

---

# **Amazon Relational Database Service**

**User Guide**

**API Version 2012-04-23**



# Amazon Relational Database Service: User Guide

Welcome .....	1
Introduction to Amazon Relational Database Service .....	3
Amazon RDS Terminology and Concepts .....	5
DB Instances .....	5
DB Security Groups .....	11
Backup and Restoration .....	12
DB Parameter Groups .....	14
DB Instance Monitoring .....	14
Amazon RDS Events .....	15
AWS Identity and Access Management .....	15
Amazon RDS and Amazon Virtual Private Cloud (VPC) .....	15
Amazon RDS Database Engines .....	19
MySQL Database Engine .....	20
Oracle Database Engine .....	23
Microsoft SQL Server Database Engine .....	26
Signing up for Amazon RDS .....	30
Setting up the Command Line Tools .....	31
Using the Amazon RDS API .....	36
Controlling User Access to Your AWS Account .....	36
Making API Requests .....	40
Using the Query API .....	40
Using the SOAP API .....	43
Available Libraries .....	46
Troubleshooting Applications .....	47
Using Amazon RDS .....	48
Creating and Modifying DB Instances .....	49
Creating a DB Instance Running the MySQL Database Engine .....	50
Creating a DB Instance Running the Oracle Database Engine .....	54
Creating a DB Instance Running the Microsoft SQL Server Database Engine .....	60
Connecting to a DB Instance Running the MySQL Database Engine .....	64
Connecting to a DB Instance Running the Oracle Database Engine .....	66
Connecting to a DB Instance Running the Microsoft SQL Server Database Engine .....	68
Rebooting a DB Instance .....	73
Modifying a DB Instance .....	74
Modifying a DB Instance Running the MySQL Database Engine .....	74
Modifying a DB Instance Running the Oracle Database Engine .....	75
Deleting a DB Instance .....	77
Sizing Your Amazon RDS DB Instance .....	80
Scaling CPU and Memory for a DB Instance .....	82
Scaling DB Instance Storage .....	84
Adjusting the Preferred Maintenance Window .....	87
Working with Reserved DB Instances .....	90
Using Amazon RDS with Amazon Virtual Private Cloud (VPC) .....	99
Creating a Virtual Private Cloud (VPC) .....	99
Creating a DB Subnet Group .....	102
Creating a DB Instance in a VPC .....	105
Connecting to a DB Instance Running in a VPC .....	109
Importing Data to a DB Instance .....	117
Importing Data to a Microsoft SQL Server Instance .....	117
Preparing to Import Data into Your SQL Server DB Instance .....	117
Import Logins to Your DB Instance .....	119
Import the Data .....	120
Cleaning Up .....	122
Backing Up and Restoring DB Instances .....	125
Creating a DB Snapshot .....	126
Restoring From a DB Snapshot .....	128
Working With Automated Backups .....	130
Restoring a DB Instance to a Specified Time .....	133

DB Parameter Groups .....	135
Working with DB Parameter Groups .....	136
DB Security Groups .....	145
Working with DB Security Groups .....	146
Monitoring DB Instances .....	155
Viewing DB Instance Metrics .....	156
Working with Amazon RDS Events .....	158
Viewing Amazon RDS Events .....	159
Amazon RDS Technical FAQ .....	161
General Information FAQ .....	161
Billing .....	164
Reserved Instances .....	165
Multi-AZ Deployments .....	167
Hardware and Scaling .....	170
Automated Backups and Snapshots .....	172
Security and VPC .....	173
DB Parameter Groups .....	176
Replication .....	177
MySQL Database Engine .....	178
Oracle Database Engine .....	184
SQL Server Database Engine .....	188
Appendix: Common DBA Tasks for MySQL .....	192
Appendix: Common DBA Tasks for Oracle .....	196
Appendix: Common DBA Tasks for Microsoft SQL Server .....	204
Appendix: Oracle Character Sets Supported in Amazon RDS .....	206
Appendix: Oracle DB Engine Patch Composition .....	208
Document History .....	209
Amazon RDS Resources .....	211
Glossary .....	213

# Welcome

---

This is the *Amazon Relational Database Service User Guide*. This guide picks up where the [Amazon RDS Getting Started Guide](#) leaves off, and helps you understand the components that RDS provides and how to use them. The guide shows you how to access RDS with a web-based GUI, with command line tools, and programmatically through the RDS API.

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.

## How Do I...?

How Do I?	Relevant Sections
Get a general product overview and information about pricing	<a href="#">Amazon RDS product page</a>
Get a quick hands-on introduction to RDS	<a href="#">Amazon RDS Getting Started Guide</a>
Learn about Amazon RDS key terminology and concepts	<a href="#">Amazon RDS Terminology and Concepts (p. 5)</a>
How to get started with the command line tools.	<a href="#">Setting up the Command Line Tools (p. 31)</a>
Get started using the Query or SOAP API for EC2	<a href="#">Using the Amazon RDS API (p. 36)</a>
Find available libraries for programmatically accessing RDS	<a href="#">Available Libraries (p. 46)</a>
Get detailed information about how to use the RDS components and features, with instructions for several different interfaces	<a href="#">Using Amazon RDS (p. 48)</a>

How Do I?	Relevant Sections
Learn how to connect to a DB Instance	<a href="#">Connecting to a DB Instance Running the MySQL Database Engine (p. 64)</a>

# Introduction to Amazon Relational Database Service

---

## Topics

- [What is Amazon Relational Database Service? \(p. 3\)](#)
- [Advantages of Amazon Relational Database Service \(p. 3\)](#)

## 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 multiple industry-standard relational databases and manages common database administration tasks.

Amazon RDS gives you access to the capabilities of a familiar MySQL or Oracle database server. This means that most of the code, applications, and tools you already use today with your existing MySQL or Oracle databases work with Amazon RDS without modification. Amazon RDS automatically backs up your database and maintains the database software that powers your DB Instance. Amazon RDS is flexible: you can scale your database instance's compute resources and storage capacity to meet your application's demand. As with all our products, there are no up-front investments, and you pay only for the resources you use.

## Advantages of Amazon Relational Database Service

Amazon RDS provides the following advantages:

- **Accelerated Deployment**—Amazon RDS reduces friction as you move from project conception to deployment.

You can use simple API calls to access the capabilities of a production-ready relational database without worrying about infrastructure provisioning or installing and maintaining database software.

- **Managed**—Amazon RDS handles generic database management tasks so you can pursue higher value application development or database refinements.

- **Compatible**—With Amazon RDS, you get native access to a MySQL or Oracle database server.

This means Amazon RDS works with most of your existing tools, applications, and drivers. There's no need to change any code.

- **Scalable**—With a simple API call you can scale the compute and storage resources available to your database to meet your business needs and application load.
- **Reliable**—Amazon RDS runs on the same highly reliable infrastructure used by other Amazon Web Services.

Amazon RDS's automated backup service automatically manages the backup of your database, letting you restore to any point within a retention period you specify.

- **Designed for use with other AWS products**—Amazon RDS is tightly integrated with our other products. For example, applications running in Amazon EC2 experience low-latency database access to Amazon RDS.
- **Secure**—Amazon RDS provides web service interfaces to configure firewall settings that control network access to your database instances and allows SSL connections to your DB Instances.
- **Inexpensive**—You pay a low rate for the resources you actually consume.

There are no long-term contracts or up-front commitments to use Amazon RDS.

# Amazon RDS Terminology and Concepts

---

## Topics

- [DB Instances \(p. 5\)](#)
- [DB Security Groups \(p. 11\)](#)
- [Backup and Restoration \(p. 12\)](#)
- [DB Parameter Groups \(p. 14\)](#)
- [DB Instance Monitoring \(p. 14\)](#)
- [Amazon RDS Events \(p. 15\)](#)
- [AWS Identity and Access Management \(p. 15\)](#)
- [Amazon RDS and Amazon Virtual Private Cloud \(VPC\) \(p. 15\)](#)

This chapter introduces you to Relational Database Service terminology and concepts. Many of the concepts introduced in this chapter are explored in greater depth in later chapters.

## DB Instances

This section groups database instance concepts and terms.

### DB Instance

A *DB Instance* is an isolated database environment running in the cloud. A DB Instance can contain multiple user-created databases, and can be accessed using the same tools and applications as a stand-alone database instance.



#### Note

Amazon RDS supports access from any standard SQL client application. Amazon RDS does not allow direct host access via Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection.

DB Instances are simple to create and modify with the Amazon RDS command line tools, APIs, or the AWS Management Console.

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.

For more information on working with DB Instances, go to [Creating and Modifying DB Instances \(p. 49\)](#).

## DB Instance Class

After you have selected a DB engine (MySQL or Oracle), you can create a DB Instance from a particular DB Instance class that has the compute power, memory, bandwidth, and storage capacity that best meets your needs. For pricing information, go to [Amazon Relational Database Service \(Amazon RDS\)](#).

For each DB Instance class, you can select from 5GB to 1TB of associated storage capacity. One elastic compute unit (ECU) provides CPU capacity equivalent to a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

The following table describes the instance classes that are available. Except where noted, each instance class is available for all DB engines.

Designation	Description
db.m1.small	Small DB Instance: 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, Moderate I/O Capacity
db.m1.large	Large DB Instance: 7.5 GB memory, 4 ECUs (2 virtual cores with 2 ECUs each), 64-bit platform, High I/O Capacity
db.m1.xlarge	Extra Large DB Instance: 15 GB of memory, 8 ECUs (4 virtual cores with 2 ECUs each), 64-bit platform, High I/O Capacity. <b>MySQL only.</b>
db.m2.xlarge	High-Memory Extra Large Instance: 17.1 GB memory, 6.5 ECU (2 virtual cores with 3.25 ECUs each), 64-bit platform, High I/O Capacity
db.m2.2xlarge	High-Memory Double Extra Large DB Instance: 34 GB of memory, 13 ECUs (4 virtual cores with 3.25 ECUs each), 64-bit platform, High I/O Capacity
db.m2.4xlarge	High-Memory Quadruple Extra Large DB Instance: 68 GB of memory, 26 ECUs (8 virtual cores with 3.25 ECUs each), 64-bit platform, High I/O Capacity

## DB Instance Identifier

The DB Instance identifier is a customer-supplied identifier for a DB Instance. This identifier specifies a particular 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.

## Database Name

The definition of the term *Database Name* depends on the database engine in use.

- For the MySQL database engine, the Database Name is the name of a database hosted in your Amazon DB Instance. An Amazon DB Instance can host multiple databases. 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.

## Database Engine

The Amazon Relational Database Service is designed to eliminate the maintenance overhead associated with running a relational database service, while not affecting the actual usage of the database engine. The Amazon RDS service is designed to support multiple database engines.

For more information on specific database engine implementations, go to [Amazon RDS Database Engines](#) (p. 19).

## DB Instance Maintenance

Periodically, the Amazon RDS system performs maintenance on the DB Instance. This maintenance occurs during a user-definable *maintenance window*, and can include planned patch rollouts for the database engine or the operating system, as well as implementing pending changes to storage or CPU class for the DB Instance. Maintenance typically requires a short downtime for single-AZ DB Instances.



### Note

For Multi-AZ deployments, your downtime during system maintenance will often be shorter than for Single-AZ deployments, because a failover may occur instead of a full offline compute node replacement. Not all maintenance will trigger a failover, because some maintenance can be performed with a shorter outage window than a failover causes. For more information on Multi-AZ deployments, see [Multi-AZ Deployments](#) (p. 8).

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 time blocks for each Region from which the default maintenance windows are assigned.

Region	Time Block
US East (Northern Virginia) Region	03:00-11:00 UTC
US West (Northern California) Region	06:00-14:00 UTC
US West (Oregon) Region	06:00-14:00 UTC
EU (Ireland) Region	22:00-06:00 UTC
Asia Pacific (Singapore) Region	14:00-22:00 UTC
Asia Pacific (Tokyo) Region	17:00-03:00 UTC
South America (São Paulo) Region	00:00-08:00 UTC

## Regions and Availability Zones

Amazon cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, these data center facilities are located in several different physical locations. These locations are categorized by *Regions* and *Availability Zones*.

Regions are large and widely dispersed into separate geographic locations. Availability Zones are distinct locations within a Region that are engineered to be isolated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same Region.



### Important

Each Region is completely independent. Any Amazon RDS activity you initiate (e.g. creating database instances or listing available database instances) runs only in your current default Region. The default Region can be changed by setting the EC2\_REGION environment variable, or be overridden by using the `--url` parameter with the command line interface. See [Common Options for API Tools](#) for more information.

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.

Region	Endpoint
US East (Northern Virginia) Region	<a href="https://rds.us-east-1.amazonaws.com">https://rds.us-east-1.amazonaws.com</a>
US West (Northern California) Region	<a href="https://rds.us-west-1.amazonaws.com">https://rds.us-west-1.amazonaws.com</a>
US West (Oregon) Region	<a href="https://rds.us-west-2.amazonaws.com">https://rds.us-west-2.amazonaws.com</a>
EU (Ireland) Region	<a href="https://rds.eu-west-1.amazonaws.com">https://rds.eu-west-1.amazonaws.com</a>
Asia Pacific (Singapore) Region	<a href="https://rds.ap-southeast-1.amazonaws.com">https://rds.ap-southeast-1.amazonaws.com</a>
Asia Pacific (Tokyo) Region	<a href="https://rds.ap-northeast-1.amazonaws.com">https://rds.ap-northeast-1.amazonaws.com</a>
South America (São Paulo) Region	<a href="https://rds.sa-east-1.amazonaws.com">https://rds.sa-east-1.amazonaws.com</a>

If you do not explicitly specify an endpoint, the US-East (Northern Virginia) Region endpoint is the default.

## Multi-AZ Deployments

You can run your DB Instance as a Multi-AZ deployment. When you select this option, Amazon automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary DB Instance is synchronously replicated across Availability Zones to the standby replica to provide data redundancy, eliminate I/O freezes during system backups, enhance availability during planned system maintenance, and help protect your databases against DB Instance failure and Availability Zone disruption.

In the event of a planned or unplanned outage of your primary DB Instance, Amazon RDS automatically switches to the standby replica. The automatic failover mechanism simply changes the canonical name record (CNAME) of the main DB Instance to point to the standby DB Instance.

Amazon RDS handles the failover 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

You can create a Multi-AZ deployment by simply specifying the Multi-AZ option 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.

Multi-AZ deployments are not a scaling solution for reads and do not allow you to use the standby replica to serve read traffic.

## Read Replicas

The Amazon RDS Read Replica feature lets you create a DB Instance as a replica of another DB Instance. Any updates to the data on the source DB Instance are replicated to the Read Replica DB Instances using the built-in replication feature of MySQL. You can use the Read Replica feature to scale beyond the capacity constraints of a single DB Instance for read-heavy database workloads.



### Note

The Amazon RDS Read Replica feature is currently supported only with the MySQL database engine.

The Amazon RDS Read Replica feature is not currently supported for DB Instances running in an Amazon Virtual Private Cloud (VPC).

Using the Read Replica feature is easy; simply create a Read Replica DB Instance using AWS Management Console or the new `CreateDBInstanceReadReplica` API or `rds-create-db-instance-read-replica` command line tool and specify the source DB Instance that you want to replicate. You can create up to five Read Replicas per DB Instance.

When you initiate the creation of a Read Replica, Amazon RDS takes a snapshot of the source DB Instance and begins replication. As a result, you will experience a brief I/O suspension on your source DB Instance as the snapshot occurs (this I/O suspension is mitigated if the source DB Instance is a Multi-AZ deployment, because snapshots are taken from the Multi-AZ standby).

You can use the standard `DescribeDBInstances` API or the `rds-describe-db-instances` command line tool to return a list of all the DB Instances you have deployed (including Read Replicas), or click the **DB Instances** tab of the Amazon RDS Management Console.

Unlike Multi-AZ deployments, updates are applied to Read Replica DB Instances using asynchronous replication, and are subject to *replication lag*.

Due to replication lag, recent database updates made to a source DB Instance might not be present on associated Read Replicas in the event of an unplanned outage on the source DB Instance; because of this, Read Replicas are not intended to protect against DB Instance or Availability Zone failure.



### Note

Replication lag can be monitored as a CloudWatch metric. For more information on monitoring, see [DB Instance Monitoring \(p. 14\)](#).

To combine the availability and durability of multiple Availability Zones with the scaling benefits of Read Replicas, you can set a Multi-AZ deployment as the source DB Instance for your Read Replicas.

You can simply delete Read Replica DB Instances when you no longer need them.



### Note

A Read Replica will stay active and continue accepting read traffic even after its corresponding source DB Instance has been deleted. You must explicitly delete the Read Replica DB Instance.

## 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.

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.

You can use the command line tools, the API, or the AWS Management Console to list and purchase available Reserved DB Instance *offerings*. The three types of Reserved DB Instance offerings are based on DB instance class, duration, and whether or not the Reserved DB instance is Single-AZ or Multi-AZ.

## Reserved DB Instance Offerings

*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.

The following table summarizes the differences between the Reserved DB Instances offering types.

### Reserved Instance Offerings

Offering	Upfront Cost	Usage Fee	Advantage
Heavy Utilization	Highest	Lowest hourly fee. Applied to the whole term whether or not you're using the Reserved DB Instance.	Lowest overall cost if you plan to utilize your Reserved DB Instances more than 79 percent of a 3-year term.
Medium Utilization	Average	Hourly usage fee charged for each hour you use the DB Instance.	Suitable for elastic workloads or when you expect moderate usage, more than 40 percent of a 3-year term.
Light Utilization	Lowest	Hourly usage fee charged. Highest fees of all the offering types, but they apply only when you're using the Reserved DB Instance.	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.

For more information on working with Reserved DB Instances, go to [Working with Reserved DB Instances](#) (p. 90).

## Related Topics

- [Creating a DB Instance Running the MySQL Database Engine](#) (p. 50)
- [Scaling CPU and Memory for a DB Instance](#) (p. 82)
- [Scaling DB Instance Storage](#) (p. 84)
- [Deleting a DB Instance](#) (p. 77)

## DB Security Groups

Amazon RDS allows you to control access to your DB Instances using *DB Security Groups*. A DB Security Group acts like a firewall controlling network access to your DB Instance. By default, network access is turned off to your DB Instances. If you want your applications to access your DB Instance you can allow access from specific EC2 security groups or IP ranges. Once ingress is configured, the same rules apply to all DB Instances associated with that DBSecurityGroup.



### Important

Please ensure you authorize only specific IP ranges or EC2 security groups. We highly discourage authorizing broad IP ranges (for example, 0.0.0.0/0).

Note that you cannot use Amazon RDS DB Security Groups to restrict access to your Amazon EC2 instances. Similarly, you cannot apply an Amazon EC2 security group to your Amazon RDS DB Instance.

For more information on working with DB Security Groups, go to [Working with DB Security Groups \(p. 146\)](#).

## Related Topics

- [Working with DB Security Groups \(p. 146\)](#)

# Backup and Restoration

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-created snapshots 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 and user-initiated backup operations on Single-AZ DB Instances.

## Automated Backup

Automated backup is an Amazon RDS feature that automatically creates a backup of your database. This backup occurs during a daily user-configurable period of time known as the *backup window*. Backups created during the backup window are retained for a user-configurable number of days (the *backup retention period*).

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.

You can specify the daily backup window for the automated backup when you create or modify your DB 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 a 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.

Region	Time Block
US East (Northern Virginia) Region	03:00-11:00 UTC
US West (Northern California) Region	06:00-14:00 UTC
US West (Oregon) Region	06:00-14:00 UTC
EU (Ireland) Region	22:00-06:00 UTC
Asia Pacific (Singapore) Region	14:00-22:00 UTC
Asia Pacific (Tokyo) Region	17:00-03:00 UTC
South America (São Paulo) Region	00:00-08:00 UTC

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



**Note**

Automated backups are enabled by default for a new DB Instance.

For more information on working with automated backups, go to [Working With Automated Backups \(p. 130\)](#).

## 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 \(p. 133\)](#).



**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 [Multi-AZ Deployments \(p. 8\)](#).

## DB Snapshots

DB Snapshots are user-initiated backups of a DB Instance. DB Snapshots are retained until they are deleted by the user. For more information on working with DB Snapshots, see [Creating a DB Snapshot \(p. 126\)](#) and [Restoring From a DB Snapshot \(p. 128\)](#).

## Automated Backups with Unsupported Storage Engines

Amazon RDS automated backups and DB Snapshots are currently supported for only the InnoDB storage engine. 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

1	Stop all activity to your MyISAM tables (that is, close all sessions)
2	Lock and flush each of your MyISAM tables

3	Issue a <code>CreateDBSnapshot</code> API call, or use the RDSCLI <code>rds-create-db-snapshot</code> command. 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.
---	---

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;`).

## Related Topics

- [Creating a DB Snapshot \(p. 126\)](#)
- [Restoring From a DB Snapshot \(p. 128\)](#)
- [Working With Automated Backups \(p. 130\)](#)

## DB Parameter Groups

Amazon RDS allows you to control the database 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 particular DB Parameter Group name. This default group contains database engine defaults and Amazon RDS system defaults based on the engine, compute class, and allocated storage of the instance. 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. Once associated, all DB Instances that use a particular DB Parameter Group get all parameter updates to that DB Parameter Group.



### Note

Some database engine parameters are constrained or disabled in the context of an RDS DB Instance. For more information, please see [Engine-Specific Parameter Exceptions for RDS DB Instances \(p. 20\)](#).

## Related Topics

- [Working with DB Parameter Groups \(p. 136\)](#)

## DB Instance Monitoring

Amazon RDS collects performance information for all DB Instances. Using the Amazon CloudWatch web service APIs, you can access CPU, storage, and database connections metrics.

For a complete list of Amazon RDS metrics, go to [Amazon RDS Dimensions and Metrics](#) in the [Amazon CloudWatch Developer Guide](#).

## Related Topics

- [Viewing DB Instance Metrics \(p. 156\)](#)

## Amazon RDS Events

Amazon RDS logs 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 from the log using the `rds-describe-events` command or the `DescribeEvents` API.

### Related Topics

- [Viewing Amazon RDS Events \(p. 159\)](#)

## AWS Identity and Access Management

Amazon Relational Database Service integrates with AWS Identity and Access Management (IAM), a service that 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

For example, you can use IAM with Amazon RDS to control which Users in your AWS Account can create or modify DB Instances for your AWS Account.

For general information about IAM, go to:

- [Identity and Access Management \(IAM\)](#)
- [AWS Identity and Access Management Getting Started Guide](#)
- [Using AWS Identity and Access Management](#)

For specific information about how you can control User access to Amazon Relational Database Service, go to [Integrating with Other AWS Products](#) in *Using AWS Identity and Access Management*.

## Amazon RDS and Amazon Virtual Private Cloud (VPC)



### Important

You should have an understanding of how Amazon VPC works before reading this section. Please refer to the [Amazon Virtual Private Cloud documentation](#) for detailed information about Amazon VPC.

Amazon Virtual Private Cloud enables you to create a virtual network in the AWS cloud. With a Virtual Private Cloud (VPC), you can define a virtual network that closely resembles a traditional data center. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of routing and access control lists.

The basic functionality of Amazon RDS is the same when using Amazon VPC; Amazon RDS manages backups, software patching, automatic failure detection and recovery whether your DB Instances are deployed inside or outside a VPC.



#### Note

Read replicas and multi-AZ deployments are not supported for DB Instances running in a VPC.

Amazon RDS DB Instances deployed outside a VPC are assigned an external IP address (to which the Endpoint/DNS Name resolves) that provides connectivity from EC2 or the Internet. In Amazon VPC, Amazon RDS DB instances only have a private IP address (within a subnet that you define), and a random IP address from that subnet is chosen and assigned to the DB Instance's DNS name.

## Prerequisites

Following are the pre-requisites necessary to create a DB Instances within a VPC:

- You need to have a VPC set up with at least one subnet created in every Availability Zone in the Region you want to deploy your DB Instance.
- You need to have a DB Subnet Group defined for your VPC.
- You need to have a DB Security Group defined for your VPC (or you can use the default provided).
- You should allocate adequately large CIDR blocks to each of your subnets so that there are spare IP addresses for Amazon RDS to use during maintenance activities, including failover and compute scaling.

When creating a DB Instance in VPC, you will need to create or 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.

## DB Subnet Groups

A DB Subnet Group for a VPC is a collection of subnets (typically private) that you may want to designate for your backend RDS DB Instances. Each DB Subnet Group should have at least one subnet for every Availability Zone in a given Region. When creating a DB Instance in VPC, you will need to 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.

You must specify a subnet for every Availability Zone while defining a DB Subnet Group. In case 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.

While creating a DB Instance in a VPC, Amazon RDS associates an Elastic Network Interface to your DB Instance using the IP address selected from your DB Subnet Group. However, we strongly recommend you use the DNS Name to connect to your DB Instance as the underlying IP address can change during failover.

## Levels of Privacy

When you create a VPC, you can configure it based on the level of privacy you want. In the most private scenario, you can attach only a *virtual private gateway*, and create an IPsec tunnel between your VPC and home network. In this scenario, your EC2 instances have no direct exposure to the Internet.

You can configure your VPC to be somewhere in between, with both a virtual private gateway and an Internet gateway. Here, some instances could receive Internet traffic (e.g., web servers), whereas others could remain unexposed (e.g., database servers). This is a common scenario for running a multi-tier web application in the AWS cloud.

These different scenarios are discussed in more detail in the Amazon VPC documentation.

## Routing and Security

You can configure routing in your VPC to control where traffic flows (e.g., to the Internet gateway, virtual private gateway, etc). With an Internet gateway, your VPC has direct access to other AWS products such as Amazon Simple Storage Service (Amazon S3). If you choose to have only a virtual private gateway with a connection to your home network, you can route your Internet-bound traffic over the VPN and control its egress with your security policies and corporate firewall. In the latter case, you incur additional bandwidth charges when accessing AWS products over the Internet.

You can use *DB Security Groups*, *network ACLs*, and *VPC Security Groups* to help secure the instances in your VPC. Security groups act like a firewall at the instance level, whereas network ACLs are an additional layer of security that act at the subnet level.



### Note

If you associate a VPC to a DB Security Group, all the access rules within that DB Security Group should be either from VPC Security Groups or IP ranges. EC2 Security Groups and VPC Security Groups are not interchangeable.

DB Instances deployed within an Amazon VPC can be accessed by Amazon EC2 Instances deployed in the same VPC. If these EC2 Instances are deployed in a public subnet with associated Elastic IPs, you can access the EC2 Instances via the internet.



### Note

We strongly recommend you use the DNS Name to connect to your DB Instance as the underlying IP address can change during failovers.

DB Instances deployed within a VPC can be accessed from the Internet or from EC2 Instances outside the VPC via *bastion hosts* that you can launch in your public subnet or via VPN. You will need to set up a public subnet with an EC2 instance that acts as a SSH Bastion. This public subnet must have an internet gateway and routing rules that allow traffic to be directed via the SSH host, which must then forward requests to the private IP address of your RDS DB instance. You can also set up a VPN Gateway that extends your corporate network into your VPC, and allows access to the RDS DB instance in that VPC.



### Important

Please refer to the [Amazon Virtual Private Cloud documentation](#) for detailed information about Amazon VPC.

For more information on using Amazon RDS with Amazon Virtual Private Cloud, go to [Using Amazon RDS with Amazon Virtual Private Cloud \(VPC\)](#) (p. 99).

## Amazon VPC Documentation

Amazon VPC has its own set of documentation to describe how to create and use your VPC. The following table gives links to the Amazon VPC guides.

<b>Description</b>	<b>Documentation</b>
How to get started using Amazon VPC	<a href="#">Amazon Virtual Private Cloud Getting Started Guide</a>
How to use Amazon VPC through the AWS Management Console	<a href="#">Amazon Virtual Private Cloud User Guide</a>
Complete descriptions of all the Amazon VPC commands	<a href="#">Amazon Elastic Compute Cloud Command Line Reference</a> (the Amazon VPC commands are part of the Amazon EC2 reference)
Complete descriptions of the Amazon VPC API actions, data types, and errors	<a href="#">Amazon Elastic Compute Cloud API Reference</a> (the Amazon VPC API actions are part of the Amazon EC2 reference)
Information for the network administrator who needs to configure the gateway at your end of an optional IPsec VPN connection	<a href="#">Amazon Virtual Private Cloud Network Administrator Guide</a>

# Amazon RDS Database Engines

---

This section covers some special considerations when running specific database engines with the Amazon RDS service.

## Topics

- [MySQL Database Engine \(p. 20\)](#)
- [Oracle Database Engine \(p. 23\)](#)
- [Microsoft SQL Server Database Engine \(p. 26\)](#)

# MySQL Database Engine

## General Information

### Engine-Specific Parameter Exceptions for 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.

## SSL Support

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:

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



#### **Note**

For more information on SSL connections with MySQL, go to the [MySQL documentation](#).

## Security

Managing users and privileges in DB Instances works the same way as in stand-alone database instances. Commands such as `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE`, and `SET PASSWORD` work just as they do in stand-alone databases, as does directly modifying database schema tables.

When you create a database 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
- select
- show databases
- show view
- trigger
- update



#### Note

Although it is possible to delete the master user on the DB Instance, it is not recommended. To recreate the master user, use the `ModifyDBInstance` 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 RDS commands `rds_kill` and `rds_kill_query` are provided to allow you to terminate user sessions or queries on DB Instances.

## DB Engine Version Management

DB Engine Version Management is a feature of Amazon RDS that enables you to control when and how the database engine software running your DB Instances is patched and upgraded. This feature gives you the flexibility to maintain compatibility with specific MySQL 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.

Taking advantage of the DB Engine Version Management feature of Amazon RDS is easily accomplished using the **ModifyDBInstance** API call, **rds-modify-db-instance** command line utility, or the AWS

Management Console. Your DB Instances are upgraded to minor patches by default (you can override this setting).



**Note**

Automatic patching works only for minor upgrades; for example, from version 5.1.42 to 5.1.45. Automatic patching will not work for major upgrades (for example, from version 5.1 to 5.5).

# Oracle Database Engine

This section discusses specific details for implementing the Oracle database engine on Amazon RDS.

## General Information

### Common DBA Tasks

In order to provide you with a secure and stable managed database 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 provides wrapper procedures for many common DBA tasks that require advanced privileges. For more information, see [Appendix: Common DBA Tasks for Oracle \(p. 196\)](#).

## Engine Features

The following list shows a subset of the key Oracle database engine features that are currently supported by Amazon RDS. For a complete list of features supported by Oracle database engine, 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
- Advanced Queuing
- Distributed Queries/Transactions
- Text
- Materialized Views
- Import/Export and sqlldr Support

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

- Real Application Clusters (RAC)
- Data Guard / Active Data Guard
- Oracle Enterprise Manager
- Automated Storage Management
- Data Pump
- Streams

- Oracle Application Express
- XML DB
- Java Support
- Locator
- Spatial

## 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

## DB Engine Version Management

DB Engine Version Management is a feature of Amazon RDS that enables you to control when and how the database engine software running your DB Instances is patched and upgraded. This feature gives you the flexibility to maintain compatibility with database engine patch versions, test new patch versions to ensure they work effectively with your application before deploying in production, and perform version upgrades on your own terms and timelines.



### Note

Amazon RDS periodically aggregates official Oracle database patches using an Amazon RDS-specific DB Engine version. To see a list of which Oracle patches are contained in an Amazon RDS Oracle-specific engine version, go to [Appendix: Oracle DB Engine Patch Composition](#) (p. 208) or see the [Amazon RDS Technical FAQ](#) (p. 161).

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 a major upgrade be applied to your database.

## 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.

# Microsoft SQL Server Database Engine

This section discusses specific details and limitations for Amazon RDS DB Instances running the Microsoft SQL Server database engine.

## General Information

### Common DBA Tasks

In order to provide you with a secure and stable managed database 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 provides wrapper procedures for many common DBA tasks that require advanced privileges. For more information, see [Appendix: Common DBA Tasks for Microsoft SQL Server \(p. 196\)](#).

## Engine Features

The following list shows a subset of the key SQL Server database engine features that are currently supported by Amazon RDS. For a complete list of features supported by the SQL Server database engine, go to [Features Supported by the Editions of SQL Server 2012](#).

- 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)
- Safe CLR
- Full-text search
- Spatial and location features

Microsoft SQL Server database features that are not currently supported include the following:

- SQL Server Agent
- Maintenance Plans
- Database Tuning Advisor
- Database Mail
- SSL
- Transparent Data Encryption
- 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)
- Distributed Queries
- WCF Data Services
- FILESTREAM support
- Policy-Based Management
- SQL Server Audit

Currently, Amazon RDS supports SQL Server 2008 R2. The following database editions are available:

- License included at no additional cost: SQL Server Express Edition, SQL Server Web Edition, and SQL Server Standard Edition



**Note**

In accordance with Microsoft 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](#).

- Bring your own license (license mobility): SQL Server Standard Edition and SQL Server Enterprise Edition

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 250 GB for the Standard and Enterprise Editions.

The maximum number of databases on a single Microsoft SQL Server DB Instance is 30.

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.

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.

