PRODUCTS & SERVICES

Amazon RDS

Product Details

Instance Types

Pricing

FAQs

Reserved Instances

Getting Started

Customer Highlights

Partners

Resources

Amazon RDS SLA

RELATED LINKS

Amazon Aurora

Amazon RDS for MySQL

Amazon RDS for MariaDB

Amazon RDS for Oracle

Amazon RDS for PostgreSQL

Amazon RDS for SQL Server

Release Notes

Community Forum

Documentation

Manage Your Resources

Amazon RDS FAQs

- General
- Database Instances
- Database Engine Versions
- Billing
- Free Tier
- Reserved Instances
- Hardware and Scaling
- Automatic Backups and Database Snapshots
- Security
- Database Configuration
- Multi-AZ Deployments and Read Replicas
- Enhanced Monitoring

You can also refer to the FAQs page for each Amazon RDS database engine for engine-specific questions.

Amazon Aurora | MySQL | MariaDB | Oracle | PostgreSQL | SQL Server

General

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. It

Get Started with AWS for Free

Create a Free Account

AWS Free Tier includes 750hrs of Micro DB instance each month for one year, 20GB of Storage, and 20GB for backups with Amazon Relational Database Service (RDS).

View AWS Free Tier Details »

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 can automatically back up your database and keep your database software up to date with the latest version. You benefit from the flexibility of being able to easily scale the compute resources or storage capacity associated with your relational database instance. 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: Which relational database engines does Amazon RDS support?

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

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 up and running, Amazon RDS automates common administrative tasks such as performing backups and patching the software that powers your database. With optional Multi-AZ deployments, Amazon RDS also manages synchronous data replication across Availability Zones with 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?

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. Using one of our many relational database AMIs on Amazon EC2 allows you to manage 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 Cloud Databases with AWS 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 if you do not already have one. After you are signed up, please refer to the Amazon RDS documentation, which includes our Getting Started Guide.

Amazon RDS is part of the AWS Free Tier so that new AWS customers can get started with a managed database service in the cloud for free.

Database Instances

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, Amazon RDS APIs, and AWS Command Line Interface. You can run one or more DB instances, and each DB instance can support one or more databases or database schemas, depending on engine type.

Q: How do I create a DB instance?

DB instances are simple to create, using either the AWS Management Console, Amazon RDS APIs, or AWS Command Line Interface. To launch a DB instance using the AWS Management Console, click "RDS," then the **Launch DB Instance** button on the **Instances** tab. From there, you can specify the parameters for your DB instance including DB engine and version, license model, instance type, storage type and amount, and master user credentials.

You also have the ability to change your DB instance's backup retention policy, preferred backup window, and scheduled maintenance window. Alternatively, you can create your DB instance using the CreateDBInstance API or 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, DescribeDBInstances API or describe-db-instances command. Using this endpoint you can construct the connection string required to connect directly with your DB instance using your favorite database tool or programming language. In order to allow network requests to your running DB instance, you will need to authorize access. For a detailed explanation of how to construct your connection string and get started, please refer to our Getting Started Guide.

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" model. All 40 can be used for Amazon Aurora, MySQL, MariaDB, PostgreSQL and Oracle under the "BYOL" model. Note that RDS for SQL Server has a limit of 30 databases on a single DB instance.

If your application requires more DB instances, you can request additional DB instances via this 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 into an Amazon RDS DB instance?

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, full backup files (.bak files) 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 or the Data Import Guide for Oracle or the Data Import Guide for SQL Server or the Data Import Guide for PostgreSQL.

In addition, AWS Database Migration Service can help you migrate databases to AWS easily and securely.

Q: What is a maintenance window? Will my DB instance be available during maintenance events?

The Amazon RDS maintenance window is your opportunity to control when DB instance modifications, database engine version upgrades, and software patching occurs, in the event they are requested or required. If a maintenance event is scheduled for a given week, it will be initiated during the maintenance window you identify.

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), database engine version upgrades, and required software patching. Required software 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, the ModifyDBInstance API or the modify-db-instance command. 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. Please refer to the Amazon RDS User Guide for more information on maintenance operations.

Q: What should I do if my queries seem to be running slow?

For production databases we encourage you to enable Enhanced Monitoring, which provides access to over 50 CPU, memory, file system, and disk I/O metrics. You can enable these features on a per-instance basis and you can choose the granularity (all the way down to 1 second). 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 DB instance, refer to the Amazon RDS User Guide.

