Week 4

Elastics Load Balancers, Auto Scaling and AWS database Service

Elastic Load Balancing

- Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones.
- It monitors the health of its registered targets, and routes traffic only to the healthy targets.
- Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
- **Load balancer benefits**
    - ✓ A load balancer distributes workloads across multiple servers. Using a load balancer increases the availability and fault tolerance of your applications.
    - ✓ You can add and remove servers from your load balancer as your needs change, without disrupting the overall flow of requests to your applications.
    - ✓ You can configure health checks, which monitor the health of the compute resources, so that the load balancer sends requests only to the healthy ones.

How Elastic Load Balancing works

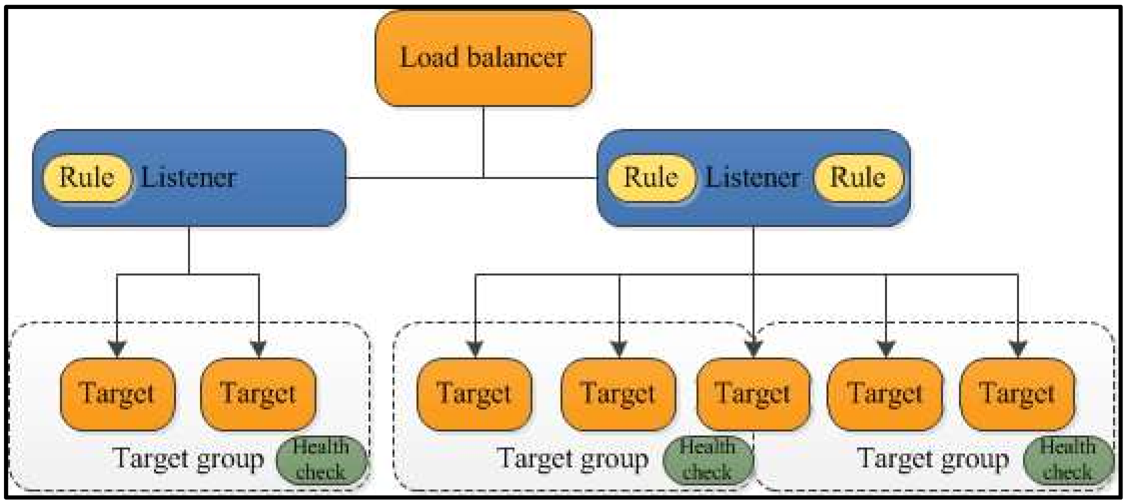
- You configure your load balancer to accept incoming traffic by specifying one or more listeners.
- A listener is a process that checks for connection requests.
- It is configured with a protocol and port number for connections from clients to the load balancer.
- Likewise, it is configured with a protocol and port number for connections from the load balancer to the targets.
- Elastic Load Balancing supports the following types of load balancers:
    - ✓ Application Load Balancers
    - ✓ Network Load Balancers

- ✓ Gateway Load Balancers
- ✓ Classic Load Balancers

Application Load Balancer

- An Application Load Balancer functions at the application layer, the 7th layer of the OSI model.
- Application Load Balancer support the HTTP, and HTTPS protocols.
- A *load balancer* serves as the single point of contact for clients. The load balancer distributes incoming traffic across multiple targets. You add one or more listeners to your load balancer.
- A *listener* checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to a target group.
- A *target group* routes requests to one or more registered targets, such as EC2 instances, using the protocol and the port number that you specify. Network Load Balancer target groups support the TCP, UDP, TCP_UDP, and TLS protocols. You can register a target with multiple target groups. You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group
- You can configure the routing algorithm used at the target group level. The default routing algorithm is round robin; alternatively, you can specify the least outstanding requests routing algorithm.

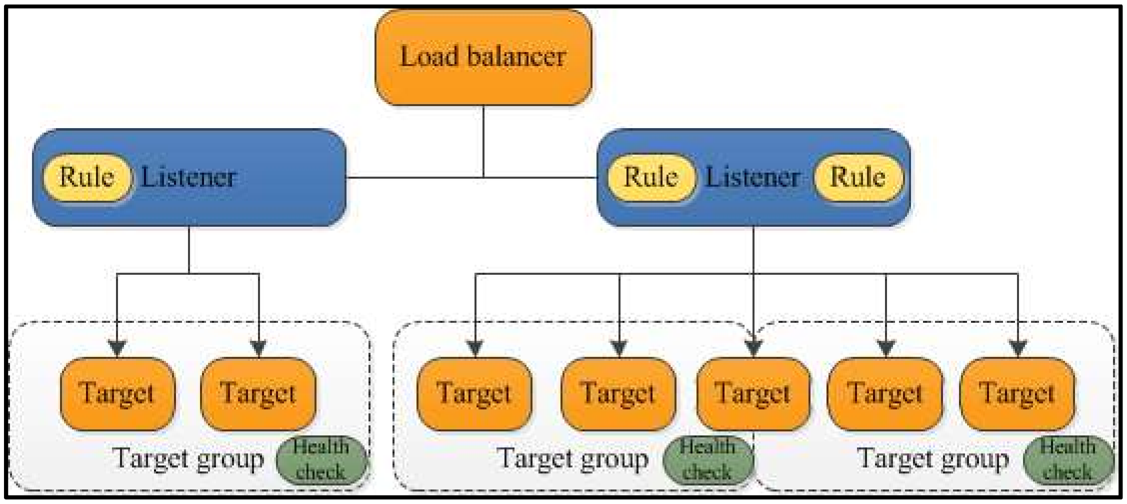
The following diagram illustrates working of ALB. Notice that each listener contains a default rule, and one listener contains another rule that routes requests to a different target group. One target is registered with two target groups.



