# AWS Database Services

## Database Primer

* Database systems and engines can be grouped into two broad categories: **Relational Database Management Systems** (RDBMS) and **NoSQL (or non-relational) databases**.

### Relational Databases

* Relational databases provide a common interface that lets users read and write from the database using commands or queries written using Structured Query Language (SQL).
* With relational databases, it is important to note that the structure of the table (such as the number of columns and data type of each column) must be defined prior to data being added to the table.
* A relational database can be categorized as either an **Online Transaction Processing (OLTP)** or **Online Analytical Processing (OLAP)** database system, depending on how the tables are organized and how the application uses the relational database.
* **Amazon Relational Database Service (Amazon RDS)** significantly simplifies the setup and maintenance of OLTP and OLAP databases.
* Amazon RDS provides support for six popular relational database engines: **MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MariaDB, and Amazon Aurora.**

### Data Warehouses

* A data warehouse is a central repository for data that can come from one or more sources.
* This data repository is often a specialized type of relational database that can be used for reporting and analysis via OLAP.
* Organizations typically use data warehouses to compile reports and search the database using highly complex queries.
* Data warehouses are also typically updated on a batch schedule multiple times per day or per hour, compared to an OLTP relational database that can be updated thousands of times per second.
* **Amazon RDS** is often used for OLTP workloads, but it can also be used for OLAP.
* **Amazon Redshift** is a high-performance data warehouse designed specifically for OLAP use cases.

### NoSQL Databases

* NoSQL databases are instead often **key/value stores** or document stores with flexible schemas that **can evolve over time or vary.**
* Today, many application teams use **Hbase, MongoDB, Cassandra, CouchDB, Riak, and Amazon DynamoDB** to store large volumes of data with high transaction rates.
* A common use case for NoSQL is managing user session state, user profiles, shopping cart data, or time-series data.

# Amazon Relational Database Service (Amazon RDS)

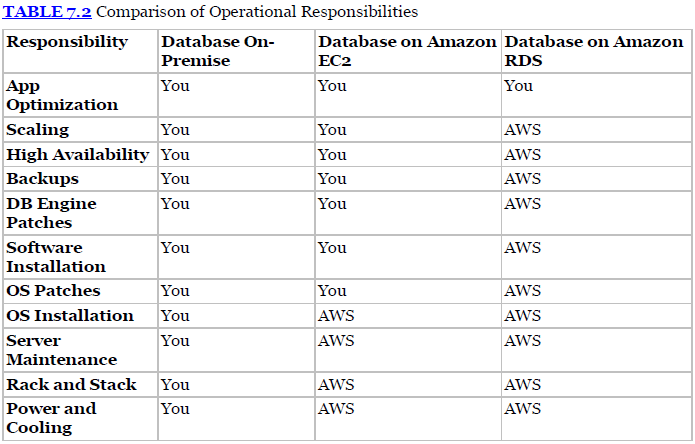
* Amazon RDS is a service that simplifies the setup, operations, and scaling of a relational database on AWS.
* Amazon RDS helps you to **streamline the installation of the database software** and also the provisioning of infrastructure capacity.
* Amazon RDS **exposes a database endpoint** to which client software can connect and execute SQL.
* Amazon RDS **does not provide shell access to Database (DB) Instances**, and it **restricts access to certain system procedures and tables that require advanced privileges**

## Database (DB) Instances

* The Amazon RDS service itself provides an Application Programming Interface (API) that lets you create and manage one or more **DB Instances**.
* A DB Instance is **an isolated database environment deployed in your private network segments in the cloud.**
* A DB Instance **can contain multiple different databases**, all of which you create and manage within the DB Instance itself by executing SQL commands with the Amazon RDS endpoint.
* The range of DB Instance classes extends **from a db.t2.micro with 1 virtual CPU (vCPU) and 1 GB of memory, up to a db.r3.8xlarge with 32 vCPUs and 244 GB of memory.**
* As your needs change over time, you can change the instance class and the balance of compute of memory, and Amazon RDS will migrate your data to a larger or smaller instance class.
* A **DB parameter group** acts as a container for engine configuration values that can be applied to one or more DB Instances.
* A **DB option group** acts as a container for engine features, which is empty by default.
* Existing databases can be migrated to Amazon RDS using native tools and techniques that vary depending on the engine.
* You can also use the **AWS Database Migration Service**, which gives you a graphical interface that simplifies the migration of both schema and data between databases.
* AWS Database Migration Service also **helps convert databases from one database engine to another.**

## Operational Benefits

* With Amazon RDS, you cannot use Secure Shell (SSH) to log in to the host instance and install a custom piece of software.
* You can, however, connect using SQL administrator tools or use DB option groups and DB parameter groups to change the behaviour or feature configuration for a DB Instance.



