**Encrypt unencrypted RDS Instnace**

**Objectives:**

**1.** Signin to AWS Management Console

**2.** Create RDS MySQL instance (***uncheck - enabling encrypt option***)

**3.** Create a snapshot

**4.** Make a copy of the snapshot and encrypt it

**5.** Restore DB Instance from the encrypted snapshot

**6.** Change the name of the original DB Instance

**7.** Change the name of the Restored DB Instance to the original DB Instance name

**8.** Delete the original RDS Instance and snapshot

**9.** Delete AWS Resources

**Promoting read replica as Primary DB**

**Encrypting ReadReplicas**

You can't have an encrypted read replica of an unencrypted DB instance or an unencrypted read replica of an encrypted DB instance. Encrypted read replicas must be encrypted with the same KMS key as the source DB instance when both are in the same AWS Region.

**RDS:**

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

Amazon RDS DB instances

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

DB instance classes

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

## DB instance class types

Amazon RDS supports DB instance classes for the following use cases:

* [General-purpose](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html#Concepts.DBInstanceClass.Types.general-purpose)
* [Memory-optimized](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html#Concepts.DBInstanceClass.Types.memory)
* [Burstable-performance](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html#Concepts.DBInstanceClass.Types.burstable)

Amazon RDS DB instance storage

* DB instances for Amazon RDS for MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server use Amazon Elastic Block Store (Amazon EBS) volumes for database and log storage.

## Amazon RDS storage types

Amazon RDS provides three storage types: General Purpose SSD (also known as gp2 and gp3), Provisioned IOPS SSD (also known as io1), and magnetic (also known as standard).

The following list briefly describes the three storage types:

* **General Purpose SSD** – General Purpose SSD volumes offer cost-effective storage that is ideal for a broad range of workloads running on medium-sized DB instances. General Purpose storage is best suited for development and testing environments.
* **Provisioned IOPS SSD** – Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that require low I/O latency and consistent I/O throughput. Provisioned IOPS storage is best suited for production environments.

.

* **Magnetic** – Amazon RDS also supports magnetic storage for backward compatibility. We recommend that you use General Purpose SSD or Provisioned IOPS SSD for any new storage needs. The maximum amount of storage allowed for DB instances on magnetic storage is less than that of the other storage types.

# Regions, Availability Zones, and Local Zones

[**PDF**](https://docs.aws.amazon.com/pdfs/AmazonRDS/latest/UserGuide/rds-ug.pdf#Concepts.RegionsAndAvailabilityZones)[**RSS**](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rdsupdates.rss)

Amazon cloud computing resources are hosted in multiple locations world-wide. These locations are composed of AWS Regions, Availability Zones, and Local Zones. Each AWS Region is a separate geographic area. Each AWS Region has multiple, isolated locations known as Availability Zones.

By using Local Zones, you can place resources, such as compute and storage, in multiple locations closer to your users. Amazon RDS enables you to place resources, such as DB instances, and data in multiple locations. Resources aren't replicated across AWS Regions unless you do so specifically.

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


   AWS Region
  

# Creating an Amazon RDS DB instance

The basic building block of Amazon RDS is the DB instance, where you create your databases. You choose the engine-specific characteristics of the DB instance when you create it. You also choose the storage capacity, CPU, memory, and so on, of the AWS instance on which the database server runs.

In **Templates**, choose the template that matches your use case. If you choose **Production**, the following are preselected in a later step:

* **Multi-AZ** failover option
* **Provisioned IOPS SSD (io1)** storage option
* **Enable deletion protection** option

# Creating Amazon RDS resources with AWS CloudFormation

Amazon RDS is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you want (such as DB instances and DB parameter groups), and AWS CloudFormation provisions and configures those resources for you.

When you use AWS CloudFormation, you can reuse your template to set up your RDS resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

# Connecting to an Amazon RDS DB instance

After Amazon RDS provisions your DB instance, use any standard client application or utility for your DB engine to connect to the DB instance. In the connection string, specify the DNS address from the DB instance endpoint as the host parameter. Also, specify the port number from the DB instance endpoint as the port parameter.

