Amazon RDS FAQs

* [General](#general)
* [Billing](#billing)
* [Free Tier](#free-tier)
* [Reserved Instances](#reserved-instances)
* [Hardware and Scaling](#hardware-scaling)
* [Automatic Backups and Database Snapshots](#automated-backups-database-snapshots)
* [Security](#security)
* [DB Parameter Groups](#db-parameter-groups)
* [Multi-AZ Deployments and Read Replicas](#Multi-AZ_Deployments)
* [Enhanced Monitoring](#enhanced-monitoring)
* [MySQL](#mysql)
* [Oracle](#oracle)

General [(back to top)](#Amazon_RDS_FAQs)

Q: What is Amazon RDS?

Amazon Relational Database Service (Amazon RDS) is a managed service that makes it easy to set up, operate, and scale a relational database in [the cloud](https://aws.amazon.com/what-is-cloud-computing/). 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, MariaDB, Oracle, SQL Server, or PostgreSQL database. This means that the code, applications, and tools you already use today with your existing 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](https://console.aws.amazon.com/). In addition, Amazon RDS makes it easy to use replication 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.

Q: 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](https://console.aws.amazon.com/), Amazon RDS APIs, and Command Line Tools. You can run [one or more DB Instances](https://aws.amazon.com/rds/faqs/#8), and each DB Instance can support [one or more databases or database schemas](https://aws.amazon.com/rds/faqs/#134), depending on engine type.

Q: 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](https://aws.amazon.com/rds/faqs/#36), 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.

Q: When would I use Amazon RDS vs. Amazon EC2 Relational Database AMIs vs. Amazon DynamoDB?

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 DynamoDB](https://aws.amazon.com/dynamodb/) is a fully managed NoSQL database service that offers fast and predictable performance with seamless scalability; and using one of our many relational database AMIs on [Amazon EC2](https://aws.amazon.com/ec2/) 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](https://aws.amazon.com/running_databases/) for guidance on which solution is best for you.

Q: How do I get started with Amazon RDS?

To sign up for Amazon RDS, you must have an Amazon Web Services account. [Create an account](https://portal.aws.amazon.com/gp/aws/developer/registration/index.html) if you do not already have one. After you are signed up, please refer to the [Amazon RDS documentation](http://aws.amazon.com/documentation/rds/), which includes our [Getting Started Guide](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.html).

You can launch a DB Instance within minutes by using the [AWS Management Console](https://console.aws.amazon.com/) or [Amazon RDS APIs](http://docs.aws.amazon.com/AmazonRDS/latest/APIReference/Welcome.html).

Q: How do I create a DB Instance?

DB Instances are simple to create, using either the [AWS Management Console](https://console.aws.amazon.com/), [Amazon RDS APIs](http://docs.aws.amazon.com/AmazonRDS/latest/APIReference/Welcome.html), or [Command Line Tools](http://docs.aws.amazon.com/AmazonRDS/latest/CommandLineReference/Welcome.html). To launch a DB Instance using the AWS Management Console, click "RDS," then the "Launch a DB Instance" button on the "Amazon RDS" tab. From there, you can specify the fundamental parameters for your DB instance:

* DB engine: Amazon Aurora, MariaDB, MySQL, Oracle, Microsoft SQL Server, PostgreSQL
* DB engine version (optional)
* License Model (optional)
* [DB Instance type](http://aws.amazon.com/rds/details/#DB_Instance_Classes)
* Amount of allocated storage (in GB)
* Whether your DB Instance should run as a Multi-AZ deployment
* Storage type
* DB Instance identifier
* Master user name
* Master user password

You also have the ability to change your DB Instance’s [backup retention policy](https://aws.amazon.com/rds/faqs/#26), [preferred backup window](https://aws.amazon.com/rds/faqs/#25), and [scheduled maintenance window](https://aws.amazon.com/rds/faqs/#12). Alternatively, you can create your DB Instance using the CreateDBInstance API or rds-create-db-instance command.

Q: 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](https://console.aws.amazon.com/) 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](https://aws.amazon.com/rds/faqs/#31). For a detailed explanation of how to construct your connection string and get started, please refer to our[Getting Started Guide](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.html).

Q: How many DB Instances can I run with Amazon RDS?

By default, customers are allowed to have up to a total of 40 Amazon RDS DB instances. Of those 40, up to 10 can be Oracle or SQL Server DB Instances under the ["License Included"](https://aws.amazon.com/rds/faqs/#111) model. All 40 can be used for Amazon Aurora, MySQL, MariaDB, Oracle, SQL Server, or PostgreSQL under the "BYOL" model. If your application requires more DB Instances, you can request additional DB Instances via [this](https://aws.amazon.com/contact-us/request-to-increase-the-amazon-rds-db-instance-limit/)request form.

Q: How many databases or schemas can I run within a DB Instance?

* RDS for Amazon Aurora: No limit imposed by software
* RDS for MySQL: No limit imposed by software
* RDS for MariaDB: No limit imposed by software
* RDS for Oracle: 1 database per instance; no limit on number of schemas per database imposed by software
* RDS for SQL Server: 30 databases per instance
* RDS for PostgreSQL: No limit imposed by software

Q: 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; Data Pump, import/export or SQL Loader for Oracle; Import/Export wizard or Bulk Copy Program (BCP) for SQL Server; or pg\_dump for PostgreSQL. For more information on data import and export, please refer to the [Data Import Guide for MySQL](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/MySQL.Procedural.Importing.html) or the [Data Import Guide for Oracle](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Oracle.Procedural.Importing.html) or the [Data Import Guide for SQL Server](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/SQLServer.Procedural.Importing.html) or the [Data Import Guide for PostgreSQL](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/PostgreSQL.Procedural.Importing.html).

Q: Which relational database engines does Amazon RDS support?

Amazon RDS supports Amazon Aurora, MySQL, MariaDB, Oracle, SQL Server, and PostgreSQL database engines.

Amazon RDS for MySQL currently supports MySQL 5.5, 5.6 and 5.7 (Community Edition) with InnoDB as the default database storage engine. Amazon RDS for MariaDB currently supports MariaDB 10.0 and 10.1. Amazon RDS for Oracle currently supports Oracle Database 11gR2 and 12c. Amazon RDS for SQL Server currently supports 2008 R2, SQL Server 2012 (SP2) and SQL Server 2014. Amazon RDS for PostgreSQL currently supports PostgreSQL 9.3, 9.4 and 9.5.

