ELASTIC COMPUTE CLOUD (EC2)

Amazon EC2 provides scalable computing capacity in the AWS cloud.
You cans use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking and manage storage.
Amazon EC2 enables you to scale up or scale down the instance.
Amazon EC2 is having two storage options i.e EBS and instance store.
Preconfigured templates are available known as amazon machine image.
By default when you create an EC2 account with amazon your account is limited to a maximum of 20 instances per EC2 region with two default high I/O instances.

Types of EC2 instances:

1. General purpose
2. Compute optimized
3. Memory optimized
4. Storage optimized
5. Accelerated computing or GPU
6. Storage optimized
7. High memory optimized

1. **General purpose:**
   General purpose instances provide a balance of compute, memory and networking resources and can be used for a variety of workloads.
   There are 3 series are available in general purpose instance:
   a. A series: A1
   b. M series: M4, M5, M5a, M5d, M5ad (large)
   c. T series: T2 (free tier eligible), T3, T3a
   Instances are available in four sizes: Nano, Small, Medium, Large

   a. A series: A1-instances:
   A1 instances are ideally suited for scale out workloads that are supported by the ARM Ecosystem.
   ARM ecosystem of software provides customers a wide range of products to get to market faster than the competition. ARM development boards are the ideal platform for accelerating development and reducing the risk of new Soc design. Micro service is a distinct method to developing software systems that tries to focus on building single function modules with well-defined interfaces and operations. Cache is a high speed data storage layer which stores a subset of data typically transient in nature, so that future request fir that data are served up faster.
   These instances are well suited for the following applications:
   1. Web server
   2. Containerized micro services

3. Caching fleets
4. Distributed data stores
5. Application that requires ARM instruction set

b. **M series: M4, M5, M5a, M5d, M5ad**
M4 instance:
The new M4 instances features a custom Intel Xeon E5-2676 v3 Haswell processor optimized specifically for EC2.
vCPU- 2 to 40 (max)
RAM- 8GB to 160GB(max)
Instance storage: EBS only (root volume storage)

M5, M5a, M5d and M5ad instances:
These instances provide an ideal cloud infra, offering a balance of compute, memory and networking resources for a broad range of applications.

Used in : gaming server, webserver, small and medium database
vCPU- 2 to 96(max)
RAM- 8 to 384(max)
Instance storage- EBS and NVMe SSD

c. **T series: T2, T3, T3a instances:**
These instances provide a baseline level of CPU performance with the ability to burst to a higher level when required by your workload. An unlimited instances can sustain high CPU performance for any period of time whenever required.

vCPU- 2 to 8
RAM- 0.5 to 32 GB

Used for:
i. Website and web app
ii. Code repositories
iii. Development, build, test
iv. Micro services

2. **Compute optimized:**
Compute optimized are ideal for compute bound applications that benefits from high performance processors.

C Series:
Three types are available: C4, C5, C5n [C3- previous instance]

C4:
C4 instances are optimized for compute intensive workloads and deliver very cost effective high performance at a low price per complete ratio.

vCPU- 2 to 36
RAM- 3.75 to
60GB
Storage- EBS only
Network BW- 10 Gbps
Usecase: web server, batch processing, MMO gaming, Video encoding

Note:

C5 support max 25 EBS volumes
C5 use Elastic Network Adaptor
C5 uses new EC2 Hypervisor

3. **Memory Optimized:**
Memory optimized instances are designed to deliver fast performance for workloads that large data sets in memory.

There are 3 series are available:
R series, X series, Z series

A. R Series:
R4, R5, R5a, R5ad, R5ad
High performance, relational (MySQL) and NoSQL (MongoDB, Casssandra) databases.
Distributed web scale cache stores that provide in memory caching of key volume type data.
Used in financial services,
Hadoop vCPU- 2 to 96
RAM- 16 768GB
Instance storage- EBS only and NVMe SSD

B. X Series:
X1, X1e instances:
Well suited for high performance database, memory intensive enterprise application, relational database workload, SAP HANA.
Electronic design automation
vCPU- 4 to 128
RAM- 122 to 3904GB
Instance storage- SSD

C. Z1d instance:

High frequency Z1d delivers a sustained all core frequency of up to 4.0 GHz, the fastest of any cloud instances.
AWS Nitro System, Xeon processor, up to 1.8 TB of instances storage.
vCPU- 2 to 48
RAM- 16 to 384 GB
Storage- NVM SSD

Use case: electronic design automation and certain database workloads with high per-core licensing cost.

4. **Storage optimized:**

Storage optimized instances are designed for workloads that require high, sequential Read and Write access to very large data sets on local storage. They are optimized to deliver tens of thousands of low latency, random I/O operations per second (IOPS) to application.

It is of three types:

A. D series- D2 instance
B. H series- H1 instance
C. I series- I3 and I3en instance

A. D2 instance:

Massive parallel processing (MPP) data warehouse.
Map reduce and Hadoop distributed computing.
Log or data processing
app vCPU- 4 to 36
RAM- 30.5 to 244GB
Storage- SSD

B. H series- H1 instance:

This family features up to 16GB of HDD based local storage, high disk throughput and balance of compute and memory.
Well suited for app requiring sequential access to large amounts of data on direct attached instance storage.
Application that requires high throughput access to large quantities of data.
vCPU- 8 to 64
RAM- 32 to 256GB
Storage- HDD

C. **I3 and I3en instances:**

High frequency online transaction processing system (OLTP)
Relational databases:
NoSQL database
Distributed file
system
Data warehousing application

vCPU- 2 to 96
RAM- 16 to
768GB
Local storage- NVMe SSD
Networking performance- 25 Gbps to 100 Gbps
Sequential throughput:
Read- 16GBps
Write- 6.4 GBps (I3)
8GBps (I3en)

5. **Accelerated Computing Instances**

Accelerated computing instance families use hardware accelerators or co-processors to perform some functions such as floating point number calculations, graphics processing or data pattern matching more efficiently than is possible in software running on CPUs.