The endpoint is unique for each DB instance, and the values of the port and user can vary. The following list shows the most common port for each DB engine:

* MariaDB – 3306
* Microsoft SQL Server – 1433
* MySQL – 3306
* Oracle – 1521
* PostgreSQL – 5432

# Working with parameter groups

Database parameters specify how the database is configured. For example, database parameters can specify the amount of resources, such as memory, to allocate to a database.

You manage your database configuration by associating your DB instances and Multi-AZ DB clusters with parameter groups. Amazon RDS defines parameter groups with default settings. You can also define your own parameter groups with customized settings.

A DB parameter group acts as a container for engine configuration values that are applied to one or more DB instances. DB cluster parameter groups apply to Multi-AZ DB clusters only. In a Multi-AZ DB cluster, the settings in the DB cluster parameter group apply to all of the DB instances in the cluster. The default DB parameter group for the DB engine and DB engine version is used for each DB instance in the DB cluster.

# Creating an Amazon ElastiCache cluster using Amazon RDS DB instance settings

ElastiCache is a fully managed, in-memory caching service that provides microsecond read and write latencies that support flexible, real-time use cases. ElastiCache can help you accelerate application and database performance. You can use ElastiCache as a primary data store for use cases that don't require data durability, such as gaming leaderboards, streaming, and data analytics. ElastiCache helps remove the complexity associated with deploying and managing a distributed computing environment. You can use the Amazon RDS console for creating ElastiCache clusters.

Amazon ElastiCache works with both the Redis and Memcached engines.

# Stopping an Amazon RDS DB instance temporarily

Suppose that you use a DB instance intermittently, for temporary testing, or for a daily development activity. If so, you can stop your Amazon RDS DB instance temporarily to save money. While your DB instance is stopped, you are charged for provisioned storage (including Provisioned IOPS). You're also charged for backup storage, including manual snapshots and automated backups within your specified retention window. However, you're not charged for DB instance hours.

# Automatically connecting an EC2 instance and a DB instance

You can use the Amazon RDS console to simplify setting up a connection between an Amazon Elastic Compute Cloud (Amazon EC2) instance and a DB instance. Often, your DB instance is in a private subnet and your EC2 instance is in a public subnet within a VPC. You can use a SQL client on your EC2 instance to connect to your DB instance . The EC2 instance can also run web servers or applications that access your private DB instance .

# Automatically connecting a Lambda function and a DB instance

You can use the Amazon RDS console to simplify setting up a connection between a Lambda function and a DB instance. Often, your DB instance is in a private subnet within a VPC. The Lambda function can be used by applications to access your private DB instance.


        
        Automatically connect a DB instance with a Lambda function
      

You can set up the connection between your Lambda function and your DB instance through RDS Proxy to improve your database performance and resiliency. Often, Lambda functions make frequent, short database connections that benefit from connection pooling that RDS Proxy offers. You can take advantage of any AWS Identity and Access Management (IAM) authentication that you already have for Lambda functions, instead of managing database credentials in your Lambda application code.


        
        Automatically connect a DB instance with a Lambda function through RDS Proxy
      

# Modifying an Amazon RDS DB instance

You can change the settings of a DB instance to accomplish tasks such as adding **additional storage or changing the DB instance class**.

# Maintaining a DB instance

Periodically, Amazon RDS performs maintenance on Amazon RDS resources. Maintenance most often involves updates to the following resources in your DB instance:

* Underlying hardware
* Underlying operating system (OS)
* Database engine version

Updates to the operating system most often occur for security issues. You should do them as soon as possible.

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

Deferred DB instance modifications that you have chosen not to apply immediately are also applied during the maintenance window. For example, you might choose to change the DB instance class or parameter group during the maintenance window. Such modifications that you specify using the **pending reboot** setting don't show up in the **Pending maintenance** list

## Viewing pending maintenance

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


            Offline patch available
        

If no maintenance update is available for a DB instance, the column value is **none** for it.

If a maintenance update is available for a DB instance, the following column values are possible:

* **required** – The maintenance action will be applied to the resource and can't be deferred indefinitely.
* **available** – The maintenance action is available, but it will not be applied to the resource automatically. You can apply it manually.
* **next window** – The maintenance action will be applied to the resource during the next maintenance window.
* **In progress** – The maintenance action is in the process of being applied to the resource.

