CIDR IP range on a DB security group

- Use the command `rds-revoke-db-security-group-ingress` to modify a DB security group.

```bash
PROMPT>rds-revoke-db-security-group-ingress <mydbsecuritygroup> --cidr-ip 192.168.1.1/27
```

The command should produce output similar to the following:

```
SECGROUP 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 `RevokeDBSecurityGroupIngress` 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>
```

Related Topics

- Amazon RDS Security Groups (p. 74)
Working with Reserved DB Instances

Topics
- Things You Should Know About Reserved DB Instances (p. 342)
- Describing Available Reserved DB Instance Offerings (p. 343)
- Purchasing a Reserved DB Instance (p. 347)
- Describing Reserved DB Instances (p. 349)
- Related Topics (p. 352)

Things You Should Know About Reserved DB Instances

Reserved DB instances let you make a one-time up-front payment for a DB instance and reserve the DB instance for a one- or three-year term at significantly lower rates. You can use the command line tools, the API, or the AWS Management Console to list and purchase available Reserved DB instance offerings.

Reserved Instances are available in three varieties—Heavy Utilization, Medium Utilization, and Light Utilization—that enable you to optimize your Amazon RDS costs based on your expected utilization. For more information about reserved instance types, see Amazon RDS Reserved Instances.

Heavy Utilization Reserved DB instances enable workloads that have a consistent baseline of capacity or run steady-state workloads. Heavy Utilization Reserved DB instances require the highest up-front commitment, but if you plan to run more than 79 percent of the Reserved DB instance term you can earn the largest savings (up to 58 percent off of the On-Demand price). Unlike the other Reserved DB instances, with Heavy Utilization Reserved DB instances, you pay a one-time fee, followed by a lower hourly fee for the duration of the term regardless of whether or not your DB instance is running.

Medium Utilization Reserved DB instances are the best option if you plan to leverage your Reserved DB instances a substantial amount of the time, but want either a lower one-time fee or the flexibility to stop paying for your DB instance when you shut it off. This offering type is equivalent to the Reserved DB instance offering available before the 2011-12-19 API version of Amazon RDS. Medium Utilization Reserved DB instances are a more cost-effective option when you plan to run more than 40 percent of the Reserved Instance term. This option can save you up to 49 percent off of the On-Demand price. With Medium Utilization Reserved DB instances, you pay a slightly higher one-time fee than with Light Utilization Reserved DB instances, and you receive lower hourly usage rates when you run a DB instance.

Light Utilization Reserved DB instances are ideal for periodic workloads that run only a couple of hours a day or a few days per week. Using Light Utilization Reserved DB instances, you pay a one-time fee followed by a discounted hourly usage fee when your DB instance is running. You can start saving when your instance is running more than 17 percent of the Reserved DB instance term, and you can save up to 33 percent off of the On-Demand rates over the entire term of your Reserved DB instance.

Remember that discounted usage fees for Reserved Instance purchases are tied to instance type and Availability Zone. If you shut down a running DB instance on which you have been getting a discounted rate as a result of a Reserved DB instance purchase, and the term of the Reserved DB instance has not yet expired, you will continue to get the discounted rate if you launch another DB instance with the same specifications during the term. Your up-front payment for a reserved instance will reserve the resources for your use; since these resources are reserved for you, you will be billed for the resources regardless of if you use them.

The following table summarizes the differences between the Reserved DB instances offering types.
Reserved Instance Offerings

<table>
<thead>
<tr>
<th>Offering</th>
<th>Upfront Cost</th>
<th>Usage Fee</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heavy Utilization</td>
<td>Highest</td>
<td>Lowest hourly fee. Applied to the whole term whether or not you're using the Reserved DB instance.</td>
<td>Lowest overall cost if you plan to utilize your Reserved DB instances more than 79 percent of a 3-year term.</td>
</tr>
<tr>
<td>Medium Utilization</td>
<td>Average</td>
<td>Hourly usage fee charged for each hour you use the DB instance.</td>
<td>Suitable for elastic workloads or when you expect moderate usage, more than 40 percent of a 3-year term.</td>
</tr>
<tr>
<td>Light Utilization</td>
<td>Lowest</td>
<td>Hourly usage fee charged. Highest fees of all the offering types, but they apply only when you're using the Reserved DB instance.</td>
<td>Highest overall cost if you plan to run all of the time, however lowest overall cost if you anticipate you will use your Reserved DB instances infrequently, more than about 15 percent of a 3-year term.</td>
</tr>
</tbody>
</table>

**Describing Available Reserved DB Instance Offerings**

Before you purchase a reserved DB instance, you can get information about available reserved DB instance offerings.

This example shows how to get pricing and information about available reserved DB Instance offerings.

**AWS Management Console**

**To get pricing and information about available reserved DB Instances**

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, click the Reserved DB Purchases link.
3. Click Purchase Reserved DB Instance.
4. Use the **Product Description** drop-down list box to select the DB engine and licensing type.
5. Select the DB instance class from the **DB Instance Class** drop-down list box.
6. Select whether or not you want a Multi-AZ deployment from the **Multi-AZ Deployment** drop-down list box.
7. Select the length of time you want the DB instance reserved from the **Term** drop-down list box.
8. Select the offering type from the **Offering Type** drop-down list box.
9. Information is displayed after you select the offering type. When you have selected the reserved instance you want, click **Continue**.
10. The summation screen shows you the instance information and cost. Click the X in the upper-right corner of the page to avoid incurring any charges.

CLI

To get information about reserved DB Instances
• Enter the following command at a command prompt:

```
PROMPT>rds-describe-reserved-db-instances-offerings  --headers
```

This call returns output similar to the following:

<table>
<thead>
<tr>
<th>OFFERING</th>
<th>OfferingId</th>
<th>Class</th>
<th>Multi-AZ</th>
<th>Duration</th>
<th>Fixed Price</th>
<th>Usage Price</th>
<th>Description</th>
<th>Offering Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>438012d3-4052-4cc7-b2e3-8d3372e0e706</td>
<td>db.m1.large</td>
<td>y</td>
<td>1y</td>
<td>1820.00 USD</td>
<td>0.368 USD</td>
<td>mysql</td>
<td>Medium Utilization</td>
</tr>
<tr>
<td></td>
<td>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</td>
<td>db.m1.small</td>
<td>n</td>
<td>1y</td>
<td>227.50 USD</td>
<td>0.046 USD</td>
<td>mysql</td>
<td>Medium Utilization</td>
</tr>
<tr>
<td></td>
<td>123456cd-ab1c-47a0-bfa6-12345667232f</td>
<td>db.m1.small</td>
<td>n</td>
<td>1y</td>
<td>162.00 USD</td>
<td>0.00 USD</td>
<td>mysql</td>
<td>Heavy Utilization</td>
</tr>
</tbody>
</table>

Recurring Charges:

<table>
<thead>
<tr>
<th>Amount</th>
<th>Currency</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.123</td>
<td>USD</td>
<td>Hourly</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OFFERING</th>
<th>OfferingId</th>
<th>Class</th>
<th>Multi-AZ</th>
<th>Duration</th>
<th>Fixed Price</th>
<th>Usage Price</th>
<th>Description</th>
<th>Offering Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>123456cd-ab1c-37a0-bfa6-12345667232d</td>
<td>db.m1.large</td>
<td>y</td>
<td>1y</td>
<td>700.00 USD</td>
<td>0.00 USD</td>
<td>mysql</td>
<td>Heavy Utilization</td>
</tr>
<tr>
<td></td>
<td>123456cd-ab1c-17d0-bfa6-12345667234e</td>
<td>db.m1.xlarge</td>
<td>n</td>
<td>1y</td>
<td>4242.00 USD</td>
<td>2.42 USD</td>
<td>mysql</td>
<td>Light Utilization</td>
</tr>
</tbody>
</table>

Recurring Charges:

<table>
<thead>
<tr>
<th>Amount</th>
<th>Currency</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.25</td>
<td>USD</td>
<td>Hourly</td>
</tr>
</tbody>
</table>

API

To get information about available reserved DB Instances

• Call DescribeReservedDBInstancesOfferings.

Example

```
https://rds.amazonaws.com/
?Action=DescribeReservedDBInstancesOfferings
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2012-01-18T18%3A31%3A36.118Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

This call returns output similar to the following:

```
<DescribeReservedDBInstancesOfferingsResponse xmlns="http://rds.amazonaws.com/doc/2012-01-15/">
<DescribeReservedDBInstancesOfferingsResult>
  <ReservedDBInstancesOfferings>
    <ReservedDBInstancesOffering>
      <Duration>31536000</Duration>
    </ReservedDBInstancesOffering>
  </ReservedDBInstancesOfferings>
</DescribeReservedDBInstancesOfferingsResult>
</DescribeReservedDBInstancesOfferingsResponse>
```
<OfferingType>Medium Utilization</OfferingType>
<CurrencyCode>USD</CurrencyCode>
<RecurringCharges/>
<FixedPrice>1820.0</FixedPrice>
<ProductDescription>mysql</ProductDescription>
<UsagePrice>0.368</UsagePrice>
<MultiAZ>true</MultiAZ>
<ReservedDBInstancesOfferingId>438012d3-4052-4cc7-b2e3-8d3372e0706</ReservedDBInstancesOfferingId>
=DBInstanceClass>db.m1.large</DBInstanceClass>
</ReservedDBInstancesOffering>
<ReservedDBInstancesOffering>
<Duration>31536000</Duration>
<OfferingType>Medium Utilization</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>
<ReservedDBInstancesOffering>
<Duration>31536000</Duration>
<OfferingType>Heavy Utilization</OfferingType>
<CurrencyCode>USD</CurrencyCode>
<RecurringCharges>
<RecurringCharge>
<RecurringChargeFrequency>Hourly</RecurringChargeFrequency>
<RecurringChargeAmount>0.123</RecurringChargeAmount>
</RecurringCharge>
</RecurringCharges>
<FixedPrice>162.0</FixedPrice>
<ProductDescription>mysql</ProductDescription>
<UsagePrice>0.0</UsagePrice>
<MultiAZ>false</MultiAZ>
<ReservedDBInstancesOfferingId>TEMP-DELETE-1</ReservedDBInstancesOfferingId>
=DBInstanceClass>db.m1.small</DBInstanceClass>
</ReservedDBInstancesOffering>
<ReservedDBInstancesOffering>
<Duration>31536000</Duration>
<OfferingType>Heavy Utilization</OfferingType>
<CurrencyCode>USD</CurrencyCode>
<RecurringCharges>
<RecurringCharge>
<RecurringChargeFrequency>Hourly</RecurringChargeFrequency>
<RecurringChargeAmount>1.25</RecurringChargeAmount>
</RecurringCharge>
</RecurringCharges>
<FixedPrice>700.0</FixedPrice>
<ProductDescription>mysql</ProductDescription>
<UsagePrice>0.0</UsagePrice>
<MultiAZ>true</MultiAZ>
<ReservedDBInstancesOfferingId>TEMP-DELETE-2</ReservedDBInstancesOfferingId>
Purchasing a Reserved DB Instance

This example shows how to purchase a Reserved DB Instance Offering.

Important
Following the examples in this section will incur charges on your AWS account.

AWS Management Console

This example shows purchasing a specific Reserved DB Instance Offering, 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f, with a Reserved DB Instance ID of myreservationID.

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, click the Reserved DB Instances link.
3. Click the Purchase Reserved DB Instance button.
4. Select the DB Engine type from the Product Description drop-down list box.
5. Select the DB Instance Class from the DB Instance Class drop-down list box.
6. Select whether or not you want a Multi-AZ deployment from the Multi-AZ Deployment drop-down list box.
7. Select length of time you want the DB Instance reserved from the Term drop-down list box.
8. Select the offering type from the Offering Type drop-down list box.
9. You can optionally enter a Reserved DB Instance ID in the Reserved DB ID text box.
10. Click the Continue button.
11. Click the Yes, Purchase button to proceed and purchase the Reserved DB Instance.
This example shows purchasing a specific Reserved DB Instance Offering, 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f, with a Reserved DB Instance ID of myreservationID.

## CLI

To purchase a reserved DB Instance

- Enter the following command at a command prompt:

  ```bash
  PROMPT>rds-purchase-reserved-db-instances-offering 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f -i myreservationID
  ```

  The command returns output similar to the following:

<table>
<thead>
<tr>
<th>ReservationId</th>
<th>Class</th>
<th>Multi-AZ</th>
<th>Start Time</th>
<th>Duration</th>
<th>Fixed Price</th>
<th>Usage Price</th>
<th>Count</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>myreservationid</td>
<td>db.m1.small</td>
<td>y</td>
<td>2011-12-19T00:30:23.247Z</td>
<td>1y</td>
<td>455.00 USD</td>
<td>0.092 USD</td>
<td>1</td>
<td>payment-pending</td>
<td>mysql Medium Utilization</td>
</tr>
</tbody>
</table>

## API

This example shows purchasing a specific reserved DB Instance offering, 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f, with a reserved DB Instance ID of myreservationID.

To purchase a reserved DB Instance

- Call `PurchaseReservedDBInstancesOffering` with the following parameters:

  ```json
  {  
    "ReservedDBInstancesOfferingId": "649fd0c8-cf6d-47a0-bfa6-060f8e75e95f",  
    "ReservedDBInstanceCount": 1,  
    "ReservedDBInstanceID": "myreservationID"  
  }
  ```

  The API call returns output similar to the following:

<table>
<thead>
<tr>
<th>ReservationId</th>
<th>Class</th>
<th>Multi-AZ</th>
<th>Start Time</th>
<th>Duration</th>
<th>Fixed Price</th>
<th>Usage Price</th>
<th>Count</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>myreservationid</td>
<td>db.m1.small</td>
<td>y</td>
<td>2011-12-19T00:30:23.247Z</td>
<td>1y</td>
<td>455.00 USD</td>
<td>0.092 USD</td>
<td>1</td>
<td>payment-pending</td>
<td>mysql Medium Utilization</td>
</tr>
</tbody>
</table>
Describing Reserved DB Instances

You can get information about Reserved DB instances for your AWS account.

AWS Management Console

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, click the Reserved DB Instances link.

The Reserved DB instances for your account appear in the My Reserved DB Instances list. You can click any of the Reserved DB instances in the list to see detailed information about the Reserved DB instance in the detail pane at the bottom of the console.

**CLI**

**To get information about Reserved DB Instances for your AWS account**

- Enter the following command at a command prompt:

```
PROMPT>rds-describe-reserved-db-instances --headers
```

This command should return output similar to the following:

<table>
<thead>
<tr>
<th>RESERVATION</th>
<th>ReservationId</th>
<th>Class</th>
<th>Multi-AZ</th>
<th>Start Time</th>
<th>Duration</th>
<th>Fixed Price</th>
<th>Usage Price</th>
<th>Count</th>
<th>State</th>
<th>Description</th>
<th>Offering Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ki-real-ri-test5</td>
<td>db.m1.small</td>
<td>y</td>
<td>1y</td>
<td>2011-12-09T23:37:44.720Z</td>
<td>455.00 USD</td>
<td>0.092 USD</td>
<td>1</td>
<td>retired</td>
<td>mysql</td>
<td>Medium Utilization</td>
<td></td>
</tr>
</tbody>
</table>

**API**

**To get information about Reserved DB Instances for your AWS account**

- Call DescribeReservedDBInstances.
Example

https://rds.amazonaws.com/
?Action=DescribeReservedDBInstances
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2012-01-15T17%3A48%3A21.746Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>

The API returns output similar to the following:

```
<DescribeReservedDBInstancesResponse xmlns="http://rds.amazonaws.com/doc/2012-01-15/">
<ReservedDBInstances>
    <ReservedDBInstance>
        <OfferingType>Medium Utilization</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <ProductDescription>mysql</ProductDescription>
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</ReservedDBInstancesOfferingId>
        <MultiAZ>false</MultiAZ>
        <State>payment-failed</State>
        <DBInstanceId>myreservationid</DBInstanceId>
        <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>Medium Utilization</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <ProductDescription>mysql</ProductDescription>
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</ReservedDBInstancesOfferingId>
        <MultiAZ>false</MultiAZ>
        <State>payment-failed</State>
        <DBInstanceId>myreservationid2</DBInstanceId>
        <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>
</DescribeReservedDBInstancesResponse>
```

Amazon Relational Database Service User Guide
Describing Reserved DB Instances
Related Topics

- How You Are Charged for Amazon RDS (p. 4)
Amazon RDS supported two VPC platforms: **EC2-VPC** and **EC2-Classic**. The EC2-VPC platform has a default VPC where all new DB instances are created unless you specify otherwise. The EC2-Classic platform does not have a default VPC, but as with either platform, you can create your own VPC and specify that a DB instance be located in that VPC. If you are a new customer to Amazon RDS or if you are using a region you have not previously used, you are most likely working with the EC2-VPC platform.

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 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 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.

Amazon Virtual Private Cloud (VPC) is an AWS service and this section only covers VPC topics that directly affect access to a DB instance. You should read the Amazon VPC documentation to familiarize yourself with all the features of a VPC. For more information about Amazon VPC, see the table below that provides several links to the Amazon VPC documentation.

The following topics in this section apply to working with DB instances inside a VPC:

**Topics**

- Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 354)
- Working with a DB Instance in a VPC (p. 355)
- Working with DB Subnet Groups (p. 356)
- Hiding a DB instance in a VPC from the Internet (p. 356)
- Creating a DB Instance in a VPC (p. 357)

Amazon VPC has its own set of documentation that describes how to create and use a VPC. The following table shows links to the Amazon VPC guides.

<table>
<thead>
<tr>
<th>Description</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to get started using Amazon VPC</td>
<td>Amazon Virtual Private Cloud Getting Started Guide</td>
</tr>
<tr>
<td>How to use Amazon VPC through the AWS Management Console</td>
<td>Amazon Virtual Private Cloud User Guide</td>
</tr>
<tr>
<td>Complete descriptions of all the Amazon VPC commands</td>
<td>Amazon Elastic Compute Cloud Command Line Reference</td>
</tr>
</tbody>
</table>

(the Amazon VPC commands are part of the Amazon EC2 reference)
Determining Whether You are Using the EC2-VPC or EC2-Classic Platform

You can create a DB instance on two different platforms: 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 EC2-Classic platform is the original platform used by Amazon RDS; if you want to use a VPC with this platform, you must create it using Amazon VPC. The EC2-VPC platform provides a default VPC (virtual private cloud) where a new DB instance is created, and you must use either an EC2 or VPC security group to provide access to the DB instance. If you are a new Amazon RDS customer or you are creating DB instances in a region you have not used before, you are likely on the EC2-VPC platform.

The following illustration shows how you access a DB instance when it is on the EC2-VPC platform and in a default VPC:

You can tell which platform your AWS account in a given region is using by looking at the RDS console home page.

If Supported Platforms indicates VPC, 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 an EC2 or VPC security group.
If **Supported Platforms** indicates *EC2, VPC*, 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. Note that you can create a VPC on the *EC2-Classic* platform, but one is not created for you by default as it is on the *EC2-VPC* platform.

**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 one subnet in at least two of the 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 (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.

• If you want to launch a DB instance in a VPC, the instance tenancy attribute of the VPC must be set to default. For information about instance tenancy in a VPC, see Using EC2 Dedicated Instances in the Amazon Virtual Private Cloud User Guide. All default VPCs have the instance tenancy attribute set to default.

The most common scenarios for using a VPC are documented at Scenarios for Using Amazon VPC. Each of these scenarios have a link to a detailed explanation of the scenario. At the end of the section is a section called Implementing the Scenario that gives you instructions on how to create a VPC for that scenario. For detailed instructions on creating a VPC, see the Amazon Virtual Private Cloud User Guide.

### 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 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 could create a VPC that has a public subnet and a private subnet. Amazon EC2 instances that function as web servers could be deployed in the public subnet, and the Amazon RDS DB instances would be deployed in the private subnet. In such a deployment, only the web servers have access to the DB instances.

When you launch a DB instance inside any VPC (including a default VPC), you can designate whether the DB instance you create has a DNS that resolves to a public IP address by using the PubliclyAccessible parameter. This parameter lets you designate whether there is public access to the DB instance, even if the VPC has a public IP address. 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. If you want a DB instance in a VPC to be publicly accessible, you must also enable the VPC attributes DNS hostnames and DNS resolution. For more information about creating a VPC, see Working with a DB Instance in a VPC (p. 355).

The following illustration shows the Publicly Accessible parameter in the Launch DB Instance Wizard.
Creating a DB Instance in a VPC

DB instances in a VPC require slightly more setup than a DB instance not in a VPC, but the added flexibility is often worth it. If your account has a default VPC, you can begin with step 3 in this tutorial since the VPC and DB subnet group have been created for you. If your AWS account does not have a default VPC or if you do not have a default VPC in a particular region, you can create a VPC using the Amazon VPC service and launch a DB instance in the VPC. If you don’t know if you have a default VPC, see the topic Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 354)

You should have a clear understanding of the type of scenario you intend for your VPC. The most common scenarios for using a VPC are documented at Scenarios for Using Amazon VPC. Each of these scenarios have a link to a detailed explanation of the scenario and at the end of the section is a link called Implementing the Scenario which gives you instructions on how to create a VPC for that scenario. Follow these instructions to create a VPC.

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.

Once you have a VPC, there are four steps to create a DB instance in a VPC.

- Step 1: Adding Subnets to a VPC (p. 357)
- Step 2: Creating a DB Subnet Group (p. 358)
- Step 3: Creating a VPC Security Group (p. 359)
- Step 4: Creating a DB Instance in a VPC (p. 359)

Step 1: Adding Subnets to a VPC

Once you have created a VPC, you need to create a subnet in the VPC in at least two of the Availability Zones of the region. You will use these subnets when you create a DB subnet group.

If you have a default VPC, a subnet is automatically created for you in each Availability Zone of the region.
Step 2: Creating 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.

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 example, you create a DB subnet group and add the subnets you created for your VPC.

AWS Management Console

Create a DB subnet group

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click Subnet Groups in the navigation pane on the left side of the console window.
3. Click the Create DB Subnet Group button.
4. Type the name of your DB subnet group in the Name text box.
5. Type a description for your DB subnet group in the Description text box.
6. Select the VPC that you created from the VPC ID drop-down list box.
7. Type a description for your DB subnet group in the Description text box.
8. In the Add Subnet(s) to this Subnet Group section, click the add all the subnets link.
9. Click Yes, Create.
10. Click Close.

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 3: Creating a VPC Security Group

Before you create your DB instance, you must create a VPC security group to associate with your DB instance. You can find out how to create a VPC security group in the Amazon VPC documentation.

Step 4: Creating a DB Instance in a 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.

AWS Management Console

Create an Amazon RDS DB instance in a VPC

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click Instances in the navigation pane on the left side of the console window.
3. Click Launch DB Instance.
4. Click the Select button for the DB engine that you want to use.
5. On the DB Instance Details page, select your DB instance configuration. Click Next Step.
6. On the Additional Configuration page, select the VPC that you want to use from the Choose a VPC drop-down list box.
7. From the Subnet Group drop-down list box, select the DB subnet group that you created.
8. From the VPC Security Group drop-down list box, select the VPC security group that you created. Click Next Step.
9. In the Management Options page, choose the options you want, and then click the Next Step.
10. In the Review page, click Launch DB Instance.
11. On the confirmation page, click Close.
12. You can see your DB instance in the DB instances list of the RDS console. Click on it to view the details pane. Once your DB instance is running, make note of the endpoint shown in the details pane; you will use this to connect to your DB instance.
Working with Provisioned IOPS Storage

Topics
- Things You Should Know About Provisioned IOPS Storage (p. 361)
- Modifying a DB Instance to Use Provisioned IOPS Storage (p. 365)
- Modifying the IOPS and Storage Settings for a DB Instance That Uses Provisioned IOPS Storage (p. 367)
- Creating a DB Instance that Uses Provisioned IOPS Storage (p. 369)
- Creating a Read Replica that Uses Provisioned IOPS Storage (p. 371)

Things You Should Know About Provisioned IOPS Storage

For any 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 option that delivers fast, predictable, and consistent throughput performance. When you create a DB instance, you specify an IOPS rate and storage space allocation. Amazon RDS provisions that IOPS rate and storage for the lifetime of the DB instance or until you change it. Provisioned IOPS storage is optimized for I/O intensive, online transaction processing (OLTP) workloads that have consistent performance requirements.

Note
You cannot decrease standard storage or Provisioned IOPS storage allocated for a DB instance.

You can create a DB instance that uses Provisioned IOPS storage by using the AWS Management Console, the Amazon RDS API, or the Command Line Interface (CLI). You specify the IOPS rate and the amount of storage that you require. You can provision a MySQL, PostgreSQL, or Oracle DB instance with up to 30,000 IOPS and 3 TB of allocated storage. You can provision a SQL Server DB instance with up to 10,000 IOPS and 1 TB of allocated storage.

Note
Your actual realized IOPS may vary from the value that you specify depending on your database workload, DB instance size, and the page size and channel bandwidth that are available for your DB engine. For more information, see Factors That Affect Realized IOPS Rates (p. 362).

The ratio of the requested IOPS rate to the amount of storage allocated is important. The ratio of IOPS to storage, in GB, for your DB instances should be between 3:1 and 10:1 for MySQL, PostgreSQL, and Oracle DB instances. For SQL Server DB instances, the ratio should be 10:1. For example, you could start by provisioning an Oracle DB instance with 1000 IOPS and 200 GB storage (a ratio of 5:1). You could then scale up to 2000 IOPS with 200 GB of storage (a ratio of 10:1), 3000 IOPS with 300 GB of storage, and up to the maximum for an Oracle DB instance of 30,000 IOPS with 3 TB (3000 GB) of storage.

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

<table>
<thead>
<tr>
<th>Database Engine</th>
<th>Range of Provisioned IOPS</th>
<th>Range of Storage (GB)</th>
<th>Range of IOPS to Storage (GB) Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>1000 - 30,000 IOPS</td>
<td>100 GB - 3 TB</td>
<td>3:1 - 10:1</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>1000 - 30,000 IOPS</td>
<td>100 GB - 3 TB</td>
<td>3:1 - 10:1</td>
</tr>
</tbody>
</table>

API Version 2013-09-09

Amazon Relational Database Service User Guide

Working with Provisioned IOPS Storage
You can modify an existing Oracle or MySQL DB instance to use Provisioned IOPS storage, and you can modify Provisioned IOPS storage settings.

## Factors That Affect Realized IOPS Rates

Your actual realized IOPS rate may vary from the amount that you provision depending on page size and network bandwidth, which are determined in part by your DB engine. It is also affected by DB instance size and database workload.

### Page Size and Channel Bandwidth

The theoretical maximum IOPS rate is also a function of database I/O page size and available channel bandwidth. MySQL uses a page size of 16 KB, while Oracle, PostgreSQL (default), and SQL Server use 8 KB. On a DB instance with a full duplex I/O channel bandwidth of 1000 megabits per second (Mbps), the maximum IOPS for page I/O is about 6,250 IOPS in each direction (input/output channel) for 16 KB I/O and 12,500 IOPS in each direction for 8 KB I/O. A workload consisting of 50% reads and 50% writes could reach 12,500 IOPS with 16 KB I/O or 25,000 IOPS with 8 KB I/O.

If traffic on one of the channels reaches capacity, available IOPS on the other channel cannot be reallocated. As a result, the attainable IOPS rate will be less than the provisioned IOPS rate.

Each page read or write constitutes one I/O operation. Database operations that read or write more than a single page will use multiple I/O operations for each database operation. IO requests larger than 16 KB are treated as more than one IO for the purposes of PIOPS capacity consumption. A 20 KB IO request will consume 1.25 IOs, a 24 KB request will consume 1.5 IOs, a 32 KB request will consume 2 IOs, and so on. The IO request is not split into separate IOs; all IO requests are presented to the storage device unchanged. For example, if the database submits a 128 KB IO request, it goes to the storage device as a single 128 KB IO request, but it will consume the same amount of PIOPS capacity as eight 16 KB IO requests.

The following table shows the page size and the theoretical maximum IOPS rate for each DB engine. IOPS rates are based on the m2.4xlarge instance class (for Oracle and SQL Server) or the cr1.8xlarge instance class (for MySQL and PostgreSQL) with full duplex and a workload that is perfectly balanced between reads and writes. The SQL Server limit of 10,000 is due to the current storage limit of 1 TB and the current maximum IOPS to storage ratio of 10:1.

<table>
<thead>
<tr>
<th>DB Engine</th>
<th>Page Size</th>
<th>Maximum IOPS Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>16 KB</td>
<td>20,000</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>8 KB</td>
<td>25,000</td>
</tr>
<tr>
<td>Oracle</td>
<td>8 KB</td>
<td>25,000</td>
</tr>
<tr>
<td>SQL Server</td>
<td>8 KB</td>
<td>10,000</td>
</tr>
</tbody>
</table>
Note
If you provision an IOPS rate that is higher than the maximum or that is higher than your realized
IOPS rate, you may still benefit from reduced latency and improvements in overall throughput.

DB Instance Class

If you are using Provisioned IOPS storage, we recommend that you use the db.m3.xlarge, db.m3.2xlarge,
db.m1.large, db.m1.xlarge, db.m2.2xlarge, or db.m2.4xlarge DB instance classes. These instance classes
are optimized for Provisioned IOPS storage; other instance classes are not. You can also effectively use
the high-memory-cluster instance class db.cr1.8xlarge.

<table>
<thead>
<tr>
<th>DB Instance Classes Optimized for Provisioned IOPS</th>
<th>Available Network Bandwidth</th>
<th>Maximum 16k IOPS Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>db.m3.xlarge</td>
<td>500 Mbps</td>
<td>4000</td>
</tr>
<tr>
<td>db.m1.large</td>
<td>500 Mbps</td>
<td>4000</td>
</tr>
<tr>
<td>db.m2.2xlarge</td>
<td>500 Mbps</td>
<td>4000</td>
</tr>
<tr>
<td>db.m3.2xlarge</td>
<td>1000 Mbps</td>
<td>8000</td>
</tr>
<tr>
<td>db.m1.xlarge</td>
<td>1000 Mbps</td>
<td>8000</td>
</tr>
<tr>
<td>db.m2.4xlarge</td>
<td>1000 Mbps</td>
<td>8000</td>
</tr>
</tbody>
</table>

Database Workload

System activities such as automated backups, DB snapshots, and scale storage operations may consume
some I/O, which will reduce the overall capacity available for normal database operations. If your database
design results in concurrency issues, locking, or other forms of database contention, you may not be able
to directly use all the bandwidth that you provision.

If you provision IOPS capacity to meet your peak workload demand, during the non-peak periods, your
application will probably consume fewer IOPS on average than provisioned.

To help you verify that you are making the best use of your Provisioned IOPS storage, we have added
a new CloudWatch Metric called Disk Queue Depth. If your application is maintaining an average queue
depth of approximately 5 outstanding I/O operations per 1000 IOPS that you provision, you can assume
that you are consuming the capacity that you provisioned. For example, if you provisioned 10,000 IOPS,
you should have a minimum of 50 outstanding I/O operations in order to use the capacity you provisioned.

Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes

For production OLTP use cases, we recommend that you use Multi-AZ deployments for enhanced fault
tolerance and Provisioned IOPS storage for fast and predictable performance. In addition to Multi-AZ
deployments, Provisioned IOPS storage complements the following features:

• Amazon VPC for network isolation and enhanced security.
• Read Replicas – The type of storage on a read replica is independent of that on the master DB instance.
  For example, if the master DB instance uses standard storage, you can add read replicas that use
  Provisioned IOPS storage and vice versa. If you use standard storage–based read replicas with a
  master DB instance that uses Provisioned IOPS storage, the performance of your read replicas may
differ considerably from that of a configuration in which both the master DB instance and the read
  replicas are using Provisioned IOPS storage.
• DB Snapshots – If you are using a DB instance that uses Provisioned IOPS storage, you can use a DB snapshot to restore an identically configured DB instance, regardless of whether the target DB instance uses standard storage or Provisioned IOPS storage. If your DB instance uses standard storage, you can use a DB snapshot to restore only a DB instance that uses standard storage.

• You can use Provisioned IOPS storage with any DB instance class that runs the DB engine you want; however, smaller DB instance classes will not consistently make the best use Provisioned IOPS storage. We recommend that you use Provisioned IOPS Optimized storage.

**Provisioned IOPS Storage Costs**

Because Provisioned IOPS storage reserves resources for your use, you are charged for the resources whether or not you use them in a given month. When you use Provisioned IOPS storage, you are not charged the monthly Amazon RDS I/O charge. If you prefer to pay only for I/O that you consume, a DB instance that uses standard storage may be a better choice. For Amazon RDS pricing information, see the Amazon RDS product page.

**Getting the Most out of Amazon RDS Provisioned IOPS**

Using Provisioned IOPS storage increases the number of IO requests the system is capable of processing concurrently. Increased concurrency allows for decreased latency since IO requests spend less time in a queue. Decreased latency allows for faster database commits, which improves response time and allows for higher database throughput.

For example, consider a heavily loaded OLTP database provisioned for 10,000 Provisioned IOPS that runs consistently at the channel limit of 105 Mbps throughput for reads. The workload isn’t perfectly balanced, so there is some unused write channel bandwidth. The instance would consume less than 10,000 IOPS and but would still benefit from increasing capacity to 20,000 Provisioned IOPS.

Increasing Provisioned IOPS capacity from 10,000 to 20,000 doubles the system’s capacity for concurrent IO. Increased concurrency means decreased latency, which allows transactions to complete faster, so the database transaction rate increases. Read and write latency would improve by different amounts and the system would settle into a new equilibrium based on whichever resource becomes constrained first.

It is possible for Provisioned IOPS consumption to actually decrease under these conditions even though the database transaction rate can be much higher. For example, you could see write requests decline accompanied by an increase in write throughput. That’s a good indicator that your database is making better use of group commit. More write throughput and the same write IOPS means log writes have become larger but are still less than 16 KB. More write throughput and fewer write IO means log writes have become larger and are averaging larger than 16 KB since those IO requests consume more than one IO of Provisioned IOPS capacity.

**Provisioned IOPS Storage Support in the CLI and Amazon RDS API**

The Amazon RDS CLI supports Provisioned IOPS storage in the following commands:

- `rds-create-db-snapshot` – The output shows the IOPS value.
- `rds-create-db-instance` – Includes the input parameter `iops`, and the output shows the IOPS rate.
- `rds-modify-db-instance` – Includes the input parameter `iops`, and the output shows the IOPS rate.
- `rds-restore-db-instance-from-db-snapshot` – Includes the input parameter `iops`, and the output shows current IOPS rate. If **Apply Immediately** was specified, the output also shows the pending IOPS rate.
- `rds-restore-db-instance-to-point-in-time` – Includes the input parameter `iops`, and the output shows the IOPS rate.
The Amazon RDS API supports Provisioned IOPS storage in the following actions:

- **CreateDBInstance** – Includes the input parameter `iops`, and the output shows the IOPS rate.
- **CreateDBInstanceReadReplica** – Includes the input parameter `iops`, and the output shows the IOPS rate.
- **CreateDBSnapshot** – The output shows the IOPS rate.
- **ModifyDBInstance** – Includes the input parameter `iops`, and the output shows the IOPS rate.
- **RestoreDBInstanceFromDBSnapshot** – Includes the input parameter `iops`, and the output shows current IOPS rate. If **Apply Immediately** was specified, the output also shows the pending IOPS rate.
- **RestoreDBInstanceToPointInTime** – Includes the input parameter `iops`, and the output shows the IOPS rate.

### Modifying a DB Instance to Use Provisioned IOPS Storage

You can use the Amazon RDS console, the Amazon RDS API, or the Command Line Interface (CLI) to modify a DB instance that uses standard storage to use Provisioned IOPS storage. You must specify either a value for allocated storage or specify both allocated storage and IOPS values. You may need to modify the amount of allocated storage in order to maintain the required ratio between IOPS and storage. For more information about the required ratio between IOPS and storage, see the **Things You Should Know About Provisioned IOPS Storage** (p. 361).

**Note**
You cannot modify an existing SQL Server DB instance to use Provisioned IOPS storage.

Your DB instance will be unavailable for a short time when you initiate a conversion from Provisioned IOPS storage to standard storage or vice versa. For DB instances in a single Availability Zone, the DB instance could be unavailable for a few minutes when the conversion is initiated. For multi-AZ deployments, the time the DB instance is unavailable is limited to the time it takes for a failover operation to complete, which is typically takes less than two minutes. Although your DB instance is available for reads and writes during the conversion, you may experience degraded performance until the conversion process is complete. This process can take several hours.

**Important**
DB instances that have had one or more scale storage operations (modifications to increase storage capacity) applied prior to March 15, 2013 may experience a one-time, extended migration time. The duration of the migration depends on several factors such as database load, storage size, storage type (standard or PIOPS), amount of IOPS provisioned (if any), and number of prior scale storage operations. Typical migration times will be under 24 hours, but can take up to several days in some cases. During the migration, the DB instance will be available for use, but may experience performance degradation. **While the migration takes place, nightly backups will be suspended and no other Amazon RDS operations can take place, including Modify, Reboot, Delete, Create Read Replica, and Take DB Snapshot.** After this one-time extended migration time, all future storage operations for the DB instance will proceed at a faster rate.

### AWS Management Console

To modify a DB instance to use Provisioned IOPS storage
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. From the navigation pane on the Amazon RDS Console, click **DB Instances**.
3. Select the DB instance that you want to modify.
4. In the **Instance Actions** dropdown list, click **Modify**.
5. Select the **Use Provisioned IOPS** check box. Enter values for **Allocated Storage** and **Provisioned IOPS**. The values that you enter must maintain the required ratio between the minimum allocated storage value and the Provisioned IOPS value. The ratios can be between 3:1 to 10:1 for Oracle and MySQL DB instances, and 10:1 for SQL Server.
6. To immediately initiate conversion of the DB instance to use Provisioned IOPS storage, select the **Apply Immediately** check box. If the check box is cleared (the default), the changes will be applied during the next maintenance window.
7. When the settings are as you want them, click **Modify**.

### CLI

To modify a DB instance to use Provisioned IOPS storage

Use the `rds-modify-db-instance` command. Set the following parameters:

- `--allocated-storage` – Amount of storage to be allocated for the DB instance, in gigabytes.
- `--iops` – The IOPS rate for the DB instance, expressed in I/O operations per second.
- `--apply-immediately` – Set to `True` to initiate conversion immediately. If `False` (the default), the conversion is applied during the next maintenance window.
API

Use the ModifyDBInstance action. Set the following parameters:

- **AllocatedStorage** – Amount of storage to be allocated for the DB instance, in gigabytes.
- **Iops** – The new IOPS rate for the DB instance, expressed in I/O operations per second.
- **ApplyImmediately** – Set to True if you want to initiate conversion immediately. If False (the default), the conversion is applied during the next maintenance window.

Modifying the IOPS and Storage Settings for a DB Instance That Uses Provisioned IOPS Storage

You can modify the settings for an Oracle, PostgreSQL, or MySQL DB instance that uses Provisioned IOPS storage by using the AWS Management Console, the Amazon RDS API, or the Command Line Interface (CLI). You must specify the IOPS rate and the amount of storage that you require. You can choose from 1000 IOPS and 100GB of storage up to 30,000 IOPS and 3 TB (3000 GB) of storage, depending on your database engine. You cannot reduce the amount of allocated storage from the value currently allocated for the DB instance. For more information, see the Things You Should Know About Provisioned IOPS Storage (p. 361).

**Note**
You cannot modify the IOPS rate or allocated storage settings for a SQL Server DB instance.

AWS Management Console

To modify 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, click **DB Instances**.
   
   **Note**
   To filter the list of DB instances, in the **Search DB Instances**… box, type a text string that Amazon RDS will use to filter the results. Only DB instances whose name contains the string will appear.
3. Select the DB instance with Provisioned IOPS storage that you want to modify.
4. In the **Instance Actions** dropdown list, click **Modify**.
5. On the **Modify DB Instance** page, enter the value that you want for either **Allocated Storage** or **Provisioned IOPS**.
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 indicating the range of values required for the other parameter.

6. To apply the changes to the DB instance immediately, select the Apply Immediately check box. If you leave the check box cleared, the changes will be applied during the next maintenance window.

7. Click Continue.

8. Review the parameters that will be changed, and click Modify DB Instance to complete the modification.

The new value for allocated storage or for provisioned IOPS appears in the Pending Values column.

---

**CLI**

To modify the Provisioned IOPS settings for a DB instance

Use the `rds-modify-db-instance` command. Set the following parameters:

- `--allocated-storage` – Amount of storage to be allocated for the DB instance, in gigabytes.
- `--iops` – The new amount of Provisioned IOPS for the DB instance, expressed in I/O operations per second.
--apply-immediately – Set to True to initiate conversion immediately. If False (the default), the conversion is applied during the next maintenance window.

API

To modify the Provisioned IOPS settings for a DB instance

Use the ModifyDBInstance action. Set the following parameters:

- AllocatedStorage – Amount of storage to be allocated for the DB instance, in gigabytes.
- Iops – The new IOPS rate for the DB instance, expressed in I/O operations per second.
- ApplyImmediately – Set to True if you want to initiate conversion immediately. If False (the default), the conversion will be applied during the next maintenance window.

Creating a DB Instance that Uses Provisioned IOPS Storage

You can create a DB instance that uses Provisioned IOPS by setting several parameters when you launch the DB instance. You can use the AWS Management Console, the Amazon RDS API, or the Command Line Interface (CLI). For more information about the settings you should use when creating a DB instance, see Creating a DB Instance Running the MySQL Database Engine (p. 82), Creating a DB Instance Running the Oracle Database Engine (p. 138), or Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 207).

AWS Management Console

To create a new DB instance that uses Provisioned IOPS storage

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. From the Amazon RDS Console, click Launch DB Instance.
3. In the Launch RDS DB Instance Wizard, on the Engine Selection page, click the Select button next to the DB Engine that you want.
4. On the DB Instance Details page, enter the required values. For more information on the settings you should use when creating a DB instance, see Creating a DB Instance Running the MySQL Database Engine (p. 82), Creating a DB Instance Running the Oracle Database Engine (p. 138), or Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 207).
5. Select the Use Provisioned IOPS check box. The values for Allocated Storage and Provisioned IOPS change to reflect the minimum allocated storage value and the proportional provisioned IOPS value. You can change these values but the ratio between provisioned IOPS and allocated storage must be in a range between 3:1 and 10:1 for MySQL and Oracle instances, while SQL Server requires a ratio of 10:1.
6. When the settings are as you want them, click **Continue**. Enter the remaining values to create the DB instance.

**CLI**

To create a new DB instance that uses Provisioned IOPS storage

Use the `rds-create-db-instance` command. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `--allocated-storage` - Amount of storage to be allocated for the DB instance, in gigabytes.
- `--iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.

**API**

To create a new DB instance that uses Provisioned IOPS storage

Use the `CreateDBInstance` action. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `AllocatedStorage` - Amount of storage to be allocated for the DB instance, in gigabytes.
- `Iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.
Creating a Read Replica that Uses Provisioned IOPS Storage

You can create a MySQL read replica that uses Provisioned IOPS storage. You can create a read replica that uses Provisioned IOPS storage by using a source DB instance that uses either standard storage or Provisioned IOPS storage.

AWS Management Console

To create a read replica DB instance that uses Provisioned IOPS storage

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, click **DB Instances**.
3. Select the MySQL DB instance with Provisioned IOPS storage that you want to use as the source for the read replica, and then click **Instance Actions** and select **Create Read Replica**.
4. On the **Create Read Replica DB Instance** page, type a DB instance identifier for the read replica.

5. Click **Yes, Create Read Replica**.

CLI

To create a read replica DB instance that uses Provisioned IOPS

Use the **rds-create-db-instance-read-replica** command. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- **--allocated-storage** - Amount of storage to be allocated for the DB instance, in gigabytes.
- **--iops** - The new IOPS rate for the DB instance, expressed in I/O operations per second.

API

To create a read replica DB instance that uses Provisioned IOPS

Use the **CreateDBInstanceReadReplica** action. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- **AllocatedStorage** - Amount of storage to be allocated for the DB instance, in gigabytes.
• **Iops** - The new IOPS rate for the DB instance, expressed in I/O operations per second.
Adjusting the Preferred Maintenance Window

Every DB Instance has a weekly maintenance window during which any system changes are applied. If you don't specify a preferred maintenance window when you create the DB Instance, Amazon RDS assigns a 30-minute maintenance window on a randomly selected day of the week. The 30-minute maintenance window is selected at random from an 8-hour block of time per region. The following table lists the default maintenance windows for each Region.

<table>
<thead>
<tr>
<th>Region</th>
<th>Time Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Northern Virginia)</td>
<td>03:00-11:00 UTC</td>
</tr>
<tr>
<td>Region</td>
<td></td>
</tr>
<tr>
<td>US West (Northern California)</td>
<td>06:00-14:00 UTC</td>
</tr>
<tr>
<td>Region</td>
<td></td>
</tr>
<tr>
<td>US West (Oregon) Region</td>
<td>06:00-14:00 UTC</td>
</tr>
<tr>
<td>Region</td>
<td></td>
</tr>
<tr>
<td>EU (Ireland) Region</td>
<td>22:00-06:00 UTC</td>
</tr>
<tr>
<td>Region</td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Tokyo) Region</td>
<td>17:00-03:00 UTC</td>
</tr>
<tr>
<td>Region</td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Sydney) Region</td>
<td>12:00-20:00 UTC</td>
</tr>
<tr>
<td>Region</td>
<td></td>
</tr>
<tr>
<td>South America (São Paulo) Region</td>
<td>00:00-08:00 UTC</td>
</tr>
<tr>
<td>Region</td>
<td></td>
</tr>
<tr>
<td>AWS GovCloud (US) Region</td>
<td>06:00-14:00 UTC</td>
</tr>
</tbody>
</table>

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 that are being applied 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. Launch the AWS Management Console.
   a. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
   b. Click on the DB Instances link in the Navigation panel on the left side of the console display.
      The My Instances list appears.
   c. Right-click on the DB Instance in the My DB Instances list and select Modify from the drop-down menu.
      The Modify DB Instance window appears.
2. Type the maintenance window into the Maintenance Window text box using the format "day:hour:minute-day:hour:minute".

   **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.

3. Click the **OK** button.

   Changes to the maintenance window take effect immediately.

**CLI**

To adjust the preferred maintenance window

- Use the `rds-modify-db-instance` command with the following parameters:

```bash
PROMPT>rds-modify-db-instance mydbinstance --preferred-maintenance-window Tue:04:00-Tue:04:30
```

This command produces output similar to the following.

```
DBINSTANCE mydbinstance 2009-10-22T18:10:15.274Z db.m1.large mysql
60
master available mydbinstance.clouwupjnvmq.us-east-1.rds.amazonaws.com
3306 us-east-1a 1 n 5.1.57 general-public-license
SECGROUP default active
PARAMGRP default.mysql5.1 in-sync
```

**API**

To adjust the preferred maintenance window

- Call `ModifyDBInstance` with the following parameters:

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

https://rds.amazonaws.com/
?Action=ModifyDBInstance
&DBInstanceIdentifier=mydbinstance
&PreferredMaintenanceWindow=Tue:04:00-Tue:04:30
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-14T17%3A48%3A21.746Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>

Related Topics

- DB Instance Maintenance (p. 53)
- DB Instance (p. 43)
- DB Instance Class (p. 47)
Monitoring a DB Instance

There are several ways you can track the performance and health of a database or a DB instance. You can:

• Use the free Amazon CloudWatch service to monitor the performance and health of a DB instance.
• Subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB snapshot, DB parameter group, or DB security group.
• View, download, or watch database log files using the Amazon RDS console or Amazon RDS APIs. You can also query some database log files that are loaded into database tables.
• Use the AWS CloudTrail service to record AWS calls made by your AWS account. The calls are recorded in log files and stored in an Amazon S3 bucket.

Topics
• Viewing DB Instance Metrics (p. 377)
• Using Amazon RDS Event Notification (p. 380)
• Viewing Amazon RDS Events (p. 395)
• Working with Database Log Files (p. 397)
• Logging Amazon RDS API Calls Using CloudTrail (p. 415)
Viewing DB Instance Metrics

Amazon RDS and CloudWatch are integrated so you can gather and monitor a variety of metrics. You can view CloudWatch metrics using the RDS console, CLI, or API.

In the following example, you use CloudWatch to gather storage space statistics for an Amazon RDS DB instance for the past hour.

AWS Management Console

To view usage and performance statistics 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, click DB Instances.
3. Select the check box for the DB instance you want to monitor.
4. Click Show Monitoring at the top of the window.

Graphs showing the metrics for the selected DB instance display in this tab.
Tip
You can use the **Time Range** drop-down list box to select the time range of the metrics represented by the graphs.
You can click on any of the graphs to bring up a more detailed view of the graph that allows you to apply additional metric-specific filters to the metric data.

**CLI**

Note
The following CLI example requires the CloudWatch command line tools. For more information on CloudWatch and to download the developer tools, go to the Amazon CloudWatch product page. Note that 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.
For a complete list of Amazon RDS metrics, go to Amazon RDS Dimensions and Metrics in the Amazon CloudWatch Developer Guide.

**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="DBInstanceId=mydbinstance" --statistics= Average
  --namespace="AWS/RDS" --start-time 2009-10-16T00:00:00 --end-time 2009-10-16T00:02:00
  ```

