






# 9



## Other AWS Storage Options


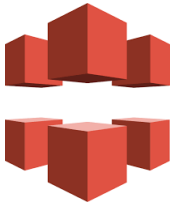
AWS offers a variety of highly available, scalable, reliable, and secure storage services to address various organizational needs. It provides a rich web console for all of the services, which is easy to access and navigate. It's easy to use UI complements efficient services for quickly performing day-to-day administrative. AWS also provides a set of API and CLI interfaces. You can use API and CLI to perform advanced operations or to automate various tasks using customized applications and scripts. The following table describes a number of storage and backup services provided by AWS.

AWS service	Description
<div>Amazon S3</div> <div></div>	<p>S3 is a cloud-based object storage over the internet. It is ideally suggested for storing static content such as graphics files, documents, log files, audio, video, compressed files, and so on. Virtually, any type of data in any file format can be stored on S3. Currently, permissible object size in S3 is 0 bytes to 5 TB. Objects in S3 are stored in a bucket. A bucket is a logical unit in S3 which is just like a folder. Buckets are created at root level in S3 with a globally unique name. You can store objects and also folders inside a bucket. Any number of objects can be stored in each bucket. There is a soft limit of 100 buckets per account in S3.</p> <p><b>Common usage:</b> S3 can be used for content storage and distribution, static website hosting, big data object store, backup and archival, storing application data as well as for DR. Using Java Script SDK and DynamoDB, you can also host dynamic applications in S3.</p>

<p>Amazon Glacier</p> 	<p>Glacier is a highly secure, durable, and a very low-cost cloud storage service for archiving data and taking long-term backups. Each file or an object stored in Amazon Glacier is called an archive. These stored archives are immutable, which means that contents of the archive cannot be modified. If required, another version of the archive can be stored and the existing version can be deleted. Size of each archive can range from 1 byte to 40 TB. With the help of the S3 lifecycle rules, objects from S3 can be automatically transferred to Glacier. These archives can be logically isolated in containers called vaults. A maximum of 1,000 vaults per account per region can be created.</p> <p>A few important characteristics:</p> <ul style="list-style-type: none"><li>• Very economical for storing long term archival data, which is rarely accessed</li><li>• Retrieval incurs charges and may take a minimum of 3 to 4 hours or more depending on the size of the data</li><li>• Amazon charges early deletion fees if data is deleted within three months from the date of storing</li></ul> <p><b>Common usage:</b> Glacier can be mainly used for data archival. It is widely used for media asset archiving, healthcare information archiving, regulatory and compliance archiving, scientific data storage, digital preservation, magnetic tape replacement, and so on. It is rarely retrieved for audit or other business purposes.</p>
<p>Amazon EFS</p> 	<p>AWS EFS is a simple to use and scalable file storage, which can be used with EC2 instances. It is a fully managed storage service from AWS which can be used for storing GBs to TBs of data. EFS volumes can be mounted and accessed by multiple EC2 instances at the same time. It uses the <b>Network File System versions 4.1 (NFSv4.1)</b> protocol. When using any EFS volume for the first time, you simply need to mount and format it to the desired filesystem. Subsequently, you can mount this volume on other EC2 instances directly and start using it. EFS volumes can also be accessed from on-premise environments using Direct Connect. You cannot access it from an on-premise environment over VPN connectivity. EFS is available in two modes, General Purpose mode and Max I/O mode.</p> <p><b>Common usage:</b> EFS is designed to provide very high disk throughput. It can be used for big data and analytics, media, content management, web serving, and home directories.</p>

<p>Amazon EBS</p> 	<p>EBS is a persistent, block-level storage service from Amazon. Persistent storage is a type of storage which retains the data stored on it even after power to the device is turned off. Block level storage is a type of storage which can be formatted to support a specific filesystem like NFS, NTFS, SMB, or VMFS. EBS volumes can be attached to an EC2 instance. Because of its persistent nature, data on EBS volume remains intact even after restarting or stopping an EC2 instance.</p> <p>There are five variants of EBS:</p> <ol style="list-style-type: none"><li>1) General Purpose SSD (gp2)</li><li>2) Provisioned IOPS SSD (io1)</li><li>3) Throughput optimized HDD (st1)</li><li>4) Cold HDD (sc1)</li><li>5) Magnetic (Standard)</li></ol> <p>Each of these variants, differs in terms of price and performance. EBS volumes are connected as a network storage to an EC2 instance. It can be sized from 1 GB to 16 TB. You can take a snapshot of an EBS volume. A Snapshot is a point-in-time backup of an EBS volume. Snapshots can be used to restore the volume as and when required.</p> <p><b>Common usage:</b> EBS volumes can be used as a root partition and for installing operating systems. It is also used for storing enterprise applications, application data, and databases.</p>
<p>Amazon EC2 instance store</p> 	<p>Instance store is a temporary block-level storage service from Amazon. Unlike EBS, instance store is temporary in nature. Data stored in instance store volume is deleted when EC2 instance is either restarted, stopped, or terminated. Instance store volumes are directly attached to the underlying hosts where an EC2 instance is provisioned. Instance store volumes are faster in comparison to EBS, however, it is a temporary data store. Performance of the instance store volume attached to an EC2 instance, size of each of the volumes, and the number of such volumes that can be attached to an EC2 instance, depending on the EC2 instance type.</p> <p><b>Common usage:</b> It is widely used to store swap files, temporary files, or in applications where good disk throughput is required but data persistence is not required.</p>

<p>AWS Storage Gateway</p> 	<p>AWS Storage Gateway is a hybrid storage service which connects on-premise environments with cloud storage using a software appliance. It seamlessly connects on-premise environments with Amazon's block-level and object-level storage services such as EBS, S3, and Glacier. Storage Gateway uses standard storage protocols such as NFS and iSCSI. It provides low-latency for exchanging data from on-premise to S3, Glacier, or EBS volumes and vice versa. Storage Gateway can provide high performance for frequently accessed data by caching them at source in on-premise environment.</p> <p><b>Common usage:</b> Storage Gateway can be configured for use as a file server in conjunction with S3. It can also be used as a virtual tape library for backup on S3 and virtual tape shelf for archival on Glacier. It can also be configured to be used as a local iSCSI volume. Storage Gateway can also be handy for transferring data from on-premise environments to AWS or transferring the data from AWS to on-premise environments.</p>
<p>AWS Snowball</p> 	<p>AWS Snowball is a petabyte-scale level data transport solution that uses physical appliances to transfer large-scale of data from on-premise environments to the AWS cloud and vice versa. A single Snowball appliance can transport up to 80 TB of data. Snowball comes in two sizes, 50 TB and 80 TB. Data can be copied over to multiple physical appliances and transported to and from an AWS. Transferring large scale data over the internet can take a significant amount of time depending upon the size of data. The purpose of the Snowball service is to minimize the data transfer time by transferring the data using a physical medium rather than transferring data over the internet. Snowball can efficiently compress, encrypt, and transfer data from the on-premise host to the intended Snowball device. Once the data is copied over to one or more snowball devices, these devices are transported back to the nearest AWS data center. Subsequently, AWS transfers data from Snowball devices to S3.</p> <p><b>Common usage:</b> Snowball is used for rapidly and securely transferring bulk data between on-premise data centers and the AWS cloud at a very economical rate.</p>

<p>AWS Snowmobile</p> 	<p>AWS Snowmobile is an exabyte-scale data transport solution that uses physical containers to transfer extremely large-scale of data from on-premise environment to the AWS cloud and vice versa. A Snowmobile container literally comes in a truck which can transfer up to 100 PB of data per snowmobile. The truck carries a high cube shipping container which is 45 foot long, 8 foot wide, and 9.6 foot tall. If your data is more than 100 PB, you can ask Amazon for more than one Snowmobile. At a time, more than one Snowmobile can be connected to the on-premise network for transferring data. When connected to an on-premise network, Snowmobile appears as a standard NFS mounting point on the network. It may require up to 350 KW of power supply. Once the data is transferred from the on-premise network to the Snowmobile, it returns to the nearest AWS data center in the region and subsequently, the data is transferred to the S3 of the respective customer account.</p> <p><b>Common usage:</b> Snowmobile is used for rapidly and securely transferring extremely large scale data between the on-premise data center and the AWS cloud at a very economical rate.</p>
<p>Amazon CloudFront</p> 	<p>Amazon CloudFront is a <b>Content Delivery Network (CDN)</b> offered by AWS. It is a system of distributed servers spread across edge locations. It is mainly used for caching static content such as web page, stylesheets, client-side scripts, images, and so on. It can also speed up dynamic content distribution. When a user hits a URL which is served through CloudFront, it routes the user request to the nearest edge location. The nearest edge location gives minimum latency in serving the request and provides the best possible performance to the user.</p> <p><b>Common usage:</b> CloudFront is used for providing seamless performance on delivery of a website or web application for a user base spread across multiple geographic locations. It can be used for distributing software or other large files, streaming media files, offering large downloads, and delivering live events.</p>

S3, Glacier, EBS, EC2 instance store, and CloudFront are elaborated in other relevant chapters. Subsequent section of this chapter touches up on EFS, AWS Storage Gateway, AWS Snowball, and AWS Snowmobile.

## Amazon EFS

AWS EFS is a simple to use and scalable file storage which can be used with EC2 instances. It is a fully managed storage service from AWS which can be used for storing GBs to TBs of data. EFS volume can be mounted and accessed by multiple EC2 instances at the same time. It uses the NFSv4.1 protocol. When using any EFS volume for the first time, you simply need to mount and format it to the desired file system. Subsequently, you can mount this volume on other EC2 instances directly and start using it. EFS volumes can also be accessed from an on-premise environment using Direct Connect. You cannot access it from on-premise environment over VPN connectivity. EFS is available in two modes, General Purpose mode and Max I/O mode.

In the industry, it is a common requirement to share file systems across the network, which be used as a common data source. EFS is a simple, secure, fully managed, scalable, and reliable block storage, to fulfill common file storage requirements. For using EFS with Linux EC2 instance, you may require installing the latest NFS packages. AWS recommends using the NFSv4.1 client on EC2 instances. Unlike EBS, EFS does not require provisioning a fixed volume size in advance. Being a managed service, you can store as much data as you need and pay only for what you use.



Currently, EFS does not support Windows-based EC2 instances.

An EFS volume is created at the VPC level. At the time of creating an EFS volume, you can specify the AZ from where it can be accessed. EC2 instances from all selected AZ within the same VPC can access the EFS volume. Optionally, you can add tags to your EFS volume. It is recommended to provide a relevant and meaningful name to your EFS volume along with tags for better identification and reference. While creating an EFS volume, it is essential to select the type of EFS volume. Types of EFS volume are General Purpose and Max I/O. The default EFS volume type is General Purpose. Once an EFS volume is successfully created, it returns a DNS endpoints. You can mount the EFS volumes on an EC2 instance or on-premises environment using the endpoint. Remember, you can mount EFS volume on an on-premise network only if you use Direct Connect.