# Renaming a DB instance

You can rename a DB instance by using the AWS Management Console, the AWS CLI modify-db-instance command, or the Amazon RDS API ModifyDBInstance action. Renaming a DB instance can have far-reaching effects. The following is a list of considerations before you rename a DB instance.

* When you rename a DB instance, the endpoint for the DB instance changes, because the URL includes the name you assigned to the DB instance. You should always redirect traffic from the old URL to the new one.
* When you rename a DB instance, the old DNS name that was used by the DB instance is immediately deleted, although it could remain cached for a few minutes. The new DNS name for the renamed DB instance becomes effective in about 10 minutes. The renamed DB instance is not available until the new name becomes effective.
* You cannot use an existing DB instance name when renaming an instance.
* All read replicas associated with a DB instance remain associated with that instance after it is renamed. For example, suppose you have a DB instance that serves your production database and the instance has several associated read replicas. If you rename the DB instance and then replace it in the production environment with a DB snapshot, the DB instance that you renamed will still have the read replicas associated with it.
* Metrics and events associated with the name of a DB instance are maintained if you reuse a DB instance name. For example, if you promote a read replica and rename it to be the name of the previous primary DB instance, the events and metrics associated with the primary DB instance are associated with the renamed instance.
* DB instance tags remain with the DB instance, regardless of renaming.
* DB snapshots are retained for a renamed DB instance.

# Rebooting a DB instance

You might need to reboot your DB instance, usually for maintenance reasons. For example, if you make certain modifications, or if you change the DB parameter group associated with the DB instance, you must reboot the instance for the changes to take effect.

###### Note

If a DB instance isn't using the latest changes to its associated DB parameter group, the AWS Management Console shows the DB parameter group with a status of **pending-reboot**. The **pending-reboot** parameter groups status doesn't result in an automatic reboot during the next maintenance window. To apply the latest parameter changes to that DB instance, manually reboot the DB instance.

If the Amazon RDS DB instance is configured for Multi-AZ, you can perform the reboot with a failover. An Amazon RDS event is created when the reboot is completed. If your DB instance is a Multi-AZ deployment, you can force a failover from one Availability Zone (AZ) to another when you reboot. When you force a failover of your DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone, and updates the DNS record for the DB instance to point to the standby DB instance. As a result, you need to clean up and re-establish any existing connections to your DB instance. Rebooting with failover is beneficial when you want to simulate a failure of a DB instance for testing, or restore operations to the original AZ after a failover occurs.

# Working with DB instance read replicas

A read replica is a read-only copy of a DB instance. You can reduce the load on your primary DB instance by routing queries from your applications to the read replica. In this way, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.

After you create a read replica from a source DB instance, the source becomes the primary DB instance. When you make updates to the primary DB instance, Amazon RDS copies them asynchronously to the read replica.

# Tagging Amazon RDS resources

You can use Amazon RDS tags to add metadata to your Amazon RDS resources. You can use the tags to add your own notations about database instances, snapshots, Aurora clusters, and so on. Doing so can help you to document your Amazon RDS resources. You can also use the tags with automated maintenance procedures.

In particular, you can use these tags with IAM policies. You can use them to manage access to RDS resources and to control what actions can be applied to the RDS resources. You can also use these tags to track costs by grouping expenses for similarly tagged resources.

## Using tags to produce detailed billing reports

You can also use tags to track costs by grouping expenses for similarly tagged resources.

Use tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of combined resources, organize your billing information according to resources with the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services.

Working with storage for Amazon RDS DB instances

To specify how you want your data stored in Amazon RDS, choose a storage type and provide a storage size when you create or modify a DB instance. Later, you can increase the amount or change the type of storage by modifying the DB instance.