It is of 3 types:

A. F series- F1 instance
B. P series- P2 and P3 instance
C. G series- G2 and G3 instance

a. **F1 instance**:

F1 instances offers customizable hardware acceleration with field programmable gate arrays.(FPGA)
Each FPGA contains 2.5 million logic elements and 6800 DSP (Digital Processing Unit) engines.
Designed to accelerate computationally intensive algorithms such as data flow or highly parallel operations.
F1 provides local NVM SSD storage.

vCPU- 8 to 64
FPGA- 1 to 8
RAM- 122 to 976GB
Storage- NVMe SSd

Used in- genomics research, financial analytics, real time video processing and big data search.

b. P2 and P3 Instance:

It uses NVIDIA Tesla GPUs.
Provide high bandwidth networking.
Up to 32GB of memory per GPUs which makes them ideal for deep learning and computational fluid dynamics.

P2 instance	**P3 instance**
vCPU- 4 to 64	vCPU- 8 to 96
GPU- 1 to 16	GPU- 1 to 8
RAM- 61 to 731GB	RAM- 61 to 768GB
GPU RAM- 12 to 192 GB	storage- SSD and
EBS Network bandwidth- 25 GBps	

Used in- machine learning, databases, seismic analysis, genomics, molecular modeling, AI, deep learning

Note: P3 support CUDA9 and OPENCL APIs,
P2 supports CUDA8 and OPENCL 1.2

c. G2 and G3 instances:

Optimized for graphics intensive application.
Well suited for app like 3D visualization.
G3 instances use NVIDIA Tesla M60 GPU and provide a cost effective, high performance platform for graphics applications.

vCPU- 4 to 64
GPU- 1 to 4
RAM- 30.5 to 488GB
GPU memory- 8 to 32 GB
Network performance- 25GBps

Used in: video creation service, 3D visualization, streaming, graphic intensive application

6. High Memory Instance:

High memory instances are purpose built to run large-in-memory databases, including production developments of SAP HANA in the cloud.
It has only on series i.e U series.

Features:

Latest generation intel Xeon Pentium 8176M processor.
6, 9, 12 TB of instance memory, the largest of any EC2 instance.
Powered by the AWS Nitro System, a combination of dedicated hardware and light weight hypervisor.

Bare metal performance with direct access to host hardware.
EBs optimized by default at no additional cost.
Model number- U-6tb1.metal, U-9tbi.metal, U-12tb1.metal
Network performance- 25 GBps
Dedicated bandwidth- 14GBps
Each instance offer 448 logical processor

Note:

High memory instances are bare metal instances and do not run on a hypervisor.
Only available under dedicated host purchasing category. (for 3 years term)
O.S directly on Hardware.

7. **Previous generation instance:**
   T1, M1, C1, CC2, M2, CR1, CG1, I2, HS1, M3, C3, R3: all instances are available now and we can purchase all of them.

EC2 PURCHASING OPTION:

There are 6 ways of purchasing options available for AWS EC2 instances, but there are 3 ways to pay for Amazon EC2 instance i.e On demand, Reserved instance and Spot instance.

You can also pay for dedicated host which provide you with EC2 instance capacity on physical servers dedicated for your use.

1. On demad
2. Dedicated instance
3. Dedicated Host
4. Spot instance
5. Scheduled instance
6. Reserved instance

1. **On-Demand Instance:**
   AWS on demand instances are virtual servers that run in AWS of AWS relational database service (RDS) and are purchased at a fixed rate per hour.
   AWS recommends using on demand instances for applications with short term irregular workloads that cannot be interrupted.
   They also suitable for use during testing and development of applications on EC2.
   With on demand instances you only pay for EC2 instances you use.

The use of on demand instances frees you from the cost and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into mush smaller variable cost.
Pricing is per instance hour consumed for each instance from the time an instance is launched until if it terminated of stopped.
Each partial instance hour consumed will be billed per second for linux instances and as a full hour for all other instance types.

2. **Dedicated Instance:**
Dedicated instances are run in a VPC on hardware that is dedicated to a single customer.
Your dedicated instances are physically isolated at the host hardware level from instances that belong to other AWS account.
Dedicated instances may share hardware with other instances from the same account that are not dedicated instance.
Pay for dedicated instances on demand save up to 70% by purchasing reserved instance or save up to 90% by purchasing spot instances.

3. **Dedicated Host:**
An Amazon EC2 dedicated host is a physical server with EC2 instance capacity fully dedicated to your use.
Dedicated host can help you address compliance requirement and reduce costs by allowing you to use your existing server bound software licenses.
Pay for a physical host that is fully dedicated to running your instances and bring your existing per socket, per core, per VM software license to reduce cost.
Dedicated host gives you additional visibility and control over how instances are placed in a physical server and you can consistently deploy your instances to the same server over time.
As a result dedicated host enables you to use your existing server bound software license and address corporate compliance and regulatory requirements.
Instances that run on a dedicated host are the same virtualized instances that you had get with traditional EC2 instance3s that use the XEN Hypervisor.
Each dedicated host supports a single instance size and type (for e.g C3.XLARGE)
Only BYOL, Amazon linux and AWS marketplace AMIs can be launched onto dedicated hosts.