Successful creation of an EFS volume also creates mount points in each AZ. EFS carries properties such as mount target ID, the file system ID, private IPv4 address, the subnet ID in which it is created and the mount target status. It is possible to mount EFS volume using a DNS name. The following *Figure 9.1* elaborates an EFS:

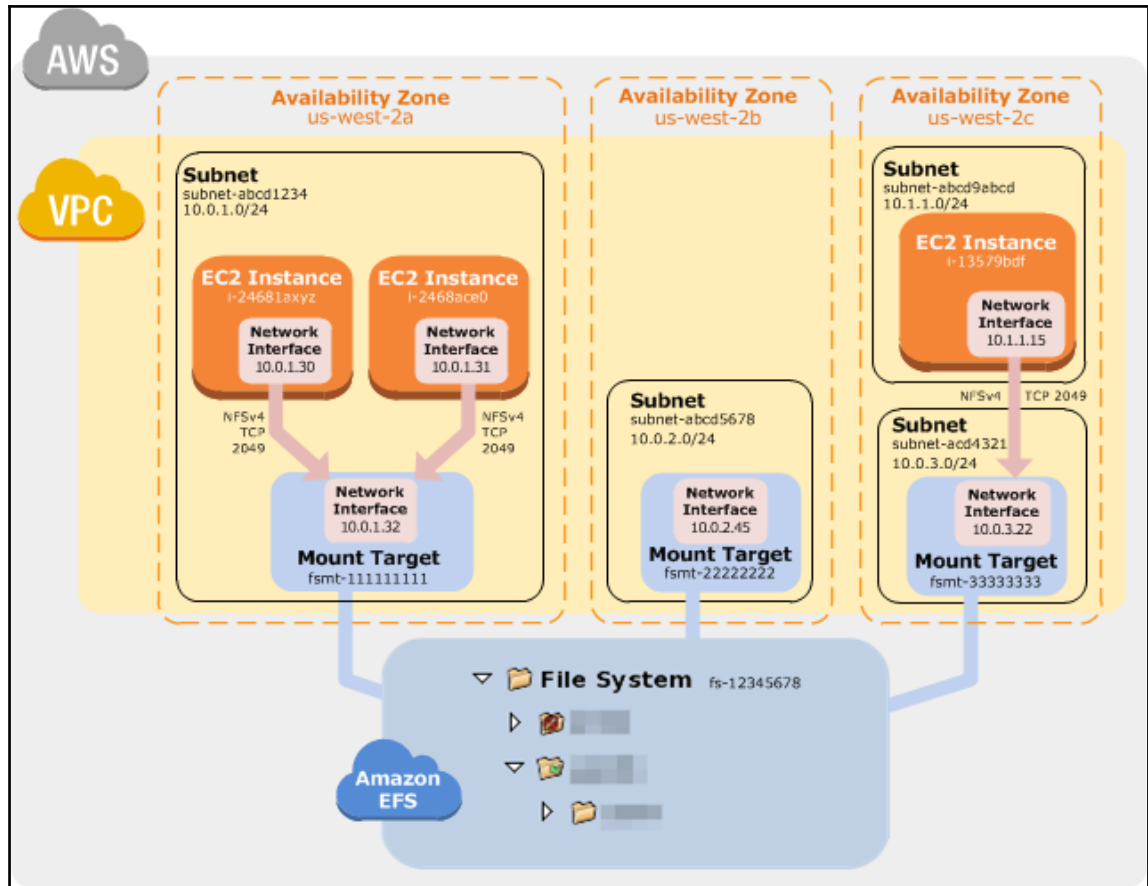


Figure 9.1: EFS

Reference URL: <https://docs.aws.amazon.com/efs/latest/ug/how-it-works.html>

An EC2 instance does not require a public or elastic IP to mount an EFS volume. You can enable or disable any existing EFS volume as required. You can perform all such changes from the **Manage file system access** option. Once you delete any EFS volume, it cannot be recovered.

Snapshots can be created for EFS volumes. It is also possible to design a backup solution using AWS Data Pipeline for copying data from one EFS volume to another EFS volume. You can also configure a copy operation schedule.

## AWS Storage Gateway

AWS Storage Gateway is a hybrid storage service provided by Amazon. With Storage Gateway services, your on-premises applications can seamlessly use AWS cloud storage. The following are some of the important points of AWS Storage Gateway:

- AWS Storage Gateway connects on-premise software appliances with the AWS cloud storage to provide a seamless integration experience and data security between the on-premises data center and the AWS storage services
- It is a scalable and cost-effective storage solution which also maintains data security
- It provides an iSCSI interface, which can be used to mount a volume as a local drive for easily integrating it with the existing backup applications
- AWS Storage Gateway uses incremental EBS snapshots for backing up data to AWS
- AWS provides a VM image for running Storage Gateway on an on-premise data center and you can also run it as an EC2 instance on AWS and in case of any issue, such as if the on-premise data center goes offline, you can deploy the gateway on an EC2 instance
- You can use a Storage Gateway hosted on an EC2 instance for DR, data mirroring, and as an application storage
- By default, Storage Gateway uploads data using SSL and provides data encryption at rest using AES-256 when the data is stored on S3 or Glacier
- Storage Gateway compresses data in-transit and at-rest for minimizing the data size



AWS Storage Gateway provides three types of solutions: file gateways, volume gateways, and tape-based gateways. A file gateway creates a file interface into Amazon S3. It allows you to access S3 using the **Network File System (NFS)** protocol. When using volume gateways, you can mount a volume as a drive in your environment. Tape-based gateways can be used similarly to a tape drive for backup.

## File gateways

A file gateway creates a file interface into Amazon S3. It allows you to access S3 using the NFS protocol. When you opt for a file gateway, a software appliance is hosted in the on-premise environment on a virtual machine running on VMware ESXi. Once the file gateway is created, it enables you to directly access S3 objects as files using an NFS volume mounted on a server.

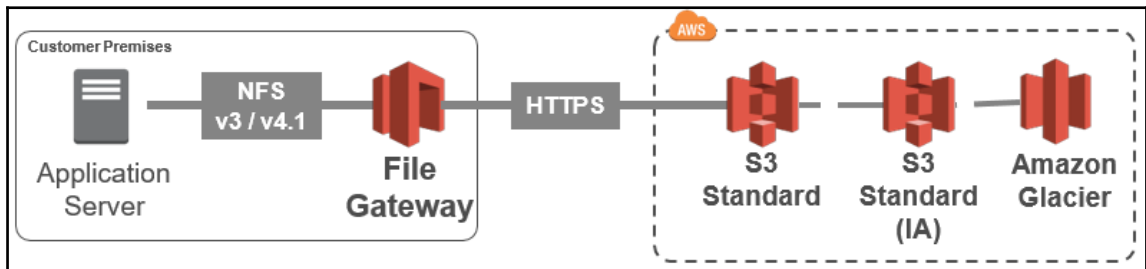


Figure 9.2: File gateway

Here is what file gateways can do for you:

- It allows you to directly store files on S3 using the NFS 3 or NFS 4.1 protocol
- You can directly retrieve files from S3 using the same NFS mount point
- It also allows you to manage S3 data with lifecycle policies, manage cross-region replication, and enable versioning on your data

## Volume gateways

When you create a volume gateway, it creates a cloud-backed storage volume, which you can mount as iSCSI devices on your on-premises servers where iSCSI stands for **Internet Small Computer System Interface**. Volume gateways stores all data securely on AWS. There are two types of volume gateway, which determine how much data is stored on-premises and how much data is stored on AWS storage and are discussed as follows:

### Gateway-cached volumes

Cached volumes enable you to store complete data on S3 and cache a copy of only frequently used data on on-premise. By reducing the amount of data stored on on-premise environment, you can reduce the overall storage cost. It also boosts performance by providing low-latency access to frequently accessed data using a cache.

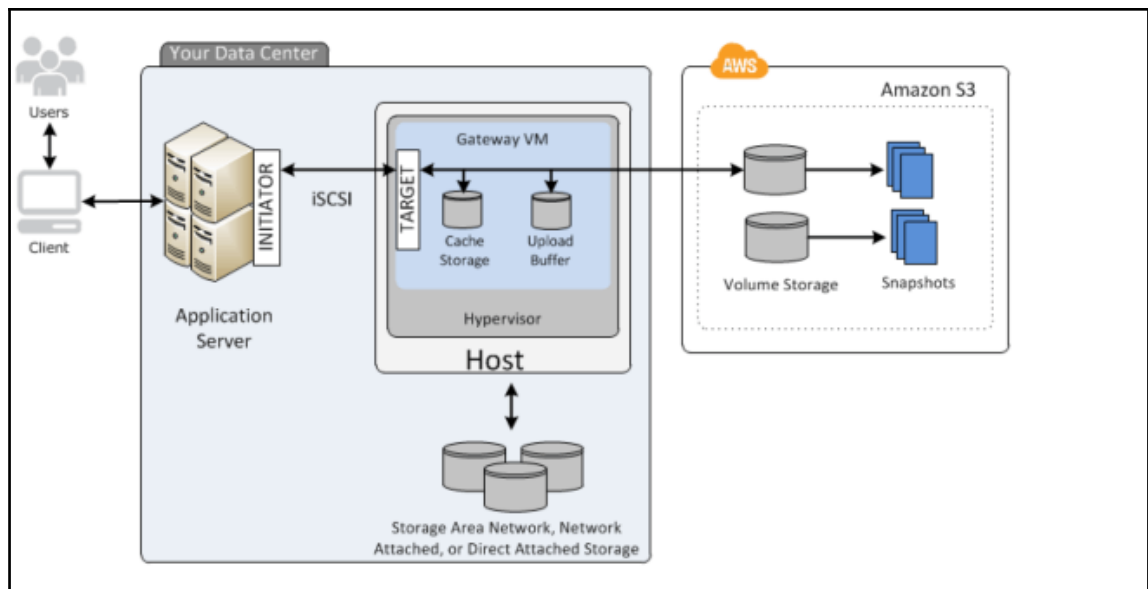


Figure 9.3: Gateway-cached volumes

The key features of gateway-cached volumes are as follows:

- A cached volume stores data in S3 and serves as a primary data storage
- It creates a local copy of frequently accessed data, which provides low-latency access for subsequent data access requests from applications
- By reducing the amount of data stored on an on-premise environment, you can reduce the overall storage cost
- You can create up to 32 gateway-cached volumes in a single storage gateway
- You can store from 1 GiB to 32 TiB in each volume with a maximum storage volume limit of 1,024 TiB (1 PiB)
- You can attach gateway-cached volumes as iSCSI devices on on-premise application servers
- You can take incremental snapshots of gateway-cached volumes
- Gateway-cached volume snapshots are stored on S3 as EBS snapshots
- Gateway-cached volume snapshots can be restored as gateway storage volume or you can create an EBS volume out of them and use it on an EC2 instance
- A maximum size of an EBS volume created out of a snapshot is 16 TiB and you cannot create an EBS volume out of the snapshot if it is more than 16 TiB in size
- AWS stores gateway-cached volume data and snapshots in Amazon S3 and the data is encrypted at rest with **server-side encryption (SSE)**; you cannot access the data with S3 API or any other tools
- Gateway VM allocates storage in two parts:
  - Cache storage
    - It serves as on-premise durable storage
    - It caches the data locally before uploading it to S3
    - It provides low-latency access to frequently accessed data
  - Upload buffer
    - Upload buffer serves as a staging location prior to uploading the data to s3
    - It uploads data on an encrypted SSL connection to AWS and stores it in an encrypted format on S3

## Gateway–stored volumes

You can use gateway-stored volumes when you need low-latency access to your entire data set. It stores all your data locally first and then asynchronously takes a point-in-time backup of this data as a snapshot to S3. It is generally used as an inexpensive off-site backup option for DR.