If you are using MySQL, you can use [Amazon RDS MySQL DB Engine Version Management](https://aws.amazon.com/rds/faqs/#75) 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](https://aws.amazon.com/rds/faqs/#118) for optional control over the patch level of your DB Instance.

If you are using SQL Server, you can use [Amazon RDS SQL Server DB Engine Version Management](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/RDSFAQ.SQLServer.html) for optional control over the patch level of your DB Instance.

Q: What is a maintenance window? Will my DB Instance be available during software maintenance?

The Amazon RDS maintenance window is your opportunity to control when DB Instance modifications (such as scaling DB Instance class) and software patching occur, in the event they 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 maintenance window you identify. Maintenance windows are 30 minutes in duration.

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](https://console.aws.amazon.com/) or by using the ModifyDBInstance API. Each of your DB Instances can have different preferred maintenance windows, if you so choose.

Running your DB Instance as a Multi-AZ deployment can further reduce the impact of a maintenance event, because Amazon RDS will perform the activity like so:

1. Perform the maintenance on the standby instance
2. Promote the standby to be the new primary
3. Perform maintenance on the old primary, which then 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 Developer Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/DeveloperGuide/). For more information on Multi-AZ mode deployments, [click here.](https://aws.amazon.com/rds/faqs/#36)

Q: Does Amazon RDS provide guidelines for support of new database engine versions, and for deprecating database engine versions that are currently supported?

This statement applies to Amazon RDS for Amazon Aurora, MySQL, MariaDB, Oracle, SQL Server, and PostgreSQL.

Over time, we plan to support additional database versions, both minor and major, for Amazon RDS's engines. The number of new version releases supported in a given year will vary based on the frequency and content of releases and patches from the engine’s vendor or core team, and the outcome of a thorough vetting of these releases and patches by our database engineering team. However, as a general guidance, we aim to support new engine versions within 3-5 months of their general availability.

Here is a general statement of Amazon RDS’s deprecation policy:

* We intend to support major version releases (e.g., MySQL 5.6) for at least 3 years after they are initially supported by Amazon RDS.
* We intend to support minor versions (e.g., MySQL 5.6.21) for at least 1 year after they are initially supported by Amazon RDS.
* From time to time, we will deprecate major or minor versions. We expect to provide a three-month grace period after the announcement of a deprecation for you to initiate an upgrade to a supported version. At the end of this grace period, an automatic upgrade will be applied to any un-upgraded instances during their scheduled maintenance windows.
* While we strive to meet these guidelines, in some cases we may deprecate specific major or minor versions sooner, such as when there are security issues.

Q: 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](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/Appendix.MySQL.CommonDBATasks.html) to learn more.

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 [Amazon RDS User Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.CommonDBATasks.html#Appendix.Oracle.CommonDBATasks.WorkingWithTracefiles).

If you are using SQL Server, you can use the client side SQL Server traces to identify slow queries. For information on accessing server side trace file data, please refer to [Amazon RDS User Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/Appendix.SQLServer.CommonDBATasks.html#Appendix.SQLServer.CommonDBATasks.WorkingWithTracefiles).

You may also want to check the CPU utilization metrics for your DB Instance via [Amazon CloudWatch](https://aws.amazon.com/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](http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2934).

Q: Why is the pricing for each RDS database engine different?

The pricing for each database engine of RDS varies because our costs are different for each. These costs include many operational components beyond software licensing. As we have demonstrated in the past with significant number of price cuts so far, we work hard to reduce costs and pass on those savings to our customers. We look forward to doing the same for our newer engines such as PostgreSQL in the future.

Billing [(back to top)](#Amazon_RDS_FAQs)

Q: 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 *(for Amazon RDS Magnetic Storage only)*
* Provisioned IOPS per month – Provisioned IOPS rate, regardless of IOPS consumed *(for Amazon RDS Provisioned IOPS (SSD) Storage only)*
* 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 10GB-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](https://aws.amazon.com/rds/pricing).

Q: 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.

Q: 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.

Q: 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.

Q: 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.](https://aws.amazon.com/rds/pricing) 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.

Q: 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 customers with a Japanese billing address, use of the Asia Pacific (Tokyo) Region is subject to Japanese Consumption Tax. [Learn more](https://aws.amazon.com/c-tax-faqs/).

Free Tier [(back to top)](#Amazon_RDS_FAQs)

Q: What does the AWS Free Tier for Amazon RDS offer?

The AWS Free Tier for Amazon RDS offer provides free use of Single-AZ Micro DB instances running MySQL, MariaDB, PostgreSQL, Oracle ("Bring-Your-Own-License (BYOL)" licensing model) and SQL Server Express Edition. The free usage tier is capped at 750 instance hours per month. Customers also receive 20 GB of database storage, 10 million I/Os and 20 GB of backup storage for free per month.

Q: For what time period will the AWS Free Tier for Amazon RDS be available to me?

New AWS accounts receive 12 months of AWS Free Tier access. Please see the [AWS Free Tier FAQs](https://aws.amazon.com/free/faqs/) for more information.

Q: Can I run more than one DB instance under the AWS Free Usage Tier for Amazon RDS?

Yes. You can run more than one Single-AZ Micro DB instance simultaneously and be eligible for usage counted under the AWS Free Tier for Amazon RDS. However, any use exceeding 750 instance hours, across all Amazon RDS Single-AZ Micro DB instances, across all eligible database engines and regions, will be billed at standard Amazon RDS prices.

For example: if you run two Single-AZ Micro DB instances for 400 hours each in a single month, you will accumulate 800 instance hours of usage, of which 750 hours will be free. You will be billed for the remaining 50 hours at the standard Amazon RDS price.

Q: Do I have access to 750 instance hours each of the MySQL, MariaDB, PostgreSQL, Oracle and SQL Server Micro DB instances under the AWS Free Tier?

No. A customer with access to the AWS Free Tier can use up to 750 instance hours of Micro instances running either MySQL, PostgreSQL, Oracle or SQL Server Express Edition. Any use exceeding 750 instance hours, across all Amazon RDS Single-AZ Micro DB instances, across all eligible database engines and regions, will be billed at [standard Amazon RDS prices](https://aws.amazon.com/rds/pricing/).