**API**

Note that 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
Example

http://monitoring.amazonaws.com/
?SignatureVersion=2
&Action=GetMetricStatistics
&Version=2009-05-15
&StartTime=2009-10-16T00:00:00
&EndTime=2009-10-16T00:02:00
&Period=60
&Statistics.member.1=Avere
&Dimensions.member.1="DBInstanceIdentifier=mydbinstance"
&Namespace=AWS/RDS
&MeasureName=FreeStorageSpace
&Timestamp=2009-10-15T17%3A48%3A21.746Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>

Related Topics

• Using Amazon RDS Event Notification (p. 380)
• Working with Database Log Files (p. 397)
Using Amazon RDS Event Notification

Topics

- Amazon RDS Event Categories and Event Messages (p. 381)
- Subscribing to Amazon RDS Event Notification (p. 385)
- Listing Your Amazon RDS Event Notification Subscriptions (p. 388)
- Modifying an Amazon RDS Event Notification Subscription (p. 389)
- Adding a Source Identifier to an Amazon RDS Event Notification Subscription (p. 391)
- Removing a Source identifier from an Amazon RDS Event Notification Subscription (p. 392)
- Listing the Amazon RDS Event Notification Categories (p. 393)
- Deleting an Amazon RDS Event Notification Subscription (p. 394)

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 security group, or for a DB parameter group. For example, if you subscribe to the Backup category for a given DB instance, you will be 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 will be notified when the DB security group is changed. You will also receive notification when an event notification subscription changes.

Event notifications are sent to the addresses you provide when you create the subscription. You may 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 setting the Enabled radio button to No in the Amazon RDS console or by setting the Enabled parameter to false using the CLI or Amazon RDS API.

Note

Amazon RDS event notifications using SMS text messages are currently available for topic 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 Amazon Resource Name (ARN) of an Amazon SNS topic to identify each subscription. The Amazon RDS console will create the ARN for you when you create the subscription. If you use the CLI or API, you have to 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, 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, click 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 will 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, 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.

Event categories for a DB instance source type include: Availability, Backup, Configuration Change, Creation, Deletion, Failover, Failure, Low Storage, Maintenance, Notification, Recovery, and Restoration.

Event categories for a DB snapshot source type include: Creation, Deletion, and Restoration.

Event categories for a DB security group source type include: Configuration Change and Failure.

Event category for a DB parameter group source type is Configuration Change.

The following table shows the event category and a list of events when a DB instance is the source type.

<table>
<thead>
<tr>
<th>Category</th>
<th>Amazon RDS Event ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>RDS-EVENT-0006</td>
<td>The DB Instance is restarting and will be unavailable until the restart is complete.</td>
</tr>
<tr>
<td>Availability</td>
<td>RDS-EVENT-0004</td>
<td>The DB Instance has shut down.</td>
</tr>
<tr>
<td>Availability</td>
<td>RDS-EVENT-0022</td>
<td>An error has occurred while restarting MySQL.</td>
</tr>
<tr>
<td>Backup</td>
<td>RDS-EVENT-0001</td>
<td>A manual backup of the DB instance has started.</td>
</tr>
<tr>
<td>Backup</td>
<td>RDS-EVENT-0002</td>
<td>A manual backup of the DB instance is complete.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0009</td>
<td>The DB instance has been added to a security group.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0024</td>
<td>The DB instance is being converted to a Multi-AZ DB instance.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0030</td>
<td>The DB instance is being converted to a Single-AZ DB instance.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0012</td>
<td>The DB instance class for this DB instance is being changed.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0018</td>
<td>The current storage settings for this DB instance is being changed.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0011</td>
<td>A parameter group for this DB instance has changed.</td>
</tr>
<tr>
<td>Category</td>
<td>Amazon RDS Event ID</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0028</td>
<td>Automatic backups for this DB instance have been disabled.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0032</td>
<td>Automatic backups for this DB instance have been enabled.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0033</td>
<td>There are [count] users that match the master user name. Users not tied to a specific host have been reset.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0025</td>
<td>The DB instance has been converted to a Multi-AZ DB instance.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0029</td>
<td>The DB instance has been converted to a Single-AZ DB instance.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0014</td>
<td>The DB instance class for this DB instance has changed.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0017</td>
<td>The storage settings for this DB instance has changed.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0010</td>
<td>The DB instance has been removed from a security group.</td>
</tr>
<tr>
<td>Configuration Change</td>
<td>RDS-EVENT-0016</td>
<td>The master password for the DB instance has been reset.</td>
</tr>
<tr>
<td>Creation</td>
<td>RDS-EVENT-0005</td>
<td>A DB instance is being created.</td>
</tr>
<tr>
<td>Deletion</td>
<td>RDS-EVENT-0003</td>
<td>The DB instance is being deleted.</td>
</tr>
<tr>
<td>Failover</td>
<td>RDS-EVENT-0034</td>
<td>Amazon RDS is not attempting a requested failover because a failover recently occurred on the DB instance.</td>
</tr>
<tr>
<td>Failover</td>
<td>RDS-EVENT-0013</td>
<td>A Multi-AZ failover that resulted in the promotion of a standby instance has started.</td>
</tr>
<tr>
<td>Failover</td>
<td>RDS-EVENT-0015</td>
<td>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.</td>
</tr>
<tr>
<td>Failover</td>
<td>RDS-EVENT-0049</td>
<td>A Multi-AZ failover has completed.</td>
</tr>
<tr>
<td>Failover</td>
<td>RDS-EVENT-0050</td>
<td>A Multi-AZ activation has started after a successful instance recovery.</td>
</tr>
<tr>
<td>Failover</td>
<td>RDS-EVENT-0051</td>
<td>A Multi-AZ activation is complete. Your database should be accessible now.</td>
</tr>
<tr>
<td>Category</td>
<td>Amazon RDS Event ID</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Failure</td>
<td>RDS-EVENT-0031</td>
<td>The DB instance has failed. We recommend that you begin a point-in-time-restore for the DB instance.</td>
</tr>
<tr>
<td>Failure</td>
<td>RDS-EVENT-0036</td>
<td>The DB instance is in an incompatible network. Some of the specified subnet IDs are invalid or do not exist.</td>
</tr>
<tr>
<td>Failure</td>
<td>RDS-EVENT-0035</td>
<td>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.</td>
</tr>
<tr>
<td>Failure</td>
<td>RDS-EVENT-0058</td>
<td>Error while creating Statspack user account PERFSTAT. Please drop the account before adding the Statspack option.</td>
</tr>
<tr>
<td>Low Storage</td>
<td>RDS-EVENT-0007</td>
<td>The allocated storage for the DB instance has been exhausted. To resolve this issue, you should allocate additional storage for the DB instance.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>RDS-EVENT-0026</td>
<td>Offline maintenance of the DB instance is taking place. The DB instance is currently unavailable.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>RDS-EVENT-0027</td>
<td>Offline maintenance of the DB instance is complete. The DB instance is now available.</td>
</tr>
<tr>
<td>Notification</td>
<td>RDS-EVENT-0044</td>
<td>Operator-issued notification. For more information, see the event message.</td>
</tr>
<tr>
<td>Notification</td>
<td>RDS-EVENT-0045</td>
<td>An error has occurred in the read replication process. For more information, see the event message. For information on troubleshooting read replica errors, see Troubleshooting a Read Replica Problem (p. 116).</td>
</tr>
<tr>
<td>Notification</td>
<td>RDS-EVENT-0046</td>
<td>The read replica has resumed replication. This message appears when you first create a read replica, and when replication has resumed after an error or after replication was stopped.</td>
</tr>
<tr>
<td>Notification</td>
<td>RDS-EVENT-0047</td>
<td>The read replication process has been stopped because replication errors have occurred for more than 30 days. The read replica is still accessible for read operations but cannot synchronize with the master instance. We recommend that you delete the read replica and create a new one. For information on troubleshooting a broken read replica, see Troubleshooting a Read Replica Problem (p. 116).</td>
</tr>
<tr>
<td>Notification</td>
<td>RDS-EVENT-0048</td>
<td>Patching of the DB instance has been delayed.</td>
</tr>
<tr>
<td>Notification</td>
<td>RDS-EVENT-0054</td>
<td>The MySQL storage engine you are using is not InnoDB, which is the recommended MySQL storage engine for Amazon RDS. For information on MySQL storage engines, see Amazon RDS Supported Storage Engines (p. 79).</td>
</tr>
<tr>
<td>Category</td>
<td>Amazon RDS Event ID</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Notification</td>
<td>RDS-EVENT-0055</td>
<td>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. For information on recommended best practices, see Amazon RDS Basic Operational Guidelines (p. 6).</td>
</tr>
<tr>
<td>Recovery</td>
<td>RDS-EVENT-0020</td>
<td>Recovery of the DB instance has started. Recovery time will vary with the amount of data to be recovered.</td>
</tr>
<tr>
<td>Recovery</td>
<td>RDS-EVENT-0021</td>
<td>Recovery of the DB instance is complete.</td>
</tr>
<tr>
<td>Recovery</td>
<td>RDS-EVENT-0023</td>
<td>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.</td>
</tr>
<tr>
<td>Recovery</td>
<td>RDS-EVENT-0052</td>
<td>Recovery of the Multi-AZ instance has started. Recovery time will vary with the amount of data to be recovered.</td>
</tr>
<tr>
<td>Recovery</td>
<td>RDS-EVENT-0053</td>
<td>Recovery of the Multi-AZ instance is complete.</td>
</tr>
<tr>
<td>Read Replicas</td>
<td>RDS-EVENT-0062</td>
<td>Replication on the read replica was manually stopped.</td>
</tr>
<tr>
<td>Replication</td>
<td>RDS-EVENT-0063</td>
<td>Replication on the read replica was reset.</td>
</tr>
<tr>
<td>Restoration</td>
<td>RDS-EVENT-0008</td>
<td>The DB instance has been restored from a DB snapshot.</td>
</tr>
<tr>
<td>Restoration</td>
<td>RDS-EVENT-0019</td>
<td>The DB instance has been restored from a point-in-time backup.</td>
</tr>
</tbody>
</table>

The following table shows the event category and a list of events when a DB parameter group is the source type.

### Categories and Events for the DB Parameter Group Source Type

<table>
<thead>
<tr>
<th>Category</th>
<th>RDS Event ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change</td>
<td>RDS-EVENT-0037</td>
<td>The parameter group was modified.</td>
</tr>
</tbody>
</table>

The following tables shows the event category and a list of events when a DB security group is the source type.

### Categories and Events for the DB Security Group Source Type

| API Version 2013-09-09 | 384 | Amazon Relational Database Service User Guide | Amazon RDS Event Categories and Event Messages |
The security group has been modified.

The Amazon EC2 security group owned by [user] does not exist; authorization for the security group has been revoked.

The following tables show the event category and a list of events when a DB snapshot is the source type.