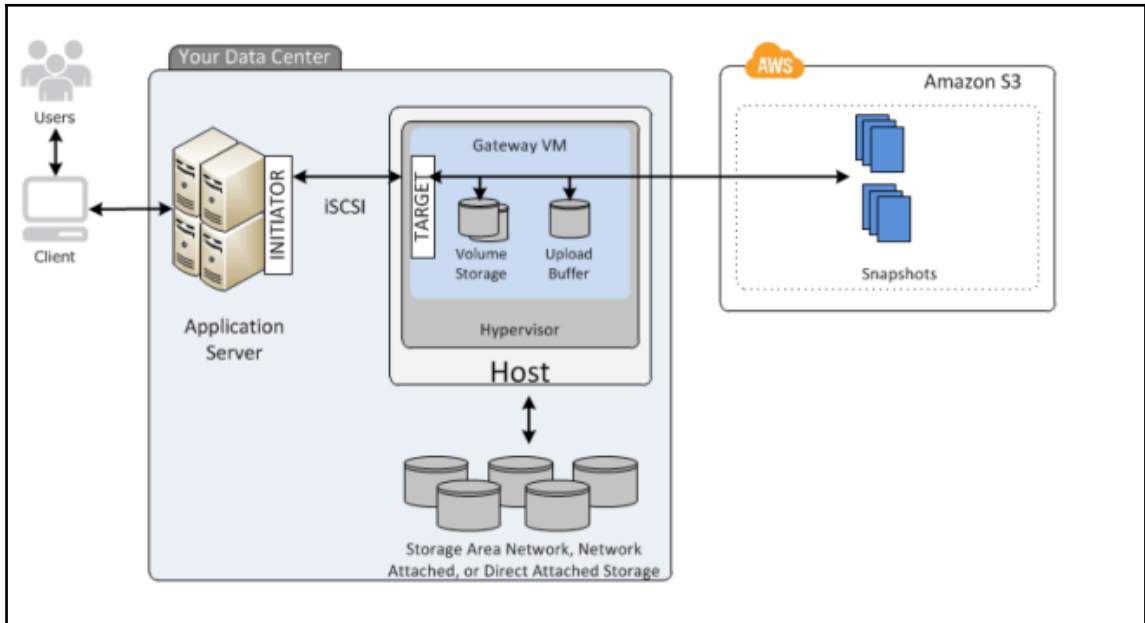


Figure 9.4: Gateway stored volume

The key features of a gateway-stored volume are as follows:

- It maintains the entire data set locally, providing low-latency access to the data
- It stores all your data locally first and then asynchronously takes a point-in-time backup of this data as a snapshot to S3
- You can attach gateway-stored volumes as iSCSI devices on on-premise application servers
- It supports up to 12 gateway-stored volumes per storage gateway applications
- Each gateway stored volume can be from 1 GiB to 16 TiB in size with a total volume storage limit of 192 TiB

- Gateway-stored volumes can be restored as an EBS volume on EC2 instance
- Gateway-stored volume snapshots can be restored as gateway storage volume or you can create an EBS volume out of it and use it on an EC2 instance
- A maximum size of an EBS volume created out of a snapshot is 16 TiB and you cannot create an EBS volume out of the snapshot if it is more than 16 TiB in size
- AWS stores gateway-stored volume data and snapshots in Amazon S3 and the data is encrypted at rest with SSE; you cannot access the data with S3 API or any other tools
- Gateway VM allocates storage in two parts:
  - Volume storage
    - It is used for storing actual data
    - You can map it to an on-premise **DAS (Direct-Attached Storage)** or **SAN (Storage Area Network)**
  - Upload buffer
    - Upload buffer serves as a staging location, before uploading the data to S3
    - It uploads data on an encrypted SSL connection to AWS and stores it in an encrypted format on S3

## Tape-based storage solutions

A tape gateway serves as a replacement for an on-premise tape drive for backup purposes. It stores data on Amazon Glacier for long-term archival. It provides a virtual tape, which can scale based on requirement. It also reduces the burden of managing the physical tape infrastructure.

There are two types of tape-based storage solutions: **Virtual Tape Library (VTL)** and **Virtual Tape Shelf (VTS)**.

## VTL

VTL is a scalable and cost effective virtual tape infrastructure, which seamlessly integrates with your existing backup software.

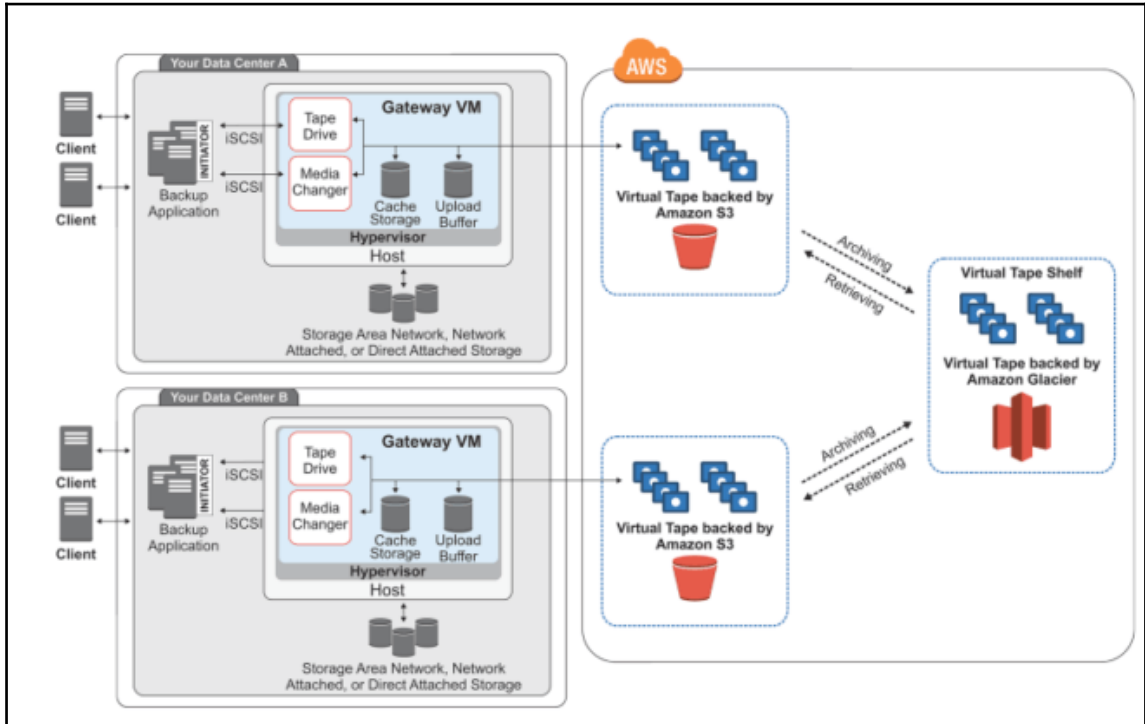


Figure 9.5: Gateway-Virtual tape library

- It provides a low-cost and long-duration archival option in Glacier.
- It provides a virtual tape infrastructure, which can scale based on requirements. It also reduces the burden of managing the physical tape infrastructure.
- It allows you to continue using your existing tape-based backup software for storing data on virtual tape cartridges, which can be created on a gateway-VTL.
- Each gateway-VTL is equipped with preconfigured media changer and tape drives. These are made available to existing backup applications as iSCSI devices. You can add tape cartridges as needed for archiving the data.

- A Gateway VTL contains the following components :
  - Virtual tape
    - Virtual tape emulates a physical tape cartridge wherein the data is stored in AWS storage solutions
    - You can have up to, 1500 tapes in each gateway or up to 150 TiB of total tape data
    - Each tape can store from 100 GiB to a maximum of 2.5 TiB of data
  - VTL
    - VTL emulates a physical tape library wherein the data is stored in S3
    - When a backup software writes data to the gateway, at first the data is stored locally and, subsequently, asynchronously uploaded to virtual tapes in S3
  - VTS
    - VTS works just like an offsite tape holding facility
    - In VTL, data is stored on S3 whereas VTS stores data in Glacier
    - As it uses Glacier for data archival, it becomes an extremely low-cost data archival option
    - VTS resides in the same region where the Storage Gateway is created and there is always only one VTS irrespective of the number of gateways created in an AWS account
    - The gateway moves a virtual tape to VTS when the backup software ejects a tape
    - You can retrieve tapes from VTS only after retrieving the tapes from VTL and it takes around 24 hours for the tapes to be available in the VTL

- Gateway allocates storage in two parts:
  - Cache storage
    - It serves as an on-premise durable storage
    - It caches the data locally before uploading it to S3
    - It provides low-latency access to frequently accessed data
  - Upload buffer
    - Upload buffer serves as a staging location, prior to uploading the data to S3
    - It uploads data on an encrypted SSL connection to AWS and stores it in an encrypted format on S3

## AWS Snowball

AWS Snowball comes in a hardware form and can be used with the AWS dashboard or API. It is available in two different sizes, 50 TB and 80 TB. It can be used to transfer **petabytes (PB)** of data into and from AWS S3. Dedicated Snowball software is made available by AWS to perform data transfer in a compressed, encrypted, and secure manner. You can attach multiple AWS Snowball devices at the same time to on-premises network backbone. Perform the following steps to obtain AWS Snowball:

- Sign in to your AWS account and create a job inside the AWS Snowball management console. While creating a job, you need to provide information such as shipping details to receive Snowball device(s), job details mentioning the region, the AWS S3 bucket name, and so on. You also need to provide security details such as ARN of the AWS IAM role and master key from AWS KMS.



- Once the job is created, the Snowball device is shipped to the given shipping address. The following *Figure 9.6* illustrates a Snowball device:



Figure 9.6: Snowball device and its features

Reference URL:

<https://image.slidesharecdn.com/clouddatamigration1272016final-160127210855/95/aws-january-2016-webinar-series-cloud-data-migration-6-strategies-for-getting-data-into-aws-14-638.jpg?cb=1466106757>

- Once the device is received, connect it to the network. It has two panels, one in the front and another in the back. Flipping the front panel on the top gives access to the E Ink-based touch screen to operate. Network and power cables can be connected on the back side.
- When a Snowball is connected to an on-premise network, it becomes ready to transfer data. Snowball requires credentials to start a data transfer job. These credentials can be retrieved from the AWS dashboard or API. These credentials are encrypted with a manifest file and unlock code. Without the manifest file and unlock code, it is not possible to communicate with the Snowball. AWS provides a Snowball client, to transfer data from on-premise to the Snowball device.
- It is highly recommended that you do not delete the on-premise copy of the data until the data is successfully migrated to the AWS S3 bucket.

- Once the job is complete, disconnect the device from the network and return the device to the shipping address displayed on the display panel. When you create a job, at that time only regional shipping carriers are assigned. For India, Amazon logistics and for rest of the world, UPS are the shipping carrier partners.
- Once the device is shipped back to AWS, the job progress status indicating the movement of data from Snowball to AWS S3 bucket can be tracked on the AWS dashboard or through APIs.