Q: Is the AWS Free Tier for Amazon RDS available in all AWS Regions?

The AWS Free Tier for Amazon RDS is available in all AWS Regions except GovCloud (US).

Q: How am I billed when my instance-hour usage exceeds the Free Tier benefit?

You are billed at standard Amazon RDS prices for instance hours beyond what the Free Tier provides. See the [Amazon RDS pricing page](https://aws.amazon.com/rds/pricing/)for details.

Reserved Instances [(back to top)](#Amazon_RDS_FAQs)

Q: What is a Reserved Instance (RI)?

Reserved Instances give you the option to reserve capacity within a datacenter and in turn receive a significant discount on the hourly charge for instances that are covered by the reservation. There are three RI payment options  -- No Upfront, Partial Upfront, All Upfront -- which enable you to balance the amount you pay upfront with your effective hourly price.

Q: 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 reserve capacity for a machine in a datacenter and in return receive a lower effective hourly usage rate (compared with On-Demand DB Instances) for the duration of the term.

Q: 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: 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.

Q: 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.

Q: How many Reserved Instances can I purchase?

You can purchase up to 40 Reserved DB Instances. If you wish to run more than 40 DB Instances, please complete the [Amazon RDS DB Instance request form](https://aws.amazon.com/contact-us/request-to-increase-the-amazon-rds-db-instance-limit/).

Q: 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.

Q: 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.

Q: 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.

Q: 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.

Q: 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.

Q: 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.

Q: 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.

Q: Can I cancel a reservation?

No, you cannot cancel your Reserved DB Instance and the one-time payment (if applicable) is not refundable. You will continue to pay for every hour during your Reserved DB Instance term regardless of your usage.

Q: How do the payment options impact my bill?

When you purchase an RI under the All Upfront payment option, you pay for the entire term of the RI in one upfront payment. You can choose to pay nothing upfront by choosing the No Upfront option. The entire value of the No Upfront RI is spread across every hour in the term and you will be billed for every hour in the term, regardless of usage. The Partial Upfront payment option is a hybrid of the All Upfront and No Upfront options. You make a small upfront payment, and you are billed a low hourly rate for every hour in the term regardless of usage.

Hardware and Scaling [(back to top)](#Amazon_RDS_FAQs)

Q: 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 the [Amazon RDS DB Instance Sizing Guide](http://aws.amazon.com/articles/2936?_encoding=UTF8&jiveRedirect=1).

Q: 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 API or the [AWS Management Console](https://console.aws.amazon.com/) (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. Please note that when you modify your DB Instance class or allocated storage, your requested changes will be applied during your specified maintenance window. Alternately, you can use the “apply-immediately” flag to apply your scaling requests immediately. Bear in mind that any other pending system changes will be applied as well.

Monitor the compute and storage resource utilization of your DB Instance, for no additional charge, via 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](https://console.aws.amazon.com/) or using the Amazon CloudWatch APIs. To learn more about monitoring your active DB Instances, read the[Amazon RDS Monitoring Guide](http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2934).

Please note that for SQL Server, because of the extensibility limitations of striped storage attached to a Windows Server environment, Amazon RDS does not currently support increasing storage. While we plan to support this functionality in the future, we recommend you to provision storage based on anticipated future storage growth. In the interim, 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. Please refer to the [data import guide for SQL Server](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/ImportData.SQLServer.html) for more information.

Q: 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. For MySQL and Oracle, for an existing 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](https://console.aws.amazon.com/), the [rds-modify-db-instance command](http://docs.aws.amazon.com/AmazonRDS/latest/CommandLineReference/CLIReference-cmd-ModifyDBInstance.html), or the [ModifyDBInstance API](http://docs.amazonwebservices.com/AmazonRDS/2010-07-28/APIReference/index.html?API_ModifyDBInstance.html).

However, for SQL Server, because of the extensibility limitations of striped storage attached to a Windows Server environment, Amazon RDS does not currently support increasing storage.

For more information, see [Storage for Amazon RDS](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html).

Q: 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.

Q: 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.

Q: What is Amazon RDS General Purpose (SSD) Storage?

Amazon RDS General Purpose (SSD) Storage is suitable for a broad range of database workloads that have moderate I/O requirements. With the baseline of 3 IOPS/GB and ability to burst up to 3,000 IOPS, this storage option provides predictable performance to meet the needs of most applications.

Q: What is Amazon RDS Provisioned IOPS (SSD) Storage?

Amazon RDS Provisioned IOPS (SSD) Storage is an SSD-backed storage option designed to deliver fast, predictable, and consistent I/O performance. With Amazon RDS Provisioned IOPS (SSD) Storage, you specify an IOPS rate when creating a DB Instance, and Amazon RDS provisions that IOPS rate for the lifetime of the DB Instance. Amazon RDS Provisioned IOPS (SSD) Storage is optimized for I/O-intensive, transactional (OLTP) database workloads. For more details, please see the [Amazon RDS User Guide](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html).

Q: What is Amazon RDS Magnetic Storage?

Formerly known as Standard storage, Amazon RDS Magnetic Storage is useful for small database workloads where data is accessed less frequently.

Q: How do I choose among the Amazon RDS storage types?

Choose the storage type most suited for your workload.

* High-performance OLTP workloads: Amazon RDS Provisioned IOPS (SSD) Storage
* Database workloads with moderate I/O requirements: Amazon RDS General Purpose (SSD) Storage
* Small database workloads with infrequent I/O: Amazon RDS Magnetic Storage

Q: What are the minimum and maximum IOPS supported by Amazon RDS?

The IOPS supported by Amazon RDS varies by database engine. For more details, please see the [Amazon RDS User Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/USER_PIOPS.html).

Automated Backups and Database Snapshots [(back to top)](#Amazon_RDS_FAQs)

Q: 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 thirty five 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](https://console.aws.amazon.com/) 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](https://console.aws.amazon.com/) or CreateDBSnapshot API and are kept until you explicitly delete them with the 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.

Please 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.

Q: 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 Developer Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/DeveloperGuide/).

Q: 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 few minutes). During the backup window, storage I/O may be briefly suspended while the backup process initializes (typically under a few seconds) and you may experience a brief period of elevated latency. There is no I/O suspension for Multi-AZ DB deployments, since the backup is taken from the standby.