## 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 \(p. 136\)](#).)
- 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

## DB Engine Version Management

Amazon RDS provides DB Engine Version Management so that you can deploy patches and upgrades to your DB Instances on your own terms and timelines. For example, you can maintain compatibility with database engine patch versions, and you can test new patches to ensure that they work with your application before you deploy them to your production environment.



### Note

Amazon RDS periodically aggregates official Microsoft SQL Server database patches and assigns an Amazon RDS-specific DB Engine version. The current supported version is cumulative update package 3 for SQL Server 2008 R2 Service Pack 1. For information about this page, go to [the corresponding Microsoft Knowledge Base article](#).

To take advantage of the DB Engine Version Management feature of Amazon RDS, you can use the **ModifyDBInstance** action or the **rds-modify-db-instance** command line utility. By default, your DB Instances are upgraded to minor patches. You can, however, override this setting. You can also explicitly specify that a major upgrade be applied to your database.

## 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.

# Signing up for Amazon RDS

---

To use the Amazon Relational Database Service, 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.

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> and click **Sign Up for Amazon RDS Now**.
2. Follow the on-screen instructions.

# Setting up the Command Line Tools

---

## Topics

- [Prerequisites \(p. 31\)](#)
- [Getting the Command Line Tools \(p. 32\)](#)
- [Setting Up the Tools \(p. 33\)](#)
- [Overriding the Default Region \(p. 34\)](#)
- [Providing Credentials for the Tools \(p. 34\)](#)

This section describes the prerequisites for running the command line tools, where to get the command line tools, how to set up the tools and their environment, and includes a series of common examples of tool usage.

## Prerequisites

This document assumes you can work in a Linux/UNIX or Windows environment. The Amazon RDS command line tools also work correctly on Mac OS X (which resembles the Linux and UNIX command environment), but no specific Mac OS X instructions are included in this guide.

As a convention, all command line text is prefixed with a generic `PROMPT>` command line prompt. The actual command line prompt on your machine is likely to be different. We also use `$` to indicate a Linux/UNIX specific command and `C:\>` for a Windows specific command. The example output resulting from the command is shown immediately thereafter without any prefix.

## The Java Runtime Environment

The command line tools used in this guide require Java version 5 or later to run. Either a JRE or JDK installation is acceptable. To view and download JREs for a range of platforms, including Linux/UNIX and Windows, go to <http://java.sun.com/j2se/1.5.0/>.

## Setting the Java Home Variable

The command line tools depend on an environment variable (`JAVA_HOME`) to locate the Java Runtime. This environment variable should be set to the full path of the directory that contains a sub directory named `bin` which in turn contains the executable `java` (on Linux and UNIX) or `java.exe` (on Windows) executable.

### To set the Java Home variable

1. Set the Java Home variable.
  - On Linux and UNIX, using the following command:

```
$ export JAVA_HOME=<PATH>
```

- On Windows, using the following command:

```
C:\> set JAVA_HOME=<PATH>
```

2. Confirm the path setting by running `$JAVA_HOME/bin/java -version` and checking the output.

- On Linux/UNIX, you will see output similar to the following:

```
$ $JAVA_HOME/bin/java -version
java version "1.5.0_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03)
Java HotSpot(TM) Client VM (build 1.5.0_09-b03, mixed mode, sharing)
```

- On Windows, you will see output similar to the following:

```
C:\> %JAVA_HOME%\bin\java -version
java version "1.5.0_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03)
Java HotSpot(TM) Client VM (build 1.5.0_09-b03, mixed mode, sharing)
```

## Getting the Command Line Tools

The command line tools are available as a ZIP file on the [Amazon RDS web site](#). These tools are written in Java, and include shell scripts for Windows 2000/XP/Vista, Linux/UNIX, and Mac OSX. The ZIP file is self-contained and no installation is required; simply download the zip file and unzip it to a directory on your local machine.

## Setting Up the Tools

The command line tools depend on an environment variable (`AWS_RDS_HOME`) to locate supporting libraries. You need to set this environment variable before you can use the tools. Set it to the path of the directory you unzipped the command line tools into. This directory is named `RDSCli-A.B.nnnn` (A, B and n are version/release numbers), and contains sub-directories named `bin` and `lib`.

### To set the `AWS_RDS_HOME` environment variable

- Open a command line window and enter one of the following commands to set the `AWS_RDS_HOME` environment variable.
  - On Linux and UNIX, enter the following command:

```
$ export AWS_RDS_HOME=<path-to-tools>
```

- On Windows, enter the following command:

```
C:\> set AWS_RDS_HOME=<path-to-tools>
```

To make the tools easier to use, we recommend you add the tools' BIN directory to your system PATH. The rest of this guide assumes the BIN directory is in your system path.

### To add the tools' BIN directory to your system path

- Enter the following commands to add the tools' BIN directory to your system PATH.
  - On Linux and UNIX, enter the following command:

```
$ export PATH=$PATH:$AWS_RDS_HOME/bin
```

- On Windows, enter the following command:

```
C:\> set PATH=%PATH%;%AWS_RDS_HOME%\bin
```



#### Note

The Windows environment variables are reset when you close the command window. You might want to set them permanently. Consult the documentation for your version of Windows for more information.



#### Note

Paths that contain a space must be wrapped in double quotes, for example:

```
"C:\Program Files\Java"
```

## Overriding the Default Region

By default, Amazon RDS uses the `us-east-1` region when you create DB Instances and other Amazon RDS objects. To temporarily specify a different region when entering an Amazon RDS command, you can use the `--url` or `--region` common command line options. For more information about common command line options, see the [Amazon RDS Command Line Reference](#).

To avoid having to pass the URL or region with each command, you can set the `EC2_REGION` environment variable to the appropriate region for your use.

### To override the default region

- The following example shows how to set the default region to `us-west-1`.
  - On Linux and UNIX, enter the following command:

```
$ export EC2_REGION=us-west-1
```

- On Windows, enter the following command:

```
C:\> set EC2_REGION=us-west-1
```

## Providing Credentials for the Tools

The command line tools need the AWS Access Key and Secret Access Key provided with your AWS account. You can get them using the command line or from a credential file located on your local system.

The deployment includes a template file `$(AWS_RDS_HOME)/credential-file-path.template` that you need to edit with your information. Following are the contents of the template file:

```
AWSAccessKeyId=<Write your AWS access ID>  
AWSSecretKey=<Write your AWS secret key>
```



### Important

On UNIX, limit permissions to the owner of the credential file:

```
$ chmod 600 <the file created above>
```

With the credentials file setup, you'll need to set the `AWS_CREDENTIAL_FILE` environment variable so that the Amazon RDS tools can find your information.

### To set the `AWS_CREDENTIAL_FILE` environment variable

1. Set the environment variable
  - On Linux and UNIX, update the variable using the following command:

```
$ export AWS_CREDENTIAL_FILE=<the file created above>
```

- On Windows, set the variable using the following command:

```
C:\> set AWS_CREDENTIAL_FILE=<the file created above>
```

2. Check that your setup works properly, run the following command:

```
rds --help
```

You should see the usage page for all Amazon RDS commands.

# Using the Amazon RDS API

---

## Topics

- [Controlling User Access to Your AWS Account](#) (p. 36)
- [Making API Requests](#) (p. 40)
- [Using the Query API](#) (p. 40)
- [Using the SOAP API](#) (p. 43)
- [Available Libraries](#) (p. 46)
- [Troubleshooting Applications](#) (p. 47)

## Controlling User Access to Your AWS Account

### Topics

- [Amazon RDS Group Security and IAM](#) (p. 37)
- [No Amazon RDS ARNs](#) (p. 38)
- [Amazon RDS Actions](#) (p. 38)
- [Amazon RDS Keys](#) (p. 38)
- [Example Policies for Amazon RDS](#) (p. 38)

Amazon RDS allows you to control access to your DB Instances using DB Security Groups. A DB Security Group acts like a firewall controlling network access to your DB Instance.



### Important

Amazon RDS uses database engine login names and DB Security Groups to control who has access to specific Amazon RDS DB Instances. There's no way in the IAM system to allow or deny access to a specific DB Instance.

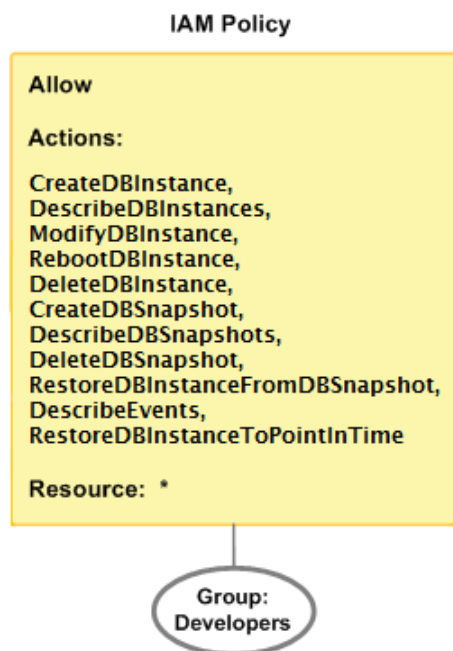
For more information about using Security Groups with Amazon RDS, refer to the [Amazon Relational Database Service Developer Guide](#) or the [Amazon Relational Database Service User Guide](#).

## Amazon RDS Group Security and IAM

Using IAM with Amazon RDS doesn't change how you use Amazon RDS DB Security Groups to grant access to DB Instances. However, you can use IAM policies to specify which Amazon RDS actions a User in your AWS Account can use with RDS resources in general. Because you can't specify a particular RDS DB Instance in the policy, you must specify \* as the resource to indicate all DB Instances in the AWS Account.

### Example 1

You could create a policy that gives the Developers group permission to use only these APIs: `CreateDBInstance`, `DescribeDBInstances`, `ModifyDBInstance`, `RebootDBInstance`, `DeleteDBInstance`, `CreateDBSnapshot`, `DescribeDBSnapshots`, `DeleteDBSnapshot`, `RestoreDBInstanceFromDBSnapshot`, `DescribeEvents`, and `RestoreDBInstanceToPointInTime`. They could then use those APIs with any DB Instance that belongs to your AWS Account. The following diagram illustrates the concept.



### Example 2

This example builds on the previous one. In addition to the IAM policy attached to the Developers group, you use the RDS command line interface to create a RDS Security Group that can access your RDS DB Instance only from a specified IP range. Anyone possessing the keys for your AWS Account who is accessing the Amazon RDS DB could modify the DB Instance. Also, because the Developers group has permission to use `ModifyDBInstance` with this DB Instance, any User in the Developers group can modify the DB Instance.

For examples of IAM policies that cover Amazon RDS actions, see [Example Policies for Amazon RDS](#) (p. 38).

## No Amazon RDS ARNs

Because you can't specify a particular Amazon RDS resource in an IAM policy, Amazon RDS has no ARNs. When writing a policy to control access to Amazon RDS actions, you use `*` as the resource. For more information about ARNs, go to [ARNs](#) in the [AWS Identity and Access Management \(IAM\) documentation](#).

## Amazon RDS Actions

In an IAM policy, you can specify any and all actions that Amazon RDS offers. Each action name must be prefixed with the lowercase string `rds:`. For example: `rds:ModifyDBInstance`, `rds:DescribeDBInstance`, or `rds:*` (for all Amazon RDS actions). For a list of the actions, refer to the Query API or SOAP API action names in the [Amazon Relational Database Service API Reference](#).

## Amazon RDS Keys

Amazon RDS implements the following policy keys, but no others. For more information about policy keys, go to [Condition](#) in the in the [AWS Identity and Access Management \(IAM\) documentation](#).

### AWS-Wide Policy Keys

- `aws:CurrentTime` (for date/time conditions)
- `aws:EpochTime` (the date in epoch or UNIX time, for use with date/time conditions)
- `aws:SecureTransport` (Boolean representing whether the request was sent using SSL)
- `aws:SourceIp` (the requester's IP address, for use with IP address conditions)
- `aws:UserAgent` (information about the requester's client application, for use with string conditions)

If you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, we evaluate the instance's public IP address to determine if access is allowed.

For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.

The key names are case insensitive. For example, `aws:CurrentTime` is equivalent to `AWS:currenttime`.

## Example Policies for Amazon RDS

This section shows a few simple policies for controlling User access to Amazon RDS.



### Note

In the future, Amazon RDS might add new actions that should logically be included in one of the following policies, based on the policy's stated goals.

### Example 1: Allow a Network Admin group to only be able to access the APIs related to Amazon RDS Security Groups

In this example, we create a policy that gives access to the relevant actions and attach it to the group. The resource is stated as "\*", because you can't specify a particular Amazon RDS resource in an IAM policy.

```
{
  "Statement": [ {
    "Effect": "Allow",
    "Action": [ "rds:CreateDBSecurityGroup",
               "rds>DeleteDBSecurityGroup",
               "rds:DescribeDBSecurityGroup",
               "rds:AuthorizeDBSecurityGroupIngress",
               "rds:RevokeDBSecurityGroupIngress" ],
    "Resource": "*"
  }
]
```

### Example 2: Allow managers to only be able to list the current Amazon RDS resources in the AWS Account

In this example, we create a policy that lets managers use the Amazon RDS actions with `Describe` in the name.

```
{
  "Statement": [ {
    "Effect": "Allow",
    "Action": "rds:Describe*",
    "Resource": "*"
  }
]
```

### Example 3: Allow a DBA to access a select set of Amazon RDS actions

In this example, we create a policy that gives access to the relevant actions for DBAs and attach it to the group. As with the other examples, the resource is stated as "\*", because you can't specify a particular Amazon RDS resource in an IAM policy.

```
{
  "Statement": [ {
    "Effect": "Allow",
    "Action": [ "rds:CreateDBSnapshot",
               "rds:DescribeDBSnapshots",
               "rds>DeleteDBSnapshot",
               "rds:ModifyDBInstance",
               "rds:RebootDBInstance",
               "rds:DescribeDBInstances",
               "rds:DescribeEvents",
               "rds:ModifyDBParameterGroup",
               "rds:DescribeDBParameterGroups",
               "rds:DescribeDBParameters",
               "rds:ResetDBParameterGroup",
               "rds:DescribeEngineDefaultParameters" ],
    "Resource": "*"
  }
]
```

## Making API Requests

### Endpoints

For information about this product's regions and endpoints, go to [Regions and Endpoints](#) in the Amazon Web Services General Reference.

If no endpoint is explicitly specified, the US-East (Northern Virginia) Region endpoint is used by default.

## Using the Query API

### Topics

- [Query Parameters](#) (p. 40)
- [Query Request Authentication](#) (p. 41)

### Query Parameters

HTTP Query-based requests are HTTP requests that use the HTTP verb GET or POST and a Query parameter named *Action*.

Each Query request must include some common parameters to handle authentication and selection of an action.

Some operations take lists of parameters. These lists are specified using the *param.n* notation. Values of *n* are integers starting from 1.

## Query Request Authentication

You can only send Query requests over HTTPS and you must include a signature in every Query request. This section describes how to create the signature. The method described in the following procedure is known as *signature version 2*.

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.



### 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

1	The sender constructs a request to AWS.
2	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.
3	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.
4	AWS uses the Access Key ID to look up the Secret Access Key.
5	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.
6	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.



### 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" +  
CanonicalizedQueryString <from the preceding step>
```

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  
&Timestamp=2010-05-10T17%3A09%3A03.726Z  
&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\n  
rds.amazonaws.com\n  
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  
  &Timestamp=2010-05-10T17%3A09%3A03.726Z  
  &Signature=<URLEncode(Base64Encode(Signature))>  
  &SignatureVersion=2  
  &SignatureMethod=HmacSHA256  
  &AWSAccessKeyId=<Your AWS Access Key ID>
```

## Using the SOAP API

### Topics

- [WSDL and Schema Definitions \(p. 43\)](#)
- [Programming Language Support \(p. 44\)](#)
- [Request Authentication \(p. 44\)](#)
- [The Response Structure \(p. 46\)](#)
- [Web Services References \(p. 46\)](#)

## 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 \(p. 46\)](#).



### 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:

Region	WSDL Location
US East (Northern Virginia) Region	<a href="https://rds.us-east-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl">https://rds.us-east-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl</a>

Region	WSDL Location
US West (Northern California) Region	<a href="https://rds.us-west-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl">https://rds.us-west-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl</a>
US West (Oregon) Region	<a href="https://rds.us-west-2.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl">https://rds.us-west-2.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl</a>
EU (Ireland) Region	<a href="https://rds.eu-west-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl">https://rds.eu-west-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl</a>
Asia Pacific (Singapore) Region	<a href="https://rds.ap-southeast-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl">https://rds.ap-southeast-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl</a>
Asia Pacific (Tokyo) Region	<a href="https://rds.ap-northeast-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl">https://rds.ap-northeast-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl</a>
South America (São Paulo) Region	<a href="https://rds.sa-east-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl">https://rds.sa-east-1.amazonaws.com/doc/2012-04-23/AmazonRDSv7.wsdl</a>

## 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:

```
<DescribeDBInstances>
  <MaxRecords>100<MaxRecords>
</DescribeDBInstances>
```

To secure the request, we add the `BinarySecurityToken` element.

The secure version of the request begins with the following:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <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:Expires>2009-10-28T18:46:59.597Z</wsu:Expires>
      </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"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
    ...many, many lines of base64 encoded X.509 certificate...
</wsse:BinarySecurityToken>
<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: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>DLFQyK6lqWoJiMyC9w34siRELAM=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#id-3">
      <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>gUnvvoUezxgt56eBl2kW/y5diMk=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>OMoJJqqDnahRt/9H2n8obJolyVprpziAzlFRZ9KbdwX
JoD1Rl2sAikZ0IJW7/Vs9q8GH4JDsT2v1
UoUogKgRSWy3sU4943g1T0vhyigbUm4vNxE/qUKm
SIXx2ed/8buaF9oRib8zYDu0/qRT+QQ73rdaoyN2YRNkSi2+6P2FHmE=
</ds:SignatureValue>
  <ds:KeyInfo Id="KeyId-5992FC58FDECA60AF912567553195672">
    <wsse:SecurityTokenReference
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
      wsu:Id="STRId-5992FC58FDECA60AF912567553195703"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <wsse:Reference URI="#CertId-5992FC58FDECA60AF912567553195531"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
          xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" />
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </ds:Signature>
```

```
</wsse:Security>  
</soap:Header>
```

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:

```
<DescribeDBInstancesResponse xmlns="http://rds.amazonaws.com/admin/2009-10-16/">  
  <DescribeDBInstancesResult>  
    <DBInstances/>  
  </DescribeDBInstancesResult>  
  <ResponseMetadata>  
    <RequestId>946cda70-c3f1-11de-807a-79c03c55f7d4</RequestId>  
  </ResponseMetadata>  
</DescribeDBInstancesResponse>
```

## 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 \(p. 47\)](#)
- [Troubleshooting Tips \(p. 47\)](#)

Amazon Relational Database Service provides specific and descriptive errors to help you troubleshoot problems while interacting with the Amazon RDS API.

## Retrieving Errors

Typically, you want your application to check whether a request generated an error before you spend any time processing results. The easiest way to find out if an error occurred is to look for an *Error* node in the response from the Amazon RDS API.

XPath syntax provides a simple way to search for the presence of an *Error* node, as well as an easy way to retrieve the error code and message. The following code snippet uses Perl and the XML::XPath module to determine if an error occurred during a request. If an error occurred, the code prints the first error code and message in the response.

```
use XML::XPath;

my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error
code: ",
    $xp->findvalue("//Error[1]/Code"), "\n", " ",
    $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

## Troubleshooting Tips

We recommend the following processes to diagnose and resolve problems with the Amazon 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>.
- 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 <http://developer.amazonwebservices.com/connect/forum.jspa?forumID=60>

# Using Amazon RDS

---

## Topics

- [Creating and Modifying DB Instances \(p. 49\)](#)
- [Importing Data to a DB Instance \(p. 117\)](#)
- [Backing Up and Restoring DB Instances \(p. 125\)](#)
- [DB Parameter Groups \(p. 135\)](#)
- [DB Security Groups \(p. 145\)](#)
- [Monitoring DB Instances \(p. 155\)](#)
- [Working with Amazon RDS Events \(p. 158\)](#)

This section covers the Amazon RDS operations you are most likely to use, and provides procedural instruction and examples.

## Creating and Modifying DB Instances

The following scenarios cover basic operations on DB Instances.

### Topics

- [Creating a DB Instance Running the MySQL Database Engine \(p. 50\)](#)
- [Creating a DB Instance Running the Oracle Database Engine \(p. 54\)](#)
- [Creating a DB Instance Running the Microsoft SQL Server Database Engine \(p. 60\)](#)
- [Connecting to a DB Instance Running the MySQL Database Engine \(p. 64\)](#)
- [Connecting to a DB Instance Running the Oracle Database Engine \(p. 66\)](#)
- [Connecting to a DB Instance Running the Microsoft SQL Server Database Engine \(p. 68\)](#)
- [Rebooting a DB Instance \(p. 73\)](#)
- [Modifying a DB Instance \(p. 74\)](#)
- [Deleting a DB Instance \(p. 77\)](#)
- [Sizing Your Amazon RDS DB Instance \(p. 80\)](#)
- [Scaling CPU and Memory for a DB Instance \(p. 82\)](#)
- [Scaling DB Instance Storage \(p. 84\)](#)
- [Adjusting the Preferred Maintenance Window \(p. 87\)](#)
- [Working with Reserved DB Instances \(p. 90\)](#)
- [Using Amazon RDS with Amazon Virtual Private Cloud \(VPC\) \(p. 99\)](#)

## Creating a DB Instance Running the MySQL Database Engine

When you create a new DB Instance, you need to name it, set the size, determine how long you want to store backups, and give yourself a login and password.

In this example, you create a DB Instance running the MySQL database engine called *mydbinstance*, with a *Small* DB Instance class, 20 GB of storage, and automated backups enabled with a retention period of 3 days.



### Note

For information on accessing a DB Instance, refer to the [Amazon Relational Database Service Getting Started Guide](#).

## AWS Management Console

### To launch a MySQL DB Instance

1. Start the launch wizard:
  - a. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
  - b. From the Amazon RDS Console Dashboard, click **Launch DB Instance** to start the Launch RDS DB Instance Wizard.

The wizard opens on the **Engine Selection** page.

2. Click the **Select** button next to the MySQL database engine.

The wizard continues to the **DB Instance Details** page. The first page of the wizard displays a list of DB Instance Classes in the **DB Instance Class** drop-down list. The DB Instance class defines the CPU and memory capacity of your DB Instance.

3. On the **DB Instance Details** page, specify your DB Instance details as shown in the following table, then click **Continue**.

For this parameter...	...Do this:
<b>License Model</b>	Keep the default: <b>General Public License</b> . This is the only available option for DB Instances running the MySQL database engine.
<b>DB Engine Version</b>	Select <b>5.1.57 (default)</b> .
<b>DB Instance Class</b>	Select <b>db.m1.small</b> .
<b>Multi-AZ Deployment</b>	Select <b>No</b> .
<b>Auto Minor Version Upgrade</b>	Keep the default setting of <b>yes</b> for this example. The Auto Minor Version Upgrade option enables your DB Instance to receive minor engine version upgrades automatically when they become available.

**Amazon Relational Database Service User Guide**  
**Creating a DB Instance Running the MySQL Database Engine**

---

For this parameter...	...Do this:
<b>Allocated Storage</b>	You can specify how much storage in gigabytes you want initially allocated for your DB Instance. For this example, type 20.
<b>DB Instance Identifier</b>	The DB Instance is a name for your DB Instance that is unique for your account in a Region. Type <code>mydbinstance</code> in the <b>DB Instance Identifier</b> text box.
<b>Master Username</b>	Type a name for your master user in the <b>Master Username</b> text box. You use the master user name to log on to your DB Instance with all database privileges.
<b>Master Password</b>	Type a password for your master user in the <b>Master User Password</b> text box.



**Important**

You must specify a password containing from 8 to 16 alphanumeric characters only.

After you click the **Continue** button, the **Additional Configuration** page opens.

4. Provide additional configuration information for your DB Instance:
  - a. Type `mydatabase` into the **Database Name** text box.

When you're creating a DB Instance running the MySQL database engine, you provide a database name so that Amazon RDS will create a default database on your new DB Instance. If you skip this step, Amazon RDS will not create a database on your DB Instance.
  - b. Accept the default values for the rest of the parameters available on this page, and then click the **Continue** button.

After you click the **Continue** button, the **Management Options** page appears. The **Management Options** panel is where you can specify backup and maintenance options for your DB Instance.
5. For this example, accept the default values, and then click **Continue**.

After you click the **Continue** button, the **Review** panel appears.
6. Review the options for your DB Instance:
  - If you need to correct any options, click the **Back** to return to previous panels and make corrections.
  - If all your options are entered correctly, click the **Launch DB Instance** button to launch your new DB Instance.

After you click the **Launch DB Instance** button, a message displays stating that your DB Instance is being created.

This can take a few minutes to complete.
7. Click the **Close** button.

After you click the **Close** button, the **My DB Instances** panel appears. Your DB Instance appears in the list on this page with the **creating** status until your DB Instance is created and ready for use.

Once your DB instance changes to the **available** state, you need to authorize access so you can connect to it.

## CLI

### To create a MySQL DB Instance

- Use the command `rds-create-db-instance` to create a DB Instance.

```
PROMPT>rds-create-db-instance mydbinstance -s 20 -c db.m1.small -e MySQL  
- u sa -p secretpassword --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 `CreateDBInstance` with the following parameters:
  - *DBInstanceIdentifier* = `mydbinstance`
  - *DBInstanceClass* = `db.m1.small`
  - *AllocatedStorage* = `20`
  - *BackupRetentionPeriod* = `3`
  - *MasterUsername* = `sa`
  - *MasterUserPassword* = `secretpassword`

## Example

```
https://rds.amazonaws.com/  
  ?Action=CreateDBInstance  
  &DBInstanceIdentifier=mydbinstance  
  &DBInstanceClass=db.m1.small  
  &Engine=mysql  
  &MasterUserPassword=secretpassword  
  &BackupRetentionPeriod=3  
  &AllocatedStorage=20  
  &MasterUsername=sa  
  &Version=2010-06-28  
  &SignatureVersion=2  
  &SignatureMethod=HmacSHA256  
  &Timestamp=2010-08-13T19%3A36%3A35.512Z  
  &AWSAccessKeyId=<AWS Access Key ID>  
  &Signature=<Signature>
```

## Related Topics

- [DB Instance \(p. 5\)](#)
- [DB Security Groups \(p. 11\)](#)
- [Scaling CPU and Memory for a DB Instance \(p. 82\)](#)
- [Scaling DB Instance Storage \(p. 84\)](#)
- [Deleting a DB Instance \(p. 77\)](#)

## Creating a DB Instance Running the Oracle Database Engine

When you create a new DB Instance, you need to name it, set the size, determine how long you want to store backups, and give yourself a login and password.

In this example, you create a DB Instance running the Oracle database engine called *mydbinstance*, with a *Small* DB Instance class, 20 GB of storage, and automated backups enabled with a retention period of 3 days.



### Note

For information on accessing a DB Instance, refer to the [Amazon Relational Database Service Getting Started Guide](#).

## AWS Management Console

### To launch an Oracle DB Instance

1. Start the launch wizard:
  - a. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
  - b. From the Amazon RDS Console, click **Launch DB Instance**.
2. In the **Launch DB Instance Wizard**, on the **Engine Selection** page, beside the Oracle edition that you want, click **Select**.
3. On the **DB Instance Details** page, specify your DB Instance details as shown in the following table, then click **Continue**.

For this setting ...	... do this:
<b>License Model</b>	In the list provided, click the license model that you want to use. For this example, click <b>License Included</b> . For information about license models, see <a href="#">the section called "Licensing"</a> (p. 24).
<b>DB Engine Version</b>	In the list provided, click the version of the Oracle database engine that you want to use. For this example, click <b>11.2.0.2.v3 (default)</b> .
<b>DB Instance Class</b>	In the list provided, click the DB Instance class that you want to use. For this example, click <b>db.m1.small</b> . For information about instance classes, see <a href="#">the section called "DB Instance Class"</a> (p. 6).
<b>Multi-AZ Deployment</b>	If you want to deploy your DB Instance in multiple Availability Zones, click <b>yes</b> . For this example, Click <b>no</b> .
<b>Auto Minor Version Upgrade</b>	If you want your DB Instance to receive minor engine version upgrades automatically when they become available, click <b>yes</b> . For this example, accept the default setting of <b>yes</b> . Upgrades are installed only during your scheduled maintenance window.

**Amazon Relational Database Service User Guide**  
**Creating a DB Instance Running the Oracle Database**  
**Engine**

For this setting ...	... do this:
<b>Allocated Storage</b>	Specify how much storage, in gigabytes, will be initially allocated for your DB Instance. For this example, type 20.
<b>DB Instance Identifier</b>	Specify a name for your DB Instance that is unique for your account in a Region. For this example, type <code>mydbinstance</code> .
<b>Master User Name</b>	Type a name that you will use to log in to your DB Instance with all database privileges.
<b>Master User Password</b>	Type a password for your master user. The password must contain from 8 to 30 alphanumeric characters.

When all the settings are as you want them, click **Continue**.

4. On the **Additional Configuration** page, provide additional configuration information for your DB Instance:

For this setting ...	... do this:
<b>Database Name</b>	Type a name for the Oracle System ID (SID) for the DB Instance. The default value is <code>ORACL</code> . For this example, type <code>ORACLST</code> .
<b>Database Port</b>	Enter the port number that the DB Instance will open to communicate with clients. The default is 1521.
<b>Availability Zone</b>	In the list provided, click the Availability Zone where you want your DB Instance to be deployed. If you would prefer that Amazon RDS select an Availability Zone for you, accept the default value of <b>No Preference</b> .
<b>Character Set Name</b>	In the list provided, click the character set that you would like to use. The default is <code>AL32UTF8</code> , which is the Unicode 5.0 UTF-8 Universal character set. For information about character sets, see <a href="#">Appendix: Oracle Character Sets Supported in Amazon RDS</a> (p. 206).
<b>DB Parameter Groups</b>	If you have a custom DB parameter group and you want to associate it with this DB Instance, in the list provided, click the DB parameter group that you want.
<b>DB Security Groups</b>	If you have a custom DB security group and you want to associate it with this DB Instance, in the list provided, click the DB security group that you want.

When all the settings are as you want them, click **Continue**.

5. On the **Management Options** page, set the following options:

For this setting ...	... do this:
<b>Backup Retention Period</b>	Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0.

**Amazon Relational Database Service User Guide**  
**Creating a DB Instance Running the Oracle Database**  
**Engine**

For this setting ...	... do this:																
<b>Backup Window</b>	<p>Set the time range during which automated backups of your databases will occur. To accept the default backup window for your region, accept the default value, <b>No Preference</b>. To set a custom backup window, click <b>Select Window</b>, and then specify a start time in Universal Coordinated Time (UTC) and a duration in hours. Default backup windows are as follows:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Region</th> <th style="text-align: left;">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 (Singapore) Region</td> <td>14:00-22:00 UTC</td> </tr> <tr> <td>Asia Pacific (Tokyo) Region</td> <td>17:00-03:00 UTC</td> </tr> <tr> <td>South America (São Paulo) Region</td> <td>00:00-08:00 UTC</td> </tr> </tbody> </table>	Region	Time Block	US East (Northern Virginia) Region	03:00-11:00 UTC	US West (Northern California) Region	06:00-14:00 UTC	US West (Oregon) Region	06:00-14:00 UTC	EU (Ireland) Region	22:00-06:00 UTC	Asia Pacific (Singapore) Region	14:00-22:00 UTC	Asia Pacific (Tokyo) Region	17:00-03:00 UTC	South America (São Paulo) Region	00:00-08:00 UTC
Region	Time Block																
US East (Northern Virginia) Region	03:00-11:00 UTC																
US West (Northern California) Region	06:00-14:00 UTC																
US West (Oregon) Region	06:00-14:00 UTC																
EU (Ireland) Region	22:00-06:00 UTC																
Asia Pacific (Singapore) Region	14:00-22:00 UTC																
Asia Pacific (Tokyo) Region	17:00-03:00 UTC																
South America (São Paulo) Region	00:00-08:00 UTC																

**Amazon Relational Database Service User Guide**  
**Creating a DB Instance Running the Oracle Database**  
**Engine**

For this setting ...	... do this:																
<b>Maintenance Window</b>	<p>Set the time range during which system maintenance, including upgrades, will occur. To accept the default maintenance window for your region, accept the default value, <b>No Preference</b>. To set a custom backup window, click <b>Select Window</b>, and then specify a start time in UTC and a duration in hours. Default maintenance windows are as follows:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Region</th> <th style="text-align: left;">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 (Singapore) Region</td> <td>14:00-22:00 UTC</td> </tr> <tr> <td>Asia Pacific (Tokyo) Region</td> <td>17:00-03:00 UTC</td> </tr> <tr> <td>South America (São Paulo) Region</td> <td>00:00-08:00 UTC</td> </tr> </tbody> </table>	Region	Time Block	US East (Northern Virginia) Region	03:00-11:00 UTC	US West (Northern California) Region	06:00-14:00 UTC	US West (Oregon) Region	06:00-14:00 UTC	EU (Ireland) Region	22:00-06:00 UTC	Asia Pacific (Singapore) Region	14:00-22:00 UTC	Asia Pacific (Tokyo) Region	17:00-03:00 UTC	South America (São Paulo) Region	00:00-08:00 UTC
Region	Time Block																
US East (Northern Virginia) Region	03:00-11:00 UTC																
US West (Northern California) Region	06:00-14:00 UTC																
US West (Oregon) Region	06:00-14:00 UTC																
EU (Ireland) Region	22:00-06:00 UTC																
Asia Pacific (Singapore) Region	14:00-22:00 UTC																
Asia Pacific (Tokyo) Region	17:00-03:00 UTC																
South America (São Paulo) Region	00:00-08:00 UTC																

When all the settings are as you want them, click **Continue**.

6. 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**. Launching your DB Instance may take several minutes.
7. On the final page of the wizard, click **Close**.
8. You can monitor the progress of your DB Instance from the Amazon RDS console. When the state changes to **available**, you will need to authorize access in order to connect to it.

## 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.