## **AWS Snowmobile**

AWS Snowmobile is an exabyte data transfer service. This hardware come in a high cube shipping container of dimensions, 45 feet long, 8 foot wide, and 9.6 foot tall. Each Snowmobile truck can store up to 100 PB of data and at the same time multiple Snowmobile trucks can be connected to an on-premise infrastructure. It uses 256-bit encryption and a master key for encryption can be managed with AWS KMS. It comes with GPS tracking, alarm monitoring, 24/7 video surveillance, and an optional escort security vehicle while in transit.

When you request for a Snowmobile, AWS performs an assessment and subsequently transports the snowmobile to your designated location. An AWS resource configures it so that you can access it as network storage. During the entire period when the Snowmobile stays at your location, AWS personnel work with your team for assistance. The AWS personnel connect a network switch from your Snowmobile to your local network. Once the setup is ready, you can start the data transfer process from multiple sources in your network to the Snowmobile.

# 10

## AWS Relation Database Services

**AWS Relational Database Service (RDS)** is a fully managed relational database service from Amazon. RDS makes it easier for enterprises and developers who want to use a relational database in the cloud without investing much time and resources in managing the environment. AWS RDS supports six database engines: Amazon **Aurora**, **PostgreSQL**, **MySQL**, **MariaDB**, **Oracle**, and **Microsoft SQL Server**. It provides easy to use, cost-effective, and scalable relational databases in the cloud.

The advantages of Amazon RDS as follows:

- It's a fully managed service, which automatically manages backups, software and OS patching, automatic failover, and recovery.
- It also allows taking a manual backup of the database as a snapshot. Snapshots of a database can be used to restore a database as and when required.
- RDS provides fine-grained access control with the help of AWS IAM.

AWS RDS does not provide root access to the RDS instance. In short, RDS not allow the user to access the underlined host operating system. That means, you cannot login to server operating system. It also restricts access to certain system procedure and tables, which may require advance privileges.

After launching RDS in their service offerings, AWS was not providing option to stop an RDS instance for a very long time. Recently, an option to stop the RDS instance is introduced by Amazon. However, unlike EC2 instances, there are some limitations in stopping an RDS instance:

- Only a single AZ RDS instance can be stopped.

- An RDS instance can be stopped for maximum 7 consecutive days. After 7 days, the instance is automatically restarted

This way, by stopping an RDS instance, you can cut the cost for limited period of time. However, there is no limitation on restarting the instance or terminating all unused RDS DB instances to stop incurring the cost.

If a manual snapshot is not taken before terminating the RDS DB instance, it prompts you to take a final snapshot. Once an RDS DB instance is deleted, it cannot be recovered.

## Amazon RDS components

Amazon RDS components are as follows:

### DB instances

Each Amazon RDS engine can create an instance with at least one database in it. Each instance can have multiple user-created databases. Databases names must be unique to an AWS account and are called DB instance identifier. Each DB instance is a building block and an isolated environment in the cloud. These databases can be accessed using the same tools that are used to access standalone databases hosted in a data center. On top of standard tools AWS RDS instance can also accessed by the AWS Management Console, the API, and the CLI.

Each DB engine has its own version. With the help of a DB parameter group, DB engine parameters can be configured. These parameters help to configure DB instance performance. One DB parameter group can be shared among the same instances types of the same DB engine and version. These sets of allowed parameters vary according to the DB engine and its version. It is recommended to create individual DB parameter groups for each database to have legacy to fine tune as per business need each of them individually. When you choose an RDS instance type, it determines how many CPUs and memory is allocated to it. The most suitable instance type can be selected based on the performance need. Each DB instance can store a minimum of 5 GB and a maximum of 6 TB.

However, there some exceptions like Microsoft SQL Server RDS DB instances supports up to 4 TB of storage. Also, AWS periodically keeps revising this limit for different RDS engines. The minimum and maximum supported storage capacity may vary for each instance type. RDS supports magnetic, general purpose (SSD), and provisioned IOPS (SSD) storage types. RDS instances can be deployed within VPC. Based on the architectural needs, it can be deployed in a public subnet for accessing over the internet or in a private subnet for accessing it within the network.

## **Region and AZs**

AWS hosts its computing resources in data centers, spread across the Globe. Each geographical location where the data centers are located is called a region. Each region comprises multiple distinct locations that are called AZs. Amazon creates AZs in isolated locations such that a failure in one AZ does not impact other AZs in the region. AZs are interconnected with low-latency network connectivity within a region. When you launch an application in multiple AZs, it provides you with high availability and protects you from the failure of an AZ.

An RDS DB instance can be provisioned in several AZs by selecting the Multi-AZ deployment option. It can also be used for DR sites. It is advisable to create RDS in multiple AZs for avoiding single points of failure. It automatically maintains synchronous replicas across multiple AZs. RDS synchronizes DBs between primary and secondary instances. In case a primary instance fails, the load is automatically shifted to a secondary instance.

## **Security groups**

Security groups acts like a firewall. They controls access to a RDS DB instance, by specifying the allowed source port, protocol and IPs. Three types of security group can be attached with Amazon RDS DB instances – DB security groups, VPC security groups, and EC2 security groups.

In general, a DB security group is used when the RDS instance is not in the VPC. The VPC security group is used when RDS instance is within the VPC. The EC2 security group can be used with EC2 instances as well as RDS instances.

## **DB parameter groups**

Over a period when a RDS instance is used in enterprise applications, it may be required to tune certain allowed and common parameters to optimize the performance based on the data insertion and retrieval pattern. The same DB parameter group can be attached to one or more DB instances of the same engine and version type. If not specified then the default DB parameter group with default parameters will be attached. Before creating an RDS instance it is recommended to create DB parameter groups.

# DB option groups

DB options groups are used to configure RDS DB engines. With the help of the DB option groups, some of the DB engines can provide additional features for data management, database management, and can also provide additional security features. RDS supports DB options group for MariaDB, Microsoft SQL Server, MySQL, and Oracle. Before creating an RDS instance, it is recommended to create DB option groups.



Amazon RDS charges are based on instance type, running time, storage size, type and I/O requests, total backup storage size, and data in and out transfers.

# RDS engine types

Amazon RDS supports six DB engine types: **Amazon Aurora**, **MySQL**, **MariaDB**, **Microsoft SQL Server**, **Oracle**, and **PostgreSQL**. The following table helps us understand the connecting port and protocol for each of these DB instances:

Amazon RDS engine types	Default port	Protocol
Aurora DB	3306	TCP
MariaDB	3306	TCP
Microsoft SQL	1433	TCP
MySQL	3306	TCP
Oracle	1521	TCP
PostgreSQL	5432	TCP

Default port and protocol to connect with each of Amazon RDS engine types

The Amazon RDS engine for Microsoft SQL server and Oracle supports two licensing models: license included and **Bring Your Own License (BYOL)**. In case you are already invested in purchasing licenses for such databases, it can also be used as a BYOL with Amazon RDS to minimize monthly billing.



Supported instance types may vary for each Amazon RDS engine.

## Amazon Aurora DB

Amazon Aurora is a MySQL and PostgreSQL-compatible fully managed **Relational Database Management System (RDBMS)**. It provides a rare combination of performance and reliability like commercial databases and the cost effectiveness of open source databases. Amazon RDS also provides push-button migration tools to convert your existing Amazon RDS for MySQL applications to Amazon Aurora. It is also possible to use the code, tools, and applications you use today with your existing PostgreSQL databases with Aurora (PostgreSQL).

Creating an Amazon Aurora DB instance will create a DB cluster. It may consist of one or more instances along with a cluster volume to manage the data. These clusters consist of two types of instance: Primary instance and Aurora Replica. Actually the Aurora cluster volume is a virtual database storage volume of type SSD and it spans across multiple AZs in the same region. Each AZ will have a copy of the cluster data. Each Aurora cluster grows automatically as the amount of data in the database grows. It can grow up to 64 TB. Table size is limited to the cluster volume size hence the table can grow up to 64 TB in size:

- **Primary instance:** Performs read, writes, and modifies data to the cluster volume. Each Aurora DB cluster has one primary instance.

- **Aurora Replica:** Performs only read operations. Each Aurora DB cluster supports up to 15 Aurora Replicas plus one primary instance. Amazon RDS Aurora instance availability can be increased by spreading Aurora Replicas across multiple AZs. The following *Figure 10.1* helps to understand this:

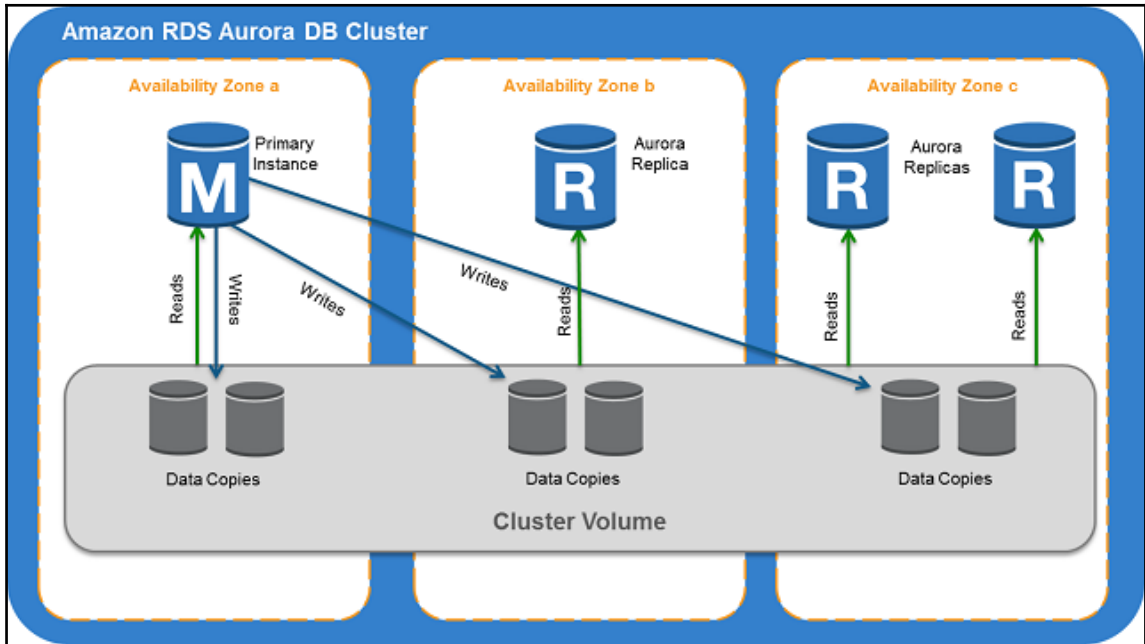


Figure 10.1: Amazon RDS Aurora primary and replica

Reference URL: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Overview.html>

With the help of various endpoints such as the cluster endpoint, reader endpoint, and instance endpoint it is possible to connect to the Aurora DB cluster. Each endpoint consists of a domain name and port separated by a colon and both are discussed as follows:



An endpoint is a URL to access an AWS resource. It can be used to access the DB instance from an application, script, or as a CNAME in a DNS.

- **Cluster endpoint:** To connect with primary instances to perform data read, write, and modification operations. The primary instance also has its own endpoint. An advantage of the cluster endpoint is, it always points to the current primary instance.