Q: Where are my automated backups and DB Snapshots stored and how do I manage their retention?

Amazon RDS DB snapshots and automated backups are stored in S3.

You can use the [AWS Management Console](https://console.aws.amazon.com/) 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.

Q: What happens to my backups and DB Snapshots if I delete my DB Instance?

When you delete a DB Instance, you can create a final DB Snapshot upon deletion; if you do, you can use this DB Snapshot to restore the deleted DB Instance at a later date. Amazon RDS retains this final user-created DB Snapshot along with all other manually created DB Snapshots after the DB Instance is deleted. Refer to the [pricing page](https://aws.amazon.com/rds/pricing/) for details of backup storage costs.

Automated backups are deleted when the DB Instance is deleted. Only manually created DB Snapshots are retained after the DB Instance is deleted.

Security [(back to top)](#Amazon_RDS_FAQs)

Q: What is Amazon Virtual Private Cloud (VPC) and why may 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 webservers 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](http://docs.amazonwebservices.com/AmazonVPC/latest/UserGuide/).

Q: How is using Amazon RDS inside a VPC different from using it on the EC2-Classic platform (non-VPC)?

The basic functionality of Amazon RDS is the same regardless of whether VPC is used. Amazon RDS manages backups, software patching, automatic failure detection, read replicas and recovery whether your DB Instances are deployed inside or outside 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). You can configure a VPC to make an Amazon RDS DB instance in that VPC publicly accessible; see the [VPC documentation](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Internet_Gateway.html) for more information. For more information about the differences between EC2-Classic and EC2-VPC, see the [EC2 documentation](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-vpc.html#differences).

Q: 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.

Please note that, we strongly recommend you use the DNS Name to connect to your DB Instance as the underlying IP address can change (e.g., during a failover).

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.

Q: 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](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/USER_VPC.html).

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 refer to the [Getting Started Guide for Amazon VPC](http://docs.amazonwebservices.com/AmazonVPC/latest/GettingStartedGuide/).
* 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.

Q: How do I control network access to my DB Instance(s)?

Visit the [Security Groups](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.RDSSecurityGroups.html) section of the Amazon RDS User Guide to learn about the different ways to control access to your DB Instances.

Q: How do I secure Amazon RDS DB Instances running within my VPC?

VPC 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.

Q: 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, or using Amazon RDS's Publicly Accessible option:

* 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.
* To use public connectivity, simply create your DB Instances with the Publicly Accessible option set to yes. With Publicly Accessible active, your DB Instances within a VPC will be fully accessible outside your VPC by default. This means you do not need to configure a VPN or bastion host to allow access to your instances.

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](http://docs.amazonwebservices.com/AmazonVPC/latest/UserGuide) for more details.

We strongly recommend you use the DNS Name to connect to your DB Instance as the underlying IP address can change (e.g., during failover).

Q: 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.

Q: 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.

Q: 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.

Q: 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. Removing subnets from an existing DB Subnet Group can cause unavailability for instances if they are running in a particular AZ that gets removed from the subnet group.

At the present time, updating an existing DB Subnet Group does not change the current subnet of the deployed DB instance; an instance-type scale operation is required. Explicitly changing the DB Subnet Group of a deployed DB instance is not currently allowed.

Q: 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.

Q: 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](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/) for the list of restricted privileges and the corresponding alternatives to perform administrative tasks that may require these privileges.

For SQL Server, a user that creates a database is granted the "db\_owner" role. Please refer to the [Amazon RDS User Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/RDSFAQ.SQLServer.html) for the list of restricted privileges and the corresponding alternatives to perform administrative tasks that may require these privileges.

Q: 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.

Q: 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 configuring [Security Groups](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.RDSSecurityGroups.html). You can authorize access for only the specific IPs, IP ranges, or subnets corresponding to servers in your own data center.

Q: Can I encrypt connections between my application and my DB Instance using SSL?

Yes, this option is currently supported for the MySQL, MariaDB, SQL Server, PostgreSQL, and Oracle engines.

Amazon RDS generates an SSL certificate for each DB Instance. Once an encrypted connection is established, data transferred between the DB Instance and your application will be encrypted during transfer.

While SSL offers security benefits, be aware that SSL encryption is a compute-intensive operation and will increase the latency of your database connection. 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.

For details on establishing an encrypted connection with Amazon RDS, please visit Amazon RDS's [MySQL User Guide](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_MySQL.html#MySQL.Concepts.SSLSupport), [MariaDB User Guide](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_MariaDB.html" \l "MariaDB.Concepts.SSLSupport), [SQL Server User Guide](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.html#SQLServer.Concepts.General.SSL), [PostgreSQL User Guide](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_PostgreSQL.html#PostgreSQL.Concepts.General.SSL) or [Oracle User Guide](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Oracle.html#Oracle.Concepts.SSL). To learn more about how SSL works with these engines, you can refer directly to the [MySQL documentation](http://dev.mysql.com/doc/refman/5.6/en/ssl-connections.html), the [MariaDB documentation](https://mariadb.com/kb/en/mariadb/secure-connections-overview/), the [MSDN SQL Server documentation](http://msdn.microsoft.com/en-us/library/ms189067.aspx), the [PostgreSQL documentation](http://www.postgresql.org/docs/9.3/static/ssl-tcp.html), or the [Oracle Documentation](https://docs.oracle.com/database/121/DBSEG/asossl.htm)

Q: Can I require my DB instance to only accept 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

Q: Can I encrypt data at rest on my Amazon RDS databases?

Amazon RDS supports encryption at rest for all database engines, using keys you manage using [AWS Key Management Service (KMS)](https://aws.amazon.com/kms/). On a database instance running with Amazon RDS encryption, data stored at rest in the underlying storage is encrypted, as are its automated backups, read replicas, and snapshots. Encryption and decryption are handled transparently. For more information about the use of KMS with Amazon RDS, see the [Amazon RDS User's Guide](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html).

At the present time, encrypting an existing DB Instance is not supported. To use Amazon RDS encryption for an existing database, create a new DB Instance with encryption enabled and migrate your data into it.