### Categories and Events for the DB Snapshot Source Type

<table>
<thead>
<tr>
<th>Category</th>
<th>RDS Event ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation</td>
<td>RDS-EVENT-0040</td>
<td>A DB snapshot is being created.</td>
</tr>
<tr>
<td>Creation</td>
<td>RDS-EVENT-0042</td>
<td>A DB snapshot has been created.</td>
</tr>
<tr>
<td>Deletion</td>
<td>RDS-EVENT-0041</td>
<td>A DB snapshot has been deleted.</td>
</tr>
<tr>
<td>Notification</td>
<td>RDS-EVENT-0059</td>
<td>Started the copy of DB snapshot [DB snapshot name] from source region [region name].</td>
</tr>
<tr>
<td>Failure</td>
<td>RDS-EVENT-0061</td>
<td>The copy of a DB snapshot failed.</td>
</tr>
<tr>
<td>Notification</td>
<td>RDS-EVENT-0060</td>
<td>Finished the copy of DB snapshot [DB snapshot name] from source region [region name] in [time] minutes.</td>
</tr>
<tr>
<td>Restoration</td>
<td>RDS-EVENT-0043</td>
<td>A DB instance is being restored from a DB snapshot.</td>
</tr>
</tbody>
</table>

### 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 will receive all the `DB_Instance` events for the specified source. If you specify a `SourceType` but do not specify a `SourceIdentifier`, you will receive notice of the events for that source type for all your Amazon RDS sources. If you do not specify either the `SourceType` nor the `SourceIdentifier`, you will be notified of events generated from all Amazon RDS sources belonging to your customer account.
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 the Amazon RDS Console navigation pane, click Event Subscriptions.

3. In the Event Subscriptions pane, click Create Event Subscription.

4. In the Create Event Subscription dialog box, do the following:

   a. Type a name for the event notification subscription in the Name text box.

   b. Select an existing Amazon SNS Amazon Resource Name (ARN) for an Amazon SNS topic in the Send notifications to dropdown menu or click create topic to enter the name of a topic and a list of recipients.

   c. Select a source type from the Source Type dropdown menu.

   d. Select Yes to enable the subscription. If you want to create the subscription but to not have notifications sent yet, select No.

   e. Depending on the source type you selected, select the event categories and sources you want to receive event notifications for.

   f. Click Yes, Create.

5. The Amazon RDS console indicates that the subscription is being created.
To subscribe to RDS Event Notification

- Use the `rds-create-event-subscription` command.

**API**

To subscribe to Amazon RDS Event Notification

- Call `CreateEventSubscription`.
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 Amazon RDS Console navigation pane, click Event Subscriptions. The Event Subscriptions pane shows all your event notification subscriptions.

CLI

To list your current Amazon RDS event notification subscriptions

• Use the rds-describe-event-subscriptions command.

API

To list your current Amazon RDS event notification subscriptions

• Call DescribeEventSubscriptions.
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 Amazon RDS Console navigation pane, click **Event Notification**.
3. In the **DB Event Notifications** pane, select the subscription that you want to modify.
4. Make your changes to the subscription in the lower pane.
5. Click **Update**. The Amazon RDS console indicates that the subscription is being modified.

**CLI**

To modify an Amazon RDS event notification subscription

- Use the `rds-modify-event-subscription` command.
API

To modify an Amazon RDS Event

- Call ModifyEventSubscription.
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

To add a source identifier to an Amazon RDS event notification subscription

• You can easily add or remove source identifiers using the Amazon RDS console by selecting or deselecting them when modifying a subscription. See the topic `Modifying an Amazon RDS Event Notification Subscription (p. 389)` for more information.

CLI

To add a source identifier to an Amazon RDS event notification subscription

• Use the `rds-add-source-identifier-to-subscription` command.

API

To add a source identifier to an Amazon RDS event notification subscription

• Call `AddSourceIdentifierToSubscription`.
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

To remove a source identifier from an Amazon RDS event notification subscription

- You can easily add or remove source identifiers using the Amazon RDS console by selecting or deselecting them when modifying a subscription. See the topic Modifying an Amazon RDS Event Notification Subscription (p. 389) for more information.

CLI

To remove a source identifier from an Amazon RDS event notification subscription

- Use the `rds-remove-source-identifier-from-subscription` command.

API

To remove a source identifier from an Amazon RDS event notification subscription

- Call `RemoveSourceIdentifierFromSubscription`.
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

To list the Amazon RDS event notification categories

- When you create or modify an event notification subscription, the event categories are displayed in the Amazon RDS console. See the topic Modifying an Amazon RDS Event Notification Subscription (p. 389) for more information.

CLI

To list the Amazon RDS event notification categories

- Use the `rds-describe-event-categories` command.

API

To list the Amazon RDS event notification categories

- Call `DescribeEventCategories`.
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 Amazon RDS Console navigation pane, click DB Event Subscriptions.
3. In the My DB Event Subscriptions pane, click the subscription that you want to delete.
4. Click Delete.
5. The Amazon RDS console indicates that the subscription is being deleted.

CLI

To delete an Amazon RDS event notification subscription

- Use the rds-delete-event-subscription command.

API

To delete an Amazon RDS event notification subscription

- Call DeleteEventSubscription.
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 easily retrieve events for your RDS resources through the AWS Management Console, the `rds-describe-events` CLI command, or the `DescribeEvents` API.

In this example, you view all Amazon RDS events for the past 24 hours (specified in seconds). Events are retained for 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. Click **Events** in the navigation list on the left side of the window. The available events appear in a list.
3. You can use the **Filter** drop-down list box to filter the events by type, and you can use the text box to the right of the **Filter** drop-down list box to further filter your results. For example, the following screen shot shows a list of events filtered by the DB Instance event type and containing the letters "pdx."

![Filter example](image)

**CLI**

To view all Amazon RDS instance events for the past 24 hours

- Use the command `rds-describe-events` with the following parameters to view all Amazon RDS events for the past 24 hours.

```
PROMPT>rds-describe-events --duration 1440
```
To view all Amazon RDS instance events for the past 24 hours

- Call `DescribeEvents` with the following parameters:
  - `Duration` = 1440

**Example**

```plaintext
https://rds.amazonaws.com/
?Action=DescribeEvents
&Duration=1440
&MaxRecords=100
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2012-01-22T20%3A00%3A44.420Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

**Related Topics**

- Using Amazon RDS Event Notification (p. 380)
Working with Database Log Files

You can view, download, and watch database logs using the Amazon RDS console, the Command Line Interface (CLI), or the Amazon RDS API. For example, you can view, download, or watch the error log, slow query log, and general logs for MySQL. You can also view MySQL logs by directing the logs to a database table in the main database and querying that table. Note that viewing, watching, or downloading transaction logs is not supported.

There are five ways to access database log files.

- **View**: You can view the contents of a log file by using the Amazon RDS console. You can also run the `rds-describe-db-log-file` command or call the `DescribeDBLogFiles` API action to list the log files that are available for a DB instance.
- **Watch**: You can view real-time updates to log files by using the Amazon RDS console. You can also run the `rds-watch-db-logfile` command or call the `DownloadDBLogFilePortion` API action to monitor a database log file and poll to retrieve the most recent log file contents.
- **Download**: You can download the contents of a log file using the Amazon RDS console. You can also run the `rds-download-db-logfile` command to download the contents of a log file.
- **Query**: You can direct the MySQL slow query log and general log to a database table and run queries against the table to get the contents of the log file. Enabling the logs to be written to a database table can cause performance degradation.
- **Binary**: You can use the MySQL `mysqlbinlog` utility to download or stream a binary log to your local computer.

**Note**
If you cannot view the list of log files for an existing Oracle DB instance, reboot the instance to view the list.

Working with MySQL Database Log Files

You can monitor the MySQL error log, slow query log, and the general log directly through the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs. You can use the `mysqlbinlog` utility to download or stream a binary 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 below.

### 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. Each log file has the hour it was generated.
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. 317). 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.
- `log_output`: To write the general and slow query logs to the file system, set to `FILE`. The default is `TABLE`, which writes general queries to the `mysql.general_log` table, and slow queries to the `mysql.slow_log` table. To disable logging, set to `NONE`.
- `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.

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.

If either of the following conditions is met, Amazon RDS automatically rotates the slow and general logs in order to reduce the possibility of a large log either blocking database use or affecting performance:

- Disk space usage is greater than 90% of the allocated space, and a single log uses either more than 10% of the allocated storage or more than 5 GB.
- A single log uses more than 20% of the allocated disk space or more than 10 GB, regardless of total disk usage.

In either situation, Amazon RDS records the log rotation in an Amazon RDS event and sends you a notification. Note that in the future Amazon RDS could change the logic about when an automatic log rotation occurs.

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
Accessing File-based MySQL Logs

You can access the file-based MySQL logs, such as the general or slow query logs, using the Amazon RDS features for working with logs:

- For information about listing log files and viewing the contents of a log, see Viewing and Listing Database Log Files (p. 406).
- For information about downloading a log file, see Downloading a Database Log File (p. 409).
- For information about watching real time updates to a log file, see Watching a Database Log File (p. 412).

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 will 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.

Accessing MySQL 5.6 Binary Logs

You can use the mysqlbinlog utility to download or stream binary logs from Amazon RDS instances running MySQL 5.6. 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 the utility will prompt you for a password.
- To have the file downloaded in binary format, specify the --raw option.
- --result-file: Specify the local file that will 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:

```bash
mysqlbinlog --read-from-remote-server --host=MySQL56Instance1.cg034hpkmmjt.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. Specify a period in which have time to download the logs. If you set the retention period, monitor storage usage for the instance to verify that the instance starts has sufficient storage.

This example sets the retention period to 1 day:

```sql
call mysql.rds_set_configuration('binlog retention hours', 24);
```

To display the current setting, use the `mysql.rds_show_configuration` stored procedure:

```sql
call mysql.rds_show_configuration;
```

### Working with Oracle Database Log Files

You can access Oracle alert logs, audit files, and trace files by using the Amazon RDS console or APIs. These files are retained for seven days by default. Note that the Oracle database engine may rotate these logs if they get very large. If you want to retain audit or trace files for a longer period, you should download them. Storing the files locally reduces your Amazon RDS storage costs and makes more space available for your data.

The `DescribeDBLogFiles` API action that lists the Oracle log files that are available for a DB instance ignores the `MaxRecords` parameter and returns up to 1000 records.

### Switching Online Log files

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges. You can use the following Amazon RDS-specific implementations to switch online log files.

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter system switch logfile;</td>
<td>exec rdsadmin.rdsadmin_util.switch_logfile;</td>
</tr>
</tbody>
</table>
Working with Oracle Trace Files

This section describes Amazon RDS-specific procedures to create, refresh, access, and delete trace files.

Listing Files

Two procedures are available to allow access to any file within the `background_dump_dest`. The first method refreshes a view containing a listing of all files currently in the `background_dump_dest`:

```sql
exec rdsadmin.manage_tracefiles.refresh_tracefile_listing;
```

Once the view is refreshed, use the following view to access the results.

```sql
rdsadmin.tracefile_listing
```

An alternative to the previous process (available beginning with version 11.2.0.3.v1) is to use “from table” to stream non-table data in a table-like format to list DB directory contents:

```sql
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP'));
```

The following query shows text of a log file:

```sql
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','alert_xxx.log'));
```

Generating Trace Files

Since 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.

### Hanganalyze

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oradebug hanganalyze 3</code></td>
<td><code>exec rdsadmin.manage_tracefiles.hanganalyze;</code></td>
</tr>
</tbody>
</table>

### System State Dump

<table>
<thead>
<tr>
<th>Oracle Method</th>
<th>Amazon RDS Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oradebug dump systemstate 266</code></td>
<td><code>exec rdsadmin.manage_tracefiles.dump_systemstate;</code></td>
</tr>
</tbody>
</table>
Retrieving Trace Files

You can retrieve any trace file in background_dump_dest using a standard SQL query of 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 above to list the 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. The contents can be retrieved into a local file using a SQL query.

# eg: send the contents of the tracefile to a local file:
sql customer_dba/password@cust01 << EOF > /tmp/systemstatedump.txt
select * from tracefile_table;
EOF

Purging Trace Files

Tracefiles 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. Note that 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 bdump 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 bdump 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);

The following example shows how to purge all files that match a specific pattern:
exec rdsadmin.manage_tracefiles.purge_tracefiles('MYTRACE*');

Retrieving Archived Redo Logs

If you are using Oracle Database 11.2.0.2.v7 or later, you can retain archived redo logs and use log miner (DBMS_LOGMNR) to retrieve log information.

For example, the following command retains redo logs for 24 hours:

```sql
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

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.