```
PROMPT>rds-create-db-instance mydbinstance -s 20 -c db.m1.small -e oracle-  
sel  
- u sa -p secretpassword --backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE mydbinstance db.m1.small oracle-sel 20 sa creating 3 ****  
n 11.2.0.2.v3  
SECGROUP default active  
PARAMGRP default.oracle-sel-11.2 in-sync
```

## API

### To create an Oracle DB Instance

- Call `CreateDBInstance`. Calling `CreateDBInstance` with the following parameters will launch the example DB Instance:
  - `DBInstanceIdentifier` = `mydbinstance`
  - `Engine` = `oracle-sel`
  - `DBInstanceClass` = `db.m1.small`
  - `AllocatedStorage` = `20`
  - `BackupRetentionPeriod` = `3`
  - `MasterUsername` = `sa`
  - `MasterUserPassword` = `secretpassword`

### Example

```
https://rds.amazonaws.com/  
  ?Action=CreateDBInstance  
  &DBInstanceIdentifier=mydbinstance  
  &DBInstanceClass=db.m1.small  
  &Engine=oracle-se1  
  &MasterUserPassword=secretpassword  
  &BackupRetentionPeriod=3  
  &AllocatedStorage=20  
  &MasterUsername=sa  
  &Version=2012-01-15  
  &SignatureVersion=2  
  &SignatureMethod=HmacSHA256  
  &Timestamp=2011-08-13T19%3A36%3A35.512Z  
  &AWSAccessKeyId=<AWS Access Key ID>  
  &Signature=<Signature>
```

## Related Topics

- [DB Instance \(p. 5\)](#)
- [DB Security Groups \(p. 11\)](#)
- [Scaling CPU and Memory for a DB Instance \(p. 82\)](#)
- [Scaling DB Instance Storage \(p. 84\)](#)
- [Deleting a DB Instance \(p. 77\)](#)

# Creating a DB Instance Running the Microsoft SQL Server Database Engine

When you create a new DB Instance, you need to name it, set the size, determine how long you want to store backups, and give yourself a login and password.

In this example, you create a DB Instance running the Microsoft SQL Server database engine called *mymsftsqlserver*, with a *db.m1.large* DB Instance class, 250 GB of storage, and automated backups enabled with a retention period of 3 days.



## Note

For information on accessing a DB Instance, refer to the [Amazon Relational Database Service Getting Started Guide](#).

## AWS Management Console

To create a DB Instance Running the Microsoft SQL Server Database Engine

### To launch an instance

1. Start the launch wizard:
  - a. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
  - b. From the Amazon RDS Console, click **Launch DB Instance** to start the Launch RDS DB Instance Wizard.

The wizard opens on the **Engine Selection** page.

2. Click the **Select** button next to the **Microsoft SQL Server Database Standard Edition** engine.

The wizard continues to the **DB Instance Details** page. The first page of the wizard displays a list of DB Instance Classes in the **DB Instance Class** drop-down list. The DB Instance class defines the CPU and memory capacity of your DB Instance.

3. On the **DB Instance Details** page, specify your DB Instance details as shown in the following table, then click **Continue**.

For this parameter...	...Do this:
<b>License Model</b>	Select <b>License Included</b> from the drop-down list box.
<b>DB Engine Version</b>	Select <b>10.50.2789 (default)</b> .
<b>DB Instance Class</b>	Select <b>db.m1.large</b> .
<b>Multi-AZ Deployment</b>	Select <b>No</b> .
<b>Auto Minor Version Upgrade</b>	Keep the default setting of <b>yes</b> for this example. The Auto Minor Version Upgrade option enables your DB Instance to receive minor engine version upgrades automatically when they become available.

**Amazon Relational Database Service User Guide**  
**Creating a DB Instance Running the Microsoft SQL**  
**Server Database Engine**

---

For this parameter...	...Do this:
<b>Allocated Storage</b>	You can specify how much storage in gigabytes you want initially allocated for your DB Instance. For this example, type 250.
<b>DB Instance Identifier</b>	The DB Instance is a name for your DB Instance that is unique for your account in a Region. Type <code>mymsftsqlserver</code> in the <b>DB Instance Identifier</b> text box.
<b>Master Username</b>	Type a name for your master user in the <b>Master Username</b> text box. You use the master user name to log on to your DB Instance with all database privileges.
<b>Master Password</b>	Type a password for your master user in the <b>Master User Password</b> text box.



**Important**

You must specify a password containing from 8 to 128 alphanumeric characters only.

After you click the **Continue** button, the **Additional Configuration** page opens.

4. Provide additional configuration information for your DB Instance:
  - a. Type `mydatabase` into the **Database Name** text box.

When you're creating a DB Instance running the MySQL database engine, you provide a database name so that Amazon RDS will create a default database on your new DB Instance. If you skip this step, Amazon RDS will not create a database on your DB Instance.
  - b. Accept the default values for the rest of the parameters available on this page, and then click the **Continue** button.

After you click the **Continue** button, the **Management Options** page appears. The **Management Options** panel is where you can specify backup and maintenance options for your DB Instance.
5. For this example, accept the default values, and then click **Continue**.

After you click the **Continue** button, the **Review** panel appears.
6. Review the options for your DB Instance:
  - If you need to correct any options, click the **Back** to return to previous panels and make corrections.
  - If all your options are entered correctly, click the **Launch DB Instance** button to launch your new DB Instance.

After you click the **Launch DB Instance** button, a message displays stating that your DB Instance is being created.

This can take a few minutes to complete.
7. Click the **Close** button.

After you click the **Close** button, the **My DB Instances** panel appears. Your DB Instance appears in the list on this page with the **creating** status until your DB Instance is created and ready for use.

Once your DB instance changes to the **available** state, you need to authorize access so you can connect to it.

## 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.

```
PROMPT>rds-create-db-instance mymsftsqlserver -s 250 -c db.m1.large -e  
sqlserver-se  
- u sa -p secretpassword --backup-retention-period 3
```

This command should produce output similar to the following:

```
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 `CreateDBInstance` with the following parameters:
  - *DBInstanceIdentifier* = `mymsftsqlserver`
  - *Engine* = `sqlserver-se`
  - *DBInstanceClass* = `db.m1.large`
  - *AllocatedStorage* = `250`
  - *BackupRetentionPeriod* = `3`
  - *MasterUsername* = `sa`
  - *MasterUserPassword* = `secretpassword`

## Example

```
https://rds.amazonaws.com/  
  ?Action=CreateDBInstance  
  &DBInstanceIdentifier=mymsftsqlserver  
  &DBInstanceClass=db.ml.large  
  &Engine=sqlserver-se  
  &MasterUserPassword=secretpassword  
  &BackupRetentionPeriod=3  
  &AllocatedStorage=250  
  &MasterUsername=sa  
  &Version=2012-04-02  
  &SignatureVersion=2  
  &SignatureMethod=HmacSHA256  
  &Timestamp=2012-04-02T19%3A36%3A35.512Z  
  &AWSAccessKeyId=<AWS Access Key ID>  
  &Signature=<Signature>
```

## Related Topics

- [DB Instance \(p. 5\)](#)
- [DB Security Groups \(p. 11\)](#)
- [Connecting to a DB Instance Running the Microsoft SQL Server Database Engine \(p. 68\)](#)
- [Scaling CPU and Memory for a DB Instance \(p. 82\)](#)
- [Scaling DB Instance Storage \(p. 84\)](#)
- [Deleting a DB Instance \(p. 77\)](#)

## Connecting to 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 the instance. In this example, you connect to a DB Instance running the MySQL database engine using the MySQL command line tools.



### Note

For more information on using MySQL, go to the [MySQL documentation](#).

## CLI

### To connect to a DB Instance using the MySQL monitor

- Type the following command at a command prompt to connect to a DB Instance using the MySQL monitor; substitute the DNS name for your DB Instance.

```
PROMPT> mysql -h myinstance.mydnsnameexample.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)  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
  
mysql>
```

## CLI

### To connect to a DB Instance with SSL using the MySQL monitor

- Download the public key for the Amazon RDS signing certificate from <https://rds.amazonaws.com/doc/mysql-ssl-ca-cert.pem>.
- Type the following command at a command prompt to connect to a DB Instance with SSL using the MySQL monitor; substitute the DNS name for your DB Instance and the SSL certificate file name as appropriate. Enter the master user password when prompted.

```
PROMPT> mysql -h myinstance.mydnsnameexample.rds.amazonaws.com --ssl_ca=cert-mysql-ssl-ca.pem
```

You will see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

**Amazon Relational Database Service User Guide**  
**Connecting to a DB Instance Running the MySQL**  
**Database Engine**

---

```
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 \(p. 5\)](#)
- [Creating a DB Instance Running the MySQL Database Engine \(p. 50\)](#)
- [DB Security Groups \(p. 11\)](#)
- [Deleting a DB Instance \(p. 77\)](#)

## 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.



### Note

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:

```
DBINSTANCE DBInstanceId Created Class Engine
Storage
Master Username Status Endpoint Address
Port AZ Backup Retention Multi-AZ Version Read Replica
Source ID License
DBINSTANCE mydbinstance 2011-05-14T01:11:01.727Z db.m1.small mysql
20
awsuser available mydbinstance.c7abcdefghij.us-east-
1.rds.amazonaws
.com 3306 us-east-1a 1 n 5.1.57
general-public-license
```



### 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.

## Amazon Relational Database Service User Guide Connecting to a DB Instance Running the Oracle Database Engine

---

```
PROMPT>sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=oracledb.mydnsnameexample.rds.amazonaws.com)
(PORT=1521))(CONNECT_DATA=(SID=oracledb)))'
```

You will see output similar to the following.

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

SQL>
```

### Related Topics

- [DB Instance \(p. 5\)](#)
- [Creating a DB Instance Running the MySQL Database Engine \(p. 50\)](#)
- [DB Security Groups \(p. 11\)](#)
- [Deleting a DB Instance \(p. 77\)](#)

## 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.



### Note

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](#).

## AWS Management Console

### 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.

## Amazon Relational Database Service User Guide Connecting to a DB Instance Running the Microsoft SQL Server Database Engine

The screenshot shows the Amazon RDS console interface. At the top, there's a navigation bar with buttons like 'Launch DB Instance', 'Create Read Replica', 'Modify', 'Delete', 'Reboot', 'Take Snapshot', and 'Restore To Point in Time'. Below this is a table listing DB instances. One instance, 'mymssqlserver', is selected. The details for this instance are shown below, with tabs for 'Description', 'Monitoring', and 'Recent Events'. The 'Description' tab is active, displaying various properties of the instance. Two fields are circled in red: 'Port' (8443) and 'Endpoint' (mymssqlserver.c7hszkfowzmc.us-east-1.rds.amazonaws.com).

DB Instance	VPC ID	Multi-AZ	Class	Status	Alarm Status
<input checked="" type="checkbox"/> mymssqlserver		No	db.m1.large	available	none

**1 DB Instance selected**

**DB Instance: mymssqlserver**

**Description** | Monitoring | Recent Events

**DB Instance Name:** mymssqlserver | **Alarm Status:** none

**DB Engine:** sqlserver-se | **DB Engine Version:** 10.50.2789

**License Model:** license-included | **Auto Minor Ver. Upgrade:** Yes

**DB Security Groups:** default | **DB Status:** available

**DB Instance Class:** db.m1.large | **Endpoint:** mymssqlserver.c7hszkfowzmc.us-east-1.rds.amazonaws.com

**Port:** 8443 | **Zone:** us-east-1b

**Multi-AZ Deployment:** No | **DB Storage:** 500 GiB

**Master Username:** sa | **DB Name:** -

**Created Time:** 2012-03-26 11:33 PDT | **Latest Restorable Time:** -

**Backup Retention Period:** 0 | **DB Parameter Group:** default.sqlserver-se-10.5

**Backup Window:** 07:30-08:00 | **Maintenance Window:** mon:05:00-mon:05:30

**Pending Modifications:** None | **Read Replica Source:** None

**Read Replica(s):** None | **VPC ID:** -

**DB Subnet Grp Name:** - | **DB Subnet Grp Status:** -

**DB Subnet Grp Zones:** -

**DB Subnet Grp Desc:** -

**DB Subnet Grp Subnets:** -

- b. On the **Description** tab of the lower panel, note the endpoint of the DB Instance to use in the next step.
  - c. On the **Description** tab of the lower panel, note the port 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.

**Amazon Relational Database Service User Guide**  
**Connecting to a DB Instance Running the Microsoft SQL**  
**Server Database Engine**

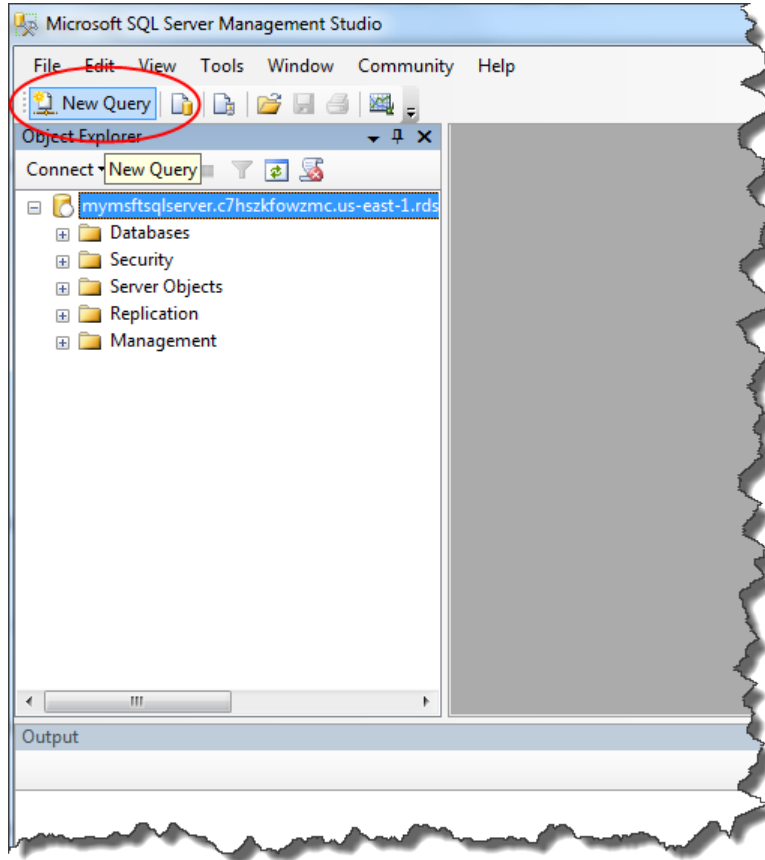
---



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, followed by a comma and then the port number of the DB Instance.
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.

**Amazon Relational Database Service User Guide**  
**Connecting to a DB Instance Running the Microsoft SQL**  
**Server Database Engine**

---

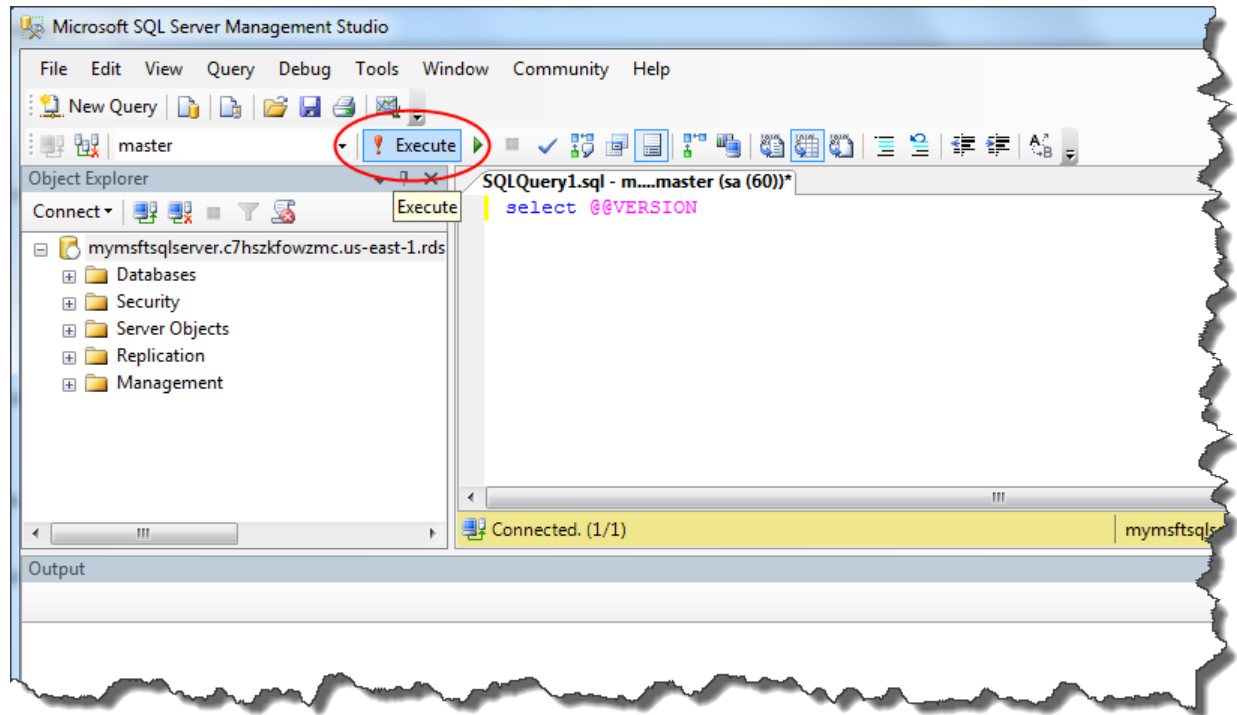


11. Type the following SQL query:

```
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.

## Amazon Relational Database Service User Guide Connecting to a DB Instance Running the Microsoft SQL Server Database Engine



### Related Topics

- [DB Instance](#) (p. 5)
- [Creating a DB Instance Running the Microsoft SQL Server Database Engine](#) (p. 60)
- [DB Security Groups](#) (p. 11)
- [Deleting a DB Instance](#) (p. 77)

## Rebooting a DB Instance

In some cases, if you modify a DB Instance or the DB parameter group associated with the instance, you must reboot the instance for the changes to take effect.

If your DB Instance is deployed in multiple Availability Zones, you can force a failover from one AZ to the other during the reboot. You might force a failover to test the availability of your DB Instance deployment or to restore operations to the original AZ after a failover occurs.

### 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. Under **My DB Instances**, right-click the DB Instance that you want to reboot, and then click **Reboot**.
3. To force a failover from one AZ to the other, in the **Reboot DB Instance** dialog box, select the **Reboot with failover?** check box.
4. 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
```

### API

#### To reboot a DB Instance

- Call `RebootDBInstance`. To force a failover from one AZ to the other, add the following parameter:
  - `ForceFailover = true`

## Modifying a DB Instance

After you have created a DB Instance, you can make changes to many configuration settings whenever you want.

### Topics

- [Modifying a DB Instance Running the MySQL Database Engine \(p. 74\)](#)
- [Modifying a DB Instance Running the Oracle Database Engine \(p. 75\)](#)

## Modifying a DB Instance Running the MySQL Database Engine

### AWS Management Console

#### To modify a MySQL DB Instance

1. Right-click the DB Instance that you want to change, and then click **Modify**.
2. In the **Modify DB Instance** dialog box, change any of the following settings that you want:

Setting	Description
DB Instance Class	In the list provided, click the DB Instance class that you want to use. For information about instance classes, see <a href="#">the section called “DB Instance Class” (p. 6)</a> .
DB Engine Version	In the list provided, click the version of the MySQL database engine that you want to use.
Multi-AZ Deployment	If you want to deploy your DB Instance in multiple Availability Zones, click <b>Yes</b> ; otherwise, click <b>No</b> .
Auto Minor Version Upgrade	If you want your DB Instance to receive minor engine version upgrades automatically when they become available, click <b>Yes</b> . Upgrades are installed only during your scheduled maintenance window.
Allocated Storage	Specify how much storage, in gigabytes, will be initially allocated for your DB Instance. The minimum allowable value is 5 GB; the maximum is 1024 GB.
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.
DB Parameter Group	If you have a custom DB parameter group and you want to associate it with this DB Instance, in the list provided, click the DB security group that you want.

Setting	Description
DB Parameter Group	If you have a custom DB parameter group and you want to associate it with this DB Instance, in the list provided, click the DB security group that you want.
DB Security Groups	If you have a custom DB security group and you want to associate it with this DB Instance, in the list provided, click the DB security group that you want.
Master User Password	Type a password for your master user. The password must contain from 8 to 30 alphanumeric characters.

3. To apply the changes immediately, select the **Apply Immediately** check box.
4. 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.

## Modifying a DB Instance Running the Oracle Database Engine

### AWS Management Console

#### To modify an Oracle DB Instance

1. Right-click the DB Instance that you want to change, and then click **Modify**.
2. In the **Modify DB Instance** dialog box, change any of the following settings that you want:

Setting	Description
DB Instance Class	In the list provided, click the DB Instance class that you want to use. For information about instance classes, see <a href="#">the section called "DB Instance Class" (p. 6)</a> .
DB Engine Version	In the list provided, click the version of the Oracle database engine that you want to use.
Multi-AZ Deployment	If you want to deploy your DB Instance in multiple Availability Zones, click <b>Yes</b> ; otherwise, click <b>No</b> .

Setting	Description
Auto Minor Version Upgrade	If you want your DB Instance to receive minor engine version upgrades automatically when they become available, click <b>Yes</b> . Upgrades are installed only during your scheduled maintenance window.
Allocated Storage	Specify how much storage, in gigabytes, will be initially allocated for your DB Instance. The minimum allowable value is 10 GB; the maximum is 1024 GB.
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.
DB Security Group	If you have a custom DB security group and you want to associate it with this DB Instance, in the list provided, click the DB security group that you want.
Master User Password	Type a password for your master user. The password must contain from 8 to 30 alphanumeric characters.

3. To apply the changes immediately, select the **Apply Immediately** check box.
4. 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`.

## API

### To modify an Oracle DB Instance

- Use the `ModifyDBInstance` action.

## Deleting a DB Instance

Deleting a DB Instance requires you to identify the instance you want to remove. You must also decide whether you want a final DB Snapshot. Without the DB Snapshot, you can't restore the DB Instance to its final state.

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.

#### 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 <https://console.aws.amazon.com/rds/>.
2. In the **My DB Instances** list, select the check box next to the DB Instance you wish to delete.
3. Click the **Delete** button or right-click the DB Instance and select **Delete** from the context menu. The **Delete DB Instance** window appears.
4. Select **No** in the **Create final snapshot?** drop-down list box.
5. Click the **OK** button.

#### 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`

## Example

```
https://rds.amazonaws.com/  
?Action=DeleteDBInstance  
&DBInstanceIdentifier=mydbinstance  
&SkipFinalSnapshot=true  
&SignatureVersion=2  
&SignatureMethod=HmacSHA256  
&Timestamp=2009-10-14T22%3A20%3A46.297Z  
&AWSAccessKeyId=<AWS Access Key ID>  
&Signature=<Signature>
```

## 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.

### 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 **My DB Instances** list, select the check box next to the DB Instance you wish to delete.
3. Click the **Delete** button or right-click the DB Instance and select **Delete** from the context menu. The **Delete DB Instance** window appears.
4. Select **Yes** in the **Create final snapshot?** drop-down list box.
5. Type the name of your final snapshot into the **Final snapshot name** text box.
6. Click the **OK** button.

### 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
```

This command should produce output similar to the following:

```
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-la 3
```

```
SECGROUP default active
```

## 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 \(p. 50\)](#)
- [DB Instance \(p. 5\)](#)

## Sizing Your Amazon RDS DB Instance

One of the advantages of Amazon RDS is that you can easily select and change the amount of compute and storage resources available to your DB Instance. This topic will assist you in the process of choosing the initial capacity of your DB Instances and show you how to change the resources available to existing DB Instances.

### Choosing an Instance Class

The compute and memory capacity of a DB Instance is determined by its DB Instance class. As a result, you can increase or decrease the CPU and memory available to a DB Instance by changing its DB Instance class. For information on available DB Instance classes and the compute and memory resources of each, see the Amazon RDS Product Page.

If you're unsure how much CPU you need, we recommend starting with the `db.m1.small` DB Instance class and monitoring CPU utilization with Amazon's CloudWatch service. If your DB Instance is CPU bound, you can easily upgrade to a larger DB Instance class using the `rds-modify-db-instance` command. For information on how to monitor the CPU utilization of your DB Instances, see the "DB Instance Monitoring" section of the Amazon RDS Developer Guide.

The following example shows how to change the DB Instance class. In this example, the `acme` DB Instance is upgraded to a `db.m1.xlarge`.

```
PROMPT> rds-modify-db-instance acme --db-instance-class db.m1.xlarge
```

Amazon RDS will perform the upgrade during the next maintenance window. If you want the upgrade to be performed now, rather than waiting for the maintenance window, specify the `--apply-immediately` option.



#### Note

Changing the DB Instance Class requires a brief outage for your DB Instance.

```
PROMPT> rds-modify-db-instance acme --db-instance-class db.m1.xlarge --apply-immediately
```

For information on displaying and setting the maintenance window for your Amazon RDS DB Instances, see the [Amazon RDS Users Guide](#).

### How Much Storage Do I Need to Allocate?

To determine the amount of storage to allocate to a DB Instance, start by estimating the storage needed under typical operating conditions (steady-state). Add to that the storage required for expected growth for 1-3 months, plus any additional storage needed for operations requiring significant amounts of temporary storage (such as batch jobs or large data loads). See the Amazon RDS Customer Data Import Guide for information on loading data into your DB Instance while minimizing temporary storage requirements.

If the data is already in an existing database, you'll already have a good measure of the storage requirements. When copying or migrating the data to Amazon RDS, allocate storage sufficient to meet the peak demand of your current database and add enough to allow for expected growth. If the data is

in flat files, a rule of thumb for estimating the amount of space needed for your DB Instance is to compute the total size of the flat files and double it. This allows space for secondary indexes and working storage.

For data stored in some other format, determine the amount of space required to store the data as flat files and use the estimating method for flat files. You can monitor the amount of allocated storage space being utilized by your DB Instance with the help of [Amazon CloudWatch](#). If you find that you need additional storage, use the **rds-modify-db-instance** command to add storage to the DB Instance. In the example below, the total storage for the `acme` DB Instance is increased to 90GB. The `--apply-immediately` option tells Amazon RDS to do it now rather than wait for a maintenance window. Amazon RDS adds storage without restarting the DB Instance and without interrupting active processes.

```
PROMPT> rds-modify-db-instance acme --allocated-storage=90 --apply-immediately
```



### Note

Because of the extensibility limitations of striped storage attached to Windows Server, Amazon RDS does not currently support modifying allocated 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](#).

Use the **rds-describe-db-instances** command to see when Amazon RDS has completed the task and use CloudWatch to verify that the additional storage is available.



### Caution

RDS cannot remove storage 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 load the data into the new DB Instance.

## Related Topics

- [DB Instance](#) (p. 5)
- [Creating and Modifying DB Instances](#) (p. 49)
- [Scaling CPU and Memory for a DB Instance](#) (p. 82)
- [Scaling DB Instance Storage](#) (p. 84)

## Scaling CPU and Memory for a DB Instance

The [DB Instance class](#) specifies the CPU and memory capacity of a DB Instance. You can scale the CPU and memory capacity of a DB Instance up or down by specifying a new DB Instance class.

In this example, you change the DB Instance class of an existing DB Instance called *mydbinstance* to *Extra Large*.

### AWS Management Console

#### To modify the CPU class for an Amazon RDS 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 **My DB Instances** list, select the check box next to the DB Instance you wish to delete.
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 the new DB Instance class from the **DB Instance Class** drop-down list box.
5. Click the **OK** button.

### CLI

#### To modify the CPU class for an Amazon RDS instance

- Use the command `rds-modify-db-instance` with the following parameters:

```
PROMPT>rds-modify-db-instance mydbinstance --db-instance-class db.m1.large
```

This command produces output similar to the following:

```
DBINSTANCE mydbinstance 2009-10-21T18:32:37.080Z db.m1.small MySQL 50
sa available mydbinstance.clouwupjnmq.us-east-1.rds.amazonaws.com 3306
us-east-1a 3 db.m1.large n 5.1.57 general-public-license
SECGROUP default active
PARAMGRP default.MySQL5.1 in-sync
```

### API

#### To modify the CPU class for an DB Instance

- Call `ModifyDBInstance` with the following parameters:
  - `DBInstanceIdentifier` = *myinstancename*
  - `DBInstanceClass` = *db.m1.large*

### Example

```
https://rds.amazonaws.com/  
?Action=ModifyDBInstance  
&DBInstanceIdentifier=mydbinstance  
&DBInstanceClass=db.m1.large  
&SignatureVersion=2  
&SignatureMethod=HmacSHA256  
&Timestamp=2009-10-14T17%3A48%3A21.746Z  
&AWSAccessKeyId=<AWS Access Key ID>  
&Signature=<Signature>
```

## Related Topics

- [DB Instance \(p. 5\)](#)
- [Adjusting the Preferred Maintenance Window \(p. 87\)](#)
- [Scaling DB Instance Storage \(p. 84\)](#)

## Scaling DB Instance Storage

If your DB instance runs out of storage space or file system resources, its status will change to *storage-full* and your DB Instance will no longer be available.



### Note

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 following example shows the output of a **rds-describe-db-instances** command for a DBInstance which has used up all its storage:

```
PROMPT> rds-describe-db-instances mydbinstance
DBINSTANCE mydbinstance 2009-12-22T23:06:11.915Z db.m1.large mysql
50 sa
storage-full mydbinstance.c1la4j4jgyph.us-east-1.rds.amazonaws.com 3306

us-east-1b 3 n 5.1.57 general-public-license
SECGROUP default active
PARAMGRP default.mysql5.1 in-sync
```

You can also check the RDS events to see if your storage space is exhausted using the `DescribeEvents` API or the **rds-describe-events** command.

The following example shows a storage-full event being returned by the **rds-describe-events** command:

```
PROMPT>rds-describe-events --source-type db-instance --source-identifier
mydbinstance
2009-12-22T23:44:14.374Z mydbinstance Allocated storage has been exhausted
db-instance
```



### Important

We highly recommend that you constantly monitor the *FreeStorageSpace* RDS metric published in CloudWatch to ensure that your DB Instance has enough free storage space. For more information on monitoring RDS DB Instances, see [Viewing DB Instance Metrics \(p. 156\)](#).

You can change the amount of disk space available to your DB Instance using the `ModifyDBInstance` API or the **rds-modify-db-instance** command.



### Important

The maximum storage for an Amazon RDS DB Instance is 1024GB.

In this example, you change the storage size of an existing DB Instance called *mydbinstance* to 1024 GB.



## Note

Your DB Instance will remain available (if it does not have a `storage-full` status) during a scale storage operation. If you want to scale storage immediately, include the `ApplyImmediately` flag in the `ModifyDBInstance` request (or the `--apply-immediately` flag with the **rds-modify-db-instance** command). If you do not specify the `ApplyImmediately`, the scale storage will occur during the next maintenance window, or after the next reboot of the DB Instance. It is only possible to increase the storage size of a DB Instance. Decreasing the storage size of a DB Instance is not permitted.

You must increase storage size in increments of at least 10%. If you specify an increase of less than 10%, the provided value will be rounded up to 10%.

## 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 **My DB Instances** list, select the check box next to the DB Instance you wish to delete.
3. Click the **Modify** button.  
The **Modify DB Instance** window appears.
4. Type the amount of storage you want (in gigabytes) into the **Allocated Storage** text box.
5. Click the **OK** button.

## CLI

- Use the command **rds-modify-db-instance** as in the following example.

```
PROMPT>rds-modify-db-instance mydbinstance --allocated-storage 1024
```

This command produces output similar to the following.