Amazon RDS for Oracle and SQL Server support those engines' Transparent Data Encryption technologies. Transparent Data Encryption in Oracle is integrated with [AWS CloudHSM](https://aws.amazon.com/cloudhsm/), which allows you to securely generate, store, and manage your cryptographic keys in single-tenant Hardware Security Module (HSM) appliances within the AWS cloud. For more information, see the Amazon RDS User's Guide sections on [Oracle](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.Options.html#Appendix.Oracle.Options.AdvSecurity) and [SQL Server](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.SQLServer.Options.html#Appendix.SQLServer.Options.TDE).

Q: How do I control the actions that my systems and users can take on specific RDS resources?

You can control the actions that your [AWS IAM](https://aws.amazon.com/iam) users and groups can take on RDS resources. You do this by referencing the RDS resources in the [AWS IAM policies](http://docs.aws.amazon.com/IAM/latest/UserGuide/PermissionsAndPolicies.html) that you apply to your users and groups. RDS resources that can be referenced in an AWS IAM policy includes DB Instances, DB Snapshots, Read Replicas, DB Security Groups, DB Option Groups, DB Parameter Groups, Event Subscriptions and DB Subnet Groups. In addition, you can tag these resources to add additional metadata to your resources. By using tagging, you can categorize your resources (e.g. "Development" DB Instances, "Production" DB Instances, "Test" DB Instances etc), and write AWS IAM policies that list the permissions (i.e. actions) that can taken on resources with the same tags. For more information, refer to [Managing Access to Your Amazon RDS Resources and Databases](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAM.html) and [Tagging Amazon RDS Resources](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Tagging.html)

Q: I wish to perform security analysis or operational troubleshooting on my RDS deployment. Can I get a history of all RDS API calls made on my account?

Yes. AWS CloudTrail is a web service that records AWS API calls for your account and delivers log files to you. The AWS API call history produced by CloudTrail enables security analysis, resource change tracking, and compliance auditing. Learn more about CloudTrail at the [AWS CloudTrail detail page](https://aws.amazon.com/cloudtrail/), and turn it on via [CloudTrail's AWS Management Console home page](https://console.aws.amazon.com/cloudtrail/home).

DB Parameter Groups [(back to top)](#Amazon_RDS_FAQs)

Q: 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 User Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/USER_WorkingWithParamGroups.html).

Q: 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, please read the [DB Parameter Group Deployment Guide](http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2935).

Q: How do I see the current setting for my parameters for a given RDS DB Parameter Group?

You can use the [AWS Management Console](https://console.aws.amazon.com/), Amazon RDS APIs, or Command Line Tools to see information about your DB Parameter Groups and their corresponding parameter settings.

Multi-AZ Deployments and Read Replicas

Q: What types of replication does Amazon RDS support and when should I use each?

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](http://aws.amazon.com/rds/faqs/#multi-az-deployments).

Multi-AZ deployments are supported for the MySQL, Oracle, PostgreSQL, and SQL Server database engines. Multi-AZ deployments for SQL Server are presently available in the US East (Northern Virginia), US West (Oregon), and EU (Dublin) AWS Regions.

To help you to scale beyond the capacity constraints of a single DB Instance for read-heavy database workloads, Amazon RDS offers Read Replicas. You can create a Read Replica of a given source DB Instance using the AWS Management Console or the 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.

Read Replicas are supported by Amazon RDS for MySQL and PostgreSQL. Unlike Multi-AZ deployments, Read Replicas for these engines use each's built-in replication technology 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 and are not designed to improve write availability.

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.

Multi-AZ Deployments [(back to top)](#Amazon_RDS_FAQs)

Q: 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 using Amazon RDS for MySQL and are looking to scale read traffic beyond the capacity constraints of a single DB Instance, you can deploy one or more [Read Replicas](https://aws.amazon.com/rds/faqs/#87).

Q: 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.

Q: 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. If you are interested in scaling read traffic beyond the capacity constraints of a single DB Instance, please see the FAQs on [Read Replicas](https://aws.amazon.com/rds/faqs/#87).

Q: 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. 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.

Q: 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.

Q: 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](https://aws.amazon.com/rds/faqs/#87).

Q: 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](https://console.aws.amazon.com/). 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.

Q: What happens when I convert my RDS instance from Single-AZ to Multi-AZ?

For the RDS MySQL, MariaDB, PostgreSQL and Oracle database engines, when you elect to convert your RDS instance from Single-AZ to Multi-AZ, the following happens:

* A snapshot of your primary instance is taken
* A new standby instance is created in a different Availability Zone, from the snapshot
* Synchronous replication is configured between primary and standby instances

As such, there should be no downtime incurred when an instance is converted from Single-AZ to Multi-AZ.

Q: What events would cause Amazon RDS to initiate a failover to the standby replica?

Amazon RDS detects and automatically recovers from the most common failure scenarios for Multi-AZ deployments so that you can resume database operations as quickly as possible without administrative intervention. Amazon RDS automatically performs a failover 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

Note: When operations such as DB Instance scaling or system upgrades like OS patching are initiated for Multi-AZ deployments, for enhanced availability, they are applied first on the standby prior to an automatic failover. As a result, your availability impact is limited only to the time required for automatic failover to complete. Note that Amazon RDS Multi-AZ deployments do not failover automatically in response to database operations such as long running queries, deadlocks or database corruption errors.

Q: 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.

Q: 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.

Failovers, as defined by the interval between the detection of the failure on the primary and the resumption of transactions on the standby, typically complete within one to two minutes. Failover time can also be affected by whether large uncommitted transactions must be recovered; the use of adequately large instance types is recommended with Multi-AZ for best results. AWS also recommends the use of Provisioned IOPS with Multi-AZ instances, for fast, predictable, and consistent throughput performance.

Q: Can I initiate a “forced failover” for my Multi-AZ DB Instance deployment?

Amazon RDS will automatically failover without user intervention under a variety of [failure conditions](https://aws.amazon.com/rds/faqs/#43). In addition, Amazon RDS provides an option to initiate a failover when rebooting your instance. You can access this feature via the AWS Management Console or when using the RebootDBInstance API call.

Q: 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. [Learn more about Multi-AZ configurations.](https://aws.amazon.com/rds/details/multi-az/)

Q: 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.

Q: 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](https://console.aws.amazon.com/) or DescribeDBInstances API.

Q: 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.