If you are using RDS for MySQL or MariaDB, you can access the slow query logs for your database to determine if there are slow-running SQL queries and, if so, the performance characteristics of each. You could set the "slow_query_log" DB Parameter and query the mysql.slow_log table to review the slow-running SQL queries. Please refer to the Amazon RDS User Guide to learn more.

If you are using RDS for 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.

If you are using RDS for 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.

Database Engine Versions

Q: Which relational database engine versions does Amazon RDS support?

For the list of supported database engine versions, please refer to the documentation for each engine:

- Amazon RDS for MySQL

- Amazon RDS for MariaDB

- Amazon RDS for PostgreSQL

- Amazon RDS for Oracle

- Amazon RDS for SQL Server

- Amazon Aurora

Q: How does Amazon RDS distinguish between "major" and "minor" DB engine versions?

Refer to the FAQs page for each Amazon RDS database engine for specifics on version numbering.

Q: Does Amazon RDS provide guidelines for support of new DB engine versions?

Over time, Amazon RDS adds support for new major and minor database engine versions. The number of new versions supported will vary based on the frequency and content of releases and patches from the engine's vendor or development organization, 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 5 months of their general availability.

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 (major and minor) when creating a new DB instance via the **Launch DB Instance** operation in the AWS Management Console or the CreateDBInstance API. Please note that not every database engine version is available in every AWS region.

Q: How do I control if and when the engine version of my DB instance is upgraded to new supported versions?

Amazon RDS strives to keep your database instance up to date by providing you newer versions of the supported database engines. After a new version of a database engine is released by the vendor or development organization, it is thoroughly tested by our database engineering team before it is made available in Amazon RDS.

We recommend that you keep your database instance upgraded to the most current minor version as it will contain the latest security and functionality fixes. Unlike major version upgrades, minor version upgrades only include database changes that are backward-compatible with previous minor versions (of the same major version) of the database engine.

If a new minor version does not contain fixes that would benefit RDS customers, we may choose not to make it available in RDS. Soon after a new minor version is available in RDS, we will set it to be the preferred minor version for new DB instances.

To manually upgrade a database instance to a supported engine version, use the **Modify DB Instance** command on the AWS Management Console or the ModifyDBInstance API and set the **DB Engine Version** parameter to the desired version. By default, the upgrade will be applied or during your next maintenance window. You can also choose to upgrade immediately by selecting the **Apply Immediately** option in the console API.

If we determine that a new engine minor version contains significant bug fixes compared to a previously released minor version, we will schedule automatic upgrades for DB instances which have the **Auto Minor Version Upgrade** setting to "Yes". These upgrades will be scheduled to occur during customer-specified maintenance windows.

We will announce scheduled upgrades on the Amazon RDS Forum and send customer e-mail notifications at least 30 days 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 minor version upgrades, you can do so by setting the Auto Minor Version Upgrade setting to "No".

In the case of RDS for Oracle and RDS for SQL Server, if the upgrade to the next minor version requires a change to a different edition, then we may not schedule automatic upgrades even if you have enabled the **Auto Minor Version Upgrade** setting. The determination on whether to schedule automatic upgrades in such situations will be made on a case-by-case basis.

Since major version upgrades involve some compatibility risk, they will not occur automatically and must be initiated by you (except in the case of major version deprecation, see below).

For more information about upgrading a DB instance to a new DB engine version, refer to the Amazon RDS User Guide.

Q: Can I test my DB instance with 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 copy of your DB instance before deciding whether or not to upgrade your original DB instance.

For more information about restoring a DB snapshot, refer to the Amazon RDS User Guide.

Q: Does Amazon RDS provide guidelines for deprecating database engine versions that are currently supported?

- We intend to support major version releases (e.g., MySQL 5.6, PostgreSQL 9.6) for at least 3 years after they are initially supported by Amazon RDS.

- We intend to support minor versions (e.g., MySQL 5.6.37, PostgreSQL 9.6.1) for at least 1 year after they are initially supported by Amazon RDS.

Periodically, we will deprecate major or minor engine versions. For major versions, this is typically when the version has moved to extended support or is no longer receiving software fixes or security updates. For minor versions, this is when a minor version has significant bugs or security issues that have been resolved in a later minor version.