```
DBINSTANCE mydbinstance 2009-10-21T18:32:37.080Z db.m1.large MySQL
50
sa available mydbinstance.clouwupjnmq.us-east-1.rds.amazonaws.com 3306
us-east-1a 3 db.m1.xlarge 1024 n 5.1.57 general-public-license
SECGROUP default active
PARAMGRP default.MySQL5.1 in-sync
```

## API

### To modify the storage size of an DB Instance

- Call `ModifyDBInstance` with the following parameters:
  - `DBInstanceIdentifier` = `myinstancename`
  - `AllocatedStorage` = `1024`

### Example

```
https://rds.amazonaws.com/  
?Action=ModifyDBInstance  
&DBInstanceIdentifier=mydbinstance  
&AllocatedStorage=1024  
&SignatureVersion=2  
&SignatureMethod=HmacSHA256  
&Timestamp=2009-10-14T17%3A48%3A21.746Z  
&AWSAccessKeyId=<AWS Access Key ID>  
&Signature=<Signature>
```

## Related Topics

- [DB Instance \(p. 5\)](#)
- [Adjusting the Preferred Maintenance Window \(p. 87\)](#)
- [Scaling CPU and Memory for a DB Instance \(p. 82\)](#)

## 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.

Region	Time Block
US East (Northern Virginia) Region	03:00-11:00 UTC
US West (Northern California) Region	06:00-14:00 UTC
US West (Oregon) Region	06:00-14:00 UTC
EU (Ireland) Region	22:00-06:00 UTC
Asia Pacific (Singapore) Region	14:00-22:00 UTC
Asia Pacific (Tokyo) Region	17:00-03:00 UTC
South America (São Paulo) Region	00:00-08:00 UTC

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:

```
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:
  - *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 \(p. 5\)](#)
- [Creating and Modifying DB Instances \(p. 49\)](#)
- [Scaling CPU and Memory for a DB Instance \(p. 82\)](#)

## Working with Reserved DB Instances

Reserved DB Instances let you make a one-time, up-front payment for a DB Instance, reserve the DB Instance for a one- or three-year term, and pay a significantly lower rate for each hour you run that instance. Reserved DB Instances are available in three types, Heavy Utilization, Medium Utilization, and Light Utilization. By choosing the size that best meets your needs, you can optimize your Amazon RDS costs based on your expected utilization. For information about reserved instance types, go to [Amazon RDS Reserved Instances](#).

You can use the command line or the API to list and purchase available reserved DB Instance offerings. Reserved DB Instance offerings are based on the DB instance class, duration, and whether or not the reserved DB instance is Single-AZ or Multi-AZ.

In this example, you see how to view available reserved DB Instance offerings, purchase an available Reserved DB Instance Offering, and list Reserved DB Instances for your account.

### 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 Instances** link.
3. Click the **Purchase Reserved DB Instance** button.
4. Use the **Product Description** drop-down list box to select the DB Engine 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. Click the **Continue** button.

The **Purchase Reserved DB Instance** dialog box shows a summary of the reserved DB Instance attributes that you've selected and the cost of the reservation.

10. Click the **Cancel** link in the upper-right corner of the dialog box 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:

OFFERING	OfferingId	Class	Multi-AZ
Duration	Fixed Price	Usage Price	Description
OFFERING	438012d3-4052-4cc7-b2e3-8d3372e0e706	db.ml.large	y
	1820.00 USD	0.368 USD	mysql
			Medium Utilization
OFFERING	649fd0c8-cf6d-47a0-bfa6-060f8e75e95f	db.ml.small	n
	227.50 USD	0.046 USD	mysql
			Medium Utilization
OFFERING	123456cd-ab1c-47a0-bfa6-12345667232f	db.ml.small	n
	162.00 USD	0.00 USD	mysql
			Heavy Utilization
	Recurring Charges:	Amount	Currency
	Recurring Charges:	0.123	USD
			Hourly
OFFERING	123456cd-ab1c-37a0-bfa6-12345667232d	db.ml.large	y
	700.00 USD	0.00 USD	mysql
			Heavy Utilization
	Recurring Charges:	Amount	Currency
	Recurring Charges:	1.25	USD
			Hourly
OFFERING	123456cd-ab1c-17d0-bfa6-12345667234e	db.ml.xlarge	n
	4242.00 USD	2.42 USD	mysql
			Light Utilization

## 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.amazon
aws.com/doc/2012-01-15/" >
  <DescribeReservedDBInstancesOfferingsResult>
    <ReservedDBInstancesOfferings>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <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-
```

```
8d3372e0e706</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</ReservedDBInstancesOf
feringId>
  <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</ReservedDBInstancesOf
feringId>
  <DBInstanceClass>db.m1.large</DBInstanceClass>
</ReservedDBInstancesOffering>
<ReservedDBInstancesOffering>
  <Duration>31536000</Duration>
  <OfferingType>Light Utilization</OfferingType>
  <CurrencyCode>USD</CurrencyCode>
  <RecurringCharges/>
  <FixedPrice>4242.0</FixedPrice>
```

```
<ProductDescription>mysql</ProductDescription>
<UsagePrice>2.42</UsagePrice>
<MultiAZ>>false</MultiAZ>
<ReservedDBInstancesOfferingId>TEMP-DELETE-3</ReservedDBInstancesOf
feringId>
  <DBInstanceClass>db.m1.xlarge</DBInstanceClass>
</ReservedDBInstancesOffering>
</ReservedDBInstancesOfferings>
</DescribeReservedDBInstancesOfferingsResult>
<ResponseMetadata>
  <RequestId>5e4ec40b-2978-11e1-9e6d-771388d6ed6b</RequestId>
</ResponseMetadata>
</DescribeReservedDBInstancesOfferingsResponse>
```

## 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.

The **Purchase Reserved DB Instance** dialog box shows a summary of the Reserved DB Instance attributes that you've selected and the payment due.

11. Click the **Yes, Purchase** button to proceed and purchase the Reserved DB Instance.

## CLI

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

- Enter the following command at a command prompt:

```
PROMPT>rds-purchase-reserved-db-instances-offering 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f -i myreservationID
```

The command returns output similar to the following:

```
RESERVATION  ReservationId      Class      Multi-AZ  Start Time
      Duration Fixed Price  Usage Price  Count  State      Description
Offering Type
RESERVATION  myreservationid  db.ml.small  y          2011-12-
19T00:30:23.247Z  1y          455.00 USD  0.092 USD  1      payment-pending
mysql          Medium Utilization
```

## 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:
  - `ReservedDBInstancesOfferingId` = *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*
  - `ReservedDBInstanceID` = *myreservationID*
  - `DBInstanceCount` = 1

## Example

```
https://rds.amazonaws.com/  
?Action=PurchaseReservedDBInstancesOffering  
&ReservedDBInstancesOfferingId=649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  
&ReservedDBInstanceId=myreservationID  
&DBInstanceCount=1  
&SignatureVersion=2  
&SignatureMethod=HmacSHA256  
&Timestamp=2012-01-14T17%3A48%3A21.746Z  
&AWSAccessKeyId=<AWS Access Key ID>  
&Signature=<Signature>
```

This call returns output similar to the following:

```
<PurchaseReservedDBInstancesOfferingResponse xmlns="http://rds.amazon  
aws.com/doc/2012-01-15/">  
  <PurchaseReservedDBInstancesOfferingResult>  
    <ReservedDBInstance>  
      <OfferingType>Medium Utilization</OfferingType>  
      <CurrencyCode>USD</CurrencyCode>  
      <RecurringCharges/>  
      <ProductDescription>mysql</ProductDescription>  
      <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-  
060f8e75e95f</ReservedDBInstancesOfferingId>  
      <MultiAZ>true</MultiAZ>  
      <State>payment-pending</State>  
      <ReservedDBInstanceId>myreservationID</ReservedDBInstanceId>  
      <DBInstanceCount>10</DBInstanceCount>  
      <StartTime>2011-12-18T23:24:56.577Z</StartTime>  
      <Duration>31536000</Duration>  
      <FixedPrice>123.0</FixedPrice>  
      <UsagePrice>0.123</UsagePrice>  
      <DBInstanceClass>db.m1.small</DBInstanceClass>  
    </ReservedDBInstance>  
  </PurchaseReservedDBInstancesOfferingResult>  
  <ResponseMetadata>  
    <RequestId>7f099901-29cf-11e1-bd06-6fe008f046c3</RequestId>  
  </ResponseMetadata>  
</PurchaseReservedDBInstancesOfferingResponse>
```

## 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:

```
RESERVATION  ReservationId      Class      Multi-AZ  Start Time
              Duration  Fixed Price  Usage Price  Count  State      Description  Of
fering Type
RESERVATION  ki-real-ri-test5  db.m1.small  y          2011-12-09T23:37:44.720Z
ly          455.00 USD    0.092 USD    1        retired  mysql        Medium
Utilization
```

## 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/">  
  <DescribeReservedDBInstancesResult>  
    <ReservedDBInstances>  
      <ReservedDBInstance>  
        <OfferingType>Medium Utilization</OfferingType>  
        <CurrencyCode>USD</CurrencyCode>  
        <RecurringCharges/>  
        <ProductDescription>mysql</ProductDescription>  
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-  
060f8e75e95f</ReservedDBInstancesOfferingId>  
        <MultiAZ>>false</MultiAZ>  
        <State>payment-failed</State>  
        <ReservedDBInstanceId>myreservationid</ReservedDBInstanceId>  
        <DBInstanceCount>1</DBInstanceCount>  
        <StartTime>2010-12-15T00:25:14.131Z</StartTime>  
        <Duration>31536000</Duration>  
        <FixedPrice>227.5</FixedPrice>  
        <UsagePrice>0.046</UsagePrice>  
        <DBInstanceClass>db.m1.small</DBInstanceClass>  
      </ReservedDBInstance>  
      <ReservedDBInstance>  
        <OfferingType>Medium Utilization</OfferingType>  
        <CurrencyCode>USD</CurrencyCode>  
        <RecurringCharges/>  
        <ProductDescription>mysql</ProductDescription>  
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-  
060f8e75e95f</ReservedDBInstancesOfferingId>  
        <MultiAZ>>false</MultiAZ>  
        <State>payment-failed</State>  
        <ReservedDBInstanceId>myreservationid2</ReservedDBInstanceId>  
        <DBInstanceCount>1</DBInstanceCount>  
        <StartTime>2010-12-15T01:07:22.275Z</StartTime>  
        <Duration>31536000</Duration>  
        <FixedPrice>227.5</FixedPrice>  
        <UsagePrice>0.046</UsagePrice>  
        <DBInstanceClass>db.m1.small</DBInstanceClass>  
      </ReservedDBInstance>  
    </ReservedDBInstances>  
  </DescribeReservedDBInstancesResult>  
</ResponseMetadata>
```

```
<RequestId>23400d50-2978-11e1-9e6d-771388d6ed6b</RequestId>  
</ResponseMetadata>  
</DescribeReservedDBInstancesResponse>
```

## Related Topics

- [Reserved DB Instances \(p. 10\)](#)

## Using Amazon RDS with Amazon Virtual Private Cloud (VPC)



### Important

You should have an understanding of how Amazon VPC works before reading this section. Please refer to the [Amazon Virtual Private Cloud documentation](#) for detailed information about Amazon VPC.

Amazon Virtual Private Cloud enables you to create a virtual network in the AWS cloud. With a Virtual Private Cloud (VPC), you can define a virtual network that closely resembles a traditional data center. You have complete control over your virtual networking environment, including the selection of your own IP address range, creation of subnets, and configuration of routing and access control lists. This section shows examples of how to use Amazon RDS with your Amazon Virtual Private Cloud.

The basic steps for running an Amazon RDS DB Instance in a VPC that will be covered in the following examples are:

- Create a VPC with a private subnet for each Availability Zone
- Create a DB Subnet Group and add to it each of the private subnets in your VPC
- Create a DB Instance in your VPC

### Topics

- [Creating a Virtual Private Cloud \(VPC\)](#) (p. 99)
- [Creating a DB Subnet Group](#) (p. 102)
- [Creating a DB Instance in a VPC](#) (p. 105)
- [Connecting to a DB Instance Running in a VPC](#) (p. 109)

## Creating a Virtual Private Cloud (VPC)

In this example, we will create a VPC with a private subnet for each Availability Zone.

### AWS Management Console

#### Create an Amazon RDS DB Instance Inside an Amazon Virtual Private Cloud

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. Create a new VPC using the **Create an Amazon Virtual Private Cloud** wizard.
  - a. Click on the **VPC Dashboard** link in the **Navigation** pane.
  - b. Click the **Get started creating a VPC** button.  
The **Create an Amazon Virtual Private Cloud** wizard appears.
  - c. Select the **VPC with Public and Private Subnets** radio button.

## Amazon Relational Database Service User Guide Using Amazon RDS with Amazon Virtual Private Cloud (VPC)

**Create an Amazon Virtual Private Cloud** Cancel X

Select a VPC configuration below:

- VPC with a Single Public Subnet Only**  
Your instances run in a private, isolated section of the AWS cloud with direct access to the Internet. Network access control lists and security groups can be used to provide strict control over inbound and outbound network traffic to your instances.
- VPC with Public and Private Subnets**  
In addition to containing a public subnet, this configuration adds a private subnet whose instances are not addressable from the Internet. Instances in the private subnet can establish outbound connections to the Internet via the public subnet using Network Address Translation.
- VPC with Public and Private Subnets and Hardware VPN Access**  
This configuration adds an IPsec Virtual Private Network (VPN) connection between your Amazon VPC and your datacenter - effectively extending your datacenter to the cloud while also providing direct access to the Internet for public subnet instances in your Amazon VPC.
- VPC with a Private Subnet Only and Hardware VPN Access**  
Your instances run in a private, isolated section of the AWS cloud with a private subnet whose instances are not addressable from the Internet. You can connect this private subnet to your corporate datacenter via an IPsec Virtual Private Network (VPN) tunnel.

**Creates:** a /16 network with two /24 subnets. Public subnet instances use Elastic IPs to access the Internet. Private subnet instances access the Internet via a Network Address Translation (NAT) instance in the public subnet. (Hourly charges for NAT instances apply)

[Continue](#)

- d. Click the **Continue** button.  
The **VPC with Public and Private Subnets** panel appears.

**Create an Amazon Virtual Private Cloud** Cancel X

**VPC with Public and Private Subnets**

Please review the information below, then click **Create VPC**.

**One VPC with an Internet Gateway**  
**IP CIDR block:** 10.0.0.0/16 (65,531 available IPs) [Edit VPC IP CIDR Block](#)

**Two Subnets**  
**Public Subnet:** 10.0.0.0/24 (251 available IPs) [Edit Public Subnet](#)  
**Availability Zone:** No Preference  
**Private Subnet:** 10.0.1.0/24 (251 available IPs) [Edit Private Subnet](#)  
**Availability Zone:** No Preference

Additional subnets can be added after the VPC has been created.

**One NAT Instance with an Elastic IP Address**  
**Instance Type:** m1.small [Edit NAT Instance Type](#)  
**Key Pair Name:** [redacted] [Edit Key Pair](#)

Note: Instance rates apply. [View rates](#).

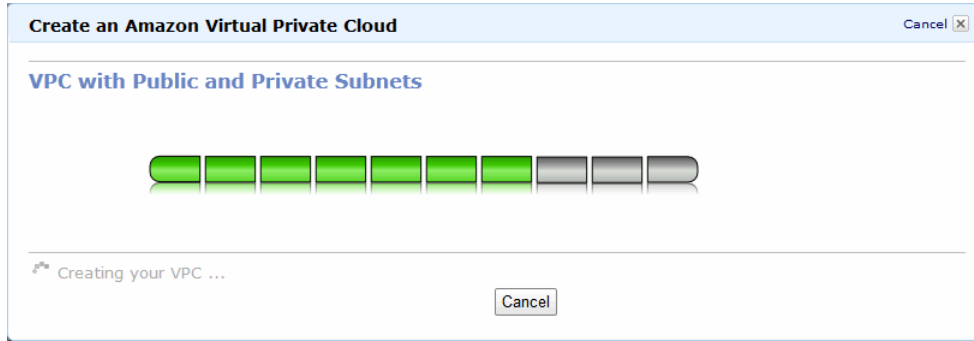
**Hardware Tenancy**  
**Tenancy:** Default [Edit Hardware Tenancy](#)

[Back](#) [Create VPC](#)

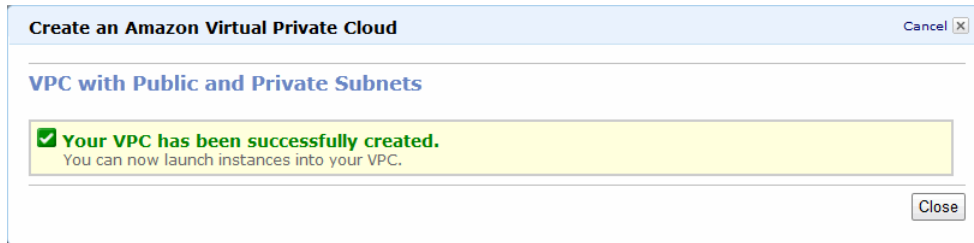
- e. Keep the default options on this panel, and click the **Create VPC** button.  
A progress bar will appear as your VPC is created.

**Amazon Relational Database Service User Guide  
Using Amazon RDS with Amazon Virtual Private Cloud  
(VPC)**

---

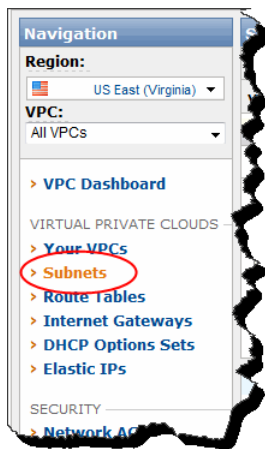


- f. After your VPC is created, a verification dialog box will appear.



Click the **Close** button.

3. Create subnets in each Availability Zone.
- a. Click on the **Subnets** link in the **Navigation** pane.



- b. Click the **Create a Subnet** button.  
The **Create Subnet** dialog appears.
- c. Create a new subnet using a CIDR address within your VPC's CIDR address range, and place it in a new Availability Zone. Repeat this step until you have a subnet created for each Availability Zone.

## Amazon Relational Database Service User Guide Using Amazon RDS with Amazon Virtual Private Cloud (VPC)

**Create Subnet** Cancel

Please use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Please note that block sizes must be between a /16 netmask and /28 netmask. You can create no more than 20 subnets per VPC. Also, please note that a subnet can be the same size as your VPC.

**VPC:** vpc-66f6350e (10.0.0.0/16)

**Availability Zone:** us-east-1c

**CIDR Block:** 10.0.3.0/24 (e.g., 10.0.0.0/24)

Cancel Yes, Create



### Note

Not all Availability Zones may be available in your Region. You will see an error message on the **Create a Subnet** dialog if you attempt to create a subnet in an Availability Zone that is not available in your Region.

4. Make a note of the public subnet that was automatically created with your VPC. You will need to know the subnet ID later when you create an EC2 instance to access your DB Instance.



### Note

Your subnet ID will be different than the one shown in the example below.

	Subnet ID	State	VPC ID	CIDR	Available IPs	Availability Zone	Route Table	Network ACL
<input type="checkbox"/>	subnet-01f43769	available	vpc-66f6350e	10.0.3.0/24	251	us-east-1d	rtb-78f63510	Default
<input type="checkbox"/>	subnet-4bf43723	available	vpc-66f6350e	10.0.2.0/24	251	us-east-1b	rtb-78f63510	Default
<input checked="" type="checkbox"/>	subnet-7ef63516	available	vpc-66f6350e	10.0.0.0/24	250	us-east-1a	rtb-73f6351b	Default
<input type="checkbox"/>	subnet-7df63515	available	vpc-66f6350e	10.0.1.0/24	251	us-east-1a	rtb-78f63510	Default
<input type="checkbox"/>	subnet-daf437b2	available	vpc-66f6350e	10.0.4.0/24	251	us-east-1e	rtb-78f63510	Default

The public subnet will have one less available IP address, since the wizard creates an EC2 NAT instance and an Elastic IP address (for which EC2 rates apply) for outbound communication to the internet from your private subnet. You can delete those resources for this example, unless you want to retain them for other use cases.

Now that you've created a VPC, you can create a DB Subnet Group and launch a DB Instance to run in the VPC. For an example, please continue to [Creating a DB Subnet Group \(p. 102\)](#).

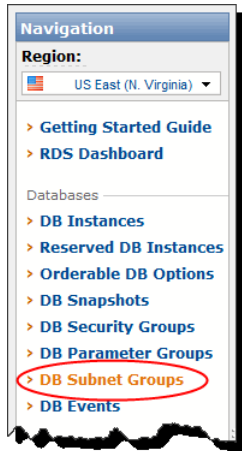
## Creating a DB Subnet Group

In this example, we will create a DB Subnet Group and add the private subnets from our VPC.

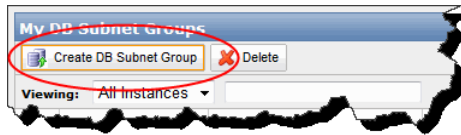
## AWS Management Console

### Create a DB Subnet Group and Associate a DB Security Group

- Create a DB Subnet Group:
  - a. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
  - b. Click **DB Subnet Groups** in the **Navigation** list on the left side of the window.



- c. Click the **Create DB Subnet Group** button.



- d. Type the name of your DB Subnet Group in the **Name** text box.
- e. Type a description for your DB Subnet Group in the **Description** text box.
- f. Select the VPC that you created from the **VPC ID** drop-down list box.
- g. Type a description for your DB Subnet Group in the **Description** text box.
- h. Click the **add all the subnets** link in the **Add Subnet(s) to this Subnet Group** section.

A screenshot of the 'Create DB Subnet Group' form in the AWS Management Console. The form includes the following fields:

- Name:** mydbsubnetgroup
- Description:** Testing
- VPC ID:** vpc-66f6350e: 10.0.0.0/16

Below these fields is a section titled 'Add Subnet(s) to this Subnet Group'. It contains a table of available subnets and an 'Add' button.

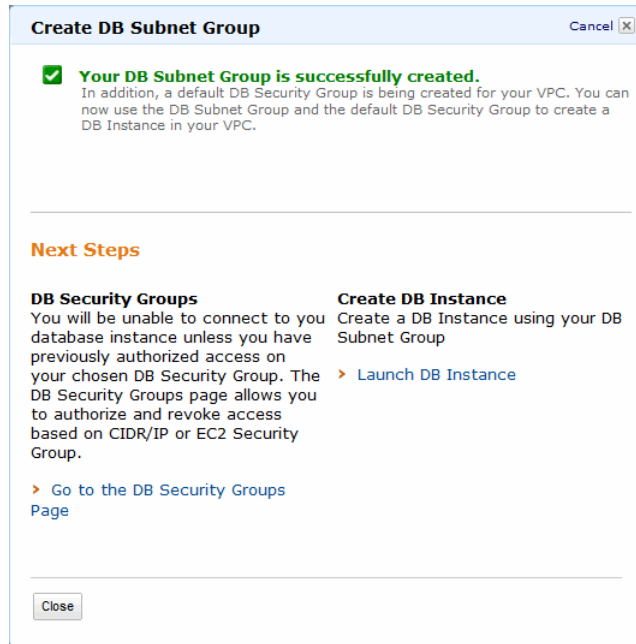
Availability Zone	Subnet IDs	CIDR Block	Action
us-east-1d	subnet-01f43769	10.0.3.0/24	remove
us-east-1b	subnet-4bf43723	10.0.2.0/24	remove
us-east-1a	subnet-7ef63516	10.0.0.0/24	remove
us-east-1a	subnet-7df63515	10.0.1.0/24	remove
us-east-1e	subnet-daf437b2	10.0.4.0/24	remove

At the bottom of the form are 'Cancel' and 'Yes, Create' buttons.

**Amazon Relational Database Service User Guide  
Using Amazon RDS with Amazon Virtual Private Cloud  
(VPC)**

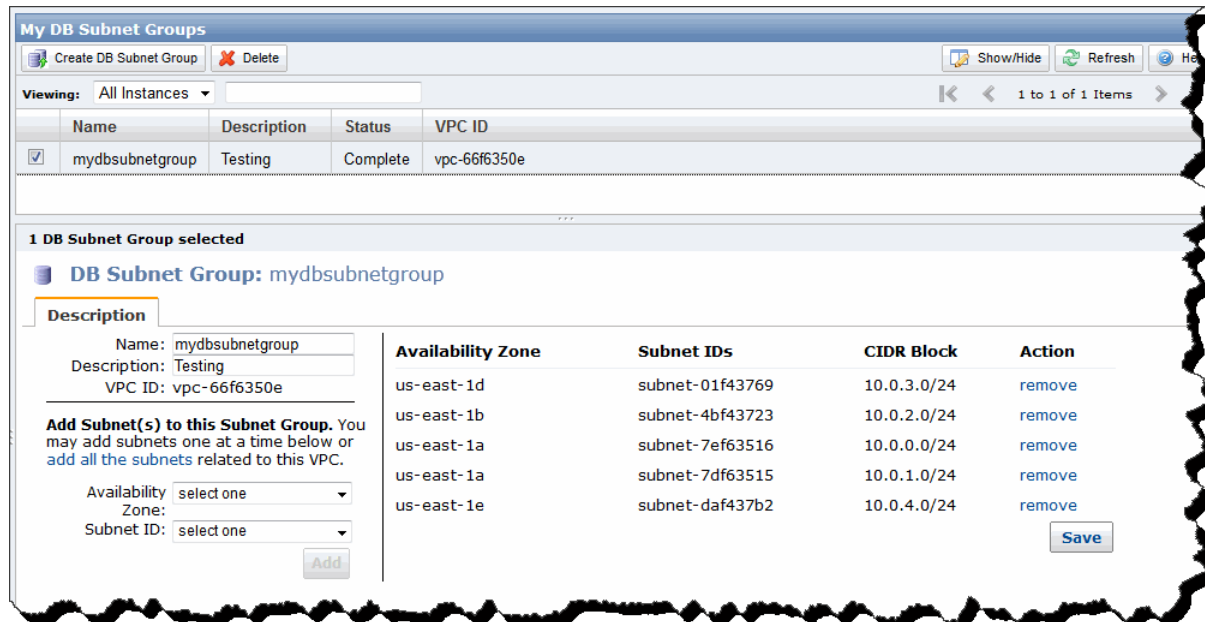
- i. Click the **Yes, Create** button.

A dialog box will appear indicating that your DB Subnet Group is created.



- j. Click the **Close** button.

Your new DB Subnet Group will now appear in the DB Subnet Groups list. You can click on it and see details, such as all of the subnets associated with this group, in the details pane at the bottom of the window.



Now that you've created a DB Subnet Group, you can launch a DB Instance to run in your VPC. For an example, continue to [Creating a DB Instance in a VPC \(p. 105\)](#).

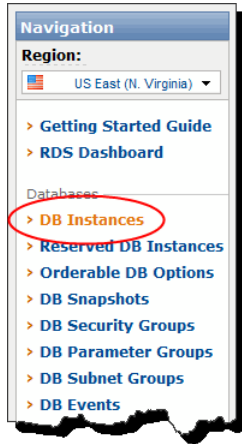
## Creating a DB Instance in a VPC

In this example, we will create a DB Instance in our VPC.

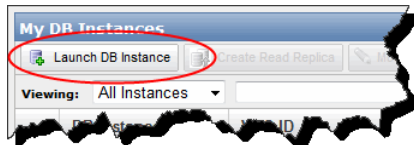
### AWS Management Console

#### Create an Amazon RDS DB Instance Inside an Amazon Virtual Private Cloud

- Create an RDS DB Instance Inside Your VPC
  - a. Click **DB Instances** in the **Navigation** list on the left side of the window.



- b. Click the **Launch DB Instance** button.

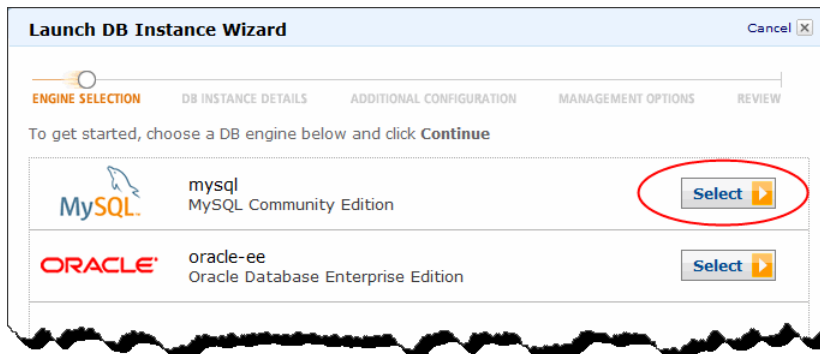


- c. Click the **Select** button next to the MySQL DB Engine.



#### Note

Amazon RDS currently supports only DB Instances running the MySQL DB Engine in an Amazon VPC.



- d. Select your DB Instance configuration on the **DB Instance Details** panel of the **Launch DB Instance Wizard**.

## Amazon Relational Database Service User Guide Using Amazon RDS with Amazon Virtual Private Cloud (VPC)

The screenshot shows the 'Launch DB Instance Wizard' window, specifically the 'DB INSTANCE DETAILS' step. The progress bar at the top indicates the current step. The 'Engine' is set to 'mysql'. The 'License Model' is 'general-public-license'. The 'DB Engine Version' is '5.5.8'. The 'DB Instance Class' is 'db.m1.small'. The 'Multi-AZ Deployment' is 'No'. The 'Auto Minor Version Upgrade' is set to 'Yes'. Below this, the user is asked to provide details for the RDS Database Instance. The 'Allocated Storage' is '5 GB'. The 'DB Instance Identifier' is 'mypcdbinstance'. The 'Master User Name' is 'awsuser'. The 'Master User Password' is masked with dots. At the bottom, there are 'Back' and 'Continue' buttons.

- e. Click the **Continue** button.  
The **Additional Configuration** panel appears.
- f. Select the VPC that you created with the **Choose a VPC** drop-down list box.
- g. Select the DB Subnet Group that you created with the **Subnet Group** drop-down list box.
- h. Select the Availability Zone for your DB Instance. In this example, we will select the us-east-1a Availability Zone, since that's where we will create the EC2 instance we will use to access the DB Instance in the VPC.

The screenshot shows the 'Launch DB Instance Wizard' window, specifically the 'ADDITIONAL CONFIGURATION' step. The progress bar at the top indicates the current step. The user is asked to provide optional additional configuration details. The 'Database Name' is 'mydb'. A note states: 'Note: if no database name is specified then no initial mysql database will be created on the DB Instance.' The 'Database Port' is '3306'. The 'Choose a VPC' is 'vpc-66f6350e'. The 'DB Subnet Group' is 'mydbsubnetgroup'. The 'Availability Zone' is 'us-east-1a'. Below this, the user is asked to select a 'DB Parameter Group' (set to 'default.mysql5.5') and 'DB Security Groups' (set to 'default:vpc-66f6350e'). At the bottom, there are 'Back' and 'Continue' buttons.

- i. Click the **Continue** button.  
The **Management Options** panel appears.

## Amazon Relational Database Service User Guide Using Amazon RDS with Amazon Virtual Private Cloud (VPC)

The screenshot shows the 'Launch DB Instance Wizard' in the 'MANAGEMENT OPTIONS' step. The progress bar at the top indicates the current step. Below the progress bar, there is a text description: 'The number of days for which automated backups are retained. Setting this parameter to a positive number enables backups. Setting this parameter to 0 disables automated backups.' Below this, the 'Backup Retention Period' is set to '1' days. Another text description follows: 'The daily time range during which automated backups are created if automated backups are enabled'. The 'Backup Window' is set to 'No Preference'. A third text description: 'The weekly time range (in UTC) during which system maintenance can occur.' The 'Maintenance Window' is also set to 'No Preference'. At the bottom, there are '< Back' and 'Continue >' buttons.

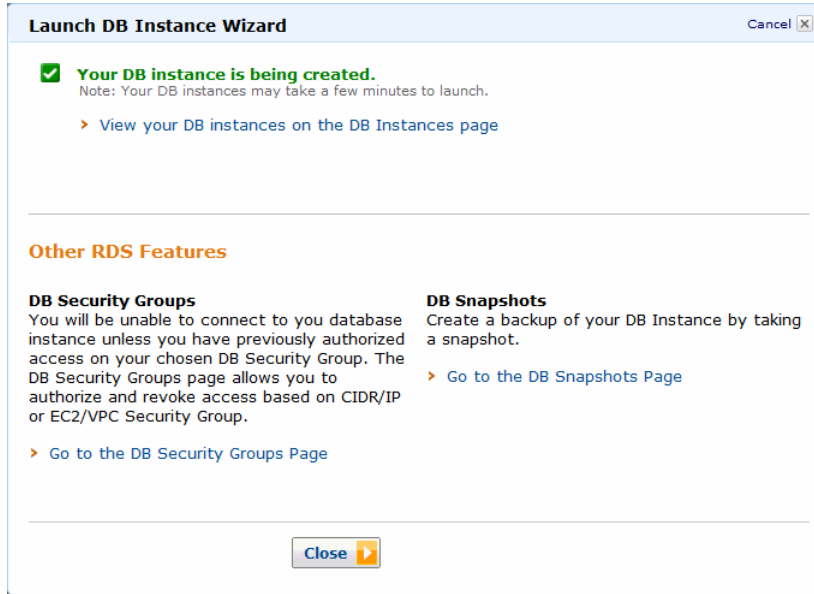
- j. For this example, keep the default options, and then click the **Continue** button.

The **Review** panel appears.