Q: 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. Please note that you may experience increased I/O latency (typically lasting a few minutes) during backups for both Single-AZ and Multi-AZ deployments.

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.

Read Replicas

Q: What does it mean to run a DB Instance as a Read Replica?

Read Replicas make it easy to take advantage of supported engines' 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 in 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 a supported engine'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 supported engines' 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](https://aws.amazon.com/rds/faqs/#36) .

Q: 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.

Q: Do I need to enable automatic backups on my DB Instance before I can create read replicas?

Yes. Enable automatic backups on your DB Instance before adding Read Replicas, by setting the backup retention period to a value other than 0. Backups must remain enabled for Read Replicas to work.

Q: Which versions of database engines support Amazon RDS Read Replicas?

*Amazon RDS for MySQL*: Automatic backups must be and remain enabled for Read Replica operations. Automatic backups are supported only for Amazon RDS Read Replicas running MySQL 5.6 and later, not 5.1 or 5.5.

*Amazon RDS for PostgreSQL*: Only instances with PostgreSQL version 9.3.5 or newer support creation of Read Replicas. Existing PostgreSQL instances prior to version 9.3.5 need to be upgraded to PostgreSQL version 9.3.5 to take advantage of Amazon RDS Read Replicas.

*Amazon RDS for MariaDB*: Automatic backups must be and remain enabled for Read Replica operations.

Q: Which storage engines are supported for use with Amazon RDS for MySQL Read Replicas?

Amazon RDS for MySQL Read Replicas require a transactional storage engine and are only supported for the InnoDB storage engine. Non-transactional MySQL storage 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 CloudWatch 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.

Q: 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 engine version (e.g., PostgreSQL 9.3.5) 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.

Q: 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.

Q: How many Read Replicas can I create for a given source DB Instance?

Amazon RDS for MySQL and PostgreSQL currently allow you to create up to five (5) Read Replicas for a given source DB Instance.

Q: Can I create a Read Replica in an AWS Region different from that of the source DB Instance?

Amazon RDS for MySQL and PostgreSQL supports cross-region Read Replicas.

Q: Do Amazon RDS Read Replicas support synchronous replication?

No. Read Replicas in Amazon RDS for MySQL and PostgreSQL are implemented using those engines' native asynchronous replication.

Q: 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 Amazon RDS Read Replicas, which employ supported engines' 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.

Q: Can I create a Read Replica with a Multi-AZ DB Instance deployment as its source?

Yes. 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.

Q: Can I make my Amazon RDS Read Replicas themselves Multi-AZ?

Amazon RDS for MySQL and PostgreSQL do not presently support this.

Q: 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).

Q: My Amazon RDS for MySQL 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 Amazon RDS for MySQL 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.](http://dev.mysql.com/doc/refman/5.1/en/binary-log.html)

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.

Q: Can I create a Read Replica of another Read Replica?

*Amazon RDS for MySQL:* You can create a second-tier Read Replica from an existing first-tier Read Replica. By creating a second-tier Read Replica, you may be able to move some of the replication load from the master database instance to a first-tier Read Replica. Please note that a second-tier Read Replica may lag further behind the master because of additional replication latency introduced as transactions are replicated from the master to the first tier replica and then to the second-tier replica.

*Amazon RDS for PostgreSQL:* Read Replicas of Read Replicas are not currently supported.

Q: 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.

Amazon RDS for MySQL can be configured to permit DDL SQL statements against a Read Replica. If you wish to enable operations other than reads for a given Read Replica,  modify the active DB Parameter Group for the Read Replica, setting the “read\_only” parameter to “0.”

Amazon RDS for PostgreSQL does not currently support the execution of DDL SQL statements against a Read Replica.

Q: Can I promote my Read Replica into a “standalone” DB Instance?

Yes. Refer to the [Amazon RDS User Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/Overview.ReadReplica.html) for more details.

Q: 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 supported engines' 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 supported engines' native 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”. Click [here](https://aws.amazon.com/rds/faqs/#98) for guidance on what to do if your Read Replica(s) fall significantly behind its source.

Q: How do I see the status of my 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 Amazon RDS Management Console.

Amazon RDS allows you to gain visibility into how far a Read Replica has fallen behind its source DB Instance. The number of seconds that the Read Replica is behind the master is published as an Amazon CloudWatch metric ("Replica Lag") available via the AWS Management Console or Amazon CloudWatch APIs. For Amazon RDS for MySQL, the source of this information is the same as that displayed by issuing a standard "Show Slave Status" MySQL command against the Read Replica. For Amazon RDS for PostgreSQL, you can use the pg\_stat\_replication view on the source DB Instance to explore replication metrics.

Amazon RDS monitors the replication status of your Read Replicas and updates the Replication State field in the AWS Management console to "Error" if replication stops for any reason (e.g., attempting DML queries on your replica that conflict with the updates made on the master database instance could result in a replication error). You can review the details of the associated error thrown by the MySQL engine by viewing the Replication Error field and take an appropriate action to recover from it. You can learn more about troubleshooting replication issues in the Troubleshooting a Read Replica Problem section of the User Guide for Amazon RDS for [MySQL](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html#USER_ReadRepl.Troubleshooting) or [PostgreSQL](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html#USER_ReadRepl.TroubleshootingPostgreSQL).

If a replication error is fixed, the Replication State changes to Replicating.

Q: 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 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.

Q: 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.

Q: Can I configure the replication between my source Amazon RDS for MySQL DB Instance and a Read Replica to use row-based replication?

You can set the binary logging format to row-based, for MySQL version 5.6 and later. 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. The MySQL documentation offers more information about the [difference between mixed-format and row-based replication](http://dev.mysql.com/doc/refman/5.1/en/replication-formats.html).

Q: 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.

Q: 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.

An Amazon RDS for MySQL 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 do so using the DeleteDBInstance API or AWS Management Console.

If you delete an Amazon RDS for PostgreSQL DB Instance that has Read Replicas, all Read Replicas will be promoted to standalone DB Instances and will be able to accept both read and write traffic. The newly promoted DB Instances will operate independently of one another. If you desire to delete these DB Instances in addition to the original source DB Instance, you must explicitly do so using the DeleteDBInstance API or AWS Management Console.

Q: Can I directly access the event logs for my Database Instance to manage my own replication?