Network Load Balancer

- A Network Load Balancer functions at the 4th layer of the OSI model.
- Network Load Balancer support the TCP, UDP, TCP_UDP, and TLS protocols.
- A *load balancer* serves as the single point of contact for clients. The load balancer distributes incoming traffic across multiple targets. You add one or more listeners to your load balancer.
- A *listener* checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to a target group.
- A *target group* routes requests to one or more registered targets, such as EC2 instances, using the protocol and the port number that you specify. Network Load Balancer target groups support the TCP, UDP, TCP_UDP, and TLS protocols. You can register a target with multiple target groups. You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group.
- You can configure the routing algorithm used at the target group level. The default routing algorithm is round robin; alternatively, you can specify the least outstanding requests routing algorithm.

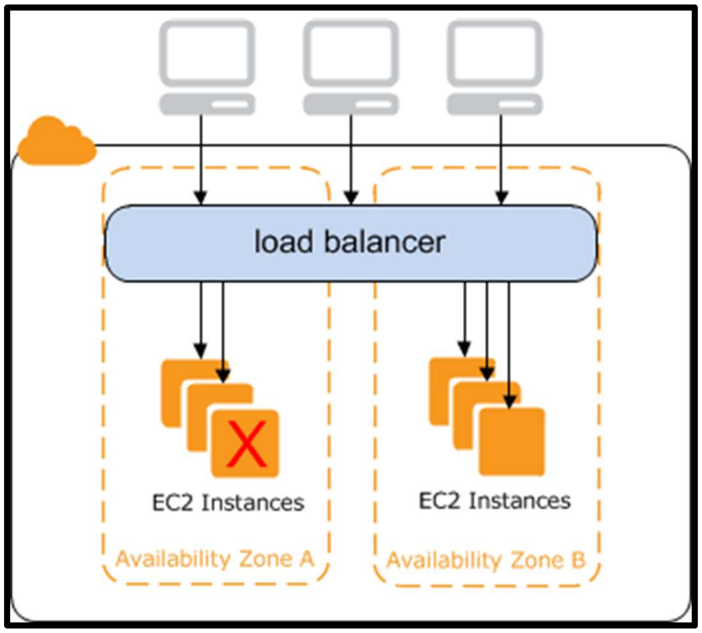
The following diagram illustrates working of NLB. Notice that each listener contains a default rule, and one listener contains another rule that routes requests to a different target group. One target is registered with two target groups.



Classic Load Balancer

- A Classic Load Balancer functions at the both 4th layer and 7th of the OSI model.
- Classic Load Balancer support the HTTP, HTTPS, TCP, UDP, TCP_UDP, and TLS protocols.
- A *load balancer* serves as the single point of contact for clients. The load balancer distributes incoming traffic across multiple targets. You add one or more listeners to your load balancer.
- A *listener* checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to one or more registered instances using the protocol and port number that you configure.
- You can configure *health checks*, which are used to monitor the health of the registered instances so that the load balancer only sends requests to the healthy instances.

The following diagram illustrates working of CLB. Notice that instances/targets are registered to the load balancer.



Gateway Load Balancer

- Gateway Load Balancers enable you to deploy, scale, and manage virtual appliances, such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems.
- A Gateway Load Balancer operates at the 3rd layer of the OSI model, the network layer.
- It listens for all IP packets across all ports and forwards traffic to the target group that's specified in the listener rule.
- You register the virtual appliances with a target group for the Gateway Load Balancer.

Elastic Load Balancers – Sticky Sessions

- By default, an Application Load Balancer routes each request independently to a registered target based on the chosen load-balancing algorithm.
- However, you can use the sticky session feature (also known as session affinity) to enable the load balancer to bind a user's session to a specific target.
- This ensures that all requests from the user during the session are sent to the same target.
- This feature is useful for servers that maintain state information in order to provide a continuous experience to clients.
- To use sticky sessions, the client must support cookies.
- Sticky sessions are enabled at the target group level. You can choose stickiness, and no stickiness across your target groups.
- Sticky sessions are not supported if cross-zone load balancing is disabled.