- **Reader endpoint:** To connect with one of the Aurora Replicas to perform read operations. This endpoint automatically loads a balanced connection across available Aurora Replicas. In case a primary instance fails, then one of the Aurora Replicas will be promoted as a primary instance and in that situation all the read requests will be dropped.
- **Instance endpoint:** To directly connect with the Primary or Aurora replica instance.

It is designed to be a highly durable, fault tolerant, and reliable and provides the following features:

- **Storage Auto-repair:** It maintains multiple copies of data in three AZs to minimize the risk of disk failure. It automatically detects the failure of a volume or a segment and fixes it to avoid data loss and point-in time recovery.
- **Survivable cache warming:** It *warms* the buffer pool page cache for known common queries, every time a database starts or is restarted after failure to provide performance. It is managed in a separate process to make it survive independently of the database crash.
- **Crash recovery:** Instantly recovers from a crash asynchronously on parallel threads to make a database open and available immediately after crash.

Amazon RDS upgrades the newer major version of Aurora to the cluster only during the system maintenance windows. Timing may vary from region to region and cluster settings. Once a cluster is updated, the database restarts and may experience a downtime for 20 to 30 minutes. It is highly recommended to configure maintenance windows setting to match an enterprise's business requirement to avoid unplanned downtime. But in the case of a minor version upgrade, Amazon RDS schedules an automatic upgrade for all Aurora DB database engines for all Aurora DB clusters. It is optional to allow that update at that scheduled time. It can be manually selected and updated at the desired schedule. Otherwise, it gets applied at the next automatic upgrade for a minor version release.



Amazon Aurora offers *lab mode*. By default it is disabled. It can be enabled for testing current instance and available features in the currently offered version. New features can be tested before applying to production instance.

## Comparison of Amazon RDS Aurora with Amazon RDS MySQL

The following table helps in understanding difference between Aurora and MySQL DB engines:

Feature	Amazon RDS Aurora	Amazon RDS MySQL
Read scaling	Supports up to 15 Aurora Replicas with minimal impact on the write performance.	Supports up to only five Read Replicas with some impact on the write operation.
Failover target	Aurora Replicas are automatic failover targets with no data loss.	Manually Read Replicas are promoted as a master DB instance with potential data loss.
MySQL version	Supports only MySQL version 5.6.	Supports MySQL version 5.5, 5.6, and 5.7.
AWS region	Not available in some regions.	Available in all regions.
MySQL storage engine	It supports only InnoDB storage engine type. Tables from other types of storage engine are automatically converted to InnoDB.	Supports both MyISAM and InnoDB.
Read replicas with a different storage engine than the master instance	MySQL (non-RDS) Read Replicas that replicate with an Aurora DB cluster can only use InnoDB.	Read Replicas can use both MyISAM and InnoDB.
Database engine parameters	Some parameters apply to the entire Aurora DB cluster and are managed by DB cluster parameter groups. Other parameters apply to each individual DB instance in a DB cluster and are managed by DB parameter groups.	Parameters apply to each individual DB instance or Read Replica and are managed by DB parameter groups.

Detailed comparison between Amazon RDS Aurora and Amazon RDS MySQL

Reference URL: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Overview.html>

## MariaDB

MariaDB is a community version of MySQL RDBMS under GNU GPL license. It maintains a high level of compatibility with MySQL.

Amazon RDS MariaDB manages versions as X.Y.Z where X.Y denotes a major version and Z is the minor version. For example, a version change from 10.0 to 10.1 is considered a major version change, while a version change from 10.0.17 to 10.0.24 is a minor version change. In general, within three to five months it will be introduced in Amazon RDS MariaDB. Amazon RDS Management Console, CLIs, or APIs can be used to perform common tasks such as creating an instance, resizing the DB instance, creating and restoring a backup, and so on.



Minor version support may not be available in all AWS regions.

Amazon RDS MariaDB supports multiple storage engines, but all of them are not optimized for recovery and durability. At present, it fully supports the **XtraDB** storage engine. It supports point in time restore and snapshot restore. It also supports the **Aria** storage type engine, but it may have a negative impact on recovery in the case of instance failure. However, to manage spatial geographical data, it is suggested to use the Aria storage type as the XtraDB storage type doesn't support.



Amazon RDS MariaDB is available in all regions except AWS GovCloud (US) (us-gov-west-1).

Amazon RDS supports two kinds of upgrade for running instances: major version upgrades and minor version upgrades. Minor version upgrades can take place automatically when auto minor version upgrade is enabled from the instance configuration options. In all other cases, upgrading minor versions or major versions requires manual upgrades.

## Microsoft SQL Server

It is possible to run Microsoft SQL Server as an RDS instance. It supports various versions of MS SQL such as from SQL Server 2008 R2 to SQL Server 2016. There are a few limitations for Microsoft SQL Server DB instances:

- Each Amazon RDS Microsoft SQL instance can have a maximum of 30 databases. Master and model databases are not counted as a database in this count.
- Some ports are reserved for internal purposes, and cannot be used for general purposes.
- It is not possible to rename a database when an RDS instance with Microsoft SQL Server is deployed in Multi-AZ mirroring.
- The minimum storage required is 20 GB with maximum 400 GB for the Web and Express edition. For the Enterprise and Standard edition a minimum of 200 GB and a maximum of 4 TB of storage is required. In case of larger storage is required, with the help of sharding across multiple DB instances this can be achieved.
- It is recommended to allocate storage based on future considerations. Once storage volume is allocated it cannot be increased due to the extensibility limitations of striped storage attached to Windows Server.
- It doesn't support some of the features of SQL Server such as SQL Server Analysis Services, SQL Server Integration Services, SQL Server Reporting Services, Data Quality Services, and Master Data Services. To use these features it is required to configure Microsoft SQL Server on an Amazon EC2 instance.
- Due to the limitations of Microsoft SQL Server, point in time restore may not work properly until the database has been dropped successfully.

Amazon RDS Microsoft SQL instances support two licensing options: License Included and BYOL. License Included mode is good for the enterprise if you have not already purchased a license. In case you have already purchased a license and are using it in an existing infrastructure, when migration to AWS cloud has been done, once the instance is running with the help of management console or CLI, BYOL can be implemented. When it is deployed in a Multi-AZ mode, the secondary instance is passive and only provides read operations until failover takes place. BYOL is supported for the following Microsoft SQL Server database editions:

- Microsoft SQL Server Standard Edition (2016, 2014, 2012, 2008 R2)
- Microsoft SQL Server Enterprise Edition (2016, 2014, 2012, 2008 R2)

It may be required to upgrade the Amazon RDS Microsoft SQL Server instance, Amazon RDS supports major version and minor version upgrades. In either type, it is essential to perform such upgrades manually. It requires downtime and the total time depends on the engine version and the size of the DB instance.

## MySQL

Amazon RDS supports various versions of MySQL. It is also compliant with many leading industry leading standards such as HIPAA, PHI, BAA, and many others. MySQL versions are organized as X.Y.Z where X.Y indicates a major version and Z indicates a minor version. Most of the major versions are supported in most of the regions, but it is recommended to check the availability of desired major and minor version in region, where you are planning to create primary and DR sites. A new version of MySQL is available with the Amazon RDS MySQL instance usually within three to five months. While upgrading MySQL to a newer version, it is possible to maintain compatibility with specific MySQL versions. Major versions can be upgraded from MySQL 5.5 to MySQL 5.6 and then MySQL 5.6 to MySQL 5.7. Usually major version upgrades complete within 10 minutes, but it may vary based on the DB instance type. Minor versions automatically get updated when AutoMinorVersionUpgrade is enabled. Amazon RDS policy on deprecation of MySQL is as follows:

- The major version is supported for three years from the release such as 5.5, 5.6, 5.7, and upcoming
- The minor version is supported for a year from release such as MySQL 5.546
- Three months of grace are provided from the date of version deprecation date



It is essential to perform an OS update (if any are available) before upgrading Amazon RDS MySQL 5.5 DB instance to MySQL 5.6 or later.

Amazon RDS MySQL 5.6 and later supports memcached in an option group. Amazon RDS MySQL also supports various storage engines, but point in time recovery is only supported by InnoDB. Amazon RDS currently does not support the following MySQL features:

- Global Transaction IDs
- Transportable table space
- Authentication plugin

- Password strength plugin
- Replication filters
- Semi-synchronous replication

It is possible to create a snapshot for an Amazon RDS MySQL instance storage volume. Each snapshot is based on the MySQL instance engine version. As it is possible to upgrade the version of an Amazon RDS MySQL instance, it is also possible to upgrade the engine version for DB snapshots. It supports DB snapshot upgrades from MySQL 5.1 to MySQL 5.5.

## Oracle

At the time of writing, the following Oracle RDBMS versions are supported by Amazon RDS Oracle engine:

- Oracle 12c, Version 12.1.0.2
- Oracle 11g, Version 11.2.0.4

Amazon RDS Oracle engine also supports the following Oracle RDBMS versions, but soon they will be deprecated:

- Oracle 12c, Version 12.1.0.1
- Oracle 11g, Version 11.2.0.3 and Version 11.2.0.2

Oracle RDS can be deployed within VPC and can perform point-in-time recovery and scheduled or manual snapshots. Optionally, it can be deployed in Multi-AZ to get high-availability and failover. At the time of creating a DB instance master user gets DBA privileges with some limitations, for example SYS user, SYSTEM user, and other DB administrative user accounts cannot be used.

Amazon RDS Oracle instances support two licensing options: License Included and BYOL. Once an instance is running with the help of management console or CLI, BYOL can be implemented. In the case of License Included, it supports the following Oracle database versions:

- Oracle Database Standard Edition One (SE1)
- Oracle Database Standard Edition Two (SE2)

BYOL supports the following license models:

- Oracle Database Enterprise Edition (EE)
- Oracle Database Standard Edition (SE)
- Oracle Database Standard Edition One (SE1)
- Oracle Database Standard Edition Two (SE2)

Amazon allows you to change Oracle RDS instance types, however, if your DB instance uses a deprecated version of Oracle, you cannot change the instance type. Such RDS instances are autocratically updated to new version based on a cut-off date provided by Amazon. For more details on supported versions, deprecated version, and cut-off date for upgrading the deprecated versions, you can check following URL: [http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP\\_Oracle.html](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Oracle.html).

It supports both major and minor version upgrades. While upgrading an Amazon RDS Oracle instance from 11g to 12c is a major version upgrade, it has to be done manually and it requires downtime. This downtime may vary based on the current engine version and size of the DB instance. While upgrading engine version it takes two snapshots. The first snapshot is taken just before upgrading, in the case of failure due to any reason it can be used to restore the database. The second snapshot is taken just after engine upgrade is completed. Once DB engine is successfully upgraded, it cannot be undone. If there is any requirement to rollback to previous version, you can create a new instance with the snapshot taken before upgrading the version. Oracle engine upgrade path may vary depending up on the current version running on the instance.

## PostgreSQL

The Amazon RDS PostgreSQL engine supports various versions of PostgreSQL. It also supports point-in-time recovery using periodically or manually taken snapshots, Multi-AZ deployment, provisioned IOPS, Read Replicas, SSL connection to DB and VPC. Applications such as *pgAdmin* or any other tool can be used to connect to PostgreSQL and run SQL queries. It is also compliant with many industry leading standards such as HIPAA, PHI, BAA, and many others.