Amazon RDS for MySQL does not currently provide access to the binary logs for your Database Instance. Similarly, Amazon RDS for PostgreSQL does not currently provide access to the WAL files for your Database Instance.

Q: 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. Click here for more information on DB Instance billing visit this[FAQ](https://aws.amazon.com/rds/faqs/#15). 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](https://aws.amazon.com/rds/pricing/) 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.

Q: How does support for Read Replicas vary among the Amazon RDS engines that support this feature?

Read Replicas in both Amazon RDS for PostgreSQL and MySQL allow you to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. There are similarities and differences in the implementations, because they leverage native engine features. See the following table for details.

|  |  |  |
| --- | --- | --- |
| Feature | PostgreSQL | MySQL |
| Maximum Read Replicas allowed per source DB Instance | 5 | 5 |
| Replication method | Asynchronous Physical | Asynchronous Logical |
| Must automatic backups be enabled for Read Replica support? | Yes | Yes |
| Engine versions for which Read Replicas are available | 9.3.5 or later | 5.6 or later |
| Promotion of Read Replica to a new standalone DB Instance | Supported | Supported |
| Creation of Indexes on Read Replica | Currently not supported | Supported |
| Creation of Backups of Read Replicas | Currently not supported | Supported |
| Chaining of Read Replicas (i.e., Read Replicas of Read Replicas) | Currently not supported | Supported |
| Cross-Region Read Replicas | Supported | Supported |

For information about replication support with the Amazon Aurora engine, see the [Amazon RDS for Aurora FAQ](https://aws.amazon.com/rds/aurora/faqs/).

Enhanced Monitoring

Q: What is Enhanced Monitoring for RDS?

A: Enhanced Monitoring for RDS gives you deeper visibility into the health of your RDS instances. Just turn on the “Enhanced Monitoring” option for your RDS instance and set a granularity and Enhanced Monitoring will collect vital operating system metrics and process information, at the defined granularity.

Q: Which metrics and processes can I monitor in Enhanced Monitoring?

A: Enhanced Monitoring captures your RDS instance system level metrics such as the CPU, memory, file system and disk I/O among others. The complete list of metrics can be found [here](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Monitoring.html).

Q: Which engines are supported by Enhanced Monitoring?

A: Enhanced Monitoring supports all RDS database engines.

Q: Which instance types are supported by Enhanced Monitoring?

A: Enhanced Monitoring supports every instance type except t1.micro and m1.small. The software uses a small amount of CPU, memory and I/O and for general purpose monitoring, we recommend switching on higher granularities for instances that are medium or larger. The default setting for Enhanced Monitoring is “off”, and you have the choice of leaving it disabled or modifying the granularity when it is on.

Q: What information can I view on the RDS dashboard?

A: You can view all the system metrics and process information for your RDS instances in a graphical format on the console. You can manage which metrics you want to monitor for each instance and customize the dashboard according to your requirements.

Q: Will all the instances in my RDS account sample metrics at the same granularity?

A: No. You can set different granularities for each instance in your RDS account. You can also choose the instances on which you want to enable Enhanced Monitoring as well as modify the granularity of any instance whenever you want.

Q: How far back can I see the historical metrics on the RDS console?

A: You can see the performance values for all the metrics for up to 1 hour ago, at a granularity of up to 1 second based on your setting.

Q: How can I visualize the metrics generated by RDS Enhanced Monitoring in CloudWatch?

A: The metrics from RDS Enhanced Monitoring are delivered into your CloudWatch Logs account. You can create metrics filters in CloudWatch from CloudWatch Logs and display the graphs on the CloudWatch dashboard. For more details, please visit the [Amazon CloudWatch](https://aws.amazon.com/cloudwatch/pricing/) page.

Q: When should I use CloudWatch instead of the RDS console dashboard?

A: You should use CloudWatch if you want to view historical data beyond what is available on the RDS console dashboard. You can monitor your RDS instances in CloudWatch to diagnose the health of your entire AWS stack in a single location. Currently, CloudWatch supports granularities of up to 1 minute and the values will be averaged out for granularities less than that.

Q: Can I set up alarms and notifications based on specific metrics?

A: Yes. You can create an alarm in CloudWatch that sends a notification when the alarm changes state. The alarm watches a single metric over a time period that you specify, and performs one or more actions based on the value of the metric relative to the specified threshold over a number of time periods. For more details on CloudWatch alarms, please visit the [Amazon CloudWatch Developer Guide](http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/AlarmThatSendsEmail.html).

Q: How do I integrate Enhanced Monitoring with my tool that I currently use?

A: RDS Enhanced Monitoring provides a set of metrics formed as JSON payloads which are delivered into your CloudWatch Logs account. The JSON payloads are delivered at the granularity last configured for the RDS instance.

There are two 2 ways you can consume the metrics via a third-party dashboard or application. Monitoring tools can use [CloudWatch Logs Subscriptions](http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/Subscriptions.html) to set up a near real time feed for the metrics. Alternatively, you can use filters in CloudWatch Logs to bridge metrics across to CloudWatch to and integrate your application with CloudWatch. Please visit [Amazon CloudWatch Documentation](http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/CWL_ES_Stream.html) for more details.

Q: How can I delete historical data?

A: Since Enhanced Monitoring delivers JSON payloads into a log in your CloudWatch Logs account, you can control its retention period just like any other CloudWatch Logs stream. The default retention period configured for Enhanced Monitoring in CloudWatch Logs is 30 days. For details on how to change retention settings, please visit [Amazon CloudWatch Developer Guide](http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/WhatIsCloudWatch.html).

Q: What impact does Enhanced Monitoring have on my monthly bills?

A: Since the metrics are ingested into CloudWatch Logs, your charges will be based on CloudWatch Logs data transfer and storage rates once you exceed CloudWatch Logs free tier. Pricing details can be found [here](https://aws.amazon.com/cloudwatch/pricing/). The amount of information transferred for an RDS instance is directly proportional to the defined granularity for the Enhanced Monitoring feature. Administrators can set different granularities for different instances in their accounts to manage costs.