Elastic Load Balancers – Cross Zone Load Balancing

- The load balancer distributes requests from clients to registered targets.
- When cross-zone load balancing is on, each load balancer node distributes traffic across the registered targets in all registered Availability Zones.
- When cross-zone load balancing is off, each load balancer distributes traffic only across the registered targets in its Availability Zone.
- With Application Load Balancers, cross-zone load balancing is always turned on at the load balancer level, and cannot be turned off.

Elastic Load Balancers – Connection Draining

- Connection draining feature ensures that Load Balancer stops sending requests to instances that are de-registering or unhealthy, while keeping the existing connections open.
- This enables the load balancer to complete in-flight requests made to instances that are de-registering or unhealthy.
- When you enable connection draining, you can specify a maximum time for the load balancer to keep connections alive before reporting the instance as de-registered.
- The maximum timeout value can be set between 1 and 3,600 seconds (the default is 300 seconds).

- When the maximum time limit is reached, the load balancer forcibly closes connections to the de-registering instance.

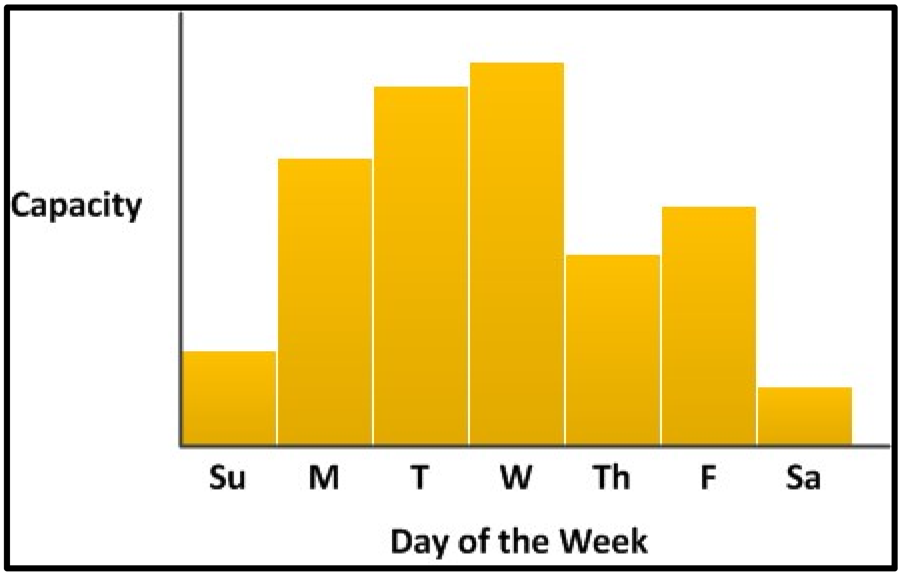
Elastic Load Balancers – SSL Certificates

- If you use HTTPS (SSL or TLS) for your front-end listener, you must deploy an SSL/TLS certificate on your load balancer.
- The load balancer uses the certificate to terminate the connection and then decrypt requests from clients before sending them to the instances.
- The SSL and TLS protocols use an X.509 certificate (SSL/TLS server certificate) to authenticate both the client and the back-end application.
- An X.509 certificate is a digital form of identification issued by a certificate authority (CA).
- You can create a certificate using AWS Certificate Manager or a tool that supports the SSL and TLS protocols.