At the time of creating an Amazon RDS PostgreSQL instance master user (super user) a system account is assigned to `rds_superuser` role with some limitations. More details about various supported PostgreSQL supported versions and their features, can be obtained from the URL: [http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP\\_PostgreSQL.html](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_PostgreSQL.html).



Amazon RDS supported and unsupported database engines can be installed and configured on Amazon EC2 instances as well. Compared to Amazon RDS, installing DB on Amazon EC2 gives more power to fine-tune database engines as gets root level access. Usually, when enterprise is looking for a managed service solution, Amazon RDS is preferred and when they are looking for more detailed fine-tuning hosting on Amazon EC2 is preferred.

## Creating an Amazon RDS MySQL DB instance

Amazon RDS MySQL DB instances can be created using Amazon Management Console, CLIs, or APIs and the steps are as follows:

1. Log in to the AWS Management Console with the appropriate user privileges and go to the Amazon RDS dashboard.
2. Select **Launch a DB Instance** as shown in the following *Figure 10.2*:

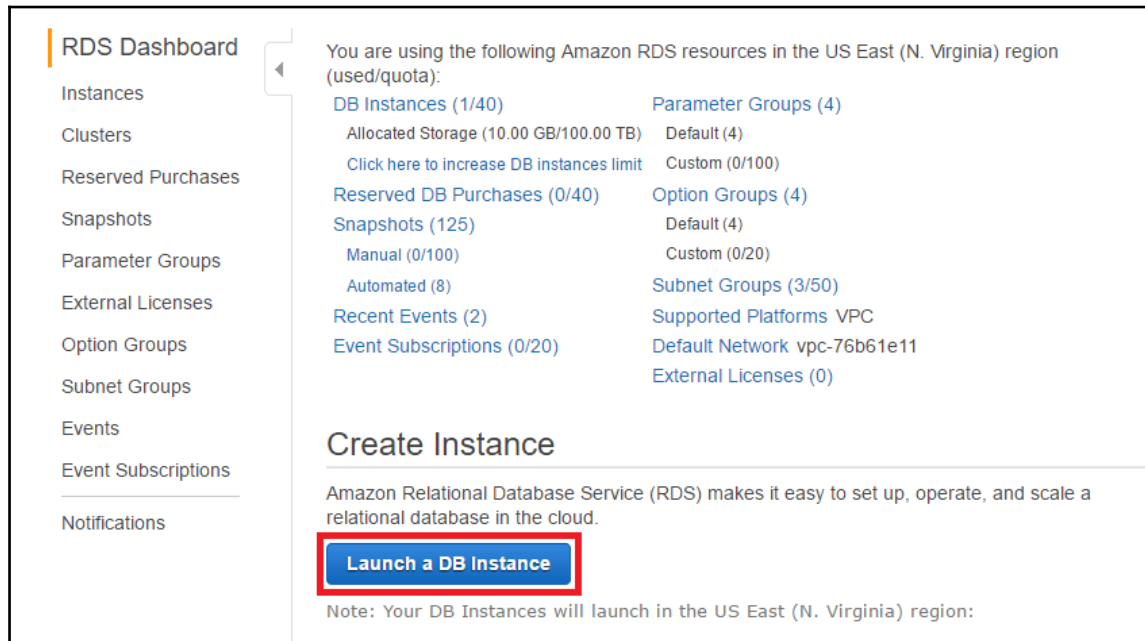


Figure 10.2: Select a DB instance



3. Select the engine type as MySQL, as shown in the following *Figure 10.3*:

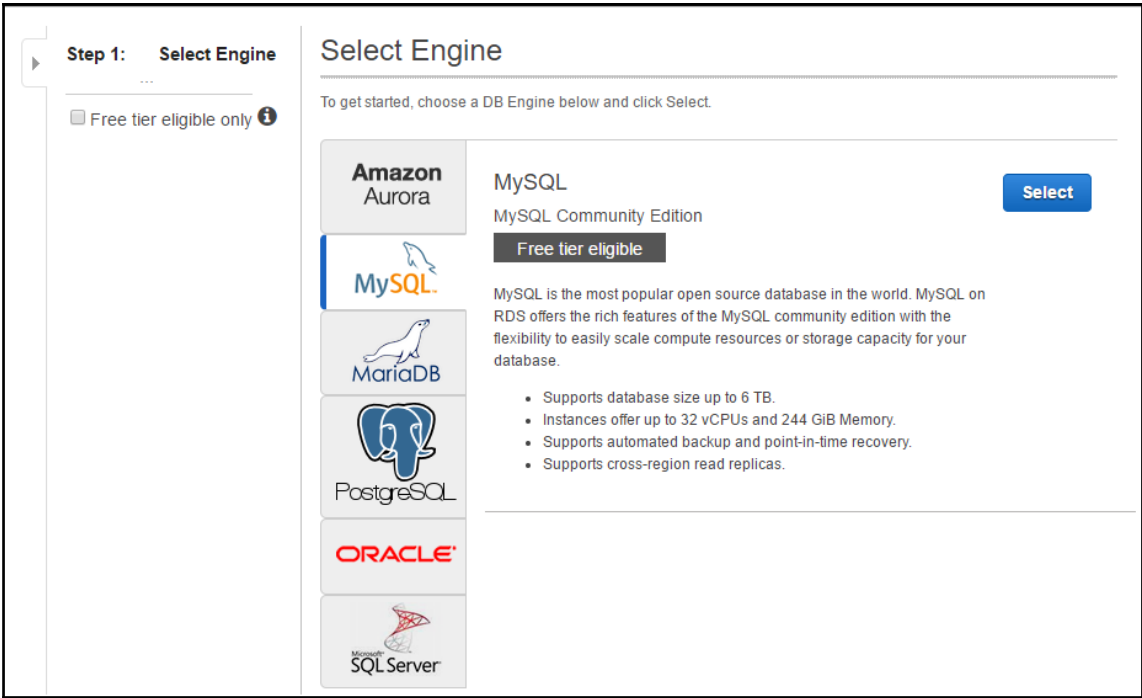


Figure 10.3: Select Amazon RDS engine type as MySQL



**Free tier** allows us to create a `t2.micro` single-AZ instance for the first year.

4. Select the **Production** type: **Dev/Test** or **MySQL Multi-AZ**, as shown in the following *Figure 10.4*. It is also suggested to switch to Amazon Aurora as it is seamlessly compatible with MySQL:

Step 1: [Select Engine](#)  
**Step 2: Production?**  
 Step 3: [Specify DB Details](#)  
 Step 4: [Configure Advanced Settings](#)

Do you plan to use this database for production purposes?

**Production**

☐ Amazon Aurora  
**Recommended**  
 MySQL-compatible, enterprise-class database at 1/10th the cost of commercial databases.

☐ MySQL  
 Use Multi-AZ Deployment and Provisioned IOPS Storage as defaults for high availability and fast, consistent performance.

**Dev/Test**

☒ MySQL  
 This instance is intended for use outside of production or under the RDS Free Usage Tier.

Billing is based on [RDS pricing](#).

[Cancel](#) [Previous](#) [Next Step](#)

Figure 10.4: Select Amazon RDS MySQL instance to deploy in a single or Multi-AZ

5. Specify Amazon RDS MySQL DB details as follows:
- **Licence Model:** At present it has only one license model: **general-public-licence**.
  - **DB Engine Version:** Amazon RDS MySQL engine supports various versions. Based on the enterprise IT requirement, the optimum and latest can be selected.
  - **DB Instance Class:** Select the RDS instance type. It decides the size of RAM, CPU, network performance, and EBS performance.
  - **Multi-AZ Deployment:** Select **Yes** to enable a standby replica of DB instance in another AZ for failover support.
  - **Storage Type:** Supports three types: **Magnetic**, **General Purpose (SSD)**, and **Provisioned IOPS (SSD)**. Storage type can be selected based on the required number of read/write operations.
  - **Allocated Storage:** Size of the storage volume to attach to the Amazon RDS DB instance.
  - **DB Instance Identifier:** It is a unique DB name for each DB within the AWS account.

- **Master Username and Master Password:** Master user is the user with the highest level of privileges within each Amazon RDS instance. It is used to create enterprise level users and grant them privileges to perform day-to-day activities and applications. It also defines passwords.

Step 1: [Select Engine](#)

Step 2: [Production?](#)

**Step 3: Specify DB Details**

Step 4: [Configure Advanced Settings](#)

Your current selection is eligible for the free tier.

[Learn More.](#)

Estimate your monthly costs for the DB Instance using the [RDS Instance Cost Calculator](#).

Specify DB Details

Free Tier

The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#).

☐ Only show options that are eligible for RDS Free Tier

Instance Specifications

DB Engine

mysql

License Model

general-public-license

DB Engine Version

MySQL 5.6.27

Review the **Known Issues/Limitations** to learn about potential compatibility issues with specific database versions.

DB Instance Class

db.t2.micro — 1 vCPU, 1 GiB RAM

Multi-AZ Deployment

No

Storage Type

General Purpose (SSD)

Allocated Storage\*

5

GB

Provisioning less than 100 GB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. [Click here](#) for more details.

Settings

DB Instance Identifier\*

learnAmazonRDSMySQL

Master Username\*

admin

Master Password\*

.....

Confirm Password\*

.....

\* Required

Cancel

Previous

Next Step

License type associated with the database engine

Figure 10.5: Amazon RDS MySQL DB instance details

[ 312 ]

6. **Configure Advanced Settings** as shown in *Figure 10.6*:

Step 1: [Select Engine](#)

Step 2: [Production?](#)

Step 3: [Specify DB Details](#)

Step 4: **Configure Advanced Settings**

### Configure Advanced Settings

Network & Security

VPC\*

Default VPC (vpc-~~abcd1234~~)

Subnet Group

default

Publicly Accessible

Yes

Availability Zone

us-east-1a

VPC Security Group(s)

Create new Security Group

Database Options

Database Name

DB1LearnRDS

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Database Port

3306

DB Parameter Group

default.mysql5.6

Option Group

default.mysql5-6

Copy Tags To Snapshots

☒

Enable Encryption

No

Backup

Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to detail [here](#).

Backup Retention Period

7 days

Backup Window

Select Window

Start Time

00 : 00 UTC

Duration

0.5 hours

Monitoring

Enable Enhanced Monitoring

No

Maintenance

Auto Minor Version Upgrade

Yes

Maintenance Window

No Preference

\* Required

Cancel

Previous

Launch DB Instance

Select to encrypt the given instance. Master key ids and aliases appear in the list after they have been created using the Key Management Service(KMS) console. [Learn More](#).