4. **Spot Instances:**
Amazon EC2 spot instances let you take advantage of unused EC2 capacity in the AWS cloud. Spot instances are available at up to 90% discount compared to on-demand prices.
You can use spot instances for various test and development workloads.

You can also have the options to hibernate, stop or terminate your spot instances when EC2 reclaims the capacity back with two minutes of notice.
Spot instances are spare EC2 capacity that can save you up 90% off of on-demand prices that AWS can interrupt with a 2 minute notification. Spot uses the same underlying EC2 instances as on-demand and reserved instances, and is best suited for flexible workloads.
You can request spot instances up to your spot limit for each region.
You can determine the status of your spot request via spot request status code and message. You can access spot request status information on the spot instance page of the EC2 console of the AWS management console.
In case of hibernate, your instance gets hibernated and RAM data persisted. In case of stop, your instance gets shutdown and RAM is cleared.
With hibernate, spot instances will pause and resume around any interruptions so your workloads can pick up from exactly where they left off.

Question: when would my spot instance get interrupted?
Ans: primary reason would be Amazon EC2 capacity requirement (e.g: on-demand or reserved instances). Secondarily, if you have chosen to set a 'max spot price' and the spot price raise above.

5. **Scheduled Instance:**

Scheduled reserve instances enable you to purchase capacity reservations that recur on a daily, weekly or monthly basis, with a specified start time and duration for one year term.
You reserve the capacity in advance so that you know it is available when you need it.
You pay for the time that the instances are scheduled even if you do not use them.
Scheduled instances are a good choice for workloads that do not run continuously but do run on a regular schedule.
Purchase instances that are always available on the specified recurring schedule for a one year term.
For example: you can use schedule instances for an application that runs during business hours of for batch processing that run at the end of the week.

6. **Reserved Instances:**

Amazon EC2 RI provides a significant discount up to 75% compared to on-demand pricing and provide a capacity reservation when used in a specific availability zone.
Reserved instances give you the option to reserve a DB instance for a one or three year term and in turn receive a significant discount compared to the on-demand instance pricing for the DB instance.

There are 3 types of RI are available such as

a. Standard RI: these provide the most significant discount up to 75% off on-demand and are best suited for steady-state usage.
b. Convertible RI: these provide a discount up to 54% and the capability to change the attributes of the RI as long as the exchange results in the creation of reserved instances of greater or equal value.
c. Scheduled RI: these are available to lunch within the time window you reserve.

Question: can I transfer a convertible or standard RI from one region to another? How Do I change the configuration of a convertible RI?
Ans: you can change the configuration of your convertible RI using the EC2 management console of the get reserved instance management quota API.

Question: do I need to pay a fee when I exchange my convertible RI?

EC2 ACCESS:

To access instances you need a key and key-pair name.
You can download the private key only once.
The public key is saved by AWS to match it to the key pair name and private key when you try to login to the EC2 instances.
Without key pair you cannot access instances via RDP or SSH (linux).
There are 20 EC2 instances soft limit per account, you can submit a request to AWS to increase it.

EC2 STATUS CHECK:

By default AWS EC2 instance performs automated status checks every one minute.
This is done on every running instance to identify any h/w of s/w issue.
Status check is built into the AWS EC2 instance.
They cannot be configured, deleted or disable.
EC2 services can send its metric data to AWS cloudwatch every 5 minutes (enable by default)
Enable detailed monitoring is chargeable and sends metric in every 1 minute.

You are not charged for EC2 instances if they are stopped however attached EBS
You area not charged for EC2 instances if they are stopped however attached EBS volumes incur charges.

When you stop an EBS Backed EC2 instance:

Instances perform a shutdown.
State changes from running to stopping.

EBS volumes remain attached to the instance.
Any data cached in RAM or instances store volume is gone.
Instances retain its private IPV4 or any IPV6 address.
Instances releases its public IPV4 address back to AWS pool.
Instances retain its elastic IP address.

EC2 TERMINATION:

When you terminate a running instance, the instance states changes as follows:
Running -> shutting down -> terminate
During the shutting down and terminated states you do not incur charges.
By default EBS root devices volumes are deleted automatically when the EC2 instances are terminated.
Any additional (non boot/boot) volumes attached to the instances by default persist after the instance is terminated.
You can modify both behaviors by modifying the "delete on termination" attributes of any EBS volumes during instance launch or while running.
Enable "EC2 termination protection" against accidental termination

EC2 METADATA:

This is the instance data that you can use to configure of manage the instance.
e.g IPV4 addresses IPV6 addresses, DNS hostname, AMI-id, instance id, instance type, local hostname, public keys, and security groups.
Metadata can be only viewed from within the instance itself i.e you have to login to the instance.
Metadata is not protected by encryption; anyone that has access to the instance can view this data.
To view instance metadata: GET http://169.254.169.254/latest/Metadata

INSTANCES USER DATA:

Data supplied by the user at instances launch in the form of a script to be executed during the instance boot.
User data is limited to 16kb.
You can change user data, by stopping EC2 first.
User data is not encrypted EC2 bare metal instances.
Non virtualized environmental.