The approximate volume of data ingested into CloudWatch Logs by Enhanced Monitoring for an instance is as shown below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Granularity | 60 seconds | 30 seconds | 15 seconds | 10 seconds | 5 seconds | 1 second |
| Data ingested in CloudWatch Logs\* (GB per month) | 0.27 | 0.53 | 1.07 | 1.61 | 3.21 | 16.07 |

MySQL [(back to top)](#Amazon_RDS_FAQs)

DB Engine Version Management

Q: 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 DB Instance (i.e. MySQL) 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](https://aws.amazon.com/rds/faqs/#77) versions as they are supported by Amazon RDS. This patching will occur during your scheduled [maintenance window,](https://aws.amazon.com/rds/faqs/#12) and will be announced on the Amazon [RDS Forum](http://developer.amazonwebservices.com/connect/forum.jspa?forumID=60&start=0) in advance. We schedule them so you can plan around them, because downtime is required to upgrade a DB engine version, even for Multi-AZ instances. 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 entires 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.

Q: 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](https://aws.amazon.com/rds/faqs/#78)) 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](https://aws.amazon.com/rds/faqs/#12) for your DB Instance.

Q: 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.

Q: 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

X = Major version, Y = Release level, Z = Version number within release series.

From the Amazon RDS standpoint, a version change would be considered major if either major version or release level is being changed. Example: going from 5.1.X -> 5.5.X. A version change would be considered minor if the version number within the release is being changed. Example: going from 5.1.45 -> 5.1.49.

As of today, Amazon RDS supports the MySQL major versions MySQL 5.1, 5.5 and 5.6. We plan to support additional MySQL major versions in the future.

Q: Does Amazon RDS provide guidelines for supporting new MySQL version releases and/or deprecating MySQL versions that are currently supported?

Yes. See the [Amazon RDS versioning guidelines](https://aws.amazon.com/rds/faqs/#versioningGuidance) in this FAQ.

Q: How do I upgrade from one major version of MySQL to another?

Review the [Upgrading a DB Instance](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_UpgradeInstance.html) section of the Amazon RDS User Guide to learn more.

Storage Engines

Q: 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](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.BackingUpAndRestoringAmazonRDSInstances.html) 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 an 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.

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

Oracle [(back to top)](#Amazon_RDS_FAQs)

Licensing and Support

Q: 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](http://www.oracle.com/us/corporate/pricing/cloud-licensing-070579.pdf).
* 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.

Q: 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 Two (SE2), Standard Edition One (SE1), Standard Edition (SE), and Enterprise Edition (EE)
* License Included: Standard Edition One (SE1)

Q: 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](http://www.oracle.com/us/corporate/pricing/cloud-licensing-070579.pdf).
* 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.

Q: 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.

Q: 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

Q: 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

X = Major version (for ex: 11.2), Y = Release level (for ex: 0.2), Z = version number within release series (for ex: 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.

From the Amazon RDS standpoint, a version change would be considered major if either major version or release level is being changed. Example: going from 11.2.0.2.Z -> 11.2.0.4.Z. A version change would be considered minor if the version number within the release is being changed. Example: going from 11.2.0.2.v2 -> 11.2.0.2.v3.

As of today, Amazon RDS supports major versions 11.2 (11g Release 2) and 12c. We plan to support additional major versions in the future.

Q: What is the patch set composition of my DB Engine Version for Oracle?

Refer to the [Amazon RDS User Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/index.html?Appendix.Oracle.PatchComposition.html) for details of the patch set composition of each DB Engine Version of Oracle.

Q: 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 when minor version upgrades are scheduled by Amazon RDS. This patching will occur during your scheduled [maintenance window,](https://aws.amazon.com/rds/faqs/#12) and will be announced on the[Amazon RDS Forum](http://developer.amazonwebservices.com/connect/forum.jspa?forumID=60&start=0) 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 entires 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.

Q: 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](https://aws.amazon.com/rds/faqs/#116)) 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](https://aws.amazon.com/rds/faqs/#12) for your DB Instance.

Q: 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.

Q: 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?

Yes. See the [Amazon RDS versioning guidelines](https://aws.amazon.com/rds/faqs/#versioningGuidance) in this FAQ.

Scaling

Q: 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 DB Instance FAQ](https://aws.amazon.com/rds/faqs/#69).

Q: Can I change the Oracle edition I'm running for a DB Instance (e.g. 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 / Features

Q: What types of replication does Amazon RDS support for Oracle?

Amazon RDS for Oracle supports [Multi-AZ deployments](https://aws.amazon.com/rds/faqs/#36) for both the License Included and Bring Your Own License (BYOL) licensing models.

Q: Does Amazon RDS use Oracle Data Guard for Multi-AZ deployments?

Oracle Data Guard is a High Availability feature available for Enterprise Edition of Oracle database. 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 supported by Amazon RDS.

Q: Will I need an additional license(s) if I use Multi-AZ deployments for my Oracle DB Instances under the "BYOL" licensing model?

Yes, we expect that you will need to use twice as many licenses for Multi-AZ deployments as you would for a corresponding Single-AZ deployment to account for the stand by DB Instance. However, you should review your Oracle Software Licensing Agreement and comply with Oracle’s licensing policies.

Q: Is Oracle RAC supported on Amazon RDS?

No, RAC is not currently supported.

Q: Which Enterprise Edition Options are supported on Amazon RDS?

Following Enterprise Edition Options are currently supported under the BYOL model:

* Advanced Security (Transparent Data Encryption, Native Network Encryption)
* Partitioning
* Management Packs (Diagnostic, Tuning)
* Advanced Compression
* Total Recall

Q: Which character sets are supported on Amazon RDS for Oracle?

Amazon RDS supports the thirty character sets in the Oracle "Recommended ASCII Database Character Sets" list. You can specify your desired character set when creating a new database instance. This is optional and the default character set is AL32UTF8. For more information, please refer to the [Amazon RDS Documentation](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/Appendix.OracleCharacterSets.html).

Q: Who manages the Oracle Wallet and Master Encryption Key when using Transparent Data Encryption on Amazon RDS?

Amazon RDS manages the Oracle Wallet and Master Encryption Key for the DB Instance.

How do I know if Amazon RDS supports a specific Oracle Database feature?

Oracle Database supports a number of [features](http://www.oracle.com/us/products/database/product-editions-066501.html) that vary with the edition of Oracle database you run. Refer to the [Amazon RDS User Guide](http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/Concepts.DBEngine.Oracle.html) to know about the Oracle features that Amazon RDS currently supports.