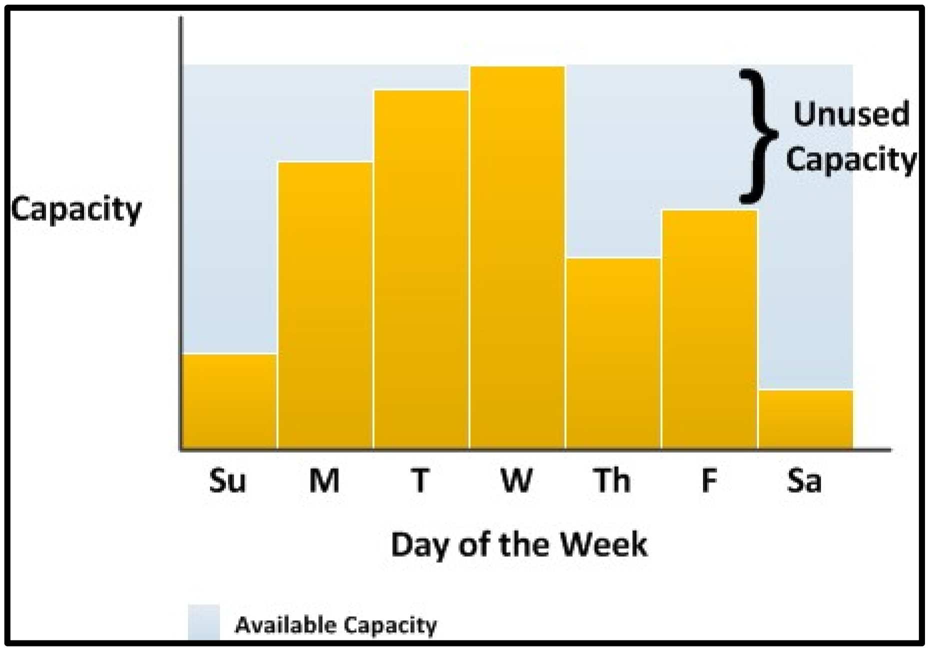
Auto Scaling: Introduction

To demonstrate some of the benefits of Amazon EC2 Auto Scaling, consider a basic web application running on AWS. During the beginning and end of the week, usage of this application is minimal. During the middle of the week, demand on the application increases significantly.

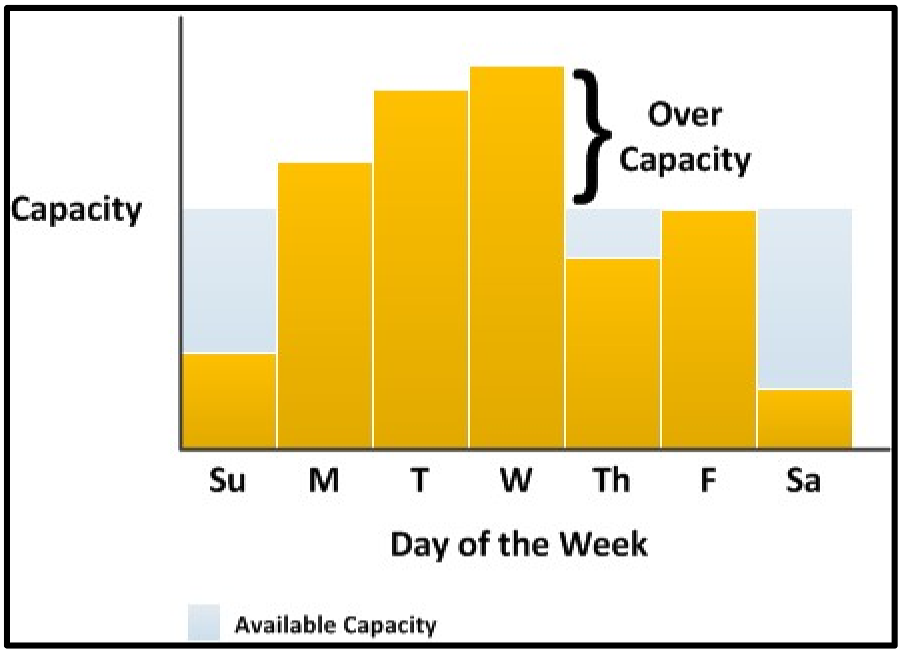
The following graph shows how much of the application's capacity is used over the course of a week.



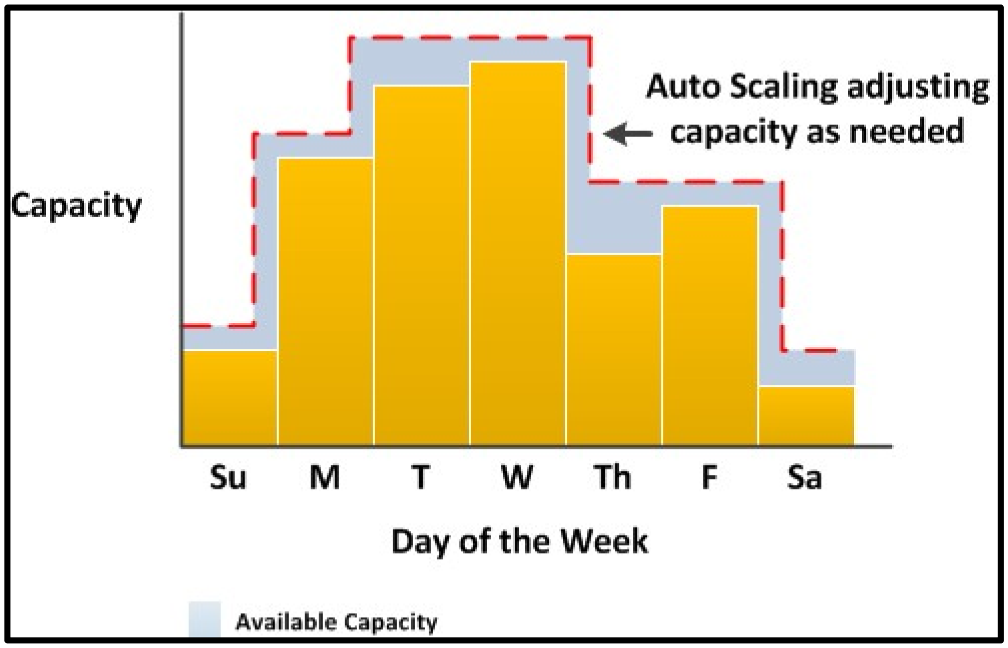
Traditionally, there are two ways to plan for these changes in capacity. The first option is to add enough servers so that the application always has enough capacity to meet demand. The drawback of this option, however, is that there are days in which the application doesn't need this much capacity. The extra capacity remains unused and, in essence, raises the cost of keeping the application running.