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. In the unlikely event that such cases occur, Amazon RDS will automatically upgrade your database engine to address the issue. Specific circumstances may dictate different timelines depending on the issue being addressed.

Q: What happens when an RDS DB engine version is deprecated?

When a <u>minor</u> version of a database engine is deprecated in Amazon RDS, we will provide a three (3) month period after the announcement before beginning automatic upgrades. At the end of the this period, all instances still running the deprecated minor version will be scheduled for automatic upgrade to the latest supported minor version during their scheduled maintenance windows.

When a <u>major</u> version of database engine is deprecated in Amazon RDS, we will provide a minimum six (6) month period after the announcement of a deprecation for you to initiate an upgrade to a supported major version. At the end of this period, an automatic upgrade to the next major version will be applied to any instances still running the deprecated version during their scheduled maintenance windows.

Once a major or minor database engine version is no longer supported in Amazon RDS, any DB instance restored from a DB snapshot created with the unsupported version will automatically and immediately be upgraded to a currently supported version.

Billing

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. db.t2.micro, db.m4.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 and Amazon Aurora 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 customer-initiated database snapshots. Increasing your backup retention period or taking additional database snapshots increases the backup storage consumed by your database.

- 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.

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 stop or delete it to avoid being billed for additional instance hours. Partial DB instance hours consumed are billed as full hours.

Q: How will I be billed for a stopped DB instance?

While your database instance is stopped, you are charged for provisioned storage (including Provisioned IOPS) and backup storage (including manual snapshots and automated backups within your specified retention window), but not for DB instance hours.

Q: Why does my 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. Multi-AZ billing is based on:

- Multi-AZ DB instance hours – Based on the class (e.g. db.t2.micro, db.m4.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. Internet data transfer in and out of your DB instance is charged the same as with a standard deployment.

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 AWS services is subject to Japanese Consumption Tax. Learn more.

Free Tier

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 General Purpose (SSD) database storage 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 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.

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 for details.

Reserved Instances

Q: What is a reserved instance (RI)?

Amazon RDS reserved instances give you the option to reserve a DB instance for
a one or three year term and in turn receive a significant discount compared to
the on-demand instance pricing for the DB instance. 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 purchase a one or three year reservation and in return receive a
lower effective hourly usage rate (compared with on-demand DB instances) for
the duration of the term. Unless you purchase reserved instances in a Region, all
DB instances will be billed at on-demand hourly rates.

Q: How do I purchase and create reserved instances?

You can purchase a reserved instance in the "Reserved Instance" section of the
AWS Management Console for Amazon RDS. Alternatively, you can use the
Amazon RDS API or AWS Command Line Interface to list the reservations
available for purchase then purchase a DB instance reservation.

Once you have made a reserved purchase, using a reserved DB instance is no
different than an On-Demand DB instance. Launch a DB instance using the same
instance class, engine and region for which you made the reservation. As long as
your reservation purchase is active, Amazon RDS will apply the reduced hourly
rate for which you are eligible to the new DB instance.

Q: Do reserved instances include a capacity reservation?

Amazon RDS reserved instances are purchased for a Region rather than for a
specific Availability Zone. As RIs are not specific to an Availability Zone, they are
not capacity reservations. This means that even if capacity is limited in one
Availability Zone, reservations can still be purchased in the Region and the

discount will apply to matching usage in any 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.

Q: What if I have an existing DB instance that I'd like to cover with a reserved instance?

Simply purchase a DB instance reservation with the same DB instance class, DB engine, Multi-AZ option 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 or describe-reserved-db-instances command. 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 operations for creating, modifying, and deleting DB instances do not distinguish between On-Demand and reserved instances. 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 DB 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, Multi-AZ deployment option, license model and Region.

A reservation for a DB engine and license model that is eligible for size-flexibility (MySQL, MariaDB, PostgreSQL, Amazon Aurora or Oracle "Bring Your Own License") will automatically apply to a running DB instance of any size within the same instance family (e.g. M4, T2, or R3) for the same database engine and

Region. In addition, the reservation will also apply to DB instances running in either Single-AZ or Multi-AZ deployment options.

For example, let's say you purchased a db.m4.2xlarge MySQL reservation. If you decide to scale up the running DB instance to a db.m4.4xlarge, the discounted rate of this RI will cover 1/2 of the usage of the larger DB instance.