```sql
select sum(blocks * block_size) bytes from v$archived_log where start_time>=sysdate-X/24 and dest_id=1;
```

Once you have retained the redo logs, you can use log miner as described in the Oracle documentation.

Previous methods for accessing alert logs and listener logs

You can view the alert and listener logs using the Amazon RDS console. You can also use the following methods to access these logs:

To access the alert log, use the following command:

```sql
select message_text from alertlog;
```

To access the listener log, use the following command:

```sql
select message_text from listenerlog;
```

**Note**
Oracle rotates the alert and listener logs when they exceed 10MB, at which point they will be unavailable from the Amazon RDS views.

Working with SQL Server Database Log Files

You can access SQL Server error logs, agent logs, and trace files by using the Amazon RDS console or APIs. Log files are rotated each day and when a database reboot occurs; a maximum of seven files are retained for each error log, agent log, and trace file. Log files are deleted after seven days. For more information on generating logs using SQL Server Agent with Amazon RDS, see Using SQL Server Agent (p. 233)

Viewing the SQL Server Error Log Using the CLI

You can use the SQL Server specific `rds_read_error_log` CLI command to view an error log.

```sql
EXEC rdsadmin.dbo.rds_read_error_log [index]
```
In the `rds_read_error_log` command, `index` corresponds to the requested error log relative to the current error log. The default value is 0, which returns the current error log. The previous log has index value 1, the one before that 2, and so on.

**Managing trace files**

This section describes Amazon RDS-specific procedures to create, refresh, access, and delete trace files.

**Generating a Trace SQL Query**

```sql
declare @rc int
declare @TraceID int
declare @maxfilesize bigint
set @maxfilesize = 5
exec @rc = sp_trace_create @TraceID output, 0, N'D:\rdsdbdata\rdstest', @maxfilesize, NULL
```

**Viewing an Open Trace**

```sql
select * from ::fn_trace_getinfo(default)
```

**Viewing Trace Contents**

```sql
select * from ::fn_trace_gettable('D:\rdsdbdata\rdstest.trc', default)
```

**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.

To view the current trace file retention period, use the `rds_show_configuration` command. At a command prompt, type the following, and then press Enter:

```
PROMPT> exec rdsadmin..rds_show_configuration;
```

To modify the retention period for trace files, use the `rds_set_configuration` command, setting the `tracefile retention` argument to the new retention period, in minutes. The following example sets the retention period to 24 hours:

```
PROMPT> exec rdsadmin..rds_set_configuration 'tracefile retention',1440;
```

For security reasons, you cannot delete a specific trace file on a SQL Server DB instance. To delete all unused tracefiles, set the `tracefile retention` argument to 0.

**Working with PostgreSQL Database Log Files**

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).
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. The log_statement parameter controls which SQL statements are logged. We recommend setting this parameter to all to log all statements; the default value is none. Alternatively, you can set this value to ddl to log all data definition statements (CREATE, ALTER, DROP, etc) or to mod to log all ddl and data-modifying statements (INSERT, UPDATE, DELETE, etc.).

The log_min_duration 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. For more information on these settings, see Error Reporting and Logging in the PostgreSQL documentation.

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. 317)

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 progres.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 parameter. The following example shows the information that is written to the progres.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. 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:
```
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 16:48:56 UTC::@::[2952]:LOG:  parameter "log_min_duration_statement" changed to "1"
```

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 Amazon RDS Command Line Interface (CLI) or APIs.

AWS Management Console

To view a database log file

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, click Instances.
3. Click the details icon next to the DB instance name that has the log file you want to view to show the DB instance details page.
4. On the DB instance details page, click **Logs**.

5. Click the **View** button for the log you want to view.
6. Click **DB Instances** at the top of the page to return to the list of DB instances.

**CLI**

To list the available database log files for a DB instance

- Use the command `rds-describe-db-log-files`.

  The following example directs a list of log files for a DB instance to a text file called `log_file_list.txt`.

  ```
  PROMPT>rds-describe-db-log-files > log_file_list.txt
  ```
API

To list the available database log files for a DB instance

• Call DescribeDBLogFiles.

Related Topics

• Monitoring a DB Instance (p. 376)
• Using Amazon RDS Event Notification (p. 380)

Downloading a Database Log File

You can use the Amazon RDS console or the Command Line Interface (CLI) to download a database log file. You cannot download log files using the Amazon RDS API.

AWS Management Console

To download a database log file

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, click Instances.
3. Click the details icon for the DB instance name that has the log file you want to view.
4. On the DB instance details page, click Logs.
5. Click the **Download** button for the log you want to download.

6. Right-click the link provided, and then select **Save Link As...** from the dropdown menu. Type the location where you want the log file to be saved, then click **Save**. Click **Close** when you are finished.
7. Click **DB Instances** at the top of the page to return to the list of DB instances.

### CLI

**To download a database log file**

- Use the command `rds-download-db-logfile`.

The following example shows how to download the contents of a log file called `error-running.log.20` and store it in a local file called `log_file.txt`.

```
$ rds-download-db-logfile MySQLDB223 --log-file-name error-running.log.20 > log_file.txt
```

### Related Topics

- **Monitoring a DB Instance** (p. 376)
- **Using Amazon RDS Event Notification** (p. 380)
Watching a Database Log File

You can monitor the contents of a log file by using the Amazon RDS console, CLI, or API.

AWS Management Console

To watch a database log file

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, click Instances.
3. Click the details icon for the DB instance name that has the log file you want to view.
4. On the DB instance details page, click Logs.
5. Click the Watch button for the log you want to watch.
6. Click **DB Instances** at the top of the page to return to the list of DB instances.

**CLI**

To watch a database log file

- Use the command `rds-watch-db-logfile`.

The following example shows how to monitor a log file for a DB instance named mysql-db1

```
PROMPT>rds-watch-db-logfile mysql-db1 --log-file-name error-running.log.20
```
API

To watch a database log file

• Call DownloadDBLogFilePortion.

Related Topics

• Monitoring a DB Instance  (p. 376)
• Using Amazon RDS Event Notification (p. 380)
Logging Amazon RDS API Calls Using CloudTrail

AWS CloudTrail is a service that logs all AWS API calls made by or on behalf of your AWS account. The logging information is stored in an Amazon S3 bucket. You can use the information collected by CloudTrail to monitor activity for your Amazon RDS DB instances. For example, you can determine whether a request completed successfully and which user made the request. To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Note

AWS CloudTrail only logs events for AWS RDS API calls. If you want to audit actions taken on your database that are not part of the Amazon RDS API, such as when a user connects to your database or when a change is made to your database schema, then you will need to use the monitoring capabilities provided by your DB engine.

Configuring CloudTrail Event Logging

CloudTrail creates audit trails in each region separately and stores them in an Amazon S3 bucket. You can configure CloudTrail to use Amazon SNS to notify you when a log file is created, but that is optional. CloudTrail will notify you frequently, so we recommend that you use Amazon SNS in conjunction with an Amazon SQS queue and handle notifications programmatically.

You can enable CloudTrail using the AWS Management Console, CLI, or API. When you enable CloudTrail logging, you can have the CloudTrail service create an Amazon S3 bucket for you to store your log files. For details, see Creating and Updating Your Trail in the AWS CloudTrail User Guide. The AWS CloudTrail User Guide also contains information on how to aggregate CloudTrail logs from multiple regions into a single Amazon S3 bucket.

There is no cost to use the CloudTrail service. However, standard rates for Amazon S3 usage apply as well as rates for Amazon SNS usage should you include that option. For pricing details, see the Amazon S3 and Amazon SNS pricing pages.

Amazon RDS Event Entries in CloudTrail Log Files

CloudTrail log files contain event information formatted using JSON. An event record represents a single AWS API call and includes information about the requested action, the user that requested the action, the date and time of the request, and so on.

CloudTrail log files include events for all AWS API calls for your AWS account, not just calls to the Amazon RDS API. However, you can read the log files and scan for calls to the Amazon RDS API using the `eventName` element.

The following example shows a CloudTrail log for a user that created a snapshot of a DB instance and then deleted that instance using the Amazon RDS console. The console is identified by the `userAgent` element. The requested API calls made by the console (`CreateDBSnapshot` and `DeleteDBInstance`) are found in the `eventName` element for each record. Information about the user (Alice) can be found in the `userIdentity` element.

```json
{
  "Records": [
    {
      "awsRegion": "us-west-2",
      "eventName": "CreateDBSnapshot",
      "eventSource": "rds.amazonaws.com",
      "eventTime": "2014-01-14T16:23:49Z",
```

Amazon Relational Database Service User Guide
Logging Amazon RDS API Calls Using CloudTrail

API Version 2013-09-09
415
For more information about the different elements and values in CloudTrail log files, see CloudTrail Event Reference in the AWS CloudTrail User Guide.

You may also want to make use of one of the Amazon partner solutions that integrate with CloudTrail to read and analyze your CloudTrail log files. For options, see the AWS partners page.
Using the Amazon RDS API

Using 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 \texttt{param.n} notation. Values of \(n\) are integers starting from 1.

For information about this product's regions and endpoints, go to Regions and Endpoints in 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 a signature version 2 or signature version 4. This section describes how to create a signature version 2. For information about creating a signature version 4, see Signature Version 4 Signing Process.

The following are the basic steps used to authenticate requests to AWS. This assumes you are registered with AWS and have an Access Key ID and Secret Access Key.

\textbf{Tip}

You can find your Access Key ID and Secret Access Key on the Security Credentials section of the AWS Your Account page.
Query Authentication Process

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The sender constructs a request to AWS.</td>
</tr>
<tr>
<td>2</td>
<td>The sender calculates the request signature, a Keyed-Hashing for Message Authentication Code (HMAC) with a SHA-1 hash function, as defined in the next section of this topic.</td>
</tr>
<tr>
<td>3</td>
<td>The sender of the request sends the request data, the signature, and Access Key ID (the key-identifier of the Secret Access Key used) to AWS.</td>
</tr>
<tr>
<td>4</td>
<td>AWS uses the Access Key ID to look up the Secret Access Key.</td>
</tr>
<tr>
<td>5</td>
<td>AWS generates a signature from the request data and the Secret Access Key using the same algorithm used to calculate the signature in the request.</td>
</tr>
<tr>
<td>6</td>
<td>If the signatures match, the request is considered to be authentic. If the comparison fails, the request is discarded, and AWS returns an error response.</td>
</tr>
</tbody>
</table>

**Note**
If a request contains a *Timestamp* parameter, the signature calculated for the request expires 15 minutes after its value. If a request contains an *Expires* parameter, the signature expires at the time specified by the *Expires* parameter.

**Calculating the request signature**

1. Create the canonicalized query string that you need later in this procedure:
   a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when Content-Type is application/x-www-form-urlencoded).
   b. URL encode the parameter name and values according to the following rules:
      i. Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).
      ii. Percent encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.
      iii. Percent encode extended UTF-8 characters in the form %XY%ZA....
      iv. Percent encode the space character as %20 (and not +, as common encoding schemes do).
   c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.
   d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).

2. Create the string to sign according to the following pseudo-grammar (the "\n" represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" + 
ValueOfHostHeaderInLowercase + "\n" + 
HTTPRequestURI + "\n" + 
```
The HTTPRequestURI component is the HTTP absolute path component of the URI up to, but not including, the query string. If the HTTPRequestURI is empty, use a forward slash (/).

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key as the key, and SHA256 or SHA1 as the hash algorithm. For more information, go to http://www.rfc.net/rfc2104.html.

4. Convert the resulting value to base64.

5. Include the value as the value of the Signature parameter in the request.

For example, the following is an example request (linebreaks added for clarity).

```
https://rds.amazonaws.com/
?Action=DescribeDBInstances
&DBInstanceIdentifier=myinstance
&Version=2010-01-01
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&AWSAccessKeyId=<Your AWS Access Key ID>
```

For the preceding Query string, you would calculate the HMAC signature over the following string.

```
GET
rds.amazonaws.com
AWSAccessKeyId=<Your AWS Access Key ID>
&Action=DescribeDBInstances
&DBInstanceIdentifier=myinstance
&Timestamp=2010-05-10T17%3A09%3A03.726Z
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Version=2009-10-16
```

The result is the following signed request.

```
https://rds.amazonaws.com/
?Action=DescribeDBInstances
&DBInstanceIdentifier=myinstance
&Version=2010-01-01
&Signature=<URLEncode(Base64Encode(Signature))>
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&AWSAccessKeyId=<Your AWS Access Key ID>
```
Using the SOAP API

Topics
- WSDL and Schema Definitions (p. 420)
- Programming Language Support (p. 420)
- Request Authentication (p. 421)
- The Response Structure (p. 422)
- Web Services References (p. 423)

WSDL and Schema Definitions

You can access the Amazon Relational Database Service using the SOAP web services messaging protocol. This interface is described by a Web Services Description Language (WSDL) document, which defines the operations and security model for the particular service. The WSDL references an XML Schema document, which strictly defines the data types that might appear in SOAP requests and responses. For more information on WSDL and SOAP, see Web Services References (p. 423).