The second option is to have enough capacity to handle the average demand on the application. This option is less expensive. However, you risk creating a poor customer experience when the demand on the application exceeds its capacity.



By adding Amazon EC2 Auto Scaling to this application, you have a third option available. You can add new instances to the application only when necessary, and terminate them when they're no longer needed. Because Amazon EC2 Auto Scaling uses EC2 instances, you only have to pay for the instances you use, when you use them. You now have a cost-effective architecture that provides the best customer experience while minimizing expenses.



Auto Scaling

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.

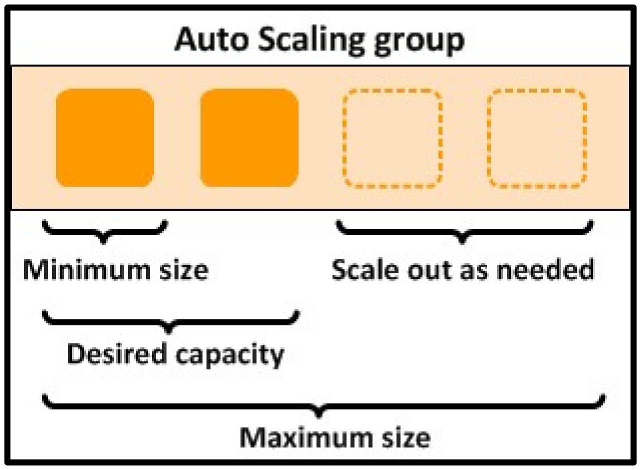
Auto Scaling Group

Auto Scaling groups are collections of Amazon EC2 instances that enable automatic scaling and also help you maintain the health and availability of your applications.

How Auto Scaling Works?

- You create collections of EC2 instances, called *Auto Scaling groups*.
- You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size.
- You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size.
- If you specify the desired capacity, either when you create the group or at any time thereafter, Amazon EC2 Auto Scaling ensures that your group has this many instances.
- If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

For example, the following Auto Scaling group has a minimum size of one instance, a desired capacity of two instances, and a maximum size of four instances. The scaling policies that you define adjust the number of instances, within your minimum and maximum number of instances, based on the criteria that you specify.



Benefits and Features

- **Better fault tolerance:** Amazon EC2 Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it.
- **Better availability:** Amazon EC2 Auto Scaling helps ensure that your application always has the right amount of capacity to handle the current traffic demand.
- **Better cost management:** Amazon EC2 Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the EC2 instances you use, you save money by launching instances when they are needed and terminating them when they aren't.

<u>Auto Scaling Policies</u>

Auto Scaling provides following policies to scale your Auto Scaling group.

- **<u>Maintain current instance levels at all times:</u>** You can configure your Auto Scaling group to maintain a specified number of running instances at all times. To maintain the current instance levels, Amazon EC2 Auto Scaling performs a periodic health check on running instances within an Auto Scaling group. When Amazon EC2 Auto Scaling finds an unhealthy instance, it terminates that instance and launches a new one.

- **<u>Manual scaling:</u>** Manual scaling is the most basic way to scale your resources, where you specify only the change in the maximum, minimum, or desired capacity of your Auto Scaling group. Amazon EC2 Auto Scaling manages the process of creating or terminating instances to maintain the updated capacity.

- **<u>Scheduled Scaling:</u>** Scaling actions are performed automatically as a function of time and date. This is useful when you know exactly when to increase or decrease the number of instances in your group.

- **<u>Dynamic Scaling:</u>** dynamic scaling, lets you define a scaling policy that dynamically resizes your Auto Scaling group to meet changes in demand. For example, let's say that you have a web application that currently runs on two instances and you want the CPU utilization of the Auto Scaling group to stay at around 50 percent when the load on the application changes. This method is useful for scaling in response to changing conditions, when you don't know when those conditions will change.

- **<u>Predictive Scaling:</u>** Use predictive scaling to increase the number of EC2 instances in your Auto Scaling group in advance of daily and weekly patterns in traffic flows.