If you are running a DB engine or license model that is <u>not</u> eligible for size-flexibility (Microsoft SQL Server or Oracle "License Included"), 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 before the end of the reservation term, your hourly usage rates for that DB instance will revert to on demand hourly rates.

For more details on about size flexibility, see the Amazon RDS User Guide.

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 or describe-reserved-db-instances-offerings command, 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 DB instance reservation can 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

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 information the about the DB instance classes available, please refer to the Amazon RDS User Guide.

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 AWS Management Console (selecting the desired DB instance and clicking the **Modify** button), the RDS API, or the AWS Command Line Interface. 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 or using the Amazon CloudWatch APIs. To learn more about monitoring your active DB Instances, read the Amazon RDS User Guide.

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 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, the ModifyDBInstance API, or the modify-db-instance command.

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.

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.

Q: What is Amazon RDS Magnetic storage?

Amazon RDS magnetic storage is useful for small database workloads where data is accessed less frequently. Magnetic storage is not recommended for production database instances.

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

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.

Automated Backups and Database Snapshots

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 7 days but can be set to up to 35 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, which is typically within the last five minutes. Alternatively, you can find the Latest Restorable Time for a DB instance by selecting it in the AWS Management Console and looking in the "Description" tab in the lower panel of the Console.

DB Snapshots are user-initiated and enable you to back up your DB instance in a known state as frequently as you wish, and then restore to that specific state at any time. DB Snapshots can be created with the AWS Management Console, CreateDBSnapshot API, or create-db-snapshot command and are kept until you explicitly delete them.

The snapshots which Amazon RDS performs for enabling automated backups are available to you for copying (using the AWS console or the 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 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, Amazon RDS enables automated backups of your DB Instance with a 7 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 100 GB of provisioned database storage over the month, we will provide 100 GB-months of backup storage at no additional charge. If you would like to extend your backup retention period beyond one day, you can do so using the CreateDBInstance API (when creating a new DB Instance) or ModifyDBInstance API (for an existing DB Instance). You can use these APIs to change the RetentionPeriod parameter from 1 to the desired number of days. For more information on automated backups, please refer to the Amazon RDS User Guide.

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, the ModifyDBInstance API, or the modify-db-instance command 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 "Snapshots" section of the Amazon RDS Console. Alternatively, you can see a list of the user-created DB Snapshots for a given DB Instance using the DescribeDBSnapshots API or describe-db-snapshots command and delete snapshots with the DeleteDBSnapshot API or delete-db-snapshot command.

Q: Why do I have more automated DB snapshots than the number of days in the retention period for my DB instance?

It is normal to have 1 or 2 more automated DB snapshots than the number of days in your retention period. One extra automated snapshot is retained to ensure the ability to perform a point in time restore to any time during the retention period. For example, if your backup window is set to 1 day, you will require 2 automated snapshots to support restores to any within previous 24 hours. You may also see an additional automated snapshot as a new automated snapshot is always created before the oldest automated snapshot is deleted.

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 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

Q: What is Amazon Virtual Private Cloud (VPC) and how does it work with Amazon RDS?

Amazon VPC lets you create a virtual networking environment in a private, isolated section of the 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 way that you can take advantage of VPC is when 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.

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

If your AWS account was created before 2013-12-04, you may be able to run Amazon RDS in an Amazon Elastic Compute Cloud (EC2)-Classic environment. The basic functionality of Amazon RDS is the same regardless of whether EC2-Classic or EC2-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. For more information about the differences between EC2-Classic and EC2-VPC, see the EC2 documentation.

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 procedure that walks you through this process, refer to Creating a DB Instance in a VPC in the Amazon RDS User Guide.

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

Visit the Security Groups section of the Amazon RDS User Guide to learn about the different ways to control access to your DB Instances.

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 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?

If your DB instance is not in a VPC, you can use the AWS Management Console to easily move your DB instance into a VPC. See the Amazon RDS User Guide for more details. You can also 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?

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.

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. View the Amazon RDS User Guide for more information.

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 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 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. 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, MariaDB User Guide, SQL Server User Guide, PostgreSQL User Guide or Oracle User Guide. To learn more about how SSL works with these engines, you can refer directly to the MySQL documentation, the MariaDB documentation, the MSDN SQL Server documentation, the PostgreSQL documentation, or the Oracle Documentation.

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). 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.