Operating system runs directly on hardware.
Suitable for licensing restricted tier-1 business critical application.
E.g : I3.metal, I5.metal, R5.metal, Z1d.metal, U-6tb1.metal

ELASTIC BLOCK STORAGE:

Most common replicated with A-Z EBS volumes attached at lunch are deleted when instance terminate.
EBS volumes attached to a running instance are not deleted when instance is terminate but are detached with data interact.

INSTANCE STORAGE:

Physically attach to the host server.
Data not lost when OS is

rebooted. Data lost when:

Underlying drive fails.
Instance is stop or terminated.
You can’t detach or attach to another instance.
Do not rely on for valuable long term data.

VIRTUAL PRIVATE CLOUD (VPC)

A virtual private cloud is a virtual network that closely resembles a traditional networking that you operate in your own data center, with the benefits of using the scalable infrastructure of AWS.
VPC is a virtual network of data center inside AWS for one client.
It is logically isolated from other virtual network in the AWS cloud.
Maximum 5 VPC can be created in one region and 200 subnets in 1 VPC.
We can allocate maximum 5 Elastic IP.
Once we created VPC, DHCP, NACL and security group will automatically created.
A VPC is confined to an AWS region and does not extend between regions.
Once the VPC is created you cannot change its CIDR block range.
If you need a different CIDR size create a new VPC.
The different subnets within a VPC cannot overlap.
You can however expand your VPC CIDR by adding new/extra IP address ranges (except GovCloud and AWS China)

Components of VPC:

A. CIDR and IP address subnets
B. Implied Router and Routing table
C. Internet Gateway
D. Security Group
E. NACL
F. Virtual Private Gateway
G. Peering Connectors
H. Elastic IP

Types of VPC:

VPC is of 2 types:

i. Default VPC
ii. Custom VPC

i. Default VPC:

Created in each AWS region when an AWS account is created.
Has default CIDR, security group, NACL and route table settings.
Has an internet gateway by default.

ii. Custom VPC:

It is a VPC and AWS account owner creates.
AWS user creating the Custom VPC can decide the CIDR.
It has its own default security group, NACL and route tables.

It doesn't have an internet gateway by default, one needs to be created if needed.

Public Subnet: if a subnet's traffic is routed to an internet gateway, the subnet is known as public subnet. If you want your instance in a public subnet to communicate with the internet over IPV4, it must have a public IPV4 address or an Elastic IP address.

Private Subnet: if a subnet doesn't have a route to the internet gateway, the subnet is known as private subnet. When you create a VPC you must specify an IPV4 CIDR block for the VPC. The allowed blocks size is between /16 to /28 networks. The first four and last IP address of subnet cannot be assigned.

The instances in the public subnet can send outbound traffic directly to the internet, but instances in private subnet can't.

For e.g: 10.0.0.0- network address

10.0.0.1- reserved by AWS for the VPC router

10.0.0.2- reserved by AWS the IP address of DNS server

10.0.0.3- reserved for future use

10.0.0.255- broadcast address

Note: AWS do not support broadcast in a VPC but reserve this address.

B. Implied Router and Routing Table:

It is the central routing function.
It connects the different AZ together and connects the VPC to the internet gateway.
You can have up to 200 route tables per VPC.
You can have up to 50 routes entries per route table.
Each subnet must be associated with only one route table at any given time only.
If you don't specify a subnet to route table association, the subnet will be associated with the default VPC route table.
You can also edit the main route table if you need but you cannot delete the main route table.
However you can make a custom route table manually become the main route table then you can delete the former main as it is no longer a main route table.
You can associate multiple subnets with the same route table.

C. Internet Gateway:

An internet gateway is a virtual router that connects a VPC to the internet.
Default VPC is already attached with an internet gateway.

If you create a new VPC then you must attach the internet gateway in order to access the internet.
Ensure that your subnet's route table points to the internet gateway.
It performs NAT between your private and public IPV4 address.
It supports both IPV4 and IPV6.

NAT Gateway:

You can use a network address translation gateway to enable instances in a private subnet to connect to the internet or other AWS services but prevent the internet from initiating a connection with those instances.
You are charged for creating and using a NAT gateway in your account. NAT gateway hourly usage and data processing rates apply. Amazon EC2 charges for data transfer also apply.
To create a NAT gateway, you must specify the public subnet in which the NAT gateway should reside.
You must also specify an elastic IP address to associate with NAT gateway when you create it.
No need to assign public IP address to your private instance.
After you have created a NAT gateway you must update the route table associated with one or more of your private subnets to point internet bound traffic to the NAT gateway. This enables instances in your private subnet to communicate with the internet.
Deleting a NAT gateway, disassociates its elastic IP address, but does not release the address from your account.

D. Security Group:

It is a virtual firewall works at ENI level.
Up to 5 security groups per EC2 instance interface can be applied.
Can only have permit rules, cannot have deny rules.
Stateful, return traffic of allowed inbound traffic is allowed even if there are no rules to allow it.

E. NACL:

It is a function performed on the implied router.
NACL is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets.
Your VPC automatically comes with a modifiable default NACL. By default, it allows all inbound and outbound IPV4 traffic and if applicable IPV6 traffic.
You can create a custom NACL and associate it with a subnet. By default each NACL denies all inbound and outbound traffic until you add rules.
Each subnet in your VPC must be associated with a NACL. If you don't explicitly associate a subnet with a NACL, the Subnet is automatically associated with the default NACL.