Databases in Cloud

- Database is a collection of related data.
- Every application needs a place to store data from users, devices, and the application itself.
- Databases are important backend systems that are used to store, manage, update, and analyze data for all types of applications.
- On-premise Database Management involves following:
    - ✓ Make sure that all your servers are equipped with the latest hardware possible
    - ✓ Continuously monitor your hardware for any faults or performance issues
    - ✓ Continuously improve your application code, and think of new features to further expand your business

AWS Database Services

- With AWS Database Services, you just have to focus on your application code and business expansion.
- The rest will be managed and taken care of by AWS, i.e., all the hardware up-gradations, security patches, and even the software up-gradations are taken care of by AWS, that too free of cost!
- With AWS Database Services,
    - ✓ You don't have to maintain a Database Maintenance team,
    - ✓ No need to buy expensive hardware for your servers, and
    - ✓ Most importantly, you can just focus on your business problems and leave everything for AWS to handle.
- Types of AWS database services:
    - ✓ Relational Databases
    - ✓ Key-Value Databases
    - ✓ In-Memory Databases
- **Relational Databases:** In relational databases, the data is usually stored in a tabular format. Relational databases particularly use structured query language (SQL) to run queries to perform operations such as insertion, updating, deletion, and more. AWS provides following relational database services.
    - ✓ Amazon RDS
    - ✓ Amazon Redshift

    - ✓ Amazon Aurora
- **Key–Value Databases:** The key–value database is a type of NoSQL database where the method of having a value attached to a key is used to store data. Meaning that the data is composed of two elements, keys and values. AWS provides following key-value database services.
    - ✓ Amazon DynamoDB
- **In-memory Databases:** This type of database is primarily based on the main memory for computer data storage. Basically, an in-memory database keeps the whole data in the RAM. Meaning that each time you access the data, you only access the main memory and not any disk.
    - ✓ Amazon ElastiCache

AWS RDS (Relational Database Service)

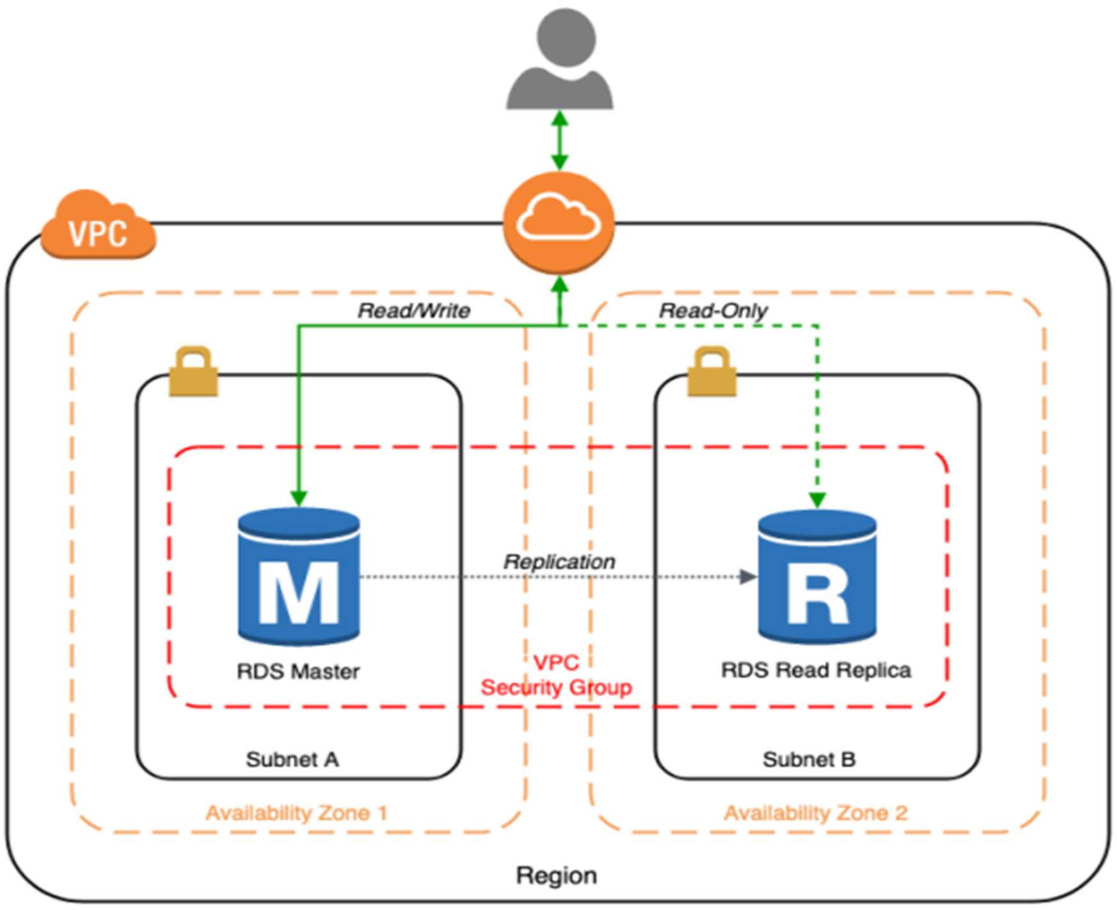
- Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.
- It provides cost-**efficient, resizable capacity for an relational database and manages common database administration tasks.**
- RDS is used to set up, operate, and scale a relational database in the cloud.
- It automates administrative tasks such as hardware provisioning, database setup, backups, and more.
- Amazon RDS currently supports MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, and Amazon Aurora database engines.
- DB Instance is the basic building block of Amazon RDS.
- A *DB instance* is an isolated database server running in the cloud.
- A DB instance can contain multiple user-created databases, and can be accessed using the same client tools and applications.