You can also add encryption to a previously unencrypted DB instance or DB cluster by creating a DB snapshot and then creating a copy of that snapshot and specifying a KMS encryption key. You can then restore an encrypted DB instance or DB cluster from the encrypted snapshot.

Amazon RDS for Oracle and SQL Server support those engines' Transparent Data Encryption technologies. Transparent Data Encryption in Oracle is integrated with AWS 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 and SQL Server.

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 users and groups can take on RDS resources. You do this by referencing the RDS resources in the AWS IAM policies 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 and Tagging Amazon RDS Resources

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, and turn it on via CloudTrail's AWS Management Console home page.

Q: Can I use Amazon RDS with applications that require HIPAA compliance?

A: Yes, all RDS database engines are HIPAA-eligible, so you can use them to build HIPAA-compliant applications and store healthcare related information, including protected health information (PHI) under an executed Business Associate Agreement (BAA) with AWS. If you already have an executed BAA, no action is necessary to begin using these services in the account(s) covered by your BAA. If you do not have an executed BAA with AWS, or have any other questions about HIPAA-compliant applications on AWS, please contact us.

Database Configuration

Q: How do I choose the right configuration parameters for my DB Instance(s)?

By default, Amazon RDS chooses the optimal configuration parameters for your DB Instance taking into account the instance class and storage capacity. However, if you want to change them, you can do so using the AWS Management Console, the Amazon RDS APIs, or the AWS Command Line Interface. 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.

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 Amazon RDS User Guide.

Q: How can I monitor the configuration of my Amazon RDS resources?

You can use AWS Config to continuously record configurations changes to Amazon RDS DB Instances, DB Subnet Groups, DB Snapshots, DB Security Groups, and Event Subscriptions and receive notification of changes through Amazon Simple Notification Service (SNS). You can also create AWS Config Rules to evaluate whether these RDS resources have the desired configurations.

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 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, the RDS API, or the AWS Command Line Interface. 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 Aurora and Amazon RDS for MySQL, MariaDB 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

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. More information about Multi-AZ deployments is in the Amazon RDS User Guide.

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.

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.

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. 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 click the "Events" section of the Amazon RDS Console or use the DescribeEvents API to return information about events related to your DB instance. You can also use Amazon RDS Event Notifications to be notified when specific DB events occur.

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. 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.

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 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.

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 Aurora: All DB Clusters.

Amazon RDS for MySQL: DB Instances with MySQL version 5.5 or newer support creation of Read Replicas. Automatic backups must be and remain enabled on the source DB Instance for Read Replica operations. Automatic backups on the replica are supported only for Amazon RDS Read Replicas running MySQL 5.6 and later, not 5.5.

Amazon RDS for PostgreSQL: DB 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: All DB Instances support creation of Read Replicas. Automatic backups must be and remain enabled on the source DB Instance for Read Replica operations.

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 on the AWS 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).

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 Aurora allows you to create up to 15 Read Replicas for a given DB Cluster.

Amazon RDS for MySQL, MariaDB and PostgreSQL currently allow you to create up to 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?

Yes, RDS supports cross-region Read Replicas.

Q: Do Amazon RDS Read Replicas support synchronous replication?

No. Read Replicas in Amazon RDS for MySQL, MariaDB and PostgreSQL are implemented using those engines' native asynchronous replication. Amazon Aurora uses a different, but still asynchronous, replication mechanism.

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 (or to a read replica, in the case of Amazon Aurora) 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 MariaDB allow you to enable Multi-AZ on Read Replicas to support disaster recovery and minimize downtime from engine upgrades. Amazon RDS for PostgreSQL does 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: Can I create a Read Replica of another Read Replica?

Amazon Aurora, Amazon RDS for MySQL and MariaDB: 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 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 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 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 or PostgreSQL.

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 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 Aurora (MySQL) Read Replica will stay active and continue accepting read traffic even after its corresponding source DB Instance has been deleted. One of the Read Replicas in the cluster will automatically be promoted as the new master, and will start accepting write traffic.

An Amazon RDS for MySQL or MariaDB 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?

With Amazon RDS for MySQL or Amazon RDS for MariaDB, you can use the mysqlbinlog utility to download or stream binary logs from your DB Instance. Amazon RDS for PostgreSQL does not currently provide access to the WAL files for your DB 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. Just like a standard DB Instance, the rate per "DB Instance hour" for a Read Replica is determined by the DB Instance class of the Read Replica – please see Amazon RDS detail page for up-to-date pricing. You are not charged for the data transfer incurred in replicating data between your source DB Instance and Read Replica.