**To monitor the amount of free storage for your DB instance so you can respond when necessary, we recommend that you create an Amazon CloudWatch alarm.**

# Multi-AZ DB instance deployments

Amazon RDS provides high availability and failover support for DB instances using Multi-AZ deployments with a single standby DB instance. This type of deployment is called a Multi-AZ DB instance deployment. Amazon RDS uses several different technologies to provide this failover support. Multi-AZ deployments for MariaDB, MySQL, Oracle, PostgreSQL, and RDS Custom for SQL Server DB instances use the Amazon failover technology. Microsoft SQL Server DB instances use SQL Server Database Mirroring (DBM) or Always On Availability Groups (AGss

In a Multi-AZ DB instance deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to a standby replica to provide data redundancy and minimize latency spikes during system backups. Running a DB instance with high availability can enhance availability during planned system maintenance. It can also help protect your databases against DB instance failure and Availability Zone disruption.

**Using the RDS console, you can create a Multi-AZ DB instance deployment by simply specifying Multi-AZ when creating a DB instance. You can use the console to convert existing DB instances to Multi-AZ DB instance deployments by modifying the DB instance and specifying the Multi-AZ option. You can also specify a Multi-AZ DB instance deployment with the AWS CLI or Amazon RDS API.**

There are two ways to modify a DB instance to be a Multi-AZ DB instance deployment:

## Modifying a DB instance to be a Multi-AZ DB instance deployment

If you have a DB instance in a Single-AZ deployment and modify it to a Multi-AZ DB instance deployment (for engines other than Amazon Aurora), Amazon RDS performs several actions:

1. Takes a snapshot of the primary DB instance's Amazon Elastic Block Store (EBS) volumes.
2. Creates new volumes for the standby replica from the snapshot. These volumes initialize in the background, and maximum volume performance is achieved after the data is fully initialized.
3. Turns on synchronous block-level replication between the volumes of the primary and standby replicas.

### Convert to a Multi-AZ DB instance deployment with the RDS console

You can use the RDS console to convert a DB instance to a Multi-AZ DB instance deployment.

# Multi-AZ DB cluster deployments

A Multi-AZ DB cluster deployment is a semisynchronous, high availability deployment mode of Amazon RDS with two readable standby DB instances. A Multi-AZ DB cluster has a writer DB instance and two reader DB instances in three separate Availability Zones in the same AWS Region. Multi-AZ DB clusters provide high availability, increased capacity for read workloads, and lower write latency when compared to Multi-AZ DB instance deployments.

Blue/Green Deploymentss

A blue/green deployment copies a production database environment in a separate, synchronized staging environment. By using Amazon RDS Blue/Green Deployments, you can make changes to the database in the staging environment without affecting the production environment. For example, you can upgrade the major or minor DB engine version, change database parameters, or make schema changes in the staging environment. When you are ready, you can promote the staging environment to be the new production database environment.

Cross-Region automated backups

By using backup replication in Amazon RDS, you can configure your RDS DB instance to replicate snapshots and transaction logs to a destination Region. When backup replication is configured for a DB instance, RDS starts a cross-Region copy of all snapshots and transaction logs when they're ready.

###### To enable backup replication for an existing DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose **Automated backups**.
3. On the **Current Region** tab, choose the DB instance for which you want to enable backup replication.
4. For **Actions**, choose **Manage cross-Region replication**.
5. Under **Backup replication**, choose **Enable replication to another AWS Region**.
6. Choose the **Destination Region**.
7. Choose the **Replicated backup retention period**.
8. If you've enabled encryption on the source DB instance, choose the **AWS KMS key** for encrypting the backups.
9. Choose **Save**.

In the source Region, replicated backups are listed on the **Current Region** tab of the **Automated backups** page. In the destination Region, replicated backups are listed on the **Replicated backups** tab of the **Automated backups** page.

**Automated backups follow these rules:**

* Your DB instance must be in the available state for automated backups to occur. Automated backups don't occur while your DB instance is in a state other than available, for example, storage\_full.
* Automated backups don't occur while a DB snapshot copy is running in the same AWS Region for the same database.

# Creating a DB snapshot

Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. Creating this DB snapshot on a Single-AZ DB instance results in a brief I/O suspension that can last from a few seconds to a few minutes, depending on the size and class of your DB instance.

Cross-Region read replicas

By using cross-Region read replicas in Amazon RDS, you can create a MariaDB, MySQL, Oracle, PostgreSQL, or SQL Server read replica in a different Region from the source DB instance.


                Cross-Region read replica configuration
            