AWS RDS Read Replica Vs Multi AZ

Amazon RDS Multi-AZ and Read Replicas maintain a copy of database but they are different in nature. Use Multi-AZ deployments for High Availability/Failover and Read Replicas for read scalability.

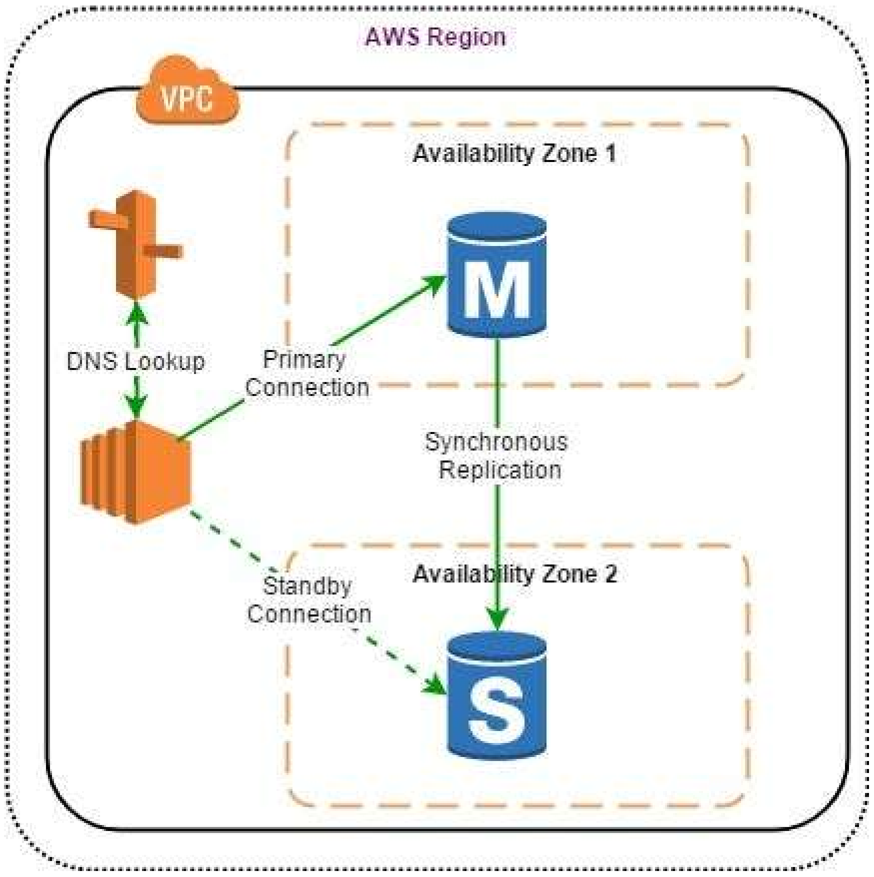
AWS Read Replica

- Read replicas allow you to have a read-only copy of your database. Just we take a snapshot of the database in AWS.
- This is like a read only copy of a database.
- Normally a Read replica is used to offload heavy read traffic for an application.
- So if you have an application that has a heavy query interface and looks for consistent number of reads, then Read replica can be used.
- When you create a Read Replica, you first specify an existing DB instance as the source.
- Then Amazon RDS takes a snapshot of the source instance and creates a read-only instance from the snapshot.
- Read Replica helps in decreasing load on the primary DB by serving read-only traffic.
- A Read Replica can be manually promoted as a standalone database instance.
- You can create Read Replicas within AZ, Cross-AZ or Cross-Region.
- You can have up to five Read Replicas per master, you can connect to each Read Replica and use them for read scaling.
- You can have Read Replicas of Read Replicas.