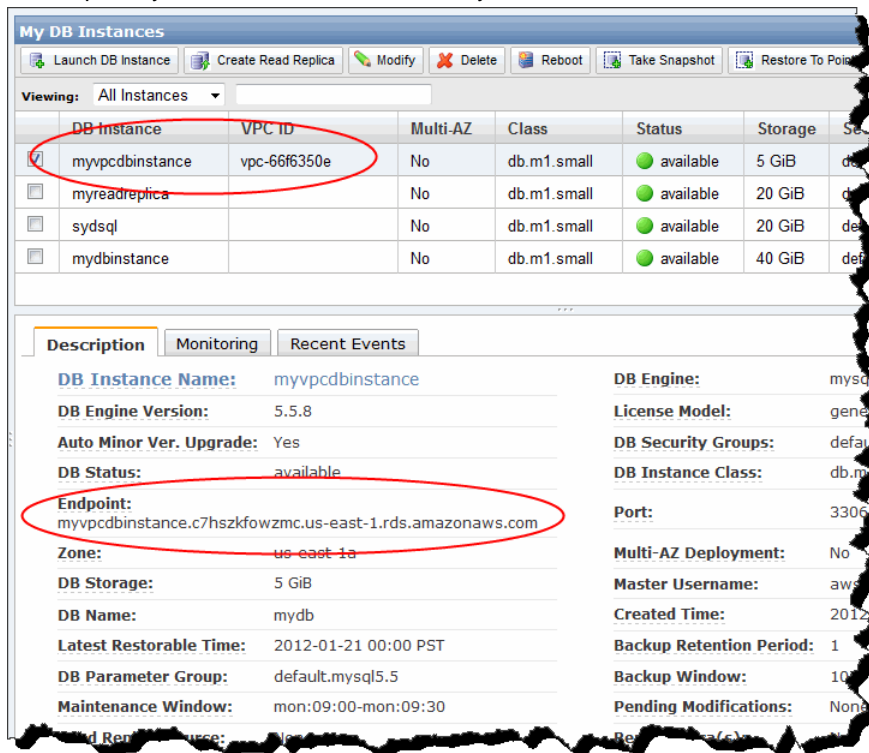
The screenshot shows the 'Launch DB Instance Wizard' in the 'REVIEW' step. The progress bar at the top indicates the current step. Below the progress bar, there is a text description: 'Please review the information below, then click Launch'. Below this, there are two sections of configuration details. The first section lists: Engine: mysql, Engine Version: 5.5.8, License Model: general-public-license, Auto Minor Ver. Upgrade: Yes, DB Instance Class: db.m1.small, Multi-AZ Deployment: No, Allocated Storage: 5, DB Instance Identifier: myvpcdbinstance, Master User Name: awsuser, and Master User Password: \*\*\*\*\*. The second section lists: Database Name: mydb, Database Port: 3306, VPC ID: vpc-66f6350e, DB Subnet Group: mydbsubnetgroup, Availability Zone: us-east-1a, DB Parameter Group: default.mysql5.5, and DB Security Group(s): default:vpc-66f6350e. Below these sections, there are: Backup Retention Period: 1, Backup Window: No Preference, and Maintenance Window: No Preference. At the bottom, there are '< Back' and 'Launch DB Instance >' buttons.

- k. Click the **Launch DB Instance** button. A confirmation dialog box will appear.

## Amazon Relational Database Service User Guide Using Amazon RDS with Amazon Virtual Private Cloud (VPC)



- I. Click the **Close** button.
- m. You can see your DB Instance in the DB Instances view 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.



You've now launched a DB Instance inside a VPC. For an example of one way to connect to your new DB Instance running in the VPC, continue to [Connecting to a DB Instance Running in a VPC \(p. 109\)](#).

## Connecting to a DB Instance Running in a VPC

This example shows how to access your DB Instance running in a VPC from an EC2 Instance.



### Important

For instructions on using SSH to connect to a Linux/UNIX instance, go to [Connect to Your Linux/UNIX Instance](#) in the [Amazon Elastic Compute Cloud Getting Started Guide](#).

## AWS Management Console

### Connect to a DB Instance Running in a VPC

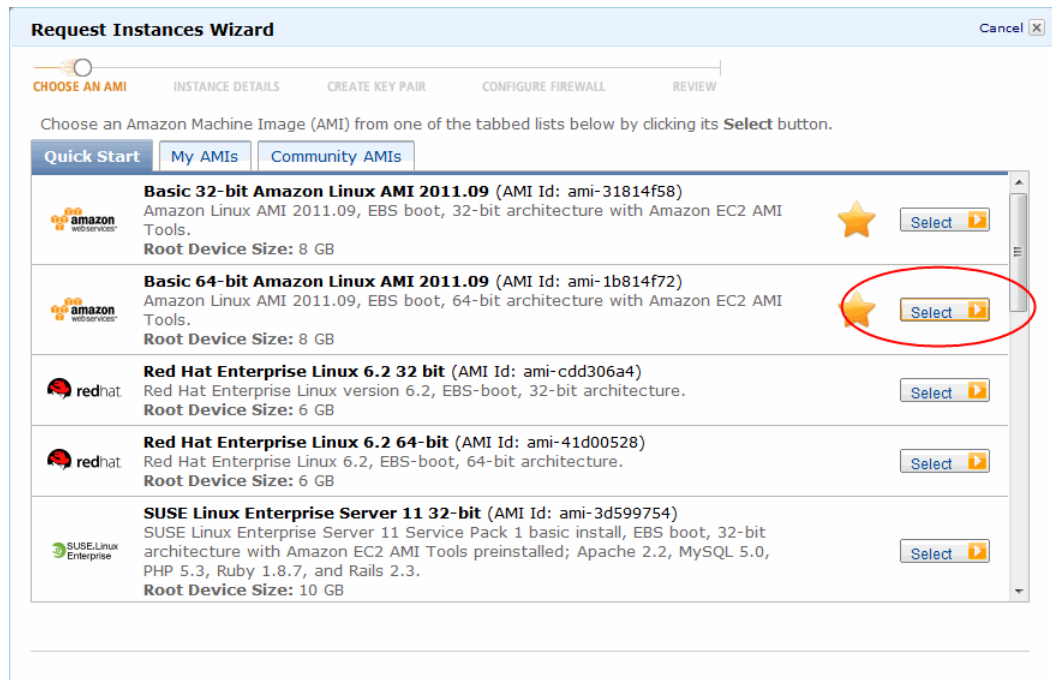
In this example, we will create an EC2 Instance in the VPC that we created earlier. This EC2 instance can be used to connect to the DB Instance running in the VPC.



### Important

For more detailed instructions on using Amazon EC2, go to the [Amazon EC2 Getting Started Guide](#) in the [Amazon EC2 documentation](#).

1. Create an EC2 Instance in your VPC:
  - a. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
  - b. Click the **Launch Instance** button.
  - c. Select a Quick Start Amazon Machine Image from the list. In this example, we will use the Basic 64-bit Amazon Linux AMI.



- d. Select an instance type using the **Instance Type** drop-down list box
- e. In the **Instance Details** panel of the **Request Instances Wizard** panel, select the **VPC** radio button, and then select the public subnet from the VPC you created in the **Subnet** drop-down list box.

## Amazon Relational Database Service User Guide Using Amazon RDS with Amazon Virtual Private Cloud (VPC)

**Request Instances Wizard** Cancel

CHOOSE AN AMI **INSTANCE DETAILS** CREATE KEY PAIR CONFIGURE FIREWALL REVIEW

Provide the details for your instance(s). You may also decide whether you want to launch your instances as "on-demand" or "spot" instances.

**Number of Instances:** 1 **Instance Type:** Large (m1.large, 7.5 GB)

**Launch Instances**

EC2 Instances let you pay for compute capacity by the hour with no long term commitments. This transforms what are commonly large fixed costs into much smaller variable costs.

**Launch into:**  EC2  VPC

**Subnet:** subnet-7ef63516 (10.0.0.0/24) us-east-1a 250 available IP addresses

**Request Spot Instances**

< Back Continue >

- f. Keep the defaults on the next panel, and click the **Continue** button.
- g. On the second **Instance Details** panel, add a name to the EC2 Instance to make it easier to find, then click the **Continue** button.

**Request Instances Wizard** Cancel

CHOOSE AN AMI **INSTANCE DETAILS** CREATE KEY PAIR CONFIGURE FIREWALL REVIEW

Add tags to your instance to simplify the administration of your EC2 infrastructure. A form of metadata, tags consist of a case-sensitive key/value pair, are stored in the cloud and are private to your account. You can create user-friendly names that help you organize, search, and browse your resources. For example, you could define a tag with key = Name and value = Webservice. You can add up to 10 unique keys to each instance along with an optional value for each key. For more information, go to [Using Tags](#) in the *EC2 User Guide*.

Key (127 characters maximum)	Value (255 characters maximum)	Remove
Name	myrdsinstance	

[Add another Tag.](#) (Maximum of 10)

- h. On the **Create Key Pair** panel, you can create a key pair or select an existing key pair. In this example, we will select an existing key pair, and then click **Continue**.



### Note

For more information about creating key pairs, go to the [Amazon EC2 Getting Started Guide](#).

## Amazon Relational Database Service User Guide Using Amazon RDS with Amazon Virtual Private Cloud (VPC)

**Request Instances Wizard** Cancel

CHOOSE AN AMI    INSTANCE DETAILS    **CREATE KEY PAIR**    CONFIGURE FIREWALL    REVIEW

Public/private key pairs allow you to securely connect to your instance after it launches. To create a key pair, enter a name and click **Create & Download your Key Pair**. You will then be prompted to save the private key to your computer. Note, you only need to generate a key pair once - not each time you want to deploy an Amazon EC2 instance.

**Choose from your existing Key Pairs**

Your existing Key Pairs\*:

Create a new Key Pair

Proceed without a Key Pair

- i. On the **Configure Firewall** panel, accept the default values, and then click the **Continue** button.



### Caution

To avoid inadvertently granting access to your DB Instances, be sure to understand how CIDR ranges work. For more information about CIDR ranges, go to the [Wikipedia Tutorial](#).

**Request Instances Wizard** Cancel

CHOOSE AN AMI    INSTANCE DETAILS    CREATE KEY PAIR    **CONFIGURE FIREWALL**    REVIEW

Security groups determine whether a network port is open or blocked on your instances. You may use an existing security group, or we can help you create a new security group to allow access to your instances using the suggested ports below. Add additional ports now or update your security group anytime using the Security Groups page.

Choose one or more of your existing Security Groups

**Create a new Security Group**

**Group Name**

**Group Description**

**Inbound Rules**

Create a new rule:

Port range:   
(e.g., 80 or 49152-65535)

Source:   
(e.g., 192.168.2.0/24, sg-47ad482e, or 1234567890/default)

TCP	Port (Service)	Source	Action
	22 (SSH)	0.0.0.0/0	Delete

- j. Review the details of the new EC2 Instance on the **Review** panel, and then click the **Launch** button.

## Amazon Relational Database Service User Guide Using Amazon RDS with Amazon Virtual Private Cloud (VPC)

**Request Instances Wizard** Cancel

CHOOSE AN AMI    INSTANCE DETAILS    CREATE KEY PAIR    CONFIGURE FIREWALL    **REVIEW**

Please review the information below, then click **Launch**.

**AMI:** Amazon Linux AMI ID ami-1b814f72 (x86\_64)  
**Name:** Basic 64-bit Amazon Linux AMI 2011.09  
**Description:** Amazon Linux AMI 2011.09, EBS boot, 64-bit architecture with Amazon EC2 AMI Tools. [Edit AMI](#)

---

**Number of Instances:** 1  
**VPC ID:** vpc-66f6350e  
**VPC Subnet:** subnet-7ef63516 (10.0.0.0/24)  
**Availability Zone:** No Preference  
**Instance Type:** Large (m1.large)  
**Instance Class:** On Demand [Edit Instance Details](#)

---

**Monitoring:** Disabled      **Termination Protection:** Disabled  
**Tenancy:** Default  
**Kernel ID:** Use Default      **Shutdown Behavior:** Stop  
**RAM Disk ID:** Use Default  
**IP Address:** Automatically Assigned  
**User Data:** [Edit Advanced Details](#)

---

**Key Pair Name:** [Edit Key Pair](#)

---

**Security Group(s):** sg-a97d63c5 [Edit Firewall](#)

[Back](#)      **Launch**

- k. Your EC2 Instance is now launching. Click the **Close** button to continue.

**Launch Instance Wizard** Cancel

**Your instances are now launching.**  
Note: Your instances may take a few minutes to launch, depending on the software you are running.

[View your instances on the Instances page](#)  
Note: To view the VPC ID and Subnet ID columns on the Instances page click the **Show/Hide** button and check the corresponding boxes.

---

**Other AWS Features**

<b>Spot Instances</b> Spot Instances enable customers to lower their Amazon EC2 costs by up to 75% by bidding on unused capacity and running instances for as long as the maximum bid exceeds the current Spot Price. <a href="#">Go to Amazon EC2 Spot Instances</a>	<b>Reserved Instances</b> Reserved Instances provide substantial savings over On-Demand instances and ensure that the capacity you need is available to you when required. <a href="#">Go to Amazon EC2 Reserved Instances</a>	<b>Suse Linux Instances</b> Suse Linux instances are a proven platform with superior reliability and security and are automatically kept up to date with Novell's security patches, bug fixes and new features. <a href="#">Go to Amazon EC2 running SUSE Linux</a>
---	--	---

**Close**

2. Define ingress rules for your DB Security Group:

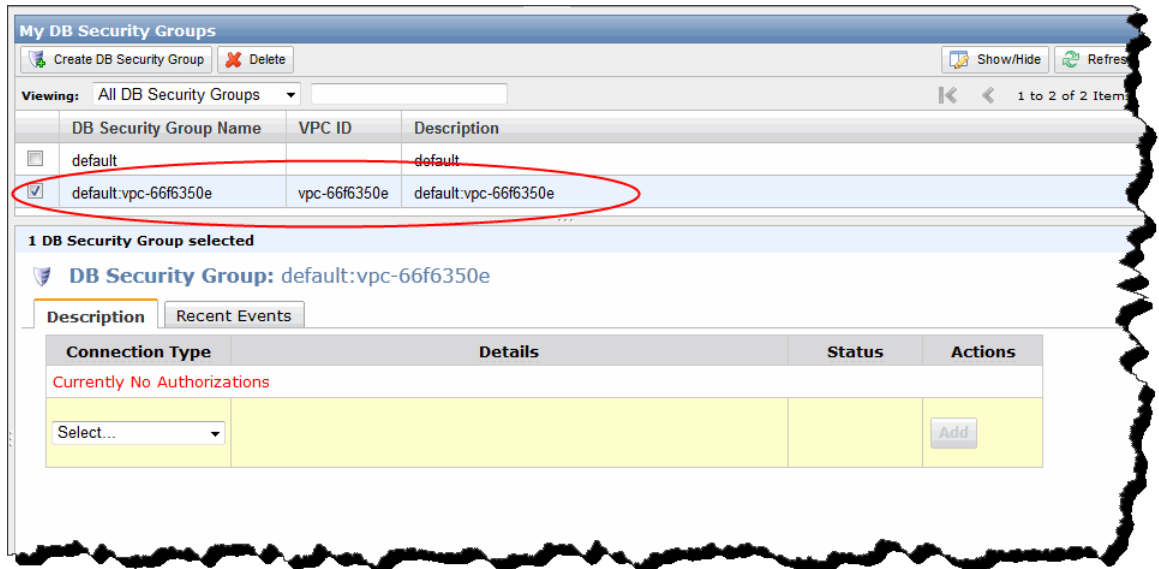
- a. When you created your EC2 instance, a default VPC Security Group was automatically created for you. You'll need to grant access to this VPC Security Group for the DB Security Group used by the DB Instance in your VPC.

In the RDS console, click on the **DB Security Groups** link in the **Navigation** panel.

**Amazon Relational Database Service User Guide**  
**Using Amazon RDS with Amazon Virtual Private Cloud**  
**(VPC)**



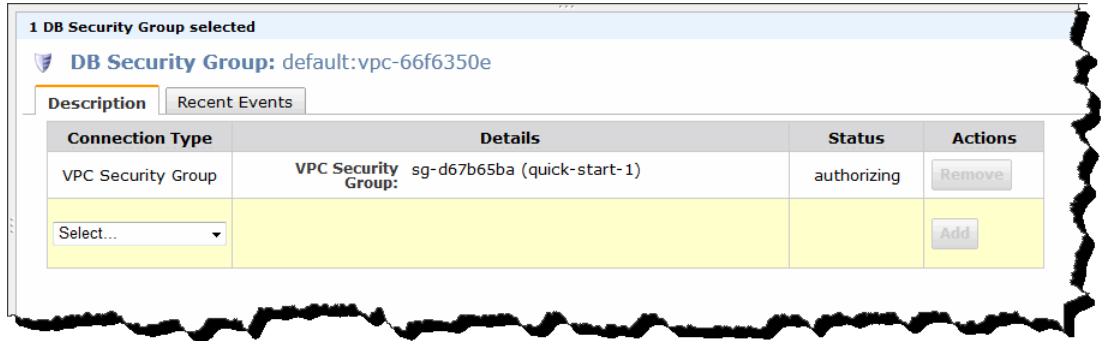
- b. Select the DB Security Group with the VPC ID that you created in the previous step (note that your VPC ID will be different than the one shown in this example).



- c. In the details panel at the bottom of the console, select VPC Security Group from the drop-down list box, and then select the VPC Security Group that includes the text 'quickstart' in the description.
- d. Click the **Add** button.

Your VPC Security Group will be displayed in the **DB Security Group** details panel with the status of 'authorizing'.

**Amazon Relational Database Service User Guide  
Using Amazon RDS with Amazon Virtual Private Cloud  
(VPC)**



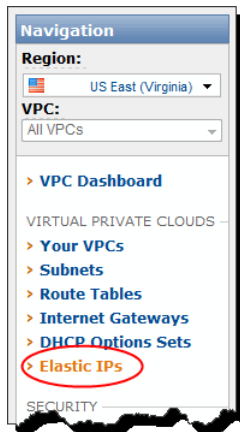
3. Assign an Elastic IP address to the EC2 Instance that you just created:



**Note**

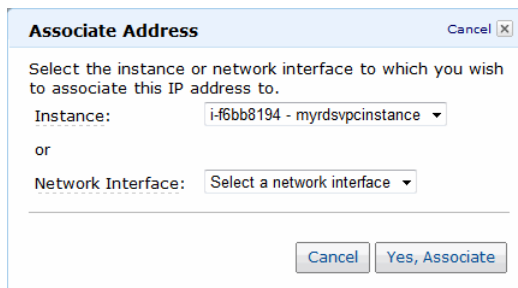
Make note of the Elastic IP address that you associate with your EC2 instance; you will use this IP address to connect to the EC2 instance.

- a. Go to the VPC console.
- b. In the **Navigation** pane, click the **Elastic IPs** link.



- c. Click the **Allocate New Address** button.  
The **Allocate New Address** dialog appears.
- d. Select VPC from the drop-down list box and click **Yes, Allocate**.
- e. Select the Elastic IP address that you just allocated from the list and click the **Associate Address** button.  
The **Associate Address** dialog box appears.
- f. Select the EC2 instance that you launched, and then click the **Yes, Associate** button.

## Amazon Relational Database Service User Guide Using Amazon RDS with Amazon Virtual Private Cloud (VPC)



**Associate Address** Cancel

Select the instance or network interface to which you wish to associate this IP address to.

Instance:

or

Network Interface:

Cancel Yes, Associate

Your EC2 instance is now launched in your Amazon VPC. You can now use the elastic IP address that you assigned and connect to this EC2 instance using SSH, and from there connect to and manage the RDS DB Instance running inside your VPC using the RDS command line interface or your choice of database management tools (such as the MySQL client tools).



### Important

For instructions about using SSH to connect to a Linux/UNIX instance, go to [Connect to Your Linux/UNIX Instance](#) in the [Amazon Elastic Compute Cloud Getting Started Guide](#).

The following example shows a connection to the DB Instance running in the VPC using the MySQL command line client.

```
[ec2-user@ip-10-0-0-168 ~]$ mysql -h myvpcdbinstance.c7hszkfowzmc.us-east-1.rds.amazonaws.com -u awsuser -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.5.8-log Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

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

mysql> select @@version \g
+-----+
| @@version |
+-----+
| 5.5.8-log |
+-----+
1 row in set (0.00 sec)
mysql> quit
Bye
[ec2-user@ip-10-0-0-168 ~]$
```



**Important**

To avoid incurring additional charges on your AWS account, be sure to delete any AWS resources you no longer wish to use after trying these examples.

## Importing Data to a DB Instance

### Topics

- [Importing Data to a Microsoft SQL Server Instance \(p. 117\)](#)
- [Preparing to Import Data into Your SQL Server DB Instance \(p. 117\)](#)
- [Import Logins to Your DB Instance \(p. 119\)](#)
- [Import the Data \(p. 120\)](#)
- [Cleaning Up \(p. 122\)](#)

## Importing Data to a Microsoft SQL Server Instance

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.

This topic explains how to import data into a SQL Server DB Instance.

### Topics

The process that we recommend is as follows:

1. Create a DB Instance. For information, see [Creating a DB Instance Running the Microsoft SQL Server Database Engine \(p. 60\)](#).
2. Before you load data into the destination DB Instance, you should do some preparation, 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 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. 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: reenable foreign key constraints and database triggers, switch the recovery model to its original state, and reenable 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 Server Agent, 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 \(p. 146\)](#).

## 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 \(p. 126\)](#).



### 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 \(p. 130\)](#).

## 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 [' + @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.

```
--Disable triggers on all tables
DECLARE @enable BIT = 0;
DECLARE @trigger SYSNAME;
DECLARE @table SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
    table_object.name table_name
FROM sysobjects trigger_object
JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
WHERE trigger_object.type = 'TR';

OPEN trigger_cursor;
FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

WHILE @@FETCH_STATUS = 0 BEGIN
    IF @enable = 1
        SET @cmd = 'ENABLE ';
    ELSE
        SET @cmd = 'DISABLE ';

    SET @cmd = @cmd + ' TRIGGER dbo.[' + @trigger + '] ON dbo.[' + @table + ']';

    EXEC (@cmd);
    FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
END

CLOSE trigger_cursor;
DEALLOCATE trigger_cursor;

GO
```

## Import Logins to Your 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 on 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:

```
p.type IN
```

Use the following instead:

```
p.type = 'U'
```

## 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 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 our 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.



### Note

If 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 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](#).

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 in 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 in 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 `out` argument to specify an output file instead of `in` to specify an input 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 dbname.schema_name.table_name out 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 out 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. 146).

## Enable Automated Backups on the Target DB Instance

For information about automated backups, see [Working With Automated Backups \(p. 130\)](#).

## 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 [' + @table_name + '] CHECK CONSTRAINT ALL';
    EXEC (@cmd);
    FETCH NEXT FROM table_cursor INTO @table_name;
END

CLOSE table_cursor;
DEALLOCATE table_cursor;
```

## Enable Database Triggers

If you disabled database triggers earlier, you can now enable them with the following script:

```
--Enable triggers on all tables
DECLARE @enable BIT = 1;
DECLARE @trigger SYSNAME;
DECLARE @table SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
    table_object.name table_name
FROM sysobjects trigger_object
JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
WHERE trigger_object.type = 'TR';

OPEN trigger_cursor;
FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

WHILE @@FETCH_STATUS = 0 BEGIN
    IF @enable = 1
        SET @cmd = 'ENABLE ';
    ELSE
        SET @cmd = 'DISABLE ';

    SET @cmd = @cmd + ' TRIGGER dbo.[' + @trigger + '] ON dbo.[' + @table + '];';
END
```

```
EXEC (@cmd);  
  FETCH NEXT FROM trigger_cursor INTO @trigger, @table;  
END  
  
CLOSE trigger_cursor;  
DEALLOCATE trigger_cursor;
```

# Backing Up and Restoring DB Instances

## Topics

- [Creating a DB Snapshot \(p. 126\)](#)
- [Restoring From a DB Snapshot \(p. 128\)](#)
- [Working With Automated Backups \(p. 130\)](#)
- [Restoring a DB Instance to a Specified Time \(p. 133\)](#)

The following scenarios cover DB Instance backup and restore operations for automated backups and user-created DB Snapshots.

## 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/>.
2. Click **DB Snapshots** in the **Navigation** list on the left side of the window.
3. Click the **Create DB Snapshot** button.  
The **Create DB Snapshot** window appears.
4. Select the DB Instance that you want to create a snapshot of from the **DB Instance** drop-down list box.
5. Type the name of the snapshot in the **Snapshot Name** text box.
6. Click the **OK** button.

## CLI

### To create a DB Snapshot

- Use the command `rds-create-db-snapshot` to create a database snapshot.

```
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
50
creating sa 5.1.57 general-public-license
```

## API

### To create a DB Snapshot

- Call `CreateDBSnapshot` with the following parameters:

- *DBSnapshotIdentifier* = mydbsnapshot
- *DBInstanceIdentifier* = mydbinstance

### Example

```
https://rds.amazonaws.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 \(p. 128\)](#)
- [Backup and Restoration \(p. 12\)](#)

## 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.



### Note

When you restore a DB Instance, only the default DB Parameter and DB Security Groups are applied. If you need custom DB Parameter or DB Security groups, you must apply them explicitly using the `ModifyDBInstance` API or the `rds-modify-db-instance` command line tool once the DB Instance is available.

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 <https://console.aws.amazon.com/rds/>.
2. Click **DB Snapshots** in the **Navigation** list on the left side of the window.
3. Click on the DB Snapshot that you want to restore from in the **My DB Snapshots** list.
4. Click the **Restore from DB Snapshot** button.  
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 mydbsnap  
shot
```

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.amazonaws.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 \(p. 126\)](#)
- [DB Snapshots \(p. 13\)](#)

## 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 (Amazon RDS does not charge for a backup 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.

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 **My DB Instances** list, select the check box next to the DB Instance you want to delete.
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. 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. Launch the AWS Management Console.
  - a. Go to the [AWS Management Console](#) web page.
  - b. Select **Amazon RDS** from the drop-down list under the **Sign in to the AWS Console** button, and then click the **Sign in to the AWS Console** button.
2. In the **My DB Instances** list, select the check box next to the DB Instance you wish to delete.
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 \(p. 133\)](#)
- [Backup and Restoration \(p. 12\)](#)

## 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.



### Note

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.

When restoring a DB Instance that is using the Oracle DB Engine 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.

In this example, you restore a database instance to a specified time, creating a new DB Instance.



### Note

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.

When you restore a DB Instance that is using the SQL Server DB Engine to a point in time, each database within that instance will be 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.

Additionally, if you are administering a DB Instance using the SQL Server DB Engine, you could change the recovery model of some databases or otherwise perform actions that break the sequence of logs that allow continuous point-in-time recovery. In some cases, this issue is detected, and the latest restorable time will stop moving forward; in other cases, such as the BULK\_LOGGED recovery model, the issue is not detected. Some issues get corrected when a snapshot is taken, either by the user or by the system during the daily backup window. It may not be possible to restore to points in time that are affected by these issues. For these reasons, Amazon RDS does not support changing the recovery model of databases.

For Microsoft SQL Server DB Instances, the OFFLINE, EMERGENCY, and SINGLE\_USER modes are not currently supported. Setting any database to one of these modes will cause the latest restorable time to stop moving ahead for all databases that are managed by the DB Instance.

## 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. Click **DB Instances** in the **Navigation** list on the left side of the window.
3. Click the **Restore To Point In Time** button.

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 \(p. 126\)](#)
- [Restoring From a DB Snapshot \(p. 128\)](#)
- [Backing Up and Restoring DB Instances \(p. 125\)](#)
- [Backup and Restoration \(p. 12\)](#)

## DB Parameter Groups

The following scenarios cover operations on DB Parameter Groups.

### Topics

- [Working with DB Parameter Groups \(p. 136\)](#)

## Working with DB Parameter Groups

A DB Parameter Group is initially created with the default parameters for the database engine used by the DB Instance. To provide custom values for any of the parameters, you must modify the group after creating it. Once you've created a DB Parameter Group, you need to associate it with your DB Instance. When you associate a new DB Parameter Group with a running DB Instance, you need to reboot the DB Instance for the new DB Parameter Group and associated settings to take effect.

In this example, you create, list, modify, and examine DB Parameter Groups.



### 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.



### Note

Some database engine parameters are constrained or disabled in the context of an RDS DB Instance. For more information, please see [Engine-Specific Parameter Exceptions for RDS DB Instances](#) (p. 20).

## Creating a DB Parameter Group

In this example, you create a new DB Parameter Group.

### 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 command `rds-create-db-parameter-group` with the following parameters:

```
PROMPT>rds-create-db-parameter-group mydbparametergroup -f MySQL5.1 -d "My
```

```
new parameter group"
```

This command should produce output similar to the following:

```
DBPARAMETERGROUP mydbparametergroup mysql5.1 My new parameter group
```

## API

### To create a DB Parameter Group

- Call `CreateDBParameterGroup` with the following parameters:
  - `DBParameterGroupName` = `mydbparamgroup`
  - `Description` = `"My new parameter group"`
  - `DBParameterGroupFamily` = `mysql5.1`

## 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>
```

This command should return a response similar to the following:

```
<CreateDBParameterGroupResponse xmlns="http://rds.amazonaws.com/admin/2012-  
01-15/">  
  <CreateDBParameterGroupResult>  
    <DBParameterGroup>  
      <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>
```

## Listing Available DB Parameter Groups

You can see which DB Parameter Groups you've created for your AWS account by listing them.



### 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 RDS DB Instance without specifying a custom MySQL 5.5 Parameter Group.

## CLI

### To list all available DB Parameter Groups for an AWS account

- Use the command `rds-describe-db-parameter-groups` to list all available DB Parameter Groups for your AWS account:

```
PROMPT>rds-describe-db-parameter-groups
```

This command should produce output similar to 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
```

## AWS Management Console

### To list all available 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 available DB Parameter Groups appears in the **My DB Parameter Groups** list.

## API

### To list all available DB Parameter Groups for an AWS account

- Call `DescribeDBParameterGroups` with no parameters.

## 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>
```

This command should return a response similar to 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</Descrip  
tion>  
        <DBParameterGroupName>default.mysql5.1</DBParameterGroupName>  
      </DBParameterGroup>  
      <DBParameterGroup>  
        <Engine>mysql5.1</Engine>  
        <Description>My new parameter group</Description>  
        <DBParameterGroupName>mydbparametergroup</DBParameterGroup  
Name>  
      </DBParameterGroup>  
    </DBParameterGroups>  
  </DescribeDBParameterGroupsResult>  
  <ResponseMetadata>  
    <RequestId>41731881-0b82-11df-9a9b-c1bd5894571c</RequestId>  
  </ResponseMetadata>  
</DescribeDBParameterGroupsResponse>
```

## Viewing Parameter Values for a DB Parameter Group

You can get a detailed listing of all parameters in a DB Parameter Group and their values.

### CLI

To view the parameter values for a specific DB Parameter Group

- Use the command `rds-describe-db-parameters` to view the parameter values for a specific DB Parameter Group:

```
PROMPT>rds-describe-db-parameters mydbparametergroup
```

This command should produce output similar to the following (the following example has been truncated):

```
DBPARAMETER allow-suspicious-udfs
engine-default boolean static false
DBPARAMETER auto_increment_increment
engine-default integer dynamic true
DBPARAMETER auto_increment_offset
engine-default integer dynamic true
(...sample truncated...)
```

## AWS Management Console

### To view the parameter values for a specific 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. Select a DB Parameter Group from the **My DB Parameter Groups** list.
4. Select the **Parameters** tab from the information panel at the bottom of the window. The list of parameters defined for the DB Parameter Group appears in this tab.

## API

### To view the parameter values for a specific DB Parameter Group

- Call `DescribeDBParameters` with the following parameters:
  - `DBParameterGroupName` = `mydbparametergroup`

### Example

```
https://rds.amazonaws.com/
?Action=DescribeDBParameters
&DBParameterGroupName=mydbparametergroup
&MaxRecords=100
&Version=2012-01-15
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-22T19%3A31%3A42.262Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

This command should return a response similar to the following (the following example has been truncated):

```
<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 </Descrip
tion>
        <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>

      (... sample truncated...)

    </Parameters>
  </DescribeDBParametersResult>
  <ResponseMetadata>
    <RequestId>99c0937a-0b83-11df-85f9-eb5c71b54ddc </RequestId>
  </ResponseMetadata>
</DescribeDBParametersResponse>
```

## Modifying a DB Parameter Group

You can modify parameters in a DB Parameter Group. These parameters are applied to DB Instances associated with the DB Parameter Group either immediately or on the next reboot of the DB Instance, depending on the type of the parameter (dynamic or static) and the apply method chosen for the parameter update.

DB Engine	Apply Immediately	Pending Reboot
MySQL	Dynamic	Dynamic and Static
Oracle	Dynamic	Dynamic and Static
SQL Server	Dynamic	Static only

In this example, you modify a parameter in a DB Parameter Group.

## CLI

### To modify a DB Parameter Group

- Use the command `rds-modify-db-parameter-group` to modify a DB Parameter Group.