You can associate a NACL with multiple subnets; however a subnet can be associated with only one NACL at a time. When you associate a NACL with a subnet the previous association is removed.
A NACL contains a numbered list of rules that we evaluate in order starting with the lowest numbered rule.
The highest number that you can use for a rule is 32766. Recommended that you start by creating rules with rule numbers that a multiple of 100, so that you can insert new rules where you need later.
It functions at the subnet level.
NACL are stateless, outbound traffic for an allowed inbound traffic must be explicitly allowed too.
You can have permit and deny rules in a NACL.

VPC Peering:

A VPC peering connection is a network connection between two VPC that enables you to route traffic between them using private IPV4 addresses or IPV6 addresses. Instances in either VPC can communicate with each other as if they are within the same network.
You can create a VPC peering connection between your own VPC or with a VPC in another AWS account. The VPC can be in different region.

VPC EndPoint: A VPC endpoint enables you to privately connect your VPC to supported AWS services, instances in your VPC do not require public IP address to communicate with resources in the service. Endpoints are virtual devices.

Difference between security group and NACL.

Security Group	**NACL**
Operate at instance level. Supports allow rules only. Stateful, return traffic is automatically allowed. Applies to an instance only.	Operates at the subnet level. It permits allow as well as deny rules. Stateless, return traffic must be explicitly allowed by rules. Applies to all instances in the subnet.

AWS STORAGE

AWS offers a complete range of cloud services to support both application and archival compliance requirements. Select from the objects, files, and block storage services as well as cloud data migration to start designing the foundation of your cloud IT environments.

Types of storage:

AWS offers 5 types of storage services such as:

1. Simple Storage Service (S3)
2. Elastic File System (EFS)
3. Elastic Block Store (EBS)
4. Glacier
5. Snowball

Difference between Object Storage and Block Storage:

Block Storage:

Block storage is suitable for transitional databases, random read/write loads and structured database storage.
Block storage divides the data to be stored in evenly sized blocks called data chunks for instance, a file can be split into evenly sized blocks before it is stored.
Data blocks stored in block storage would not contain metadata. (Data created, data modified, content type etc.)
Block storage only keeps the address (index number) where the data blocks are stored, it does not care what is in that block, just how to retrieve it when required.

Object Storage:

Object storage stores the files as a whole and does not divide them.
In object storage an object is: the file/ data itself, its Meta data, object global unique ID.
The object global unique ID is a unique identifier for the object (can be the object name itself) and it must be unique such that it can be retrieved disregarding where it's physical storage location is.
Object storage cannot be mounted as a drive.
Example of object storage solutions are Dropbox, AWS S3, Facebook.

1. **Simple Storage Service (S3):**

S3 is a storage for the internet. It has a simple web service interface for simple storing and retrieving of any amount of data, anytime from anywhere on the internet.
S3 is object based storage.
You cannot install operating system on S3.
S3 has a distributed data store architecture where objects are redundantly stored in multiple locations. (minimum 3 locations in same region)
Data is stored in bucket.
A bucket is a flat container of objects.
Maximum capacity of a bucket is 5TB.
You can create folders in your bucket (available through console)
You cannot create nested buckets.
Bucket ownership is non transferrable.
S3 bucket is region specific.
You can have up to 100 buckets per account. (may expand on request)

S3 Bucket Naming Rules:

S3 bucket names (keys) are globally unique across all AWS regions.
Bucket names cannot be change after they are created.
If bucket is deleted its name become available again to you or other account to use.
Bucket names must be at least 3 and no more than 63 characters long.
Bucket names are part of URL used to access a bucket.
Bucket name must be a series of one or more labels (xyz bucket)
Bucket names can contain lowercase, numbers and hyphen but cannot use uppercase letters.
Bucket name should not be an IP address.
Each label must start and end with a lowercase letter or a number.
By default buckets and its objects are private, and by default only owner can access the bucket.

S3 Bucket Sub-Resources:

Sub-resources of S3 bucket includes:

Lifecycle: to decide on objects lifecycle management.

Website: to hold configurations related to static website hosted in S3 buckets.

Versioning: keep objects versions as it changes (set updated)

Access Control List: bucket policies

The name is simply two parts: bucket region’s end point / bucket name

Example: for S3 bucket named mybucket in Europe west region is

https://s3-eu-west1.amazonaws.com/mybucket

S3 Objects:

An object size stored in an S3 bucket can be 0 byte to 5TB.
Each object is stored and retrieve by unique key. (ID or name)
An object in AWS S3 is uniquely identified and addressed through:

- service endpoint
- bucket name
- object key (name)
- optionally object version

Object stored in a S3 bucket in a region will never leave that region unless you specifically move them to another region or CRR.
A bucket owner can grant cross account permissions to another AWS account (or users in another account) to upload objects.
You can grant S3 bucket / object permission to:

- Individual users
- AWS account
- Make the resource public
- To all authenticate user

S3 Bucket Versioning:

Bucket versioning is a S3 bucket sub resource used to protect against accidental object/data deletion or overwrites.
Versioning can also be used for data retention and archive.
Once you enable versioning on a bucket it cannot be disabled however it can be suspended.
When enable, bucket versioning will protect existing and new objects and maintains their versions as they are updated.
Updating objects refers to PUT, POST, COPY, DELETE actions on objects.
When versioning is enable and you try to delete an object a delete marker is placed on the object.
You can still view the object and delete the marker.
If you reconsider deleting the objects you can delete the delete marker and the object will be enable again.
You will be charged for all S3 storage cost for all object versions stored.
You can use versioning with S3 lifecycle policies to delete older version or you can move them to a cheaper S3 storage (Glacier.)
Bucket version state:-

- Enabled
- Suspended

- Un-versioned

Versioning applies to all objects in a bucket and not partially applied.
Object existing before enable versioning will have a version ID or NULL.
If you have a bucket that is already versioned then you suspended versioning existing objects and their versions remain as it is.
However they will not be updated/ version further with future updates while the bucket versioning is suspended.
New objects (uploaded after suspension) they will have a version ID “null” if the same key (name) is used to stone another objects it will override the existing one.
An object deletion in a suspended versioning buckets will only delete the objects with ID “null”.