Figure 10.6: Amazon RDS MySQL DB instance advanced configuration

- **VPC:** Select a suitable network VPC.
- **Subnet Group:** Subnet selection depends on architectural design. It can be public or private based on requirements.
- **Publicly Accessible:** It should be selected as **Yes**, if you want to allow the access to the DB from the Internet. It creates a public DNS endpoint, which is Globally resolvable. Select **No** if you want this DB instance to be accessible only from within the network or VPC.
- **Availability Zone:** If you have any preference on which AZ you want to launch your instance, you choose the specific AZ as required. If you do not have any preference AZ, you can select **No Preference**. In this case, Amazon automatically launches the instance in appropriate AZ to balance the resource availability.
- **VPC Security Group(s):** Security groups act as a software firewall. One or more security groups can be attached to each Amazon RDS instance.
- **Database Name:** It can be a maximum of 64 alpha-numeric characters. For a given name, it will create a database within the DB instance. It can be blank also.
- **Database Port:** For Amazon RDS MySQL DB instances the default port is 3306. A default port list for all supported DB engines is given in a table.
- **DB Parameter Group:** It helps to configure DB engine parameters. It is recommended to create the DB parameter group before creating a DB instance. Once it is created it will appear in available drop-down list to use at the time of creating a DB instance. If it is not created before creating a DB instance then the default DB parameter group will be created. This group of parameters can be applied to one or more DB instances of the same engine type. When any dynamic parameter value is changed in the DB parameter group it gets applied immediately whether **Apply Immediately** has been enabled or not. In the case of static parameter value change to get effect, it is required to manually reboot the DB instance.
- **Option Group:** It is supported by the MariaDB, Microsoft SQL Server, MySQL, and Oracle Amazon RDS engines. With the help of option groups it is possible to fine-tune databases and manage data. Option groups can consist of two types of parameter: permanent and persistent.
- To change the persistent options value, it is required to detach the DB instance from the DB option group. When the option group is associated with any DB snapshot, then to perform point-in-time recovery using that DB snapshot it is required to create a new DB instance with the same DB options group. On the other hand, it is not possible to remove permanent options from option group. Also, an option group with permanent options cannot be detached from a DB instance.

- **Copy Tags To Snapshots:** In a backup window, it creates an instance level backup (that is, a snapshot) for the entire volume. By enabling this parameter, each snapshot will copy tags from the DB instance. These metadata can be very helpful to manage access policies.
- **Enable Encryption:** Enabling this option will encrypt data at rest in the DB instance's storage volume and subsequent snapshots. The industry standard AES-256 encryption algorithm is used. Amazon RDS automatically takes care of authentication and encrypts/decrypts data with a minimal impact on performance.
- **Backup Retention Period:** You specify snapshot retention period here in number of days. Snapshots which are older than specified number of days are automatically deleted after specified number of days. Any snapshot can be retained for maximum of 35 days.
- **Backup Window:** An automated backup time window can be specified in a UTC. During this scheduled time, everyday a snapshot will be taken. When any snapshot is aged for a backup retention period it will be automatically obsolete. It will help to achieve an organizational backup retention policy and minimize AWS billing by obsolete old snapshots.
- **Enable Enhanced Monitoring:** Amazon RDS maintains various performance metrics in an Amazon CloudWatch. The main difference between normal and enhanced monitoring is the source of the data. In the case of normal monitoring, CPU utilization of data is derived from the hypervisor. But for enhanced monitoring it is derived from the agent installed on a hypervisor. Data collection from these two sources may vary. In the case of small instance types, this difference can be bigger.
- **Auto Minor Version Upgrade:** Amazon RDS Instances can have two types of upgrade: major and minor. Major version upgrades may require down-time hence they are not performed automatically. Also it may not be possible to revert a major version upgrade. Minor version upgrades are compatible with previous versions and may not require down-time; hence it is performed automatically during scheduled maintenance.

- **Maintenance Window:** Amazon allows you to specify maintenance window. During the maintenance window, Amazon may upgrade DB instance's minor version or DB cluster's OS. Upgrade of the underlined OS or DB version may bring performance implications. Considering this, you should carefully define the maintenance window. Maintenance window definition allows you to define the starting day of the week, hour of the day, minute of the hour, and the total allocated time to perform maintenance activity. Once the maintenance activity begins and if it requires more time to complete the maintenance, it doesn't terminate in between. It stops only after completing the maintenance tasks.

## Monitoring RDS instances

Once an Amazon RDS instance is created as per the present need, it is very important to observe its performance with constantly changing business requirements and application loads. It is possible to monitor the instance's CPU utilization, DB connections, free storage space, free memory, and many other parameters. It helps to identify bottlenecks and also will give the opportunity to minimize monthly billing by reducing the resource size if it is underutilized.

An alarm can be configured to take action on a specified threshold. For example, if CPU usage is above 70% for a specified consecutive time period, then send SNS notifications to the DBA. Such an alarm can be created either from the CloudWatch dashboard or from the Amazon RDS dashboard.

To create a CloudWatch alarm from the Amazon RDS dashboard perform the following steps:

1. Go to the Amazon RDS dashboard and select the desired DB instance from the list of running DB instances.
2. Click **Show Monitoring** to get the list of supported metrics. For example, here we have selected the CPU utilization metric and selected **Create Alarm**.

3. Create an alarm by specifying the threshold and other relevant details such as the SNS topic to use to send notifications, CPU utilization threshold, consecutive time period, and alarm name as shown in the following *Figure 10.7*:

The screenshot shows the 'Create Alarm' window in the AWS RDS console. On the left is a sidebar with navigation links: Instances, Clusters, Reserved Purchases, Snapshots, Parameter Groups, External Licenses, Option Groups, Subnet Groups, Events, Event Subscriptions, and Notifications. The main area is titled 'Create Alarm' and contains the following fields:

- Send a notification to:** A text input field with a 'create topic' link to its right.
- Whenever:** A dropdown menu set to 'Average'.
- of:** A dropdown menu set to 'CPU Utilization'.
- Is:** A dropdown menu set to '>='.
- Percent:** A text input field for the threshold.
- For at least:** A text input field set to '1' followed by 'consecutive period(s) of 5 Minutes'.
- Name of alarm:** A text input field.

On the right side of the form is a line graph titled 'CPU Utilization Percent' for 'database1'. The graph shows a fluctuating line representing CPU usage over time, with labels for 05/01 08:00, 10:00, and 12:00. At the bottom right are 'Cancel' and 'Create Alarm' buttons.

Figure 10.7: Create CloudWatch alert and action from Amazon RDS dashboard.

## Creating a snapshot

A snapshot is a frozen image of the DB instance's storage volume. It helps to restore a database to a particular point-in-time. Usually, point-in-time recovery is performed when a database is corrupted or by mistake some data has been dropped (that is, deleted) to bring a database back to the last healthy state. At the time of creating an Amazon RDS instance, a daily snapshot schedule has been already configured. But sometimes it may be required to take a manual snapshot of the DB instance before performing any maintenance task on the database. Snapshot will back up an entire DB instance. It will include all databases and tables and other resources existing on it.

Creating a snapshot for a Multi-AZ DB instance doesn't bring many performance implications. But taking a snapshot for a single-AZ DB instance may suspend DB I/O for a few seconds to minutes. Manual snapshots can be taken using Amazon Management Console, CLI, or APIs. To take a manual snapshot using the management console perform the following steps:

1. Select the desired DB instance.
2. Select **Take Snapshot** from the drop-down menu **Instance Actions**, available above the list of the running RDS instances:



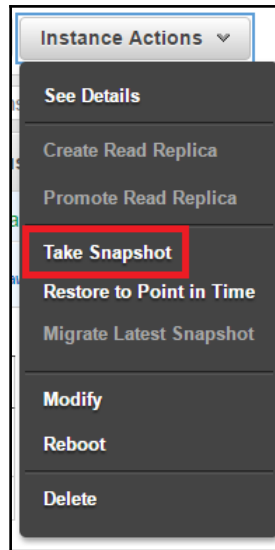


Figure 10.8: Take manual Amazon RDS DB instance snapshot

3. Provide the relevant **Snapshot Name** as shown in the following *Figure 10.9*:

A screenshot of the 'Take DB Snapshot' dialog box. The title is 'Take DB Snapshot'. Below the title, it says 'To take a snapshot of this DB instance you must provide a name for the snapshot.' There is a label 'DB Instance' followed by the text 'database1' and an information icon. Below that is a label 'Snapshot Name' followed by a text input field and an information icon. At the bottom right, there are two buttons: 'Cancel' and 'Take Snapshot' (which is highlighted in blue).

Figure 10.9: Provide Snapshot Name, while taking manual snapshot

## Restoring a DB from a snapshot

A snapshot can only be restored by creating a new instance. You cannot restore a snapshot to an existing instance. While restoring the snapshot to a new RDS instance, you can have a different storage volume type from the one used in the snapshot.

Creating an RDS DB instance from a snapshot, automatically attaches a default parameter group and security group to it. Once a DB instance is created, it is possible to change the attached parameter group and security group for that instance.

By restoring a snapshot, the same option group associated with the snapshot will get associated to the newly created RDS DB instance. Options groups are platform-specific: VPC or EC2-Classical.

Creating a RDS DB instance inside a particular VPC will link a used option group with that particular VPC. It means when the snapshot is created for that DB instance it cannot be restored in a different VPC. To do that it requires us to either attach a default options group or create a new options group and attach it to the newly created DB instance from the snapshot.

Creating a DB instance from a snapshot also requires us to provide parameters such as **DB Engine**, **Licence Model**, **DB Instance Class**, **Multi-AZ Deployment**, **Storage Type**, **DB Instance Identifier**, **VPC**, **Subnet Group**, **Publicly Accessible**, **Availability Zone**, **Database Name**, **Database Port**, **Option Groups**, and other parameters that we define at the time of creating a new Amazon RDS DB instance.



It is also possible to copy and share an Amazon RDS snapshot from one region to another and share it among multiple AWS accounts respectively. It may require us to create a DB instance from a snapshot in a different region or AWS account.

## Changing a RDS instance type

An RDS instance type is generally changed to accommodate additional resource requirement or for downgrading an existing instance type which is underutilized. For changing the instance type, perform the following steps:

1. From the list of RDS DB instances select the desired instance to modify the instance type and select **Modify** from the **Instance Actions** drop-down menu. The drop-down menu is shown in the following *Figure 10.10*:

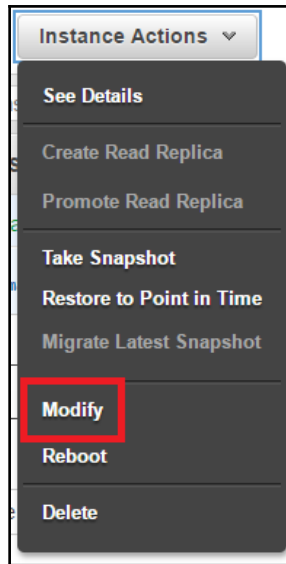


Figure 10.10: Instance Actions drop down menu to select Modify

2. Modifying a DB instance does not only allow us to change the DB instance type, it also allows us to change many other parameters that are provided at the time of creating a DB instance such as subnet group, security group, and many more options. At the end of the parameters that can be changed, an option is available to apply changes now or wait until a next maintenance window, as shown in the following *Figure 10.11*:



Figure 10.11: DB instance change parameters to Apply Immediately or wait till next maintenance window

3. If DB performance is throttling, you can change the DB instance parameters and apply them immediately. If you do not apply them immediately, the changes are automatically applied during the next maintenance window. Some modifications such as parameter group changes may require us to reboot DB instances. It is advisable to test any changes in a test environment first, before making the changes into productions environment directly.



It is best practice to test such changes in a test environment first, before making changes into productions directly.