#### 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"
```

This command should produce output similar to the following:

```
DBPARAMETERGROUP mydbparametergroup
```

## AWS Management Console

### To modify a DB Parameter Group

The AWS Console does not support this functionality. Please refer to the command line interface example.

## API

### To modify a DB Parameter Group



#### Note

Amazon RDS does not support passing multiple comma-delimited parameter values for a single parameter.

- Call `ModifyDBParameterGroup` with the following parameters:
  - `DBParameterGroupName` = `mydbparametergroup`
  - `Parameters.member.1.ParameterName` = `max_allowed_packet`

- `Parameters.member.1.ParameterValue = 1024`
- `Parameters.member.1.ApplyMethod = immediate`

### Example

```
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
```

This command should return a response similar to the following:

```
<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>
```

## Related Topics

- [DB Parameter Groups \(p. 14\)](#)

# DB Security Groups

The following scenarios cover DB Security Groups.

## Topics

- [Working with DB Security Groups \(p. 146\)](#)

## Working with DB Security Groups

A DB Security Group allows you to control access to your DB Instances. A DB Security Group acts like a firewall controlling network access to your DB Instance. By default, network access is disabled for a new DB Security Group; you must specifically authorize access to an IP range for a new DB Security Group after the DB Security Group is created.

### Creating a DB Security Group

To create a DB Security Group, you need to provide a name and a description.

In this example, you create a new DB Security Group.

#### 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. Click **DB Security Groups** in the **Navigation** list on the left side of the window.
3. Click the **Create DB Security Group** button.  
The **Create DB Security Group** window appears.
4. Type the name of the new DB Security Group in the **DB Security Group** text box.
5. Type a description for the new DB Security Group in the **Description** text box.
6. Click the **OK** button.

#### 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"
```

#### 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=2012-01-20T22%3A06%3A23.624Z  
&AWSAccessKeyId=<AWS Access Key ID>  
&Signature=<Signature>
```

## Listing Available DB Security Groups

You can list which DB Security Groups have been created for your AWS account.

In this example, you list the available DB Security Groups 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 **DB Security Groups** in the **Navigation** list on the left side of the window. The available DB Security Groups appear in the **My 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.

In this example, you view the properties of a DB Security Group.

### 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 **DB Security Groups** in the **Navigation** list on the left side of the window. The available DB Security Groups appear in the **My DB Security Groups** list.
3. Select the **Description** tab from the information panel at the bottom of the window. The list of authorizations defined for the DB Security Group appears in this tab.

### CLI

#### To view properties of a specific DB Security Group

- Use the `rds-describe-db-security-groups` to view a DB Security Group.

```
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

```
https://rds.amazonaws.com/  
?Action=DescribeDBSecurityGroups  
&DBParameterGroupName=mydbsecuritygroup  
&Version=2009-10-16  
&SignatureVersion=2  
&SignatureMethod=HmacSHA256  
&Timestamp=2009-10-16T22%3A23%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 your DB Instances. If you want your applications to access your DB Instance, you can set your DB Security Group to allow access from specific EC2 Security Groups or CIDR IP ranges. 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

To avoid inadvertently granting access to your DB Instances, be sure to understand how CIDR ranges work. For more information about CIDR ranges, go to the [Wikipedia Tutorial](#).

In this 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 **DB Security Groups** from the navigation pane on the left side of the console window.
3. In the **My DB Security Groups** list, select the check box next to the DB Security Group that you want to authorize.
4. On the **Description** tab at the bottom of the window, 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 the **Add** button.



### Tip

The AWS Management Console displays the CIDR IP to use just below the CIDR text field if you want to authorize only the machine you are currently on.



### Note

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 was just 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`

### Example

```
https://rds.amazonaws.com/  
?CIDRIP=192.168.1.10%2F27  
&DBSecurityGroupName=mydbsecuritygroup  
&Version=2009-10-16  
&Action=AuthorizeDBSecurityGroupIngress  
&SignatureVersion=2  
&SignatureMethod=HmacSHA256  
&Timestamp=2009-10-22T17%3A10%3A50.274Z  
&AWSAccessKeyId=<AWS Access Key ID>  
&Signature=<Signature>
```

## Authorizing Network Access to a DB Instance from an EC2 Instance

If you want to access your DB Instance from an EC2 instance, you must configure your DB Instance's DB Security Group with an ingress rule that allows traffic from the EC2 Instance's EC2 Security Group. In this example, you add an ingress rule to a DB Security Group for an EC2 Security Group.



### Important

- Adding an ingress rule to a DB Security Group for an EC2 Security Group only grants access to your DB Instances from EC2 Instances belonging to that EC2 Security Group.
- You can't authorize an EC2 security group that is in a different AWS Region than your DB Instance. You can authorize an IP range, or specify an EC2 security group in the same region that refers to IP address in another region.

## AWS Management Console

### To grant access to an EC2 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 **DB Security Groups** from the navigation pane on the left side of the console window.
3. In the **My DB Security Groups** list, select the check box next to the DB Security Group named **default**.
4. On the **Description** tab at the bottom of the window, select *EC2 Security Group* from the **Connection Type** drop-down list.
5. Type the name of your EC2 Security Group into the **Security Group** text box.
6. Type your AWS Account ID into the **AWS Account ID** text box.



### Note

You can find your AWS Account ID on the [AWS Security Credentials](#) page.

7. Click the **Add** button.



### Note

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 was just modified. After the ingress rule has been successfully applied, the status will change to "authorized".

## CLI

### To grant access to an EC2 security group

- Use the command `rds-authorize-db-security-group-ingress` to grant access to an 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:

```
SECGROUP  Name      Description
SECGROUP  default  default
          EC2-SECGROUP  myec2group  987654321021  authorizing
```

## API

### To authorize network access to an EC2 security group

- Call `AuthorizeDBSecurityGroupIngress` with the following parameters:
  - `EC2SecurityGroupName` = `myec2group`
  - `EC2SecurityGroupOwnerId` = `987654321021`

### Example

```
https://rds.amazonaws.com/
?Action=AuthorizeDBSecurityGroupIngress
&EC2SecurityGroupOwnerId=987654321021
&EC2SecurityGroupName=myec2group
&Version=2009-10-16
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-22T17%3A10%3A50.274Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

## Revoking Network Access to a DB Instance from an IP Range

You can easily revoke network access from a CIDR IP range to DB Instances belonging to a DB Security Group by revoking the associated CIDR IP ingress rule.

In this example, you revoke an ingress rule for a CIDR IP 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 <https://console.aws.amazon.com/rds/>.
2. Select **DB Security Groups** from the navigation pane on the left side of the console window.
3. In the **My DB Security Groups** list, select the check box next to the DB Security Group that has the ingress rule you want to revoke.
4. In the **Description** tab under the **Actions** column next to the ingress rule you would like to revoke, click the **Remove** button.



### Note

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 was just changed. 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.

```
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

- [DB Security Groups \(p. 11\)](#)

## Monitoring DB Instances

The following scenarios cover operations for monitoring RDS instances.

### Topics

- [Viewing DB Instance Metrics \(p. 156\)](#)

## Viewing DB Instance Metrics

Amazon RDS and Amazon CloudWatch are integrated so you can gather a variety of metrics. You can monitor these metrics with Amazon CloudWatch.



### Note

The following examples require the Amazon CloudWatch command line tools. For more information on Amazon CloudWatch and to download the developer tools, go to the [Amazon CloudWatch product page](#).

For a complete list of Amazon RDS metrics, go to [Amazon RDS Dimensions and Metrics](#) in the [Amazon CloudWatch Developer Guide](#).

In this example, you use Amazon CloudWatch to gather storage space statistics for an Amazon RDS DB Instance for the past hour.



### Note

The `StartTime` and `EndTime` values supplied in the examples below are for illustrative purposes. You must substitute appropriate start and end time values for your DB Instance.

## AWS Management Console

### To gather disk storage 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. Click **DB Instances** in the **Navigation** list on the left side of the window. The available DB Instances appear in the **My DB Instances** list.



### Note

You can use the text box to the right of the **Viewing** drop-down list box to further filter your results.

3. Click on the DB Instance.
4. Click on the **Monitoring** tab in the information panel at the bottom 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

### To gather disk storage statistics for a DB Instance

- Use the Amazon CloudWatch command **mon-get-stats** with the following parameters:

```
PROMPT>mon-get-stats FreeStorageSpace --dimensions="DBInstanceIdentifier=mydbinstance" --statistics= Average
--namespace="AWS/RDS" --start-time 2009-10-16T00:00:00 --end-time
2009-10-16T00:02:00
```

## API

### To gather CPU utilization statistics for a DB Instance

- Call the Amazon 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=Average
&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

- [Creating and Modifying DB Instances \(p. 49\)](#)
- [DB Instance Monitoring \(p. 14\)](#)

# Working with Amazon RDS Events

The following scenarios cover operations on Amazon RDS events.

## Topics

- [Viewing Amazon RDS Events \(p. 159\)](#)

## 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).

### 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 **DB Events** in the **Navigation** list on the left side of the window. The available events appear in the **My DB Events** list.



#### Note

You can use the **Viewing** drop-down list box to filter the events by type, and you can use the text box to the right of the **Viewing** drop-down list box to further filter your results.

### 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 RDS events for the past 24 hours.

```
PROMPT>rds-describe-events --duration 1440
```

### API

#### To view all Amazon RDS instance events for the past 24 hours

- Call `DescribeEvents` with the following parameters:
  - `Duration = 1440`

### Example

```
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

- [DB Instance Monitoring \(p. 14\)](#)
- [Amazon RDS Events \(p. 15\)](#)

# Amazon RDS Technical FAQ

---

## Topics

- [General Information FAQ \(p. 161\)](#)
- [Billing \(p. 164\)](#)
- [Reserved Instances \(p. 165\)](#)
- [Multi-AZ Deployments \(p. 167\)](#)
- [Hardware and Scaling \(p. 170\)](#)
- [Automated Backups and Snapshots \(p. 172\)](#)
- [Security and VPC \(p. 173\)](#)
- [DB Parameter Groups \(p. 176\)](#)
- [Replication \(p. 177\)](#)
- [MySQL Database Engine \(p. 178\)](#)
- [Oracle Database Engine \(p. 184\)](#)
- [SQL Server Database Engine \(p. 188\)](#)

This section contains answers to commonly asked questions.

## General Information FAQ

### What is Amazon RDS?

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity, while managing time-consuming database administration tasks, freeing you up to focus on your applications and business.

Amazon RDS gives you access to the capabilities of a familiar MySQL or Oracle database. This means that the code, applications, and tools you already use today with your existing MySQL databases should work seamlessly with Amazon RDS. Amazon RDS automatically patches the database software and backs up your database, storing the backups for a user-defined retention period. You benefit from the flexibility of being able to scale the compute resources or storage capacity associated with your relational database instance via a single API call or few clicks of the AWS Management Console. In addition, Amazon RDS makes it easy to use replication (currently supported for MySQL only) to enhance database availability, improve data durability, or scale beyond the capacity constraints of a single database instance

for read-heavy database workloads. As with all Amazon Web Services, there are no up-front investments required, and you pay only for the resources you use.

#### **What is a database instance (DB Instance)?**

You can think of a DB Instance as a database environment in the cloud with the compute and storage resources you specify. You can create and delete DB Instances, define/refine infrastructure attributes of your DB Instance(s), and control access and security via the [AWS Management Console](#), Amazon RDS APIs, and Command Line Tools. Multiple MySQL databases or Oracle Database schemas can be created on a given DB Instance.

#### **What does Amazon RDS manage on my behalf?**

Amazon RDS manages the work involved in setting up a relational database, from provisioning the infrastructure capacity you request to installing the database software. Once your database is running on its own DB Instance, Amazon RDS automates common administrative tasks, such as performing backups and patching the database software that powers your DB Instance. For optional [Multi-AZ deployments \(p. 167\)](#), Amazon RDS also manages synchronous data replication across Availability Zones and automatic failover.

Since Amazon RDS provides native database access, you interact with the relational database software as you normally would. This means you're still responsible for managing the database settings that are specific to your application. You'll need to build the relational schema that best fits your use case and are responsible for any performance tuning to optimize your database for your application's workflow.

#### **When would I use Amazon RDS vs. Amazon EC2 Relational Database AMIs vs. Amazon SimpleDB?**

Amazon Web Services provides a number of database alternatives for developers. Amazon RDS enables you to run a fully featured relational database while offloading database administration; Amazon SimpleDB provides simple index and query capabilities with seamless scalability; and using one of our many relational database AMIs on Amazon EC2 and Amazon EBS allows you to operate your own relational database in the cloud. There are important differences between these alternatives that may make one more appropriate for your use case. See [Running Databases on AWS](#) for guidance on which solution is best for you.

#### **How do I get started with Amazon RDS?**

To sign up for Amazon RDS, you must click the "Sign up for Amazon RDS" button on the Amazon RDS detail page and complete the sign-up process. You must have an Amazon Web Services account; if you do not already have one, you will be prompted to create one when you begin the Amazon RDS sign-up process. After you are signed up for RDS, please refer to the [Amazon RDS documentation](#), which includes our [Getting Started Guide](#).

#### **How do I create a DB Instance?**

DB Instances are simple to create, using either the [AWS Management Console](#), Amazon RDS API, or command line interface (CLI). For information, see [Creating and Modifying DB Instances \(p. 49\)](#)

#### **How do I access my running DB Instance?**

Once your DB Instance is available, you can retrieve its endpoint via the DB Instance description in the [AWS Management Console](#) or DescribeDBInstance API. Using this endpoint you can construct the connection string required to connect directly with your DB Instance using your favorite database tool or programming language. In order to allow network requests to your running DB Instance, you will need to authorize access. For a detailed explanation of how to construct your connection string and get started, please refer to our [Getting Started Guide](#).

#### **How many DB Instances can I run with Amazon RDS?**

By default, customers are allowed to have up to a total of 20 Amazon RDS DB instances. Of those 20, up to 10 can be Oracle DB Instances under the "License Included" model. All 20 can be used for MySQL or Oracle under the "BYOL" model. If your application requires more DB Instances, you can request additional DB Instances via [this](#) request form.

### **How do I import data to Amazon RDS?**

There are a number of simple ways to import data into Amazon RDS, such as with the mysqldump or mysqlimport utilities for MySQL, and import/export or SQL Loader for Oracle. For more information on data import and export, please refer to the [Amazon RDS Data Import Guide for MySQL](#) or to the [Amazon RDS Data Import Guide for Oracle](#).

### **Which relational database engines does Amazon RDS support?**

Amazon RDS currently supports MySQL 5.1 and 5.5 (Community Edition) with InnoDB as the default database storage engine and Oracle Database 11gR2.

If you are using MySQL, you can use Amazon RDS MySQL DB Engine Version Management for optional control over the MySQL minor version of your DB Instance.

If you are using Oracle, you can use Amazon RDS Oracle DB Engine Version Management for optional control over the patch level of your DB Instance.

### **What storage engines does Amazon RDS for MySQL support?**

The Point-In-Time-Restore and Snapshot Restore features of Amazon RDS for MySQL require a crash recoverable storage engine and are supported for 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 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. However, if you still choose to use MyISAM with Amazon RDS, following [these steps](#) 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 [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.

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

### **What is a maintenance window? Will my DB Instance be available during software maintenance?**

You can think of the Amazon RDS maintenance window as an opportunity to control when DB Instance modifications (such as scaling DB Instance class) 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) 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 should seldom require 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 modify 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 \(p. 167\)](#) 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 using the APIs or command line interface to specify your maintenance window, please refer to the [Amazon RDS User Guide](#).

#### **What should I do if my queries seem to be running slow?**

If you are using MySQL, you can access the MySQL slow query logs for your database to determine if there are slow-running SQL queries and, if so, the performance characteristics of each. You could set the "slow\_query\_log" DB Parameter and query the mysql.slow\_log table to review the slow-running SQL queries. Please refer to the [Amazon RDS User Guide](#) to learn more about working with DB Parameter Groups.

If you are using Oracle, you can use the Oracle trace file data to identify slow queries. For more information on accessing trace file data, please refer to the [Amazon RDS User Guide](#).

You may also want to check the CPU utilization metrics for your DB Instance via [Amazon CloudWatch](#). High levels of CPU utilization can reduce query performance and in this case you may want to consider scaling your DB Instance class. For more information on monitoring your CPU utilization, read the [Amazon RDS Monitoring Guide](#).

## Billing

#### **How will I be charged and billed for my use of Amazon RDS?**

You pay only for what you use, and there are no minimum or setup fees. You are billed based on:

- DB Instance hours – Based on the class (e.g. Standard Small, Large, Extra Large) of the DB Instance consumed. Partial DB Instance hours consumed are billed as full hours.
- Storage (per GB per month) – Storage capacity you have provisioned to your DB Instance. If you scale your provisioned storage capacity within the month, your bill will be pro-rated.
- I/O requests per month – Total number of storage I/O requests you have.
- Backup Storage – Backup storage is the storage associated with your automated database backups and any active database snapshots you have taken. Increasing your backup retention period or taking additional database snapshots increases the backup storage consumed by your database. Amazon RDS provides backup storage up to 100% of your provisioned database storage at no additional charge. For example, if you have 1 0GB-months of provisioned database storage, we will provide up to 10GB-months of backup storage at no additional charge. Based upon our experience as database administrators, the vast majority of databases require less raw storage for a backup than for the primary data set, meaning that most customers will never pay for backup storage. Backup storage is only free for active DB Instances.
- Data transfer – Internet data transfer in and out of your DB Instance.

For Amazon RDS pricing information, please visit [the pricing section on the Amazon RDS product page](#).

#### **When does billing of my Amazon RDS DB Instances begin and end?**

Billing commences for a DB Instance as soon as the DB Instance is available. Billing continues until the DB Instance terminates, which would occur upon deletion or in the event of instance failure.

#### **What defines billable Amazon RDS instance hours?**

DB Instance hours are billed for each hour your DB Instance is running in an available state. If you no longer wish to be charged for your DB Instance, you must terminate it to avoid being billed for additional instance-hours. Partial DB Instance hours consumed are billed as full hours.

#### **Why does additional backup storage cost more than allocated DB Instance storage?**

The storage provisioned to your DB Instance for your primary data is located within a single Availability Zone. When your database is backed up, the backup data (including transactions logs) is geo-redundantly replicated across multiple Availability Zones to provide even greater levels of data durability. The price for backup storage beyond your free allocation reflects this extra replication that occurs to maximize the durability of your critical backups.

#### **How will I be billed for Multi-AZ DB Instance deployments?**

If you specify that your DB Instance should be a Multi-AZ deployment, you will be billed according to the Multi-AZ pricing posted on [the Amazon RDS pricing page](#). Multi-AZ billing is based on:

- Multi-AZ DB Instance Hours – Based on the class (e.g. Small, Large, Extra Large) of the DB Instance consumed. As with standard deployments in a single Availability Zone, partial DB Instance hours consumed are billed as full hours. If you convert your DB Instance deployment between standard and Multi-AZ within a given hour, you will be charged both applicable rates for that hour.
- Provisioned storage (for Multi-AZ DB Instance) – If you convert your deployment between standard and Multi-AZ within a given hour, you will be charged the higher of the applicable storage rates for that hour.
- I/O requests per month – Total number of storage I/O requests you have. Multi-AZ deployments consume a larger volume of I/O requests than standard DB Instance deployments, depending on your database write/read ratio. Write I/O usage associated with database updates will double as Amazon RDS synchronously replicates your data to the standby DB instance. Read I/O usage will remain the same.
- Backup Storage – Your backup storage usage will not change whether your DB Instance is a standard or Multi-AZ deployment. Backups will simply be taken from your standby to avoid I/O suspension on the DB Instance primary.
- Data transfer – You are not charged for the data transfer incurred in replicating data between your primary and standby.

#### **Do your prices include taxes?**

Except as otherwise noted, our prices are exclusive of applicable taxes and duties, including VAT and applicable sales tax. For example, our prices for the Asia Pacific (Tokyo) Region are inclusive of Japan consumption tax.

## Reserved Instances

#### **What are Amazon RDS Reserved Instances?**

With Reserved Instances, you can now make a one-time, up-front payment to create a one or three year reservation to run your DB Instance in a specific Region and receive a significant discount off of the ongoing hourly usage charge. There are three Reserved Instance types (Light, Medium, and Heavy Utilization Reserved Instances) that enable you to balance the amount you pay upfront with your effective hourly price.

#### **How are Reserved Instances different from On-Demand DB Instances?**

Functionally, Reserved Instances and On-Demand DB Instances are exactly the same. The only difference is how your DB Instance(s) are billed; with Reserved Instances, you make a one-time up-front payment

and receive a lower ongoing hourly usage rate (compared with On-Demand DB Instances) for the duration of the term.

#### **How do I purchase and create Reserved Instances?**

You can use the "Purchase Reserved DB Instance" option in the AWS Management Console. Alternatively, you can use the API tools, you can list the reservations available for purchase with the `DescribeReservedDBInstancesOfferings` API method and then purchase a DB Instance reservation by calling the `PurchaseReservedDBInstancesOffering` method.

Creating a Reserved Instance is no different than launching an On-Demand Instance. You simply use the `rds-create-db-instance` command, the `CreateDBInstance` API, or select the Launch DB Instance option from the AWS Management Console, specifying the DB Instance class and Region for which you made the reservation. So long as your reservation purchase was successful, Amazon RDS will apply the reduced hourly rate for which you are eligible to the new DB Instance.

#### **Will there always be reservations available for purchase?**

Yes. Reserved Instances are purchased for the Region rather than for the Availability Zone. This means that even if capacity is limited in one Availability Zone, reservations can still be purchased in that Region and used in a different Availability Zone within that Region.

#### **How many Reserved Instances can I purchase?**

You can purchase up to 20 Reserved DB Instances. If you wish to run more than 20 DB Instances please complete the [Amazon RDS DB Instance request form](#).

#### **What if I have an existing DB Instance that I'd like to convert to a Reserved Instance?**

Simply purchase a DB Instance reservation with the same DB Instance class, DB Engine and License Model within the same Region as the DB Instance you are currently running and would like to reserve. If the reservation purchase is successful, Amazon RDS will automatically apply your new hourly usage charge to your existing DB Instance.

#### **If I sign up for a Reserved Instance, when does the term begin? What happens to my DB Instance when the term ends?**

Pricing changes associated with a Reserved Instance are activated once your request is received while the payment authorization is processed. You can follow the status of your reservation on the AWS Account Activity page or by using the `DescribeReservedDBInstances` API. If the one-time payment cannot be successfully authorized by the next billing period, the discounted price will not take effect.

When your reservation term expires, your Reserved Instance will revert to the appropriate On-Demand hourly usage rate for your DB Instance class and Region.

#### **How do I control which DB Instances are billed at the Reserved Instance rate?**

The Amazon RDS APIs for creating, modifying, and deleting DB Instances do not distinguish between On-Demand and Reserved Instances so that you can seamlessly use both. When computing your bill, our system will automatically apply your Reservation(s), such that all eligible DB Instances are charged at the lower hourly Reserved DB Instance rate.

#### **If I scale my Reserved Instance class up or down, what happens to my reservation?**

Each reservation is associated with the following set of attributes: DB Engine, DB Instance class, Deployment type, License Model and Region. Each reservation can only be applied to a DB Instance with the same attributes for the duration of the term. If you decide to modify any of these attributes of your running DB Instance class before the end of the reservation term, your hourly usage rates for that DB Instance will revert to on demand hourly rates. If you later modify the running DB Instance's attributes

to match those of the original reservation, or create a new DB Instance with the same attributes as your original reservation, your reserved pricing will be applied to it until the end of your reservation term.

#### **Can I move a Reserved Instance from one Region or Availability Zone to another?**

Each Reserved Instance is associated with a specific Region, which is fixed for the lifetime of the reservation and cannot be changed. Each reservation can, however, be used in any of the available AZs within the associated Region.

#### **Are Reserved Instances available for Multi-AZ Deployments?**

Yes. When you call the `DescribeReservedDBInstancesOfferings` API, simply look for the Multi-AZ options listed among the DB Instance configurations available for purchase. If you want to purchase a reservation for a DB Instance with synchronous replication across multiple Availability Zones, specify one of these offerings in your `PurchaseReservedDBInstancesOffering` call.

#### **Are Reserved Instances available for Read Replicas?**

A standard DB Instance reservation can also be applied to a Read Replica, provided the DB Instance class and Region are the same. When computing your bill, our system will automatically apply your Reservation(s), such that all eligible DB Instances are charged at the lower hourly Reserved Instance rate.

#### **Can I cancel a reservation?**

The one-time payment for Reserved Instances is not refundable. However, you can choose to terminate your DB Instance at any time, at which point you will not incur any hourly usage charges.

#### **Can I exchange my existing Reserved Instances for the lower priced Reserved Instances that came out on March 5, 2012?**

Generally, Reserved Instances are nonrefundable; however, as a special exception, if you have purchased a 1-year Reserved Instance in the past 30 days or a 3-year Reserved Instance in the past 90 days, you can exchange it for one of the new Reserved Instances. When you exchange a Reserved Instance, upfront fees will be refunded on a pro rata basis, and hourly fees for instance-hours already used will not be refunded. If you would like to exchange a Reserved Instance for one of the new ones, please contact us.

## Multi-AZ Deployments

#### **What does it mean to run a DB Instance as a Multi-AZ deployment?**

When you create or modify your DB Instance to run as a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous “standby” replica in a different Availability Zone. Updates to your DB Instance are synchronously replicated across Availability Zones to the standby in order to keep both in sync and protect your latest database updates against DB Instance failure. During certain types of planned maintenance, or in the unlikely event of DB Instance failure or Availability Zone failure, Amazon RDS will automatically failover to the standby so that you can resume database writes and reads as soon as the standby is promoted. Since the name record for your DB Instance remains the same, your application can resume database operation without the need for manual administrative intervention. With Multi-AZ deployments, replication is transparent: you do not interact directly with the standby, and it cannot be used to serve read traffic. If you are looking to scale read traffic beyond the capacity constraints of a single DB Instance, you can deploy one or more [Read Replicas](#) (p. 178).



#### **Note**

Multi-AZ deployment is not currently supported for Microsoft SQL Server DB Instances.

### **What is an Availability Zone?**

Availability Zones are distinct locations within a Region that are engineered to be isolated from failures in other Availability Zones. Each Availability Zone runs on its own physically distinct, independent infrastructure, and is engineered to be highly reliable. Common points of failures like generators and cooling equipment are not shared across Availability Zones. Additionally, they are physically separate, such that even extremely uncommon disasters such as fires, tornados or flooding would only affect a single Availability Zone. Availability Zones within the same Region benefit from low-latency network connectivity.

### **What do "primary" and "standby" mean in the context of a Multi-AZ deployment?**

When you run a DB Instance as a Multi-AZ deployment, the "primary" serves database writes and reads. In addition, Amazon RDS provisions and maintains a "standby" behind the scenes, which is an up-to-date replica of the primary. The standby is "promoted" in failover scenarios. After failover, the standby becomes the primary and accepts your database operations. You do not interact directly with the standby (e.g. for read operations) at any point prior to promotion. The Amazon RDS team is working on a separate Read Replica feature to allow you to scale read traffic beyond the capacity constraints of a single DB Instance.

### **What are the benefits of a Multi-AZ deployment?**

The chief benefits of running your DB Instance as a Multi-AZ deployment are enhanced database durability and availability. The increased availability and fault tolerance offered by Multi-AZ deployments make them a natural fit for production environments.

Running your DB Instance as a Multi-AZ deployment safeguards your data in the unlikely event of a DB Instance component failure or loss of availability in one Availability Zone. For example, if a storage volume on your primary fails, Amazon RDS automatically initiates a failover to the standby, where all of your database updates are intact. This provides additional data durability relative to standard deployments in a single AZ, where a user-initiated restore operation would be required and updates that occurred after the latest restorable time (typically within the last five minutes) would not be available.

You also benefit from enhanced database availability when running your DB Instance as a Multi-AZ deployment. If an Availability Zone failure or DB Instance failure occurs, your availability impact is limited to the time automatic failover takes to complete (typically three minutes). The availability benefits of Multi-AZ also extend to planned maintenance. For example, with automated backups, I/O activity is no longer suspended on your primary during your preferred backup window, since backups are taken from the standby. In the case of patching or DB Instance class scaling, these operations occur first on the standby, prior to automatic fail over. As a result, your availability impact is limited to the time required for automatic failover to complete.

Another implied benefit of running your DB Instance as a Multi-AZ deployment is that DB Instance failover is automatic and requires no administration. In an Amazon RDS context, this means you are not required to monitor DB Instance events and initiate manual DB Instance recovery (via the `RestoreDBInstanceToPointInTime` or `RestoreDBInstanceFromSnapshot` APIs) in the event of an Availability Zone failure or DB Instance failure.

### **Are there any performance implications of running my DB Instance as a Multi-AZ deployments?**

You may observe elevated latencies relative to a standard DB Instance deployment in a single Availability Zone as a result of the synchronous data replication performed on your behalf.

### **When running my DB Instance as a Multi-AZ deployment, can I use the standby for read or write operations?**

No, the standby replica cannot serve read requests. Multi-AZ deployments are designed to provide enhanced database availability and durability, rather than read scaling benefits. As such, the feature uses synchronous replication between primary and standby. Our implementation makes sure the primary and the standby are constantly in sync, but precludes using the standby for read or write operations. If you are interested in a read scaling solution, please see the FAQs on Read Replicas.

### **How do I set up a Multi-AZ DB Instance deployment?**

In order to create a Multi-AZ DB Instance deployment, simply click the "Yes" option for "Multi-AZ Deployment" when launching a DB Instance with the AWS Management Console. Alternatively, if you are using the Amazon RDS APIs, you would call the `CreateDBInstance` API and set the "Multi-AZ" parameter to the value "true". To convert an existing standard (single AZ) DB Instance to Multi-AZ, modify the DB Instance in the AWS Management Console or use the `ModifyDBInstance` API and set the Multi-AZ parameter to true.

### **What events would cause Amazon RDS to initiate a failover to the standby replica?**

Amazon RDS Multi-AZ deployments will detect and automatically recover from the most common failure scenarios. The service will promote the standby to primary (which serves your DB Instance operations) in the event of any of the following:

- Loss of availability in primary Availability Zone
- Loss of network connectivity to primary
- Compute unit failure on primary
- Storage failure on primary
- Scaling of the compute class of your DB Instance up or down
- Software patching

Failover typically takes on the order of three minutes. Backups are taken from the standby without failover occurring.

### **Will I be alerted when automatic failover occurs?**

Yes, Amazon RDS will emit a DB Instance event to inform you that automatic failover occurred. You can use the `DescribeEvents` to return information about events related to your DB Instance, or click the "DB Events" section of the AWS Management Console.

### **What happens during Multi-AZ failover and how long does it take?**

Failover is automatically handled by Amazon RDS so that you can resume database operations as quickly as possible without administrative intervention. When failing over, Amazon RDS simply flips the canonical name record (CNAME) for your DB Instance to point at the standby, which is in turn promoted to become the new primary. We encourage you to follow best practices and implement database connection retry at the application layer. Failover times are a function of the time it takes MySQL crash recovery to complete. Start-to-finish, failover typically completes within three minutes.

### **Can I initiate a "forced failover" for my Multi-AZ DB Instance deployment?**

You can force a failover for your multi-AZ MySQL or Oracle deployment from the AWS Management Console, the command line interface, or the Amazon RDS API. For information, see [Rebooting a DB Instance \(p. 73\)](#). Forced failover is not currently supported for Microsoft SQL Server DB Instances.

### **How do I control/configure Multi-AZ synchronous replication?**

With Multi-AZ deployments, you simply set the “Multi-AZ” parameter to true. The creation of the standby, synchronous replication, and failover are all handled automatically. This means you cannot select the Availability Zone your standby is deployed in or alter the number of standbys available (Amazon RDS provisions one dedicated standby per DB Instance primary). The standby also cannot be configured to accept database read activity (click here for more details).

**Will my standby be in the same Region as my primary?**

Yes. Your standby is automatically provisioned in a different Availability Zone of the same Region as your DB Instance primary.

**Can I see which Availability Zone my primary is currently located in?**

Yes, you can gain visibility into the location of the current primary by using the AWS Management Console or DescribeDBInstances API.

**After failover, my primary is now located in a different Availability Zone than my other AWS resources (e.g. EC2 instances). Should I be concerned about latency?**

Availability Zones are engineered to provide low latency network connectivity to other Availability Zones in the same Region. In addition, you may want to consider architecting your application and other AWS resources with redundancy across multiple Availability Zones so your application will be resilient in the event of an Availability Zone failure. Multi-AZ deployments address this need for the database tier without administration on your part.

**How do DB Snapshots and automated backups work with my Multi-AZ deployment?**

You interact with automated backup and DB Snapshot functionality in the same way whether you are running a standard deployment in a single AZ or Multi-AZ deployment. If you are running a Multi-AZ deployment, automated backups and DB Snapshots are simply taken from the standby to avoid I/O suspension on the primary. There might be increased I/O latency, typically lasting a few minutes, during backups.

Initiating a restore operation (point-in-time restore or restore from DB Snapshot) also works the same with Multi-AZ deployments as standard, single AZ deployments. New DB Instance deployments can be created with either the RestoreDBInstanceFromSnapshot or RestoreDBInstanceToPointInTime APIs. These new DB Instance deployments can be either standard or Multi-AZ, regardless of whether the source backup was initiated on a standard or Multi-AZ deployment.