S3 Bucket Versioning-MFA Delete:

Multifactor authentication delete is a versioning capacity that adds another level of security in case your account is compromised.
This adds another layer of security for the following:

- Changing your bucket’s versioning state.
- Permanently deleting on objects version.

MFA delete requires:

- Your security credentials.
- The code displayed on an approved physical or s/w based authentication device.

S3 Multipart Upload:

It is used to upload an object in parts.
Parts are uploaded independently and in parallel in any order.
It is recommended for objects sizes of 100MB or larger.
You must use it for objects larger than 5GB.
This is done though S3 multipart upload API.

Copying S3 Objects:

The copy operation creates a copy of an objects that is already stored in Amazon S3.
You can create a copy of your object up to 5GB in size a single atomic operation.
However to copy an object greater then 5GB you must use the multipart upload API.
Incur charges if copy to another region.

Use the copy operation to:

- Generate additional copies of the subjects.
- Renaming object (copy to a new name)
- Changing the copy’s storage class or encrypt it at rest.
- Move object across AWS location/region.

- Change object metadata.

STORAGE CLASSES OF AMAZON S3:

There are 6 types of storage classes of Amazon S3 is available such as:

1. Amazon S3 Standard
2. Amazon S3 Glacier Deep Archive
3. Amazon Glacier
4. Amazon S3 Standard Infrequent Access
5. Amazon S3 one-zone-IA
6. Amazon S3 Intelligent Tiering

1. Amazon S3 Standard:

S3 standard offers high durability, availability and performance object storage for frequently accessed data.
Durability is 99.999999999%.
Designed for 99.99% availability over a given year.
Supports SSL for data in transit and encryption of data at rest.
The storage cost for the object is fairly high but there is very less charge for accessing the objects.
Largest object that can be uploaded in a single PUT in 5GB.

2. Amazon S3 IA (standard):

S3-IA is for data that is accessed less frequently but requires rapid access when needed.
The storage cost is much cheaper than S3-standard almost half the price, but you are charged more heavily for accessing your objects.
Durability is 99.999999999%.
Resilient against events that impact an entire AZ.
Availability is 99.9% in a year.
Supports SSL for data in transit and encryption of data at rest.
Data that is deleted from S3-IA within 30 days will be charged for a full 30 days.
Backed with the Amazon S3 service level agreement for availability.

3. Amazon S3 Intelligent Tiering:

The S3 intelligent tiering storage class is designed to optimize cost by automatically moving data to the most cost effective access tier.

It works by storing objects in two access tiers.
If an object in the frequent access tier is accessed it is automatically moved back to the frequent access tier.
There is no retrieval fees when using the S3 intelligent tiering storage class and no additional tering fees when objects are moved between access tiers.
Same low latency and high performance of S3 standard.
Objects less than 128kb cannot move to IA.
Durability is99.999999999%.
Availability is 99.9%.

4. **Amazon One-Zone IA**

S3 one zone IA is for data that is accessed less frequently but requires rapid access when needed.
Data store is single AZ.
Ideal for those who want lower cost option of IA data.
It is good choice for storing secondary backup copies of on-premise data of easily re-creatable data.
You can use S3 lifecycle policies.
Durability is 99.999999999%.
Availability is 99.5%.
Because S3 one zone IA stores data in a single AZ, data stored in this storage class will be lost in the event of AZ destruction.

5. **Amazon S3 Glacier:**

S3 glacier is a secure, durable, low cost storage class for data archiving.
To keep cost low yet suitable for varying needs S3 glacier provides three retrieval options that ranges from a few minutes to hours.
You can upload object directly to glacier or use lifecycle policies.
Durability is 99.999999999%.
Data is resilient in the event of one entire AZ destruction.
Supports SSL for data in transit and encryption data at rest.
You can retrieve 10GB of your amazon S3 glacier data per month for free with free tier account.

6. **Amazon S3 Glacier Deep Archive:**

S3 glacier deep archive is amazon S3 cheapest storage.
Design to retain data for long period even if for 10 years.
All objects stored in S3 glacier deep archive are replicated and stored across at least at three geographically AZ.

Durability is 99.999999999%.
Ideal alternative to magnetic tape libraries.
Retrieval time within 12 hours.
Storage cost is up to 75% less than for the existing S3 glacier storage class.
Availability is 99.9%.

ELASTIC BLOCK STORE (EBS)

There are two types of block store devices are available for EC2.

1. Elastic Block Store (persistent, network attached virtual drive)
2. Instances Store Backed EC2:
    - Basically the virtual hard drive on the host allocated to this EC2 instance.
    - Limit to 10GB per device
    - Ephemeral storage (non-persistent storage)
    - The EC2 instance can't be stopped, can only be rebooted or terminated. Terminate will delete data.