**Note**
Amazon RDS supports SOAP only through HTTPS.

All schemas have a version number. The version number appears in the URL of a schema file and in a schema's target namespace. This makes upgrading easy by differentiating requests based on the version number.

The current versions of the Amazon RDS WSDL are available at the following locations:

<table>
<thead>
<tr>
<th>Region</th>
<th>WSDL Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>EU (Ireland) Region</td>
<td><a href="https://rds.eu-west-1.amazonaws.com/doc/2013-09-09/AmazonRDSv7.wsdl">https://rds.eu-west-1.amazonaws.com/doc/2013-09-09/AmazonRDSv7.wsdl</a></td>
</tr>
<tr>
<td>Asia Pacific (Singapore) Region</td>
<td><a href="https://rds.ap-southeast-1.amazonaws.com/doc/2013-09-09/AmazonRDSv7.wsdl">https://rds.ap-southeast-1.amazonaws.com/doc/2013-09-09/AmazonRDSv7.wsdl</a></td>
</tr>
<tr>
<td>Asia Pacific (Tokyo) Region</td>
<td><a href="https://rds.ap-northeast-1.amazonaws.com/doc/2013-09-09/AmazonRDSv7.wsdl">https://rds.ap-northeast-1.amazonaws.com/doc/2013-09-09/AmazonRDSv7.wsdl</a></td>
</tr>
<tr>
<td>South America (São Paulo) Region</td>
<td><a href="https://rds.sa-east-1.amazonaws.com/doc/2013-09-09/AmazonRDSv7.wsdl">https://rds.sa-east-1.amazonaws.com/doc/2013-09-09/AmazonRDSv7.wsdl</a></td>
</tr>
</tbody>
</table>

Programming Language Support

Since the SOAP requests and responses in Amazon RDS follow current standards, any programming language with the appropriate library support can be used. Languages known to have this support include C++, C#, Java, Perl, Python and Ruby.
Request Authentication

Amazon RDS complies with the current WS-Security standard, which requires you to hash and sign SOAP requests for integrity and non-repudiation. WS-Security defines profiles which are used to implement various levels of security. Secure SOAP messages use the BinarySecurityToken profile, consisting of an X.509 certificate with an RSA public key.

The following is the content of an insecure `DescribeDBInstances` operation:

```xml
<DescribeDBInstances>
  <MaxRecords>100</MaxRecords>
</DescribeDBInstances>
```

To secure the request, we add the `BinarySecurityToken` element.

The secure version of the request begins with the following:

```xml
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="Timestamp-2">
        <wsu:Created>2009-10-28T18:41:59.597Z</wsu:Created>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="CertId-5992FC58FDECA60AF912567553195531">
        ....many, many lines of base64 encoded X.509 certificate...
      </wsse:BinarySecurityToken>
    </wsse:Security>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature-1">
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      </ds:SignedInfo>
      <ds:Reference URI="#Timestamp-2">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>DLFQyK61qWoJiMyC9w34siRELAM=</ds:DigestValue>
      </ds:Reference>
    </ds:Signature>
  </soap:Header>
</soap:Envelope>
```
If you are matching this against requests generated by Amazon RDS supplied libraries, or those of another vendor, the following are the most important elements.

Elements

- **BinarySecurityToken**—Contains the X.509 certificate in base64 encoded PEM format
- **Signature**—Contains an XML digital signature created using the canonicalization, signature algorithm, and digest method
- **Timestamp**—Requests to Amazon RDS are valid within 5 minutes of this value to help prevent replay attacks

## The Response Structure

In response to a request, the Amazon RDS service returns an XML data structure that conforms to an XML schema defined as part of the Amazon RDS WSDL. The structure of an XML response is specific to the associated request.

The following is an example response:
Web Services References

For more information about using web services, go to any of the following resources:

- Web Service Description Language (WSDL)
- WS-Security BinarySecurityToken Profile

Available Libraries

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of SOAP and Query. These libraries provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. Libraries and resources are available for the following languages:

- Java
- PHP
- Python
- Ruby
- Windows and .NET

For libraries and sample code in all languages, go to Sample Code & Libraries.

Troubleshooting Applications

Topics

- Retrieving Errors (p. 423)
- Troubleshooting Tips (p. 424)

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.

```perl
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 Relational Database Service API.

- **Verify that Amazon Relational Database Service is operating normally in the region you are targeting** by visiting [http://status.aws.amazon.com](http://status.aws.amazon.com).
- **Check the structure of your request**
  Each Amazon Relational Database Service 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/](https://forums.aws.amazon.com/)
The following table describes the important changes to the documentation since the last release of the Amazon Relational Database Service User Guide.

- **API version**: 2013-09-09
- **Latest documentation update**: April 23, 2014

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>New feature</td>
<td>Updated to support upgrades from MySQL version 5.5 to version 5.6.</td>
<td>April 23, 2014</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support Oracle 11.2.0.4.</td>
<td>April 23, 2014</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support Oracle GoldenGate.</td>
<td>April 3, 2014</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support the M3 DB instance classes.</td>
<td>February 20, 2014</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support the Oracle Timezone option.</td>
<td>January 13, 2014</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support Oracle 11.2.0.3.</td>
<td>December 16, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support replication between Amazon RDS MySQL DB instances in different regions.</td>
<td>November 26, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support SQL Server transparent data encryption (TDE).</td>
<td>November 7, 2013</td>
</tr>
<tr>
<td>New API and new feature</td>
<td>Updated to support cross region DB snapshot copies; new API version, 2013-09-09.</td>
<td>October 31, 2013</td>
</tr>
<tr>
<td>New features</td>
<td>Updated to support Oracle Statspack.</td>
<td>September 26, 2013</td>
</tr>
<tr>
<td>New features</td>
<td>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.</td>
<td>September 5, 2013</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>New features</td>
<td>Updated to support the db.cr1.8xlarge DB instance class for MySQL 5.6.</td>
<td>September 4, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support replication of read replicas.</td>
<td>August 28, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support parallel read replica creation.</td>
<td>July 22, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support fine-grained permissions and tagging for all Amazon RDS resources.</td>
<td>July 8, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support MySQL 5.6 for new instances, including support for the MySQL 5.6 memcached interface and binary log access.</td>
<td>July 1, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support major version upgrades from MySQL 5.1 to MySQL 5.5.</td>
<td>June 20, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated DB parameter groups to allow expressions for parameter values.</td>
<td>June 20, 2013</td>
</tr>
<tr>
<td>New features</td>
<td>Updated to support Oracle Advanced Security features for native network encryption and Oracle Transparent Data Encryption.</td>
<td>April 18, 2013</td>
</tr>
<tr>
<td>New features</td>
<td>Updated to support major version upgrades for SQL Server and additional functionality for Provisioned IOPS.</td>
<td>March 13, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support VPC By Default for RDS.</td>
<td>March 11, 2013</td>
</tr>
<tr>
<td>New API and feature</td>
<td>Updated to support log access; new API version 2013-02-12</td>
<td>March 4, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support RDS event notification subscriptions.</td>
<td>February 4, 2013</td>
</tr>
<tr>
<td>New API and feature</td>
<td>Updated to support DB instance renaming and the migration of DB security group members in a VPC to a VPC security group.</td>
<td>January 14, 2013</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated for GovCloud support.</td>
<td>December 17, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support m1.medium and m1.xlarge DB Instance classes.</td>
<td>November 6, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support Read Replica promotion.</td>
<td>October 11, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support SSL in Microsoft SQL Server DB Instances.</td>
<td>October 10, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support Oracle micro DB Instances.</td>
<td>September 27, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated to support SQL Server 2012.</td>
<td>September 26, 2012</td>
</tr>
<tr>
<td>New API and feature</td>
<td>Updated to support provisioned IOPs. API version 2012-09-17.</td>
<td>September 25, 2012</td>
</tr>
<tr>
<td>New features</td>
<td>Updated for SQL Server support for DB Instances in VPC and Oracle support for Data Pump.</td>
<td>September 13, 2012</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated for support for SQL Server Agent.</td>
<td>August 22, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated for support for tagging of DB Instances.</td>
<td>August 21, 2012</td>
</tr>
<tr>
<td>New features</td>
<td>Updated for support for Oracle APEX and XML DB, Oracle time zones, and Oracle DB Instances in a VPC.</td>
<td>August 16, 2012</td>
</tr>
<tr>
<td>New features</td>
<td>Updated for support for SQL Server Database Engine Tuning Advisor and Oracle DB Instances in VPC.</td>
<td>July 18, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated for support for MySQL db.t1.micro DB Instances.</td>
<td>June 11, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated for support for option groups and first option, Oracle Enterprise Manager Database Control.</td>
<td>May 29, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated for support for read replicas in Amazon Virtual Private Cloud.</td>
<td>May 17, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated for Microsoft SQL Server support.</td>
<td>May 8, 2012</td>
</tr>
<tr>
<td>New features</td>
<td>Updated for support for forced failover, Multi-AZ deployment of Oracle DB Instances, and nondefault character sets for Oracle DB Instances.</td>
<td>May 2, 2012</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated for Amazon Virtual Private Cloud (VPC) Support.</td>
<td>February 13, 2012</td>
</tr>
<tr>
<td>Updated content</td>
<td>Updated for new Reserved Instance types.</td>
<td>December 19, 2011</td>
</tr>
<tr>
<td>New feature</td>
<td>Updated for Oracle engine support.</td>
<td>May 23, 2011</td>
</tr>
<tr>
<td>Updated content</td>
<td>Console updates.</td>
<td>May 13, 2011</td>
</tr>
<tr>
<td>Updated content</td>
<td>Edited content for shortened backup and maintenance windows.</td>
<td>February 28, 2011</td>
</tr>
<tr>
<td>New feature</td>
<td>Added support for MySQL 5.5.</td>
<td>January 31, 2011</td>
</tr>
<tr>
<td>New feature</td>
<td>Added support for Read Replicas.</td>
<td>October 4, 2010</td>
</tr>
<tr>
<td>New feature</td>
<td>Added support for AWS Identity and Access Management (IAM).</td>
<td>September 2, 2010</td>
</tr>
<tr>
<td>New feature</td>
<td>Added Reserved DB Instances.</td>
<td>August 16, 2010</td>
</tr>
<tr>
<td>New Feature</td>
<td>Amazon RDS now supports SSL connections to your DB Instances.</td>
<td>June 28, 2010</td>
</tr>
<tr>
<td>New Guide</td>
<td>This is the first release of <em>Amazon Relational Database Service User Guide</em>.</td>
<td>June 7, 2010</td>
</tr>
</tbody>
</table>
Amazon RDS Resources

The following table lists related resources that you'll find useful as you work with this service.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Relational Database Service API Reference</td>
<td>The API reference contains a comprehensive description of all Amazon RDS Query APIs and data types.</td>
</tr>
<tr>
<td>Amazon Relational Database Service Command Line Reference</td>
<td>The Command Line Tools Reference contains a comprehensive description of all the command line tools and their options.</td>
</tr>
<tr>
<td>Amazon RDS Technical FAQ</td>
<td>The FAQ covers the top questions developers have asked about this product.</td>
</tr>
<tr>
<td>Release notes</td>
<td>The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.</td>
</tr>
<tr>
<td>AWS Developer Resource Center</td>
<td>A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.</td>
</tr>
<tr>
<td>AWS Management Console</td>
<td>The AWS Management Console allows you to perform most of the functions of Amazon RDS without programming.</td>
</tr>
<tr>
<td>Discussion Forums</td>
<td>A community-based forum for developers to discuss technical questions related to Amazon Web Services.</td>
</tr>
<tr>
<td>Amazon RDS product information</td>
<td>The primary web page for information about Amazon RDS.</td>
</tr>
<tr>
<td>Contact Us</td>
<td>A central contact point for inquiries concerning AWS billing, account, events, abuse etc.</td>
</tr>
<tr>
<td>Conditions of Use</td>
<td>Detailed information about the copyright and trademark usage at Amazon.com and other topics.</td>
</tr>
</tbody>
</table>