## Hardware and Scaling

**How do I determine which initial DB Instance class and storage capacity are appropriate for my needs?**

In order to select your initial DB Instance class and storage capacity, you will want to assess your application’s compute, memory and storage needs. For more guidelines on picking the right DB Instance class and storage capacity, please refer to [Sizing Your Amazon RDS DB Instance](#) in the [Amazon RDS User Guide](#).

**How do I scale the compute resources and/or storage capacity associated with my Amazon RDS Database Instance?**

You can scale the compute resources and storage capacity allocated to your DB Instance with the ModifyDBInstance action or the [AWS Management Console](#) (selecting the desired DB Instance and clicking the **Modify** button). Memory and CPU resources are modified by changing your DB Instance class, and storage available is changed when you modify your storage allocation. When you modify your DB Instance class or allocated storage, your requested changes will be applied during your specified

maintenance window. Alternatively, you can use the `apply-immediately` flag to apply your scaling requests immediately. Bear in mind that any other pending system changes will also be applied.

You can monitor the compute and storage resource utilization of your DB Instance, for no additional charge, by using Amazon CloudWatch. You can access metrics such as CPU utilization, storage utilization, and network traffic by clicking the **Monitoring** tab for your DB Instance in the [AWS Management Console](#) or by using the Amazon CloudWatch API. To learn more about monitoring your active DB Instances, go to the [Amazon RDS Monitoring Guide](#).

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](#).

### What is the hardware configuration for Amazon RDS storage?

Amazon RDS uses EBS volumes for database and log storage. Depending on the size of storage requested, Amazon RDS automatically stripes across multiple EBS volumes to enhance IOPS performance. Thus on an existing MySQL or Oracle DB Instance, you may observe some I/O capacity improvement if you scale up your storage. You can scale the storage capacity allocated to your DB Instance using the [AWS Management Console](#) or the [ModifyDBInstance API](#).

For SQL Server, because of the extensibility limitations of striped storage attached to Windows Server, Amazon RDS does not currently support increasing storage.

### Will my DB Instance remain available during scaling?

The storage capacity allocated to your DB Instance can be increased while maintaining DB Instance availability. However, when you decide to scale the compute resources available to your DB Instance up or down, your database will be temporarily unavailable while the DB Instance class is modified. This period of unavailability typically lasts only a few minutes, and will occur during the maintenance window for your DB Instance, unless you specify that the modification should be applied immediately.

### How can I scale my DB Instance beyond the largest DB Instance class and maximum storage capacity?

Amazon RDS supports a variety of DB Instance classes and storage allocations to meet different application needs. If your application requires more compute resources than the largest DB Instance class or more storage than the maximum allocation, you can implement partitioning, thereby spreading your data across multiple DB Instances.

### What are the minimum and maximum limits per DB Instance for Amazon RDS storage?

The maximum storage capacity per DB Instance for all DB engines is 1024 GB, except for SQL Server Express Edition, which limits storage to 10 GB per database, or 300 GB for a single DB instance. The minimum storage capacity for each DB Instance is as follows:

Engine	Minimum storage
MySQL	5 GB
Oracle	10 GB
SQL Server Express, SQL Server Web Edition	20 GB
SQL Server Standard Edition, SQL Server Enterprise Edition	200 GB

SQL Server Express Edition limits storage to 10 GB per database, which limits the overall usable database storage for SQL Express to 300 GB per DB Instance. Any additional storage that you provision with SQL Server Express may be unusable until you upgrade to a different edition of SQL Server.

You can scale up the storage capacity of a MySQL or Oracle DB Instance by adding at least 10 percent more storage per each increment.

## Automated Backups and Snapshots

### What is the difference between automated backups and DB Snapshots?

Amazon RDS provides two different methods for backing up and restoring your DB Instance(s) automated backups and database snapshots (DB Snapshots).

The automated backup feature of Amazon RDS enables point-in-time recovery of your DB Instance. When automated backups are turned on for your DB Instance, Amazon RDS automatically performs a full daily snapshot of your data (during your preferred backup window) and captures transaction logs (as updates to your DB Instance are made). When you initiate a point-in-time recovery, transaction logs are applied to the most appropriate daily backup in order to restore your DB Instance to the specific time you requested. Amazon RDS retains backups of a DB Instance for a limited, user-specified period of time called the retention period, which by default is one day but can be set to up to eight days. You can initiate a point-in-time restore and specify any second during your retention period, up to the Latest Restorable Time. You can use the DescribeDBInstances API to return the latest restorable time for you DB Instance(s), which is typically within the last five minutes. Alternatively, you can find the Latest Restorable Time for a DB Instance by selecting it in the [AWS Management Console](#) and looking in the "Description" tab in the lower panel of the Console.

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 [AWS Management Console](#) or CreateDBSnapshot API and are kept until you explicitly delete them with the [AWS Management Console](#) or DeleteDBSnapshot API.

The snapshots which Amazon RDS performs for enabling automated backups are available to you for copying (using the AWS console or the rds-copy-db-snapshot command) or for the snapshot restore functionality. You can identify them using the "automated" Snapshot Type. In addition, you can identify the time at which the snapshot has been taken by viewing the "Snapshot Created Time" field. Alternatively, the identifier of the "automated" snapshots also contains the time (in UTC) at which the snapshot has been taken.



#### Note

When you perform a restore operation to a point in time or from a DB Snapshot, a new DB Instance is created with a new endpoint (the old DB Instance can be deleted with the AWS Management Console or DeleteDBInstance API, if so desired). This is done to enable you to create multiple DB Instances from a specific DB Snapshot or point in time.

### Do I need to enable backups for my DB Instance or is it done automatically?

By default and at no additional charge, Amazon RDS enables automated backups of your DB Instance with a 1 day retention period. Free backup storage is limited to the size of your provisioned database and only applies to active DB Instances. For example, if you have 10GB-months of provisioned database storage, we will provide at most 10GB-months of backup storage at no additional charge. If you would like to extend your backup retention period beyond one day, you can do so using the CreateDBInstance API (when creating a new DB Instance) or ModifyDBInstance API (for an existing DB Instance). You can use these APIs to change the RetentionPeriod parameter from 1 to the desired number of days. For more information on automated backups, please refer to the [Amazon RDS User Guide](#).

### **What is a backup window and why do I need it? Is my database available during the backup window?**

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. This I/O suspension typically lasts a few minutes at most. This I/O suspension is avoided with Multi-AZ DB deployments, since the backup is taken from the standby.

### **Where are my automated backups and DB Snapshots, and how do I manage their retention?**

Amazon RDS DB snapshots and automated backups are stored in Amazon Simple Storage Service (Amazon S3).

You can use the [AWS Management Console](#) or [ModifyDBInstance API](#) to manage the period of time your automated backups are retained by modifying the RetentionPeriod parameter. If you desire to turn off automated backups altogether, you can do so by setting the retention period to 0 (not recommended). You can manage your user-created DB Snapshots via the DB Snapshots section of the AWS Management Console. Alternatively, you can see a list of the user-created DB Snapshots for a given DB Instance using the DescribeDBSnapshots API and delete snapshots with the DeleteDBSnapshot API.

### **What happens to my backups and DB Snapshots if I delete my DB Instance?**

When you delete a DB Instance, you have the ability to specify whether a final DB Snapshot is created upon deletion, which enables a DB Snapshot restore of the deleted database instance at a later date. All previously created DB Snapshots of your DB Instance will be retained and billed at \$0.15 per GB-month, unless you choose to delete them with the [DeleteDBSnapshot API](#).

## **Security and VPC**

### **What is Amazon Virtual Private Cloud (VPC) and why would I want to use with Amazon RDS?**

Amazon VPC lets you create a virtual networking environment in a private, isolated section of the Amazon Web Services (AWS) cloud, where you can exercise complete control over aspects such as private IP address ranges, subnets, routing tables and network gateways. With Amazon VPC, you can define a virtual network topology and customize the network configuration to closely resemble a traditional IP network that you might operate in your own datacenter.

One of the scenarios where you may want to use Amazon RDS in VPC is if you want to run a public-facing web application, while still maintaining non-publicly accessible backend servers in a private subnet. You can create a public-facing subnet for your web servers that has access to the Internet, and place your backend RDS DB Instances in a private-facing subnet with no Internet access. For more information about Amazon VPC, refer to the [Amazon Virtual Private Cloud User Guide](#).

### **How is using Amazon RDS inside a VPC different from using it outside?**

The basic functionality of Amazon RDS is the same regardless of whether VPC is used. Amazon RDS manages backups, software patching, automatic failure detection and recovery whether your DB Instances are deployed inside or outside a VPC. However, currently, only Amazon RDS for MySQL is supported in VPC. In addition, Read Replica functionality of Amazon RDS for MySQL is currently not supported in VPC.

Amazon RDS DB Instances deployed outside a VPC are assigned an external IP address (to which the Endpoint/DNS Name resolves) that provides connectivity from EC2 or the Internet. In Amazon VPC, Amazon RDS DB instances only have a private IP address (within a subnet that you define).

### **What is a DB Subnet Group and why do I need one?**

A DB Subnet Group is a collection of subnets that you may want to designate for your RDS DB Instances in a VPC. Each DB Subnet Group should have at least one subnet for every Availability Zone in a given Region. When creating a DB Instance in VPC, you will need to 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.



### Important

Since the underlying IP address can change (for example, during a failover), we strongly recommend you use the DNS Name to connect to your DB Instance s.

For Multi-AZ deployments, defining a subnet for all Availability Zones in a Region will allow 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.

### How do I create an Amazon RDS DB Instance in VPC?

For a walk-through example of creating a DB Instance in VPC, refer to the [Amazon RDS User Guide](#).

The following are the pre-requisites necessary to create a DB Instances within a VPC:

- You need to have a VPC set up with at least one subnet created in every Availability Zone in the Region you want to deploy your DB Instance. For information on creating Amazon VPC and subnets, go to the [Getting Started Guide for Amazon VPC](#).
- You need to have a DB Subnet Group defined for your VPC.
- You need to have a DB Security Group defined for your VPC (or you can use the default provided).
- In addition, you should allocate adequately large CIDR blocks to each of your subnets so that there are enough spare IP addresses for Amazon RDS to use during maintenance activities including scale compute and failover etc.

### How do I control network access to my DB Instance(s)?

Amazon RDS allows you to control access to your DB Instances using database security groups (DB Security Groups). A DB Security Group acts like a firewall controlling network access to your DB Instance. By default, network access is turned off to your DB Instances. If you want your applications to access your DB Instance, you can set your DB Security Group to allow access from EC2 Instances with specific EC2 Security Group/VPC Security Group membership or IP ranges. This process is called ingress. Once ingress is configured for a DB Security Group, the same rules apply to all DB Instances associated with that DB Security Group. DB Security Groups can be configured with the "DB Security Groups" section of the [AWS Management Console](#) or using the Amazon RDS APIs.



### Note

If you associate a VPC to a DB Security Group, all the access rules within that DB Security Group should be either from VPC Security Groups or IP ranges. EC2 Security Groups and VPC Security Groups are not interchangeable.

Whenever you create a DB Instance in VPC with a specific DB Security Group membership, Amazon RDS creates a corresponding VPC Security Group for that DB Instance for manageability. You can identify such VPC Security Groups via their description, which would be of the format "Security Group for RDS DB Instance". Please do not update them through the EC2/VPC console or the EC2/VPC APIs. Any changes you want to do for controlling access to your DB Instance should be done via the DB Security Groups.

### How do I secure Amazon RDS DB Instances running within my VPC?

DB Security Groups can be used to help secure DB Instances within an Amazon VPC. In addition, network traffic entering and exiting each subnet can be allowed or denied via network Access Control Lists (ACLs). All network traffic entering or exiting your VPC via your IPsec VPN connection can be inspected by your on-premise security infrastructure, including network firewalls, intrusion detection and prevention systems.

### How do I connect to an RDS DB Instance in VPC?

DB Instances deployed within a VPC can be accessed by EC2 Instances deployed in the same VPC. If these EC2 Instances are deployed in a public subnet with associated Elastic IPs, you can access the EC2 Instances via the internet.

DB Instances deployed within a VPC can be accessed from the Internet or from EC2 Instances outside the VPC via VPN or bastion hosts that you can launch in your public subnet. To use a bastion host, you will need to set up a public subnet with an EC2 instance that acts as a SSH Bastion. This public subnet must have an internet gateway and routing rules that allow traffic to be directed via the SSH host, which must then forward requests to the private IP address of your RDS DB instance.



#### Important

Since the underlying IP address can change (for example, during a failover), we strongly recommend you use the DNS Name to connect to your DB Instance s.

You can also set up a VPN Gateway that extends your corporate network into your VPC, and allows access to the RDS DB instance in that VPC. Refer to the [Amazon VPC User Guide](#) for more details.

### Can I move my existing DB instances outside VPC into my VPC?

You can take a snapshot of your DB Instance outside VPC and restore it to VPC by specifying the DB Subnet Group you want to use. Alternatively, you can perform a "Restore to Point in Time" operation as well.

### Can I move my existing DB instances from inside VPC to outside VPC?

Currently, direct migration of DB Instances from inside to outside VPC is not supported. For security reasons, a DB Snapshot of a DB Instance inside VPC cannot be restored to outside VPC. The same is true with "Restore to Point in Time" functionality. If you need to move your DB Instance from inside to outside VPC, you will need to export your data from your source DB Instance in your VPC to your target DB Instance deployed outside VPC.

### What precautions should I take to ensure that my DB Instances in VPC are accessible by my application?

You are responsible for modifying routing tables and networking ACLs in your VPC to ensure that your DB instance is reachable from your client instances in the VPC.

For Multi-AZ deployments, after a failover, your client EC2 instance and RDS DB Instance may be in different Availability Zones. You should configure your networking ACLs to ensure that cross-AZ communication is possible.

### Can I change the DB Subnet Group of my DB Instance?

An existing DB Subnet Group can be updated to add more subnets either for existing Availability Zones or for new Availability Zones added since the creation of the DB Instance. However, changing the DB Subnet Group of a deployed DB instance is not currently allowed.

### What is an Amazon RDS master user account and how is it different from an AWS account?

To begin using Amazon RDS you will need an AWS developer account. If you do not have one prior to signing up for Amazon RDS, you will be prompted to create one when you begin the sign-up process. A master user account is different from an AWS developer account and used only within the context of Amazon RDS to control access to your DB Instance(s). The master user account is a native database user account which you can use to connect to your DB Instance. You can specify the master user name and password you want associated with each DB Instance when you create the DB Instance. Once you have created your DB Instance, you can connect to the database using the master user credentials. Subsequently, you may also want to create additional user accounts so that you can restrict who can access your DB Instance.

#### **What privileges are granted to the master user for my DB Instance?**

For MySQL, the default privileges for the master user 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, trigger, create user, process, show databases, grant option.

For Oracle, the master user is granted the "dba" role. The master user inherits most of the privileges associated with the role. Please refer to the [Amazon RDS User Guide](#) for the list of restricted privileges and the corresponding alternatives to perform administrative tasks that may require these privileges.

#### **Is there anything different about user management with Amazon RDS?**

No, everything works the way you are familiar with when using a relational database you manage yourself.

#### **Can programs running on servers in my own data center access Amazon RDS databases?**

Yes. You have to intentionally turn on the ability to access your database over the Internet by granting permissions to your Amazon RDS DB Security Group(s) for access over the Internet. You can authorize access for only the specific IP's, IP ranges, or subnets corresponding to servers in your own data center.

#### **Can I encrypt connections between my application and my DB Instance using SSL?**

Yes; however, this option is currently only supported for the MySQL engine.

Amazon RDS generates an SSL certificate for each DB Instance. The [Amazon RDS User Guide](#) includes instructions for establishing an encrypted connection using the default mysql client. Once an encrypted connection is established, data transferred between the DB Instance and your application will be encrypted during transfer. If you require your data to be encrypted while "at rest" in the database, your application must manage the encryption and decryption of data. Also note that SSL support within Amazon RDS is for encrypting the connection between your application and your DB Instance; it should not be relied on for authenticating the DB Instance itself.

While SSL offers security benefits, be aware that SSL encryption is a compute intensive operation and will increase the latency of your database connection. To learn more about how SSL works with MySQL, go to the [MySQL documentation](#).