EBS volume behaves like RAW, unformatted, external block storage devices that you can attached to your EC2 instance.
EBS volumes are block storage devices suitable for database style data that requires frequent reads and writes.
EBS volumes are attached to your EC2 instances through the AWS network, like virtual hard drive.
An EBS volume can attach to a single EC2 instances only at a time.
Both EBS volumes and EC2 instances must be in the same AZ.
An EBS volume data is replicated by AWS across multiple servers in the same AZ to prevent data loss resulting from any single AWS component failure.

EBS Volume Types:

1. SSD backed volume
2. HDD backed volume
3. Magnetic standard

SSD backed volume is also two types:

A. General purpose SSD (GP2)
B. Provisioned IOPS SSD (io1)

HDD backed volume is also two types:

A. Throughput optimized HDD (st1)
B. Cold HDD (SC1)

A. **General Purposed SSD (gp2)**

GP2 is the default EBS volume type for the amazon EC2 instance.
GP2 volumes are backed by SSDs.
General purpose balances both price and performances.
Ratio of 3IOPS/GB with up to 10,000 IOPS.
Boot volume having low latency.
Volume size: 1 GB to 16 GB.
Price: $0.10/ GB/month

B. **Provisioned IOPS SSD (io1)**

These volumes are ideal for both IOPS intensive and throughput intensive workloads that requires extremely low latency or for mission critical applications.
Designed for I/O intensive applications such as large relational or NoSQL databases.
Use if you need more than 10,000 IOPS.
Can provision up to 32,000 IOPS per volume.
Volume size: 4GB to 16TB
Price : $ 0.125/GB/month

C. **Throughput optimized HDD (st1)**

ST1 is backed by hard disk drives and is ideal for frequently accessed, throughput intensive workloads with large datasets.
ST1 volumes deliver performance in term of throughput, measured in MB/S.
Big data, data warehouse, log processing.
It cannot be a boot volume.
Can provisioned up to 500 IOPS per volume.
Volume size: 500GB to 16 TB
Price: $0.045/GB/month

D. **Cold HDD (SC1)**

SC1 is also backed by HDD and provides the lowest cost per GB of all EBS volume types.
Lowest cost storage for infrequent access workloads.
Used in file servers.
Cannot be a boot volume.
Can provisioned up to 250 IOPS per volume.
Volume size: 500 GB to 16TB

Price: $0.025/GB/Month

Magnetic Standard:

Lowest cost per GB of all EBS volume type that is bootable.
Magnetic volumes are ideal for workloads where data is accessed infrequently and applications where the lowest storage cost is important.
Price: $0.05/GB/month
Volume size: 1GB to 1TB
Max IOPS/volume: 40-200

EBS Snapshot of Root Volume and Non-root Volume:

EBS snapshots are point-in-time images/copies of your EBS volume.
Any data written to the volume after the snapshot process is initiated, will not be included in the resulting snapshot (but will be included in future incremental update.)
Per AWS account up to 5000 EBS volumes can be created.
Per account up to 10,000 EBS snapshots can be created.
EBS snapshots are stored on S3, however you cannot access them directly. You can only access them through EC2 APIs.
While EBS volumes are AZ specific, snapshots are region specific.
Any AZ in region can use snapshot to create EBS volume.
To migrate an EBS from one AZ to another, create a snapshot (region specific) and create an EBS volume from the Snapshot in the intended AZ.
You can create a snapshot to an EBS volume of the same or larger size than the original volumes size from which the snapshot was initially created.

You can take a snapshot of a non-root EBS volume while the volumes is in use on a running EC2 instance.
This means, you can still access it while the snapshot is being processed.
However the snapshot will only include data that is already written to your volume.
The snapshot is created immediately but it may stay in pending status until the full snapshot is completed. This may takes few hours to complete specially for the first time snapshot if a volume.
During the period when the snapshot status is pending you can still access the volume (non-root) but I/O might be slower because of the snapshot activity.
While in pending state, an in progress snapshot will not include data from ongoing reads and writes to the volume.
To take complete snapshot of your non-root EBS volume: stop of unmounts the volume.
To create a snapshot for a root EBS volume you must stop the instance first then take the snapshot.

Incremental Snapshot:

EBS snapshots are stored incrementally.
For low cost storage on S3 and a guarantee to be able to able fully restore data from the snapshot.
What you need is a single snapshot then further snapshot will only carry the changed blocks (incremental updates).
Therefore you do not need to have multiple full/complete copies of the snapshot.
You are charged for:

- Data transferred to S3 from your EBS volume you are taking snapshot.
- Snapshot stored in S3.
- First snapshot is a clone, subsequent snapshots are incremental.
- Deleting snapshot will only remove data exclusive to that snapshot.

EBS Encryption:

EBS encryption is supported on all EBS volume types and all EC2 instance families.
Snapshots of encrypted volumes are also encrypted.
Creating an EBS volume from an encrypted snapshot will result in an encrypted volume.
Data encryption at rest means encrypting data while it is stored on the data storage device.
There are many ways you can encrypt data on an EBS volume at rest, while the volume is attached to an EC2 instance:

- Use 3rd party EBS volume
- Encryption tods.
- Use encrypted EBS volumes.
- Use encrypted at the O.S level.