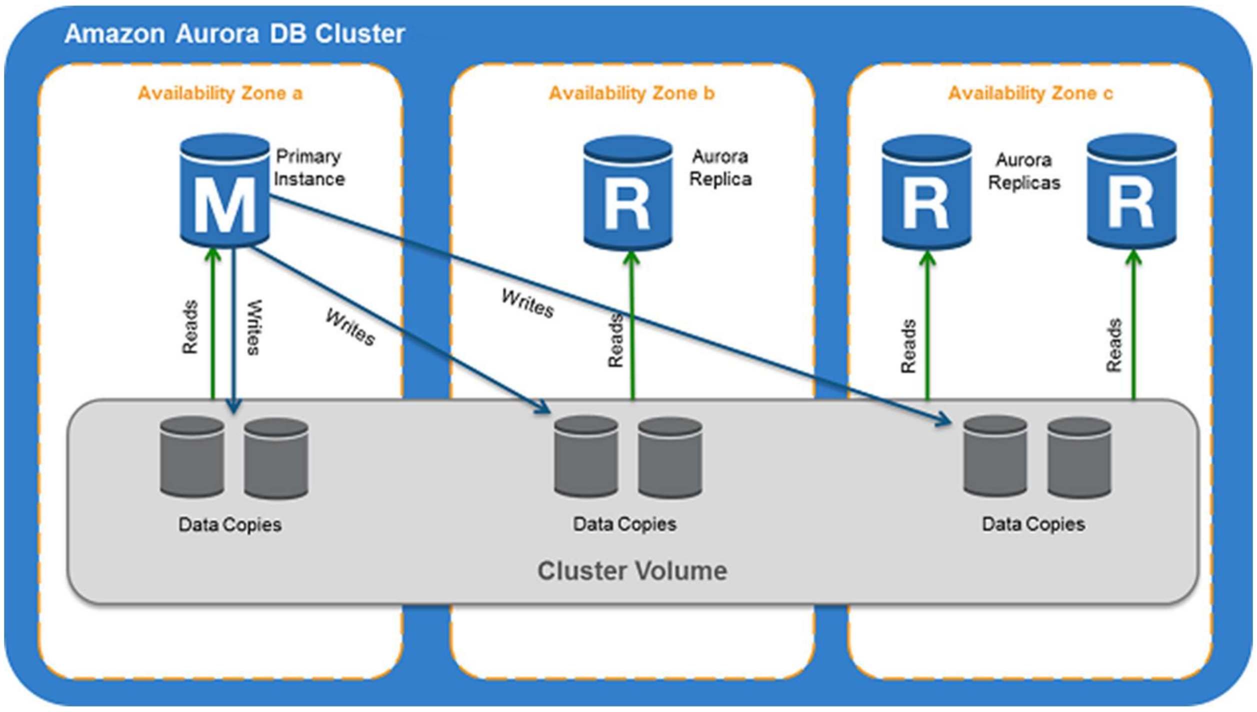
AWS RDS Multi-AZ

- In this architecture, AWS maintains a copy of the primary database in another Availability Zone. So if a failure occurs in the primary database, secondary/standby database takes over as primary database.
- The data is replicated synchronous to the standby instance.
- In a Multi-AZ deployment, it maintains a synchronous standby replica of the master DB in a different Availability Zone.
- When a problem is detected on the primary instance, it will automatically failover to the standby in the following conditions:
    - The primary DB instance fails
    - An Availability Zone outage
    - The DB instance server type is changed
    - The operating system of the DB instance is undergoing software patching.
    - A manual failover of the DB instance was initiated using Reboot with failover



Amazon Aurora

- Aurora is part of the managed database service Amazon Relational Database Service (Amazon RDS).
- Amazon Aurora (Aurora) is a fully managed relational database engine that's compatible with MySQL and PostgreSQL.
- Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL.
- You choose Aurora MySQL or Aurora PostgreSQL as the DB engine option when setting up new database servers through Amazon RDS.
- An Amazon Aurora *DB cluster* consists of one or more DB instances and a cluster volume that manages the data for those DB instances.
- An Aurora *cluster volume* is a virtual database storage volume that spans multiple Availability Zones, with each Availability Zone having a copy of the DB cluster data.
- Two types of DB instances make up an Aurora DB cluster:
    - ✓ **Primary DB instance –** Supports read and write operations, and performs all of the data modifications to the cluster volume. Each Aurora DB cluster has one primary DB instance.
    - ✓ **Aurora Replica –** Connects to the same storage volume as the primary DB instance and supports only read operations. Each Aurora DB cluster can have up to 15 Aurora Replicas in addition to the primary DB instance.



Amazon Elastic Cache

- Amazon ElastiCache is a web service that makes it easy to set up, manage, and scale a distributed in-memory data store or cache environment in the cloud.
- It provides a high-performance, scalable, and cost-effective caching solution.
- Amazon ElastiCache supports the Memcached and Redis cache engines. Each of them is an in-memory key-value store.
- Adavantages of Elastic Cache:
    - ✓ Fully Managed
    - ✓ Improves application performance
    - ✓ Highly avaialable
- Drawbacks
    - ✓ Limited to Aws Service
    - ✓ Less user friendly