Billing for a Read Replica begins as soon as the Read Replica has been successfully created (i.e. when status is listed as "active"). The Read Replica will continue being billed at standard Amazon RDS DB Instance hour rates until you issue a command to delete it.

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, MySQL, and MariaDB 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 as they leverage native engine features. See the following table for details.

Feature	PostgreSQL	MySQL	MariaDB
Maximum Read Replicas allowed per source DB Instance	5	5	5
Replication method	Asynchronous Physical	Asynchronous Logical	Asynchronous Logical
Must automatic backups be enabled for Read Replica support?	Yes	Yes	Yes
Engine versions for which Read Replicas are available	9.3.5 or later	5.5 or later	10.0 or later
Promotion of Read Replica to a new standalone DB Instance	Supported	Supported	Supported
Creation of Indexes on Read Replica	Currently not supported	Supported	Supported
Creation of Backups of Read Replicas	Currently not supported	Supported	Supported
Chaining of Read Replicas (i.e., Read Replicas of Read Replicas)	Currently not supported	Supported	Supported
Cross-Region Read Replicas	Supported	Supported	Supported
Multi-AZ Read Replicas	Supported	Supported	Supported

Enhanced Monitoring

Q: What is Enhanced Monitoring for RDS?

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 DB 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?

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.

Q: Which engines are supported by Enhanced Monitoring?

Enhanced Monitoring supports all RDS database engines.

Q: Which instance types are supported by Enhanced Monitoring?

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. For non-production DB Instances, 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?

You can view all the system metrics and process information for your RDS DB 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?

No. You can set different granularities for each DB 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?

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?

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 page.

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

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?

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.

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

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 ways you can consume the metrics via a third-party dashboard or application. Monitoring tools can use CloudWatch Logs Subscriptions 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 for more details.

Q: How can I delete historical data?

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.

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

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. 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

AWS CLOUD PRACTITIONER ESSENTIALS

Learn about AWS fundamentals to get an

WHAT'S NEW WITH AWS

Learn about the latest products, services, and more

Sign In to the Console

🐦 Twitter   f Facebook   ⊙ Podcast   🎬 Twitch   ▤ AWS Blog   📶 RSS News Feed
✉ Email Updates

AWS & Cloud Computing

What is Cloud Computing?

What is Caching?

What is NoSQL?

What is DevOps?

What is Docker?

Products & Services

Customer Success

Economics Center

Architecture Center

Security Center

What's New

Whitepapers

AWS Blog

Events

Sustainable Energy

Press Releases

AWS in the News

Analyst Reports

Legal

Solutions

Websites & Website Hosting

Business Applications

Backup & Recovery

Disaster Recovery

Data Archive

DevOps

Serverless Computing

Big Data

High Performance Computing

Mobile Services

Digital Marketing

Game Development

Digital Media

Government & Education

Health

Financial Services

Windows on AWS

Retail

Power & Utilities

Oil & Gas

Automotive

Blockchain

Manufacturing

Resources & Training

Developers

Java on AWS

JavaScript on AWS

Mobile on AWS

PHP on AWS

Python on AWS

Ruby on AWS

.NET on AWS

SDKs & Tools

AWS Marketplace

User Groups

Support Plans

Service Health Dashboard

Discussion Forums

FAQs

Documentation

Articles & Tutorials

Quick Starts

Manage Your Account

Management Console

Billing & Cost Management

Subscribe to Updates

Personal Information

Payment Method

AWS Identity & Access Management

Security Credentials

Request Service Limit Increases

Contact Us

Amazon Web Services is Hiring.

Amazon Web Services (AWS) is a dynamic, growing business unit within Amazon.com. We are currently hiring Software Development Engineers, Product Managers, Account Managers, Solutions Architects, Support Engineers, System Engineers, Designers and more. Visit our careers page to learn more.

Amazon.com is an Equal Opportunity-Affirmative Action Employer – Minority / Female / Disability / Veteran / Gender Identity / Sexual Orientation.

Language   Bahasa Indonesia   Deutsch   English   Español   Français   Italiano   Português   Tiếng Việt   Türkçe   Русский   ไทย   日本語   한국어   中文 (简体)   中文 (繁體)

Site Terms | Privacy