## Database Engines

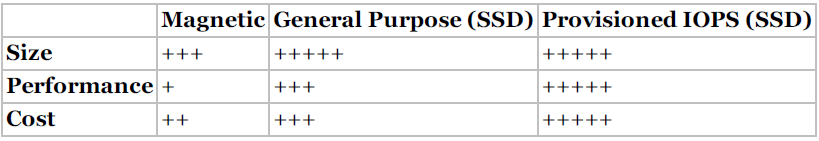
* Amazon RDS supports six database engines: MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Amazon Aurora.

### Amazon Aurora

* Amazon Aurora is a fully managed service, is MySQL compatible out of the box, and provides for increased reliability and performance over standard MySQL deployments.
* Amazon Aurora can deliver up to **five times** the performance of MySQL without requiring changes to most of your existing web applications.
* When you first create an Amazon Aurora instance, you create a **DB cluster**.
* A **DB cluster** has one or more instances and includes a cluster volume that manages the data for those instances.
* An Amazon Aurora cluster volume is a virtual database storage volume that spans multiple Availability Zones, with each Availability Zone having a copy of the cluster data.
* An Amazon Aurora DB cluster consists of two different types of instances:
  + **Primary Instance** This is the main instance, which supports both read and write workloads.
  + When you modify your data, you are modifying the primary instance.
  + Each Amazon Aurora DB cluster has **one primary instance**.
  + **Amazon Aurora Replica** This is a secondary instance that supports only read operations.
  + Each DB cluster can have **up** **to 15 Amazon Aurora Replicas** in addition to the primary instance.
  + By using multiple Amazon Aurora Replicas, you can distribute the read workload among various instances, increasing performance.

## Storage Options

* Amazon RDS is built using Amazon Elastic Block Store (Amazon EBS).
* Amazon RDS supports three storage types: Magnetic, General Purpose (Solid State Drive [SSD]), and Provisioned IOPS (SSD).



* **Magnetic** Magnetic storage, also called standard storage, offers cost-effective storage that is ideal for applications with light I/O requirements.
* **General Purpose (SSD)** General purpose (SSD)-backed storage, also called gp2, can provide faster access than magnetic storage. This storage type can provide burst performance to meet spikes and is excellent for small- to medium-sized databases.
* **Provisioned IOPS (SSD)** Provisioned IOPS (SSD) storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads that are sensitive to storage performance and consistency in random access I/O throughput.

## Backup and Recovery

* Amazon RDS provides two mechanisms for backing up the database: **automated backups and manual snapshots.**
* Each organization typically will define a **Recovery Point Objective (RPO)** **and Recovery Time Objective (RTO)** for important applications based on the criticality of the application and the expectations of the users.
* RPO is defined as the maximum period of data loss that is acceptable in the event of a failure or incident.
* RTO is defined as the maximum amount of downtime that is permitted to recover frombackup and to resume processing.
* In the event of a hardware failure, you can reduce your RTO to minutes by failing over to a secondary node.

### Automated Backups

* An **automated backup** is an Amazon RDS feature that continuously tracks changes and backs up your database.
* Amazon RDS creates a storage volume snapshot of your DB Instance, backing up the entire DB Instance and not just individual databases.
* **One day of backups** will be retained **by default**, but you can modify the retention period up to a **maximum of 35 days**.
* Keep in mind that when you delete a DB Instance, all automated backup snapshots are deleted and cannot be recovered. Manual snapshots, however, are not deleted.
* Automated backups will occur daily during a **configurable 30-minute maintenance window** called the backup window.
* Automated backups are kept for a configurable number of days, called the **backup retention period.**

### Manual DB Snapshots

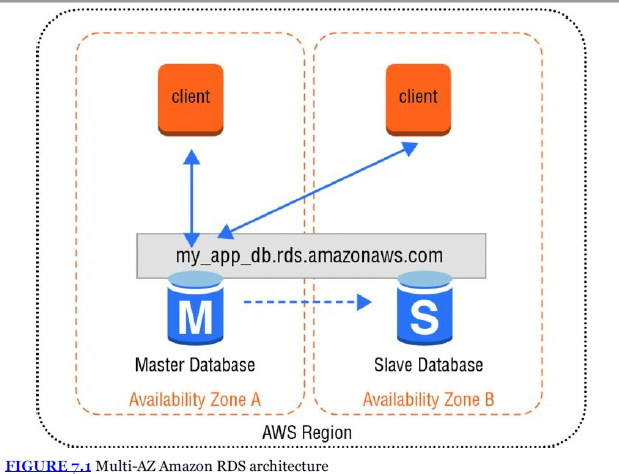
* A DB snapshot is initiated by you and can be created as frequently as you want.
* Unlike automated snapshots that are deleted after the retention period, manual DB snapshots are kept until you explicitly delete them with the Amazon RDS console or the DeleteDBSnapshot action.
* For busy databases, use Multi-AZ to minimize the performance impact of a snapshot. During the backup window, storage I/O may be suspended while your data is being backed up, and you may experience elevated latency. This I/O suspension typically lasts for the duration of the snapshot. This period of I/O suspension is shorter for Multi-AZ DB deployments because the backup is taken from the standby, but latency can occur during the backup process.