#### **Can I require my DB instance to accept only encrypted connections?**

For MySQL, after connecting to the DB Instance with the master username and password, 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
```

## DB Parameter Groups

### **How do I choose the right configuration parameters for my DB Instance(s)?**

Amazon RDS by default chooses the optimal configuration parameters for your DB Instance taking into account the DB Instance's compute resource and storage capacity. However, if you want to change them, you can do so using our configuration management APIs. Please note that changing configuration parameters from recommended values can have unintended effects, ranging from degraded performance to system crashes, and should only be attempted by advanced users who wish to assume these risks. For more information on changing parameters, please refer to the [Amazon RDS documentation](#).

#### **What are DB Parameter groups? How are they helpful?**

A database parameter group (DB Parameter Group) acts as a “container” for engine configuration values that can be applied to one or more DB Instances. If you create a DB Instance without specifying a DB Parameter Group, a default DB Parameter Group is used. This default group contains engine defaults and Amazon RDS system defaults optimized for the DB Instance you are running. However, if you want your DB Instance to run with your custom-specified engine configuration values, you can simply create a new DB Parameter Group, modify the desired parameters, and modify the DB Instance to use the new DB Parameter Group. Once associated, all DB Instances that use a particular DB Parameter Group get all the parameter updates to that DB Parameter Group. For more information on configuring DB Parameter Groups, go to the [Amazon RDS DB Parameter Group Deployment Guide](#).

#### **How do I see the current setting for my parameters for a given RDS DB Parameter Group?**

You can use the [AWS Management Console](#), Amazon RDS APIs, or Command Line Tools to see information about your DB Parameter Groups and their corresponding parameter settings.

## Replication

#### **What types of replication does Amazon RDS support and when should I use each?**

Amazon RDS supports MySQL and Oracle Database engines. A customer can choose the database engine that best meets their requirements.

Amazon RDS provides two distinct replication options to serve different purposes.

If you are looking to use replication to increase database availability while protecting your latest database updates against unplanned outages, consider running your DB Instance as a Multi-AZ deployment. When you create or modify your DB Instance to run as a Multi-AZ deployment, Amazon RDS will automatically provision and manage 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. Multi-AZ deployments utilize synchronous replication, making database writes concurrently on both the primary and standby so that the standby will be up-to-date in the event a failover occurs. While our technological implementation for Multi-AZ DB Instances maximizes data durability in failure scenarios, it precludes the standby from being accessed directly or used for read operations. The fault tolerance offered by Multi-AZ deployments make them a natural fit for production environments; to learn more about Multi-AZ deployments, please visit this [FAQ](#) section. Multi-AZ deployments are supported for the MySQL and Oracle database engines.

If you are looking to take advantage of MySQL's built-in replication to scale beyond the capacity constraints of a single DB Instance for read-heavy database workloads, Amazon RDS makes it easier with Read Replicas. You can create a Read Replica of a given “source” DB Instance using the AWS Management Console or `CreateDBInstanceReadReplica` API. Once the Read Replica is created, database updates on the source DB Instance will be propagated to the Read Replica. You can create multiple Read Replicas for a given source DB Instance and distribute your application's read traffic amongst them. Unlike Multi-AZ deployments, Read Replicas use MySQL's built-in replication and 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 (“asynchronous” replication), and replication lag can vary significantly. This means recent database updates made to a standard (non Multi-AZ) source DB Instance may not be present on associated Read

Replicas in the event of an unplanned outage on the source DB Instance. As such, Read Replicas do not offer the same data durability benefits as Multi-AZ deployments. While Read Replicas can provide some read availability benefits, they are not designed to improve write availability. Read Replicas are supported only for Amazon RDS for MySQL.

With Amazon RDS, you can use Multi-AZ deployments and Read Replicas in conjunction to enjoy the complementary benefits of each. You can simply specify that a given Multi-AZ deployment is the source DB Instance for your Read Replica(s). That way you gain both the data durability and availability benefits of Multi-AZ deployments and the read scaling benefits of Read Replicas.

## MySQL Database Engine

### Read Replicas

#### What does it mean to run a DB Instance as a Read Replica?

Read Replicas make it easy to take advantage of MySQL's built-in replication functionality to elastically scale out beyond the capacity constraints of a single DB Instance for read-heavy database workloads. You can create a Read Replica with a few clicks of the AWS Management Console or using the `CreateDBInstanceReadReplica` API. Once the Read Replica is created, database updates on the source DB Instance will be replicated using MySQL's native, asynchronous replication. You can create multiple Read Replicas for a given source DB Instance and distribute your application's read traffic amongst them. 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 associated 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.

#### When would I want to consider using an Amazon RDS Read Replica?

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.

#### Which storage engines are supported for use with Read Replicas?

Read Replicas require a transactional storage engine and are only supported for the InnoDB storage engine.

Non-transactional engines such as MyISAM might 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.

#### How do I deploy a Read Replica for a given DB Instance?

You can create a Read Replica in minutes using the standard `CreateDBInstanceReadReplica` API or a few clicks of the Amazon RDS Management Console. When creating a Read Replica, you can identify it as a Read Replica by specifying a `SourceDBInstanceIdentifier`. The `SourceDBInstanceIdentifier` is the DB Instance Identifier of the “source” DB Instance from which you wish to replicate. As with a standard DB Instance, you can also specify the Availability Zone, DB Instance class, and preferred maintenance window. The MySQL version (e.g. MySQL 5.1.50) and storage allocation of a Read Replica is inherited from the source DB Instance. When you initiate the creation of a Read Replica, Amazon RDS takes a 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 snapshot occurs. The I/O suspension typically lasts on the order of one minute, and is avoided if the source DB Instance is a Multi-AZ deployment (in the case of Multi-AZ deployments, snapshots are taken from the standby). Amazon RDS is also currently working on an optimization (to be released shortly) such that if you create multiple Read Replicas within a 30 minute window, all of them will use the same source snapshot to minimize I/O impact (“catch-up” replication for each Read Replica will begin after creation).

Amazon RDS Read Replicas are as easy to delete as they are to create; simply use the Amazon RDS Management Console or call the `DeleteDBInstance` API (specifying the `DBInstanceIdentifier` for the Read Replica you wish to delete).

When requesting the creation of a Read Replica, here are a couple of additional things to consider:

- 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 in order 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 and wait for them to complete. SELECT statements can continue running. 2. Flush and lock those tables. 3. Create the Read Replica using the `CreateDBInstanceReadReplica` API. 4. Check the progress of the Replica creation using the `DescribeDBInstances` API. Once the Replica is available unlock the tables and resume normal database operations.
- If there are any long running transactions on your source RDS instance, wait for them to complete before requesting creation of a Read Replica from that source.

### **How do I connect to my Read Replica(s)?**

You can connect to a Read Replica just as you would connect to a standard DB Instance, using the `DescribeDBInstance` API or AWS Management Console to retrieve the endpoint(s) for you Read Replica(s). If you have multiple Read Replicas, it is up to your application to determine how read traffic will be distributed amongst them.

### **How many Read Replicas can I create for a given source DB Instance?**

Amazon RDS allows you to create up to five (5) Read Replicas for a given source DB Instance.

### **Can I use a Read Replica to enhance database write availability or protect the data on my source DB Instance against failure scenarios?**

If you are looking to use replication to increase database write availability and protect recent database updates against various failure conditions, we recommend you run your DB Instance as a Multi-AZ deployment. With Read Replicas and MySQL's native, asynchronous replication, database writes occur on a Read Replica after they have already occurred on the source DB Instance, and this replication “lag” can vary significantly. In contrast, the replication used by Multi-AZ deployments is synchronous, meaning that all database writes are concurrent on the primary and standby. This protects your latest database updates, since they should be available on the standby in the event a failover is required. In addition, with Multi-AZ deployments replication is fully managed. Amazon RDS automatically monitors for DB Instance failure conditions or Availability Zone failure and initiates automatic failover to the standby if an outage occurs.

### **Can I create a Read Replica with a Multi-AZ DB Instance deployment as its source?**

Since Multi-AZ DB Instances address a different need than Read Replicas, it makes sense to use the two in conjunction for production deployments and to associate a Read Replica with a Multi-AZ DB Instance deployment. The “source” Multi AZ-DB Instance provides you with enhanced write availability and data durability, and the associated Read Replica would improve read traffic scalability.

**If my Read Replica(s) use a Multi-AZ DB Instance deployment as a source, what happens if Multi-AZ failover occurs?**

In the event of a Multi-AZ failover, any associated and available Read Replicas should automatically resume replication once failover has completed (acquiring updates from the newly promoted primary).

**My Read Replica appears “stuck” after a Multi-AZ failover and is unable to obtain or apply updates from the source DB Instance. What do I do?**

You may find in some cases that your Read Replica(s) aren’t able to receive or apply updates from their source Multi-AZ DB Instance after a Multi-AZ failover. This is because some MySQL binlog events were not flushed to disk at the time of the failover. After the failover, the Read Replica may ask for binlogs from the source that it doesn’t have. This loss of MySQL binlogs during a crash is described in the MySQL document [here](#).

Of particular relevance to this issue is the paragraph near the bottom that describes the MySQL `sync-binlog` parameter. This parameter controls how MySQL binlogs are flushed to disk, and when using InnoDB, how the binlogs and InnoDB logs may be kept in sync.

To resolve the current issue, you will need to delete the Read Replica and create a new one to replace it. To avoid this issue in the future, setting `sync-binlog=1` will greatly reduce the chance that MySQL binlogs needed by the Read Replica will be lost during a crash/failover scenario. As the MySQL documentation explains, even this doesn’t completely resolve the issue. To further reduce the likelihood of hitting this issue, set `innodb_support_xa=1`. Note that there may be a performance penalty for setting these variables. Both `sync_binlog` and `innodb_support_xa` are dynamic variables, so if you find that the performance penalty is too great, you can reset them without taking an outage.

This is ultimately a choice between performance and improving the automatic resynchronization of Read Replicas after a source Multi-AZ failover. One of the advantages of Amazon RDS Read Replicas is that they can be quickly re-instantiated when synchronization issues arise by deleting and re-creating them. As such, you don’t have to take the performance hit from setting `sync-binlog` and/or `innodb_support_xa` if manually dropping out of sync Read Replicas and re-creating them meets your needs.

**Can I create a Read Replica of another Read Replica?**

Creating a Read Replica of another Read Replica is not supported.

**Can my Read Replicas only accept database read operations?**

Read Replicas are designed to serve read traffic. However, there may be use cases where advanced users wish to complete Data Definition Language (DDL) SQL statements against a Read Replica. Examples might include adding a database index to a Read Replica that is used for business reporting, without adding the same index to the corresponding source DB Instance. If you wish to enable operations other than reads for a given Read Replica, you will need to modify the active DB Parameter Group for the Read Replica, setting the `read_only` parameter to 0.

**Can I convert my Read Replica into a primary DB Instance?**

Support for converting a Read Replica into a standard or Multi-AZ DB Instance deployment (“primary”) is not available at this time, but we expect to support this option in the future.

**Will my Read Replica be kept up-to-date with its source DB Instance?**

Updates to a source DB Instance will automatically be replicated to any associated Read Replicas. However, with MySQL's asynchronous replication technology, a Read Replica can fall behind its source DB Instance for a variety of reasons. Typical reasons include:

- Write I/O volume to the source DB Instance exceeds the rate at which changes can be applied to the Read Replica (this problem is particularly likely to arise if the compute capacity of a Read Replica is less than the source DB Instance)
- Complex or long-running transactions to the source DB Instance hold up replication to the Read Replica
- Network partitions or latency between the source DB Instance and a Read Replica

Read Replicas are subject to the strengths and weaknesses of MySQL replication. If you are using Read Replicas, you should be aware of the potential for lag between a Read Replica and its source DB Instance, or "inconsistency".

#### **How do I gain visibility into active Read Replica(s)?**

You can use the standard DescribeDBInstances API to return a list of all the DB Instances you have deployed (including Read Replicas), or simply click on the "DB Instances" tab of the AWS Management Console.

Amazon RDS allows you to gain visibility into how far a Read Replica has fallen behind its source DB Instance by issuing a standard "Show Slave Status" MySQL command against the Read Replica. The "Seconds\_Behind\_Master" data returned by "Show Slave Status" is also published as an Amazon CloudWatch metric ("Replica Lag") available via the AWS Management Console or Amazon Cloud Watch APIs.

#### **My Read Replica has fallen significantly behind its source DB Instance. What should I do?**

As discussed in the previous questions, "inconsistency" or lag between a Read Replica and its source DB Instance is common with MySQL asynchronous replication. If an existing Read Replica has fallen too far behind to meet your requirements, you can delete it and create a new one with the same endpoint by using the same DB Instance Identifier and Source DB Instance Identifier as the deleted Read Replica. Keep in mind that the re-creation process will be counter-productive at small lag levels (e.g. under five minutes of lag), and should be used with prudence (i.e. only when the Read Replica is significantly behind its source DB Instance). Also keep in mind that replica lag may naturally grow and shrink over time, depending on your source DB Instance's steady-state usage pattern.

Scaling the DB Instance class of a Read Replica may reduce replication lag in some cases, particularly if your source DB Instance class is larger than your Read Replica DB Instance class. However, Read Replicas are not guaranteed to work in all cases. There may be scenarios and usage patterns where a Read Replica can never catch up with its source after initial creation, or otherwise remains too far behind its source to meet your use case requirements.

#### **I scaled the compute and/or storage capacity of my source DB Instance, should I scale the resources for associated Read Replica(s) as well?**

For replication to work effectively, we recommend that Read Replicas have as much or more compute and storage resources as their respective source DB Instances. Otherwise replication lag is likely to increase or your Read Replica may run out of space to store replicated updates.

#### **Can I configure the replication between my source DB Instance and a Read Replica to use row-based replication?**

By default, replication is set to mixed-format (which includes both row-based and statement-based replication), which should meet the requirements of most use cases. However, if you are an advanced user and wish to designate that row-based replication is used, you can do so on a session basis by using a SQL command to set the binlog\_format for a DB Instance to "row." To learn more about the difference between mixed-format replication and row-based replication, go to the [MySQL documentation](#).

### **Can DB Snapshots or automated backups be taken of Read Replicas?**

No. If you are looking to increase database write availability by taking backups from your Read Replica instead of its source DB Instance, you can accomplish the same objective by running your DB Instance as a Multi-AZ deployment. Backups will then be initiated from the Multi-AZ standby to minimize availability impact.

### **How do I delete a Read Replica? Will it be deleted automatically if its source DB Instance is deleted?**

You can easily delete a Read Replica with a few clicks of the AWS Management Console or by passing its DB Instance identifier to the DeleteDBInstance API. A Read Replica will stay active and continue accepting read traffic even after its corresponding source DB Instance has been deleted. If you desire to delete the Read Replica in addition to the source DB Instance, you must explicitly delete the Read Replica using the DeleteDBInstance API or AWS Management Console.

### **Can I directly access the binary logs for my Database Instance to manage my own replication?**

Amazon RDS does not currently provide access to the binary logs for your Database Instance.

### **How much do Read Replicas cost? When does billing begin and end?**

A Read Replica is billed as a standard DB Instance and at the same rates. Just like a standard DB Instance, the rate per “DB Instance hour” for a Read Replica is determined by the DB Instance class of the Read Replica – please see [Amazon RDS detail page](#) for up-to-date pricing. You are not charged for the data transfer incurred in replicating data between your source DB Instance and Read Replica. Billing for a Read Replica begins as soon as the Read Replica has been successfully created (i.e. when status is listed as “active”). The Read Replica will continue being billed at standard Amazon RDS DB Instance hour rates until you issue a command to delete it.

## **DB Engine Version Management**

### **Can I control if and when the MySQL version powering Amazon RDS DB Instance is upgraded to new supported versions?**

Amazon RDS allows you to control if and when the relational database software powering your MySQL DB Instance is upgraded to new versions supported by Amazon RDS. This provides you with the flexibility to maintain compatibility with specific MySQL 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 MySQL minor versions as they are supported by Amazon RDS. This patching will occur during your scheduled maintenance window, and will be announced on the [Amazon RDS Community Forum](#) in advance. If you wish to turn off automatic version upgrades, you can do so by setting the AutoMinorVersionUpgrade parameter to “false.” Since major version upgrades involve some compatibility risk, they will not occur automatically and must be initiated by you.

This approach to database software patching puts you in the driver’s seat of version upgrades, but still offloads the work of patch application to Amazon RDS. You can learn more about version management by reading the FAQ entries that follow. Alternatively, you can reference our Developer Guide.

While DB Engine version management functionality is intended to give you as much control as possible over how patching occurs, Amazon RDS reserves the right to patch your DB Instance on your behalf in the event of a critical security vulnerability in the database software.

### **How do I specify which supported MySQL Version I would like my DB Instance to run?**

You can specify any currently supported version (minor and/or major) when creating a new DB Instance via the CreateDBInstance API. You simply pass in the desired version to the EngineVersion parameter upon create; 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 5.1) 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, simply use the DescribeDBEngineVersions API.

If you have opted out of automatically scheduled upgrades by setting the AutoMinorVersionUpgrade parameter to false but wish to manually initiate an upgrade to a supported minor version or major version release, you can do so using the ModifyDBInstance API. Simply specify the version you wish to upgrade to via the EngineVersion parameter. The upgrade will then be applied on your behalf either immediately (if the ApplyImmediately flag is set to true) or during the next scheduled maintenance window for your DB Instance.

#### **Can I test my DB Instance against a new version before upgrading?**

Yes. You can do so 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.

#### **How does Amazon RDS distinguish between major and minor version releases?**

In the context of MySQL, version numbers are organized as follows:

MySQL version = X.Y.Z

...where X denotes the major version, Y denotes the release level, and Z is the version number within the release series.

For Amazon RDS implementations, a version change would be considered major if either major version or release level is being changed; for example, going from version 5.1.x to 5.5.x. A version change would be considered minor if the version number within the release is being changed - for example, going from version 5.1.45 to version 5.1.49.

Amazon RDS currently supports the MySQL major versions MySQL 5.1 and MySQL 5.5. We plan to support additional major MySQL versions in the future.

#### **Does Amazon RDS provide guidelines for supporting new MySQL version releases and/or deprecating MySQL versions that are currently supported?**

Over time, we plan to support additional MySQL versions for Amazon RDS, both minor and major. 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.

In terms of deprecation policy:

- 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.

#### **How do I create a MySQL 5.5 DB Instance or upgrade my existing MySQL 5.1 DB Instance to MySQL 5.5?**

To create a new MySQL 5.5 DB Instance, use the **Launch DB Instance Wizard** in the AWS Management Console and select a version corresponding to MySQL 5.5, or simply call the `CreateDBInstance` API with the `EngineVersion` parameter of `5.5`.

A direct upgrade from MySQL 5.1 to MySQL 5.5 is not currently supported. However, we intend to provide this functionality in the future. Meanwhile, if you would like to port your existing MySQL 5.1 database to MySQL 5.5, you can use `mysqldump` to export your database from your existing MySQL 5.1 DB Instance and import it into a new MySQL 5.5 DB Instance.

## Oracle Database Engine

### Licensing and Support

#### What types of licensing options are available with Amazon RDS for Oracle?

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 & 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. DB Instances reside in the Amazon EC2 environment, and Oracle's licensing policy for Amazon EC2 is located [here](#).
- **License Included:** In the "License Included" service model, you do not need separately purchased Oracle licenses; the Oracle Database software has been licensed by AWS. "License Included" pricing is inclusive of software, underlying hardware resources, and Amazon RDS management capabilities.

#### Which Oracle Database Editions are available with Amazon RDS for Oracle?

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)
- License Included: Standard Edition One (SE1)

#### What are the licensing policies to use Amazon RDS for Oracle?

- BYOL: 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](#).
- License Included: In the "License Included" service model, you do not need separately purchased Oracle licenses; the Oracle Database software has been licensed by AWS.

#### How will Amazon RDS for Oracle be supported?

- BYOL: 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:** 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.

### **Can I change the licensing option for my DB Instance (e.g. from 'BYOL' to 'License Included')?**

Yes, you can change your license options. However, you will need to delete your current DB Instance with a final snapshot and create a new DB Instance from that snapshot specifying the new licensing option you need.

## **DB Engine Version Management**

### **What are Amazon RDS DB Engine Versions for Oracle and how do they relate to Oracle Patch Sets?**

In the context of Oracle, Amazon RDS DB Engine Versions are organized as follows:

DB Engine Versions for Oracle =  $x.y.z$

...where  $x$  denotes a major version (for example, 11.2),  $y$  is the release level (for example, 0.2), and  $z$  is the version number within the release series (for example: v2). For example, a DB Engine version for Oracle could be 11.2.0.2.v2

Oracle releases Database Patch Set Updates (PSU) for supported release levels on a quarterly basis. (e.g. 11.2.0.2.1). The PSUs include security fixes and additional non-security fixes recommended by Oracle.

The Amazon RDS DB Engine Versions are built with a given PSU as a baseline and may contain additional fixes beyond it.

For Amazon RDS implementations, a version change would be considered major if either major version or release level is being changed. Example: going from version 11.2.0.2.Z to version 11.2.0.4.Z. A version change would be considered minor if the version number within the release is being changed; for example, going from version 11.2.0.2.v2 to version 11.2.0.2.v3.

Amazon RDS currently supports the major version 11.2 (11g Release 2). We plan to support additional major versions in the future.

### **What is the patch set composition of my DB Engine Version for Oracle?**

Refer to the [Amazon RDS User Guide](#) for details of the patch set composition of each DB Engine Version of Oracle.

### **Can I control when my DB Instance could be upgraded to a new DB Engine Version for Oracle?**

Amazon RDS allows you to control if and when the relational database software powering your DB Instance is upgraded to new versions supported by Amazon RDS. This provides you with the flexibility to maintain compatibility with specific Oracle database 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 DB Engine Versions as they are supported by Amazon RDS. This patching will occur during your scheduled [maintenance window](#) (p. 163), and will be announced on the [Amazon RDS Community Forum](#) in advance. If you wish to turn off automatic version upgrades, you can do so by setting the "Auto Minor Version Upgrade" field to "No". Since major version upgrades involve some compatibility risk, they will not occur automatically and must be initiated by you.

This approach to database software patching puts you in the driver's seat of version upgrades, but still offloads the work of patch application to Amazon RDS. You can learn more about version management by reading the FAQ entries that follow. Alternatively, you can reference our Developer Guide.

While DB Engine version management functionality is intended to give you as much control as possible over how patching occurs, Amazon RDS may patch your DB Instance on your behalf in the event of a critical security vulnerability in the database software.

In the "License Included" model, the cost of the "Software Update License" is embedded in the hourly price, enabling access to Oracle Database software updates. However, under the BYOL model, you should have "Software Update License & Support" from Oracle to use Amazon RDS for Oracle Database.

#### **How do I specify which supported DB Engine Version I would like my DB Instance to run?**

You can specify any currently supported version ([minor and/or major \(p. 185\)](#)) when creating a new DB Instance via the **Launch DB Instance** option in the AWS Management Console or the `CreateDBInstance` API.

If you have opted out of automatically scheduled upgrades by setting the `AutoMinorVersionUpgrade` parameter to `false` but wish to manually initiate an upgrade to a supported minor version or major version release, you can do so using the `ModifyDBInstance` API. Simply specify the version you wish to upgrade to via the `EngineVersion` parameter. The upgrade will then be applied on your behalf either immediately (if the "Apply Immediately" flag is set) or during the next scheduled [maintenance window \(p. 163\)](#) for your DB Instance.

#### **Can I test my DB Instance against a new version before upgrading?**

Yes. You can do so 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.

#### **Does Amazon RDS provide guidelines for supporting new DB Engine Versions for Oracle and/or deprecating DB Engine Versions for Oracle that are currently supported?**

Over time, we plan to support additional Oracle database versions for Amazon RDS, both minor and major. The number of new version releases supported in a given year will vary based on the frequency and content of the Oracle Patch Set Updates (PSUs) 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 Oracle Database versions within 3-5 months of their General Availability.

In terms of deprecation policy:

- We intend to support major version releases for 3 years after they are initially supported by Amazon RDS.
- We intend to support minor Oracle Database versions (for example: 11.2.0.2.v2) for at least 1 year after they are initially supported by Amazon RDS.
- After a 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.

## Scaling

#### **Can I scale my DB Instance?**

For the BYOL model, you may scale your DB Instances in accordance with the terms of your Oracle license(s).

For the License Included model, DB Instances running Oracle may be scaled up and down at any point, subject to the prevailing hourly pricing for each DB Instance class.

For more information on the scaling implications of Reserved DB Instances, see our [Reserved Instances \(p. 165\)](#) FAQ.

### **Can I change the Oracle edition I'm running for a DB Instance (for example, from Oracle 11g R2 SE1 to EE)?**

For the BYOL model, you can migrate from one edition of Oracle software to another as long as you possess an unused Oracle license appropriate for the edition and class of DB Instance you plan to run. To change the edition and retain your data, you should take a snapshot of your running DB Instance and then create a new DB Instance of the desired edition from that snapshot. You should then delete the old DB Instance, unless you wish to keep it running and have the appropriate Oracle Database licenses.

For the License Included model, currently, only Oracle Standard Edition One is supported.

## **Options and Features**

### **What types of replication does Amazon RDS support for Oracle?**

Amazon RDS for Oracle supports Multi-AZ deployments for both the License Included and Bring Your Own License (BYOL) licensing models.

### **Does Amazon RDS use Oracle Data Guard for Multi-AZ deployments?**

Oracle Data Guard is a high-availability feature for the Enterprise Edition of the Oracle database engine. Amazon RDS currently uses a different synchronous replication technology and automatic failover functionality to provide Multi-AZ deployments for Oracle DB Instances. Multi-AZ deployments are available for all Oracle database editions that are supported by Amazon RDS.

### **Will I need additional licenses if I use Multi-AZ deployments for my Oracle DB Instances under the BYOL licensing model?**

Yes. We expect that you will need twice as many licenses for multi-AZ deployments as you would for a corresponding Single-AZ deployment to account for the standby DB Instance. You should review your Oracle software licensing agreement and comply with Oracle's licensing policies.

### **Is Oracle RAC supported on Amazon RDS?**

No, RAC is not currently supported.

### **Which Enterprise Edition Options are supported on Amazon RDS?**

The following Enterprise Edition Options are currently supported under the BYOL model:

- Partitioning
- Management Packs (Diagnostic, Tuning)
- Advanced Compression
- Total Recall

### **How do I know if Amazon RDS supports a specific Oracle Database feature?**

Oracle Database supports a number of [features](#) that vary with the edition of Oracle database you run. Refer to the [Amazon RDS User Guide](#) to know about the Oracle features that Amazon RDS currently supports.

## SQL Server Database Engine

### Licensing and Support

#### What types of licensing options are available with Amazon RDS for SQL Server?

Currently, the following licensing options are available for using Amazon RDS for SQL Server:

- **License Included:** In the "License Included" service model, you do not need separately purchased Microsoft SQL Server licenses. "License Included" pricing is inclusive of SQL Server software, underlying hardware resources, and Amazon RDS management capabilities.
- **License Mobility with Software Assurance (Bring Your Own License):** The Microsoft License Mobility program allows customers who already own SQL Server licenses to run SQL Server deployments on Amazon RDS. This benefit is available to Microsoft Volume Licensing (VL) customers covered by active Microsoft Software Assurance (SA) contracts. The Microsoft License Mobility program is suited for customers who prefer to use existing SQL Server licenses or purchase new licenses directly from Microsoft. Currently, SQL Server Standard Edition and SQL Server Enterprise Edition are eligible for this program; however, you should refer to Microsoft's [licensing policies](#) for the latest licensing terms.

#### How do I know if I am eligible to use License Mobility?

For information about License Mobility on Amazon Web Services, go to [Microsoft License Mobility through Software Assurance for Microsoft Windows Server Applications on AWS](#).

#### Which SQL Server versions and editions are available with Amazon RDS for SQL Server?

Currently, RDS for SQL Server supports SQL Server 2008 R2. The following Database Editions are available through RDS for SQL Server:

- **License Included:** SQL Server Express (no additional license cost), SQL Server Web and SQL Server Standard Edition.

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 [Microsoft Service Provider Usage Rights](#).

- **License Mobility (Bring Your Own License):** SQL Server Standard and SQL Server Enterprise Edition.

#### How will Amazon RDS for SQL Server be supported?

If you have an active AWS Premium Support account, you should contact AWS Premium Support for requests that are specific to either Amazon RDS or SQL Server. Amazon Web Services and Microsoft have multivendor support processes for cases that require assistance from both organizations.

#### Is there a limit on the number of SQL Server databases that I can create on one SQL Server DB Instance?

You can run a maximum of 30 databases on a single SQL Server DB Instance.

## Scaling

### Can I scale the storage of my Amazon RDS for SQL Server DB Instance?

Because of the extensibility limitations of striped storage attached to Windows Server, Amazon RDS does not currently support increasing storage. For the present, we recommend that you provision storage based on 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 import the data into it. For more information, go to the [Amazon RDS SQL Server Data Migration guide](#).

### Can I scale my DB Instance?

For the License Mobility (Bring Your Own License) model, you can scale your DB Instance according to the terms of your Microsoft SQL Server license.

For the License Included model, DB Instances that are running SQL Server can be scaled up and down at any point, subject to the prevailing hourly pricing for each DB Instance class. For more information on the scaling implications of Reserved DB Instances, see the [Reserved DB Instance FAQ](#).

### Can I change the SQL Server edition that I'm running on a DB Instance, for example, from SQL Server Web to SQL Server Standard Edition?

Yes. To change the edition and retain your data, you should take a snapshot of your running DB Instance and then create a new DB Instance of the desired edition from that snapshot. You should then delete the old DB Instance, unless you want to keep it running. If you keep it, you will continue to incur the associated charges.

If you are using the License Mobility model, you should review the Microsoft [Product Use Rights](#) for the licensing terms corresponding to the editions that you want to use.

## Options and Features

### What types of replication does Amazon RDS support for SQL Server?

Amazon RDS for MySQL supports Multi-AZ deployments and Read Replicas. Amazon RDS for Oracle supports Multi-AZ deployments, but not Read Replicas. These options are not currently available on Amazon RDS for SQL Server.

### Can I use Windows Authentication to connect to my Amazon RDS for SQL Server DB Instance?

No. Currently, only SQL Server Authentication is supported.

### What roles and permissions are available for Amazon RDS for SQL Server?

For information about roles and permissions, see [Roles and Permissions \(p. 28\)](#).

### Can I run SQL Server components such as Analysis, Reporting, Integration, or Master Data Services on my Amazon RDS for SQL Server DB Instance?

Not at present, but you can run these components on an Amazon EC2 instance pursuant to Microsoft's licensing policies and have them access your Amazon RDS for SQL Server DB Instance.

### How do I know if Amazon RDS supports a specific SQL Server Database feature?

SQL Server supports a number of features, depending on the edition of SQL Server that you are running. Amazon RDS for SQL Server supports most of the core database engine features, and you can connect to and work with your DB Instance from your favorite database clients, such as SQL Server Management Studio. For more information about currently supported features, see [Engine Features \(p. 26\)](#)

## DB Engine Version Management

### What are Amazon RDS DB engine versions for SQL Server, and how do they relate to SQL Server service packs and cumulative updates?

In the context of SQL Server, Amazon RDS DB Engine Versions are formatted as A.B.C.D.Z:

- A.B = Major version (for example, 10.50)
- C.D = Release level (for example, 2500.0)
- Z = version number within release series (for example: v2)

For example, a DB Engine version for SQL Server could be 10.50.2500.0.v2.

After the release of a major version (for example, SQL Server 2008 R2 or 10.50), Microsoft occasionally releases cumulative update packages (CUPs). In addition, the fixes in the CUPs are consolidated into Service Packs (SPs). The CUPs and SPs include security fixes and additional fixes that are recommended by Microsoft.

The Amazon RDS DB engine versions are built with a given major version or service pack as a baseline. They may also contain cumulative update packages (CUPs) and additional fixes.

For Amazon RDS, a version change would be considered major if either the major version or release level is being changed. For example, going from 10.50.1600.0.v1 to 10.50.2500.0.v1 is considered a major version change. A version change would be considered minor if the version number within the release is being changed, such as a change from 10.50.2500.0.v1 to 10.50.2500.0.v2.

Amazon RDS currently supports major version 10.5 (SQL Server 2008 R2). We plan to support additional major versions in the future.

### What is the patch set composition of my DB Engine Version for SQL Server?

At present, Amazon RDS supports SQL Server 2008 R2 with Cumulative update package 3 for SQL Server 2008 R2 Service Pack 1 (build number 10.50.2789.0). For information about this package, go to [Cumulative update package 3 for SQL Server 2008 R2 Service Pack 1](#).

### Can I control when my DB Instance is upgraded to a new DB engine version?

With Amazon RDS, you can control if and when the relational database software powering your DB Instance is upgraded to a new version that is supported by Amazon RDS. You have the flexibility to maintain compatibility with a specific SQL Server database version, to test a new version with your application before deploying in a production environment, and to perform version upgrades on your own terms and timelines.

Unless you specify otherwise, your DB Instance will be automatically upgraded to new DB Engine Versions as they are supported by Amazon RDS. These upgrades will occur during your scheduled maintenance window, and they will be announced on the Amazon RDS Forum in advance. If you want to turn off automatic version upgrades, you can do so by changing the Auto Minor Version Upgrade setting to No. Because major version upgrades involve some compatibility risk, they will occur only when you initiate them.

This approach to database software patching puts you in the driver's seat of version upgrades, but it still offloads the work of patch application to Amazon RDS.

Although DB engine version management in Amazon RDS is intended to give you as much control as possible over how patching occurs, Amazon RDS may patch your DB Instance on your behalf if the update resolves a critical security vulnerability in the database software. In addition, operating system patches are vetted by the Amazon RDS engineering team and may be applied to your DB Instances during your

maintenance window, even if you disabled automatic minor version upgrades. These maintenance events are typically announced two weeks in advance in the [Amazon RDS Forum](#).

#### **How do I specify which supported DB engine version I want my DB Instance to run?**

You can specify any currently supported major or minor version when you create a new DB Instance, whether you are using the AWS Management Console or the CreateDBInstance action.

If you have opted out of automatically scheduled upgrades but you want to manually initiate an upgrade to a supported minor or major version or major version, you can do so from the AWS Management Console by clicking the **DB Instances** link on the Amazon RDS console and using the **Modify** option corresponding to your DB Instance. Simply specify the version you wish to upgrade to with the **Engine Version** setting. The upgrade will then be applied on your behalf either immediately (if the **Apply Immediately** flag is set) or during the next scheduled [maintenance window](#) for your DB Instance.

#### **Can I test my DB Instance against a new version before upgrading?**

Yes. You can create a DB Snapshot of your existing DB Instance, restore from the DB Snapshot to create a new DB Instance, and then initiate a version upgrade for the new DB Instance. You can then experiment safely on the upgraded clone of your DB Instance before you decide whether to upgrade your production DB Instance.

#### **Does Amazon RDS provide guidelines for supporting new DB engine versions for SQL Server and/or deprecating supported versions?**

Over time, we plan to support additional major and minor SQL Server database versions for Amazon RDS. The number of new version releases supported in a given year will vary depending on the frequency and content of the SQL Server cumulative update packages (CUPs) and the outcome of a thorough vetting of the release by our database engineering team.

In terms of deprecation policy, our intentions are as follows:

- We intend to support major version releases for three years after they are initially supported by Amazon RDS.
- We intend to support minor SQL Server versions (e.g. 10.50.2500.0.v2) for at least one year after they are initially supported by Amazon RDS.
- After a 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 before we apply an automatic upgrade during your scheduled maintenance window.

# Appendix: Common DBA Tasks for MySQL

---

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. This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB Instances running the MySQL database engine.

## Managing Logs

When you create a MySQL DB Instance, both the general log and the slow query log are disabled. In order to enable logging, you create a DB parameter group and then set the `general_log` and `slow_query_log` server parameters to 1. For information about creating and modifying a DB parameter group, see [Working with DB Parameter Groups \(p. 136\)](#).

Setting `general_log` to 1 causes the `mysql.general_log` table to start accumulating all activity on the database. Similarly, setting `slow_query_log` to 1 causes the `mysql.slow_log` table to start accumulating all slow queries. A query is defined as slow if the time it takes to run exceeds the time specified by the `long_query_time` server variable. The default value is 10 seconds, but you can change it by modifying the DB parameter group that is associated with the DB Instance.

Log tables will keep growing until the respective parameters are disabled 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 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.

## Killing a Session or Query

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

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

With MySQL versions 5.1.62 and 5.5.23, 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

To verify that the error can be safely skipped, at the command prompt, you should first type the following command on the Read Replica:

```
PROMPT> show replica status\G
```

To skip the error, you can issue the following command:

```
PROMPT> mysql.rds_skip_repl_error;
```

This command has no effect if you run it on the master DB Instance or on a Read Replica that has not encountered a replication error.

If you are using a MySQL version older than 5.1.62 or 5.5.23, you can upgrade to the latest minor version within your major version of MySQL. For more information, see [Modifying a DB Instance \(p. 74\)](#).

## 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 \(p. 136\)](#).

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.

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

When GoSH is running, you can query the tables that it writes to. For example, to query the hit ratio of the InnoDB buffer pool, you would issue the following query:

```
select a.collection_end, a.collection_start, (( a.variable_Delta-
```

```
b.variable_delta)/a.variable_delta)*100 as "HitRatio"  
  from global_status_history as a join 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: Common DBA Tasks for Oracle

---

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. This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB Instances running the Oracle database engine.

## Enabling and disabling Restricted Session

Oracle Method	Amazon RDS Method
<code>alter system enable restricted session;</code>	<code>exec rdsadmin.rdsadmin_util.restricted_session(true);</code>
<code>alter system disable restricted session;</code>	<code>exec rdsadmin.rdsadmin_util.restricted_session(false);</code>

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

```
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

Oracle Method	Amazon RDS Method
<code>alter system flush shared_pool;</code>	<code>exec rdsadmin.rdsadmin_util.flush_shared_pool;</code>

## Flushing the Buffer Cache

Oracle Method	Amazon RDS Method
<code>alter system flush buffer_cache;</code>	<code>exec rdsadmin.rdsadmin_util.flush_shared_pool;</code>

## Checkpointing the Database

Oracle Method	Amazon RDS Method
<code>alter system checkpoint;</code>	<code>exec rdsadmin.rdsadmin_util.checkpoint;</code>

## Killing a Session

Oracle Method	Amazon RDS Method
<code>alter system kill session ' sid, serial#' IMMEDIATE;</code>	<code>exec rdsadmin.rdsadmin_util.kill(sid, serial#);</code>

## Switching Online Log files

Oracle Method	Amazon RDS Method
<code>alter system switch logfile;</code>	<code>exec rdsadmin.rdsadmin_util.switch_logfile;</code>

## Setting Default Tablespace

Oracle Method	Amazon RDS Method
<pre>alter database default tablespace users2;</pre>	<pre>exec rdsadmin.rdsadmin_util.alter_default_tablespace('users2');</pre>

## 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 filenames.

The following example creates a tablespace:

```
create tablespace users2;
```

The following example creates temporary tablespace:

```
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:

```
alter tablespace users2 resize 200M;
```

## 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. In cases where you want to resize their logs or add more logs, the same restrictions apply to naming physical files for 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#);
```

### Example

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

```
# 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 v$log 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.
```

## Accessing alertlogs and listenerlogs

The standard method for accessing alert logs, trace files and listener logs requires host access. Since Amazon RDS does not provide host access, the following methods to access these files are included with every DB Instance using the Oracle database engine:

To access the alert log, use the following command:

```
select message_text from alertlog;
```

To access the listener log, use the following command:

```
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. See [Workig with Trace Files \(p. 201\)](#) to access the older alertlogs and other files.

## Working with Tracefiles

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`:

```
exec rdsadmin.manage_tracefiles.refresh_tracefile_listing;
```

Once the view is refreshed, use the following view to access the results.

```
rdsadmin.tracefile_listing
```

## 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

Oracle Method	Amazon RDS Method
<code>oradebug hanganalyze 3</code>	<code>exec rdsadmin.manage_tracefiles.hanganalyze;</code>

### System State Dump

Oracle Method	Amazon RDS Method
<code>oradebug dump systemstate 266</code>	<code>exec rdsadmin.manage_tracefiles.dump_systemstate;</code>

## Retrieving Trace Files

You can retrieve any trace file in `background_dump_dest` using a standard SQL query of an 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.

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(minutes number);
```

The following example shows how to purge all files that match a specific pattern:

```
exec rdsadmin.manage_tracefiles.purge_tracefiles('MYTRACE*');
```

# Appendix: Common DBA Tasks for Microsoft 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. 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.

## Transition a Database from OFFLINE to ONLINE

SQL Server method	Amazon RDS method
ALTER DATABASE <i>name</i> SET ONLINE;	EXEC rdsadmin.dbo.rds_set_database_online <i>name</i>

## Access Error Logs

SQL Server method	Amazon RDS method
Use the Log File Viewer in SQL Server Management Studio.	EXEC rdsadmin.dbo.rds_read_error_log [ <i>index</i> ]

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.

## Manage trace files

This section describes Amazon RDS-specific procedures to create, refresh, access, and delete trace files.

## Generate a Trace SQL Query

```
declare @rc int
declare @TraceID int
declare @maxfilesize bigint
set @maxfilesize = 5
exec @rc = sp_trace_create @TraceID output, 0, N'D:\rdsdbdata\rdstest', @max
filesize, NULL
```

## View an Open Trace

```
select * from ::fn_trace_getinfo(default)
```

## View Trace Contents

```
select * from ::fn_trace_gettable('D:\rdsdbdata\rdstest.trc', default)
```

## Purge 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.

# 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.

Value	Description
AL32UTF8	Unicode 5.0 UTF-8 Universal character set (default)
JA16EUC	EUC 24-bit Japanese
JA16EUCTILDE	Same as JA16EUC except for mapping of wave dash and tilde to and from Unicode
JA16SJIS	Shift-JIS 16-bit Japanese
JA16SJISTILDE	Same as JA16SJIS except for mapping of wave dash and tilde to and from Unicode
KO16MSWIN949	Microsoft Windows Code Page 949 Korean
TH8TISASCII	Thai Industrial Standard 620-2533-ASCII 8-bit
VN8MSWIN1258	Microsoft Windows Code Page 1258 8-bit Vietnamese
ZHS16GBK	GBK 16-bit Simplified Chinese
ZHT16HKSCS	Microsoft Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001. Character set conversion is based on Unicode 3.0.
ZHT16MSWIN950	Microsoft Windows Code Page 950 Traditional Chinese
ZHT32EUC	EUC 32-bit Traditional Chinese
BLT8ISO8859P13	ISO 8859-13 Baltic

Value	Description
BLT8MSWIN1257	Microsoft Windows Code Page 1257 8-bit Baltic
CL8ISO8859P5	ISO 8859-5 Latin/Cyrillic
CL8MSWIN1251	Microsoft Windows Code Page 1251 8-bit Latin/Cyrillic
EE8ISO8859P2	ISO 8859-2 East European
EL8ISO8859P7	ISO 8859-7 Latin/Greek
EL8MSWIN1253	Microsoft Windows Code Page 1253 8-bit Latin/Greek
EE8MSWIN1250	Microsoft Windows Code Page 1250 8-bit East European
NE8ISO8859P10	ISO 8859-10 North European
NEE8ISO8859P4	ISO 8859-4 North and Northeast European
WE8ISO8859P15	ISO 8859-15 West European
US7ASCII	ASCII 7-bit American
WE8MSWIN1252	Microsoft Windows Code Page 1252 8-bit West European
AR8ISO8859P6	ISO 8859-6 Latin/Arabic
AR8MSWIN1256	Microsoft Windows Code Page 1256 8-bit Latin/Arabic
IW8ISO8859P8	ISO 8859-8 Latin/Hebrew
IW8MSWIN1255	Microsoft Windows Code Page 1255 8-bit Latin/Hebrew
TR8MSWIN1254	Microsoft Windows Code Page 1254 8-bit Turkish
WE8ISO8859P9	ISO 8859-9 West European and Turkish

# Appendix: Oracle DB Engine Patch Composition

---

This section provides information about Amazon RDS patch sets for the Oracle DB Engine.

## 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, 10018789, 9744252, 10248523, 9956713, 10356513, 9715581, 9770451, 10378005, 10170431, 10425676, 10222719, 10126094, 9591812, 10127360, 10132870, 10094201, 9443361, 10193846, 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, 11067567, 9881076, 10278372, 10040531, 10621169, 10155605, 10082277, 10356782, 10218814, 9078442, 9788588, 10157249, 9735237, 10317487, 12326246, 11707302, 10310299, 10636231, 10230571, 11065646, 12419321, 10368698, 10079168, 10013431, 10228151, 10233732, 10324526, 8223165, 10238786, 10217802, 10061015, 9953542, 9572787, 10052956, 10080579, 11699057, 12620422, 10332111, 10227288, 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

# Document History

This What's New is associated with the 2012-04-23 version of the Amazon Relational Database Service. This guide was last updated on 08 May 2012.

The following table describes the important changes since the last release of the *Amazon Relational Database Service User Guide*.

Change	Description	Release Date
New feature	Updated for Microsoft SQL Server support.	8 May 2012
New features	Updated for support for forced failover, Multi-AZ deployment of Oracle DB Instances, and nondefault character sets for Oracle DB Instances.	2 May 2012
New feature	Updated for Amazon Virtual Private Cloud (VPC) Support.	13 February 2012
Updated content	Updated for new Reserved Instance types.	19 December 2011
New feature	Updated for Oracle engine support.	23 May 2011
Updated content	Console updates.	13 May 2011
Updated content	Edited content for shortened backup and maintenance windows.	28 February 2011
New feature	Added support for MySQL 5.5.	31 January 2011
New feature	Added support for Read Replicas. For more information, see <a href="#">Read Replicas (p. 9)</a> .	04 October 2010
New feature	Added support for AWS Identity and Access Management (IAM). For more information, see <a href="#">AWS Identity and Access Management (p. 15)</a> .	02 September 2010
New feature	Added DB Engine Version Management. For more information, see <a href="#">DB Engine Version Management (p. 21)</a> .	16 August 2010
New feature	Added Reserved DB Instances. For more information, see <a href="#">Reserved DB Instances (p. 10)</a> .	16 August 2010

Change	Description	Release Date
New Feature	Amazon RDS now supports SSL connections to your DB Instances. For more information, see <a href="#">SSL Support (p. 20)</a> .	28 June 2010
New Guide	This is the first release of <i>Amazon Relational Database Service User Guide</i> .	07 June 2010

# Amazon RDS Resources

The following table lists related resources that you'll find useful as you work with this service.

Resource	Description
<a href="#">Amazon Relational Database Service Getting Started Guide</a>	The Getting Started Guide provides a quick tutorial on using the Amazon RDS console based on short, simple examples.
<a href="#">Amazon Relational Database Service API Reference</a>	The API Reference contains a comprehensive description of all Amazon RDS Query APIs and data types.
<a href="#">Amazon Relational Database Service Command Line Interface Reference</a>	The Command Line Tools Reference contains a comprehensive description of all the command line tools and their options.
<a href="#">Amazon RDS Technical FAQ</a>	The FAQ covers the top questions developers have asked about this product.
<a href="#">Release notes</a>	The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
<a href="#">AWS Developer Resource Center</a>	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
<a href="#">AWS Management Console</a>	The AWS Management Console allows you to perform most of the functions of Amazon RDS without programming.
<a href="#">Discussion Forums</a>	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
<a href="#">AWS Support Center</a>	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and Premium Support.
<a href="#">Amazon RDS product information</a>	The primary web page for information about Amazon RDS.
<a href="#">Contact Us</a>	A central contact point for inquiries concerning AWS billing, account, events, abuse etc.

Resource	Description
<a href="#">Conditions of Use</a>	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

# Glossary

---

## Glossary

---

Availability Zone	Amazon RDS uses the same locations as Amazon EC2. These locations are composed of Regions and Availability Zones.
DB engine	The database software and version running on the DB Instance.
DB name	For MySQL, this is the optionally supplied name for the initially created database on a DB Instance. For Oracle, this is the Oracle System ID (SID) used for connecting to the database on a DB Instance.
DB Instance	An isolated database environment running in the cloud. A DB Instance can contain multiple user-created databases, and can be accessed using the same tools and applications as a stand-alone database instance. Unlike a stand-alone database instance, an DB Instance cannot be accessed via an interactive login session (e.g. ssh).
DB Instance class	Used to specify the CPU and memory capacity of a DB Instance.
DB Instance identifier	Customer supplied identifier for the DB Instance that must be unique for that customer in an AWS region.
DB Parameter Group	A DB Parameter Group is a container for engine parameter values that can be applied to one or more DB Instances.
DB Security Group	A DB Security Group is a collection of ingress rules maintained by Amazon RDS. The two types of authorizations are IP ranges and EC2 security groups. IP range ingress authorizations allow access to DB Instances from the Internet. EC2 security group ingress authorizations allow access to DB Instances from EC2 instances.

DB Snapshot	A backup of a DB Instance at a point chosen by the user.
EC2 Compute Unit	An AWS standard for compute CPU and memory. One EC2 Compute Unit (ECU) provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.
endpoint	DNS name of a DB Instance.
endpoint port	Port number used by a DB Instance, see endpoint.
marker	See pagination
Multi-AZ deployment	A primary DB Instance that has a synchronous standby replica in a different Availability Zone. The primary DB Instance is synchronously replicated across Availability Zones to the standby replica.
pagination	APIs that return a potentially large list of records allow a subset to be viewed by using a <i>MaxRecords</i> and <i>Marker</i> value. The <i>MaxRecords</i> value corresponds to the maximum number of records to return, in the event <i>MaxRecords</i> is not specified a default is used. The <i>Marker</i> identifies the last record returned in the set of records (if the set is larger than <i>MaxRecords</i> ).
quota	The maximum number of DB Instances and available storage that can be used by a customer.
Single-AZ DB Instance	A standard (non-Multi-AZ) DB Instance that is deployed in one Availability Zone, without a standby replica in another Availability Zone.