## Amazon RDS and VPC

Before 2013, AWS used to support EC2-Classic. All AWS account created after 2013-1-04, it only supports EC2-VPC. If an AWS account only supports EC2-VPC, then a default VPC is created in each region and a default subnet in each AZs. Default subnets are public in nature. To meet enterprise requirements, it is possible to create a custom VPC and subnets. This custom VPC and subnet can have a custom CIDR range and can also decide which subnet can be public and which one can be private. When an AWS account only supports EC2-VPC, it has no custom VPC is created, then Amazon RDS DB instances are created inside a default VPC. Amazon RDS DB instances can also be launched into a custom VPC just like EC2 instances. Amazon RDS DB instances have the same functionality in terms of performance, maintenance, upgrading, recovery, and failover detection capability, irrespective of whether they are launched in a VPC or not.

## Amazon RDS and high availability

ELB and Auto Scaling can be used with Amazon EC2 to perform load balancing and launching or terminating an EC2 instance to match the load requirement. Auto Scaling cannot be used with Amazon RDS. Amazon RDS supports Multi-AZ deployment to provide high availability and failover. By enabling Multi-AZ deployment, Amazon RDS creates two instances of the same instance type and configuration with individual endpoints in two separate AZs. The sole purpose of another DB instance is to maintain a synchronous standby replica. The standby replica receives traffic only when failover takes place. It can not be used for load balancing or serving read-only traffic. For serving read-only traffic, read replicas can be created, which is different from creating Multi-AZ instances. At present while writing this book, Amazon RDS supports six DB engines. Four out of the six DB engines that is, Oracle, PostgreSQL, MySQL, and MariaDB can perform failover from primary DB instance to secondary DB instance using Amazon's failover mechanism. Microsoft SQL Server RDS engine uses SQL Server mirroring for high availability. Amazon Aurora cluster creates at least three copies of data across Multi-AZs within the same region, which can fulfill high availability requirement. In Amazon Aurora, in case of primary DB instance fails, one of the Aurora Replicas is promoted as a primary.

The following *Figure 10.12* helps to understand the Amazon RDS DB primary and secondary instance in a VPC where the primary instance is denoted as **M** and secondary instance is denoted as **S**:

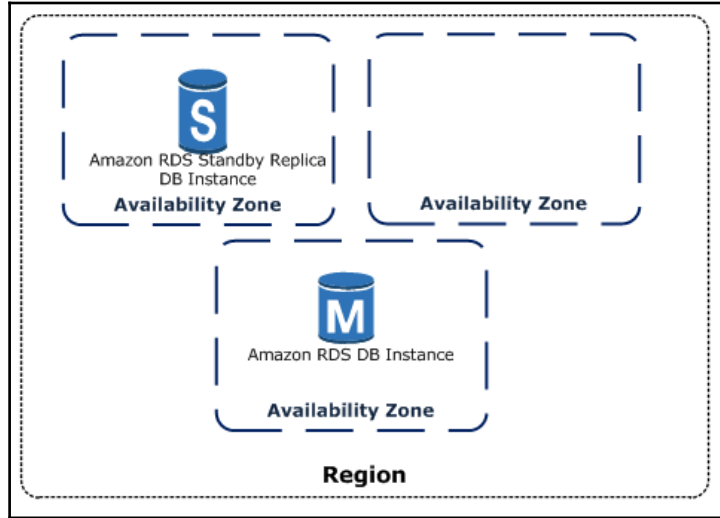


Figure 10.12: Amazon RDS DB instance in a Multi-AZ

Reference URL : <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html>



When using the BYOL licensing model, you must have a license for both the primary instance and the standby replica.

## Connecting to an Amazon RDS DB instance

Once the Amazon RDS DB instance is created, you can connect to it for performing read/write operation as well as for performing day-to-day maintenance activities. Before connecting to the DB instance, ensure that the port to connect with DB instance is allowed in the firewall or security group. Also ensure that the source IP from where you need to connect to DB instance is allowed in the security group.

## Connecting to an Amazon Aurora DB cluster

Aurora DB clusters consist of a primary instance and Aurora Replica. A separate endpoint is available for the primary instance, Aurora Read instance, or a group of Aurora Read Instances. In line with the task you want to carry out, it is possible to use any of these endpoints in scripting, application, or manually connecting them. Tools used to connect with MySQL databases can be used to connect to Amazon Aurora cluster DB instances.

You can refer to following syntax for connecting to an Aurora DB:

```
mysql -h <aurora-cluster-endpoint> --ssl-ca=[full path]rds-combined-ca-bundle.pem --ssl-verify-server-cert
```

## Connecting to a MariaDB Instance

Amazon RDS MariaDB instance up and running has a valid endpoint. It can be used with an application, client, or tool to connect with the DB instance. By default it uses port 3306 and the TCP protocol.



Amazon RDS MariaDB instances can be accessed from the `mysql` command line utility. HeidiSQL is a GUI-based utility and it can be used to connect MariaDB instances.

The following `mysql` command helps to understand syntax:

```
mysql -h <endpoit> -p 3306 -u <masteruser> -p
```



It is possible to provide a password immediately after the `-p` parameter. Best practice is not to provide it with the command, but to provide it at runtime when prompted for it. Based on the memory (instance type) the number of connection limit is derived. DB instances with higher memory have a higher connection limit. MariaDB maximum possible connection number is defined in the `max_connections` parameter.

## Connecting to a MySQL instance

By providing an Amazon RDS endpoint, MySQL DB instances can be connected using a standard MySQL client application or utility. Connecting to MySQL DB instances is similar to connecting MariaDB using a MySQL command-line tool:

```
mysql -h <endpoint> -p 3306 -u <masteruser> -p
```



Amazon RDS DB instance endpoints can be obtained from the RDS console or using CLI `describe-db-instances`.

Optionally it is also possible to use SSL encryption to connect Amazon RDS MySQL DB instance. The `--ssl-ca` parameter is used to provide a public key (`.pem`) for SSL encrypted communication.

Following two tips are repeating, same as MariaDB.



It is possible to pass a password immediately after `-p` parameter. Best practice is not to provide it with the command but to provide at runtime when prompted. The number of connection limit for a MySQL instance is dependent instance type. DB instances with higher memory have a higher connection limit. MySQL's maximum possible connection number is defined in the `max_connections` parameter.

## Connecting to an Oracle instance

**SQL\*Plus** is an Oracle command-line utility. It can be downloaded from the Oracle website. Before connecting to the Amazon RDS Oracle DB instance, it is essential to find out Amazon RDS endpoint, port, and protocol. When connecting for the first time, we must connect using master user credentials. Once Amazon RDS relevant application and real entity users are created, it can be used for day-to-day maintenance activity. The following `sqlplus` command line example helps us to understand this:

```
sqlplus 'mydbusr@ (DESCRIPTION= (ADDRESS= (PROTOCOL=TCP) (HOST=<dns name of db instance>) (PORT=<listener port>)) (CONNECT_DATA= (SID=<database name>))) '
```

Where:

- **User:** `mydbuser` could be the master user or any other valid user
- **PROTOCOL:** TCP is a protocol and it remains TCP only
- **PORT:** By default, Oracle DB can be connected on 1521
- **SID:** Database name, intended to connect where one instance may have more than one database

## RDS best practices

RDS best practices are as follows:

- Create an individual AWS IAM user to perform DBA tasks. Grant the minimum privileges required to perform day-to-day tasks. Remove unused access key and secret key. Have a strong password policy and rotate the password periodically.
- Before creating an RDS instance identify Amazon RDS essential characteristics to be specified such as VPC, security group, failover or Read Replica requirement, the region and AZs to use, and storage and backup requirements.
- Before creating an RDS instance it is recommended to create a DB options group and DB parameter group.
- Monitor Amazon RDS instance resources such as CPU, memory, and storage to avoid performance bottlenecks.
- It is recommended to keep some extra buffer in memory and a storage volume while choosing RDS instance types.
- It is recommended to test your environment for failover as it may take different time depending on the use case, instance type, and underlined data size.
- Amazon RDS provides an endpoint to connect to the RDS instance. The IP address beneath that endpoint may change after failover takes place. So if an application caches DNS IP address, set the TTL to under 30 seconds in your application environment.



# 11

## AWS DynamoDB - A NoSQL Database Service

DynamoDB is an easy to use and fully managed NoSQL database service provided by Amazon. It provides fast and consistent performance, highly scalable architecture, flexible data structure, event driven programmability, and fine-grained access control.

Before we get into much details about DynamoDB, let us understand some fundamental characteristics of RDBMS/SQL and NoSQL databases. For a long time, the developer community has been working with **Relational Database Management Service (RDBMS)** and **Structured Query Language (SQL)**. If you have used RDBMS and SQL, you will naturally want to compare and understand the fundamental differences as well as similarities between SQL and NoSQL database.

### Let us first understand what an RDBMS is

RDBMS enables you to create databases that can store related data. A database is a collection of information that stores data in database objects, called tables. A table is a collection of related data entries, which consists of columns and rows.

RDBMS enables you to create a link between these tables by establishing a relationship between them. Such a relational model helps in obtaining related information from multiple tables using SQL. You can see in the following *Figure 11.1* that there are three tables, *Employee\_Master*, *Department\_Master*, and *Emp\_Dept*, respectively. All these tables are related with a key field which is called as the primary key. In the following example, you can see how the *Emp\_Dept* table, which provides department detail for employees, is linked with the *Employee\_Master* and *Department\_Master* tables:

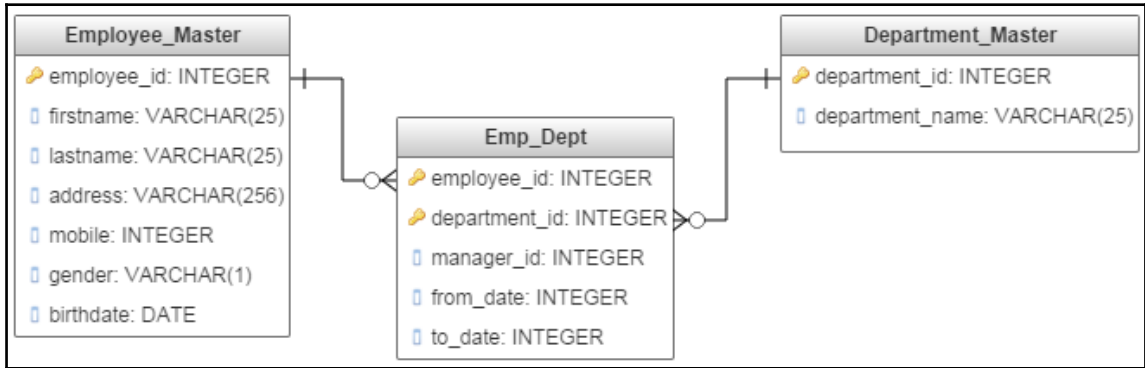


Figure 11.1: Relation between tables in an RDBMS

This is how in a nutshell, RDBMS co-relates with data stored in tables.

## What is SQL?

SQL is a standardized language to interact with relational databases. It can execute queries against a database and retrieve data from one or more tables. It can insert, update, and delete records from database tables and perform many other database-related activities. In short, SQL can help you manage your databases.

Here's a simple example of an SQL statement.

```
Select * from Employee_Master
```

The preceding example simply retrieves all the records from *Employee\_Master* database. Let's understand one more simple example of an SQL statement, which retrieves related information from multiple tables.

```
Select a.employee_id, b.firstname, b.lastname, c.department_name from
Emp_Dept a, Employee_Master b, Department_Master c where a.employee_id =
b.employee_id and a.department_id = c.department_id
```