### Recovery

* You cannot restore from a DB snapshot to an existing DB Instance; a new DB Instance is created when you restore.
* When you restore a DB Instance, only the default DB parameter and security groups are associated with the restored instance.
* As soon as the restore is complete, you should associate any custom DB parameter or security groups used by the instance from which you restored.

## High Availability with Multi-AZ

* Setting up a relational database to run in a highly available and fault-tolerant fashion is a challenging task. With Amazon RDS Multi-AZ, you can reduce the complexity involved with this common administrative task; with a single option, Amazon RDS can increase the availability of your database using replication.
* Multi-AZ allows you to place a secondary copy of your database in another Availability Zone for disaster recovery purposes.
* When you create a Multi-AZ DB Instance, a primary instance is created in one Availability Zone and a secondary instance is created in another Availability Zone.
* You are assigned a database instance endpoint such as the following: **my\_app\_db.ch6fe7ykq1zd.us-west-2.rds.amazonaws.com** This endpoint is a Domain Name System (DNS) name that AWS takes responsibility for resolving to a specific IP address.

****

* Amazon RDS automatically replicates the data from the master database or primary instance to the slave database or secondary instance using synchronous replication.
* Amazon RDS detects and automatically recovers from the most common failure scenarios for Multi-AZ deployments so that you can resume database operations as quickly as possible without administrative intervention.
* Amazon RDS automatically performs a failover in the event of any of the following:
  + Loss of availability in primary Availability Zone
  + Loss of network connectivity to primary database
  + Compute unit failure on primary database Storage failure on primary database
* The DNS name remains the same, but the Amazon RDS **service changes the CNAME to point to the standby.**
* It is important to remember that **Multi-AZ deployments are for disaster recovery only**; they are not meant to enhance database performance.
* The standby DB Instance is not available to offline queries from the primary master DB Instance.
* To improve database performance using multiple DB Instances, use read replicas or other DB caching technologies such as Amazon ElastiCache.

## Scaling Up and Out

* Amazon RDS allows you to scale compute and storage vertically, and for some DB engines, you can scale horizontally.

### Vertical Scalability

* Amazon RDS makes it easy to scale up or down your database tier to meet the demands of your application. Changes can be scheduled to occur during the next maintenance window or to begin immediately using the **ModifyDBInstance** action.
* To change the amount of compute and memory, you can select a different DB Instance class of the database.
* After you select a larger or smaller DB Instance class, Amazon RDS automates the migration process to a new class with only a short disruption and minimal effort.
* You can also increase the amount of storage, the storage class, and the storage performance for an Amazon RDS Instance.
* Each database instance **can scale from 5GB up to 6TB in provisioned storage** depending on the storage type and engine.
* Storage for Amazon RDS can be increased over time as needs grow with minimal impact to the running database.
* Storage expansion is supported for all of the database engines **except for SQL Server**

### Horizontal Scalability with Partitioning

* Partitioning a large relational database into multiple instances or shards is a common technique for handling more requests beyond the capabilities of a single instance.
* **Partitioning**, or **sharding**, allows you to scale horizontally to handle more users and requests but requires additional logic in the application layer.
* The application needs to decide how to route database requests to the correct shard and becomes limited in the types of queries that can be performed across server boundaries.
* **NoSQL databases like Amazon DynamoDB or Cassandra are designed to scale horizontally.**

### Horizontal Scalability with Read Replicas

* Another important scaling technique is to use read replicas to offload read transactions from the primary database and increase the overall number of transactions.
* There are a variety of use cases where deploying one or more read replica DB Instances is helpful. Some common scenarios include:
  + Scale beyond the capacity of a single DB Instance for read-heavy workloads.
  + Handle read traffic while the source DB Instance is unavailable. For example, due to I/O suspension for backups or scheduled maintenance, you can direct read traffic to a replica.
  + Offload reporting or data warehousing scenarios against a replica instead of the primary DB Instance.
* **Read replicas are currently supported in Amazon RDS for MySQL, PostgreSQL, MariaDB, and Amazon Aurora.**
* Updates made to the source DB Instance are asynchronously copied to the read replica.
* You can reduce the load on your source DB Instance by routing read queries from your applications to the read replica.
* **You can create one or more replicas of a database within a single AWS Region or across multiple AWS Regions.**
* To enhance your disaster recovery capabilities or reduce global latencies, you can use cross-region read replicas to serve read traffic from a region closest to your global users or migrate your databases across AWS Regions.

## Security