Encrypt data at the application level before storing it to the volume.
Use encrypt file system on the top of the EBS volume.
Encrypt volume area accessed exactly like unencrypted ones, basically encryption is handled transparently.
You can attach an encrypted and unencrypted volumes to the same EC2 instance.
Remember that the EBS volumes area not physically attached to the EC2 instance, rather they are virtually attached through the EBS infrastructure.
This means when you encrypt data on an EBS volume data is actually encrypted on the EC2 instance then transferred, encrypted to be stored on the EBS volume.
This means data in transit between EC2 and encrypted EBS volume is also encrypted.
There is no direct way to change the encryption state of the volume.
To change the state you need to follow either of the following two ways:

- Attach a new encrypted EBS volume to the EC2 instance that has the data to be encrypted.
- Mount the new volume to the EC2 instance.
- Copy the data for the un-encrypted volume to the new volume.

- Both volumes must be on the same EC2 instance.

Or

- Create a snapshot of the unencrypted volume.
- Copy the snapshot and choose encryption for the new copy, this will create an encrypted copy of the snapshot.
- Use this new copy to create an EBS volume which will be encrypt too.
- Attach the new encrypted EBS volume to the EC2 instance.

Root EBS Volume Encryption:

There is no direct way to change the encryption state of a volume.
There is an indirect work around to this:

- Launch the instance with the EBS volume required.
- Do whatever patching of install applications.
- Create an AMI from the EC2 instance.
- Copy the AMI and choose encryption while copying.
- This results it an encrypted AMI that is private (yours only).
- Use the encrypted AMI to launch new EC2 instances which will have their EBS root volume3 encrypted.

EBS Encryption Key:

To encrypt a volume or snapshot, you need an encryption key, these keys are called customer master key (CMK) and are managed by AWS key management service (KMS).
When encrypting the first EBS volume, AWS KMS creates a default CMK key.
This key is used for your first volume encryption of snapshots created from this volumes and subsequent volumes created from these snapshots.
After that each newly encrypted volume is encrypted with a unique/ separate AES-256 bit encryption key. This key is used to encrypt the volume, its snapshot and any volumes created of its snapshots.

Changing Encryption Key:

You cannot change the encryption (CMK) key used to encrypt an existing encrypted snapshot or encrypted EBS volume.
If you want to change the key, create a copy of the snapshot and specify during the copy process that you want to re-encrypt the copy with a different key.

This comes in handy when you have a snapshot that was encrypted using your default CMK key and you want to change the key in order to able to share the snapshot with other accounts.

Sharing EBS Snapshot:

By default only the account owner can create volumes from the account snapshots.
You can share your unencrypted snapshots with the AWS community by making them public.
Also you can share your unencrypted snapshots with a selected AWS account by making them private then selecting the AWS accounts to share with.
You cannot make your encrypted snapshots public.
You cannot make a snapshot of an encrypted EBS volume public on AWS.
You can share your encrypted snapshot with specific AWS account as follows:

- Make sure that you use a non-default/custom CMK key to encrypt the snapshot, not the default CMK key (AWS will not allow the sharing if default CMK is used.)
- Configure cross account permissions in order to give the account with which you want to share the snapshot access to the custom CMK key used to encrypt the snapshot.
- Without this the other account will not be able to copy the snapshots nor will be able to create volumes of the snapshots.

AWS will not allow you to share snapshots encrypted using your default CMK key.
For the AWS account with whom an encrypted snapshot is shared.

- They must first create their own copies of the snapshot.
- Then they use that copy to restore/create EBS volume.

You can make a copy of the snapshot when it has been fully saved to S3 (its status show as complete) and not during the snapshot's pending status (when data blocks are being moved to S3).
Amazon S3 server side encryption (SSE) protect the snapshot data-in-transit while copying.
You can have up to 5 snapshots copy request running in a single destination per account.

AWS AUTO SCALING

Creating group of EC2 instances that scale up or down depending on the conditions you set.

Enable elasticity by scaling horizontally through adding or terminating EC2 instances.
Auto scaling ensures that you have the right number of AWS EC2 instances for your needs at all time.
Auto scaling helps you to save cost by cutting down the number of EC2 instances when not needed and scaling out to add more instances only it is required.

Auto Scaling Components:

1. Launch Configuration : like instance type, AMI, key-pair, security group
2. Auto Scaling Group: group name, group size, VPC, subnet, health check period
3. Scaling Policy : metric type, target value

How to Balance, Attach and Detach EC2 Instances:

Balance:

If auto scaling finds that the number of EC2 instances launched by ASG into subjects AZs is not balanced (EC2 instances are not evenly distributed), auto scaling do rebalancing activity by itself.
AS always tries to balance the instances distribution across AZs.
While rebalancing, ASG launches new EC2 instances where there are less EC2 at present and then terminates the instances from the AZs that had more instances.

What causes imbalance of EC2?

If we add or remove same subnets/AZ form auto scaling.
If we manually request for EC2 termination from our ASG.
An AZ that did not have enough EC2 capacity now has enough capacity and it is one of the auto scaling group.

Attach:

We can attach a running EC2 instance to an ASG by using AWS console or CLI if the below conditions are meet:

- Instances must be on running state.
- AMI used to launch the EC2 still exists.
- Instances is not the part of another auto scaling group.
- Instances must be in the same AZ of the same group.

- If the existing EC2 instances under the ASG, plus the one to be needed, exceeds the maximum capacity of the ASG, the request will fail, EC2 instance would not be added.

Detach:

You can manually remove EC2 instances from an ASG using AWS console of CLI.
You can then manage the detached instances independently or attach it to another ASG.
When you detach an instance you have the option to decrement the ASG desired capacity.
If you do not, ASG will launch another instance to replace the one detached.
When you delete an ASG its parameter like maximum, minimum and desired capacity are all set to zero. Hence it terminates it's all EC2 instances.
If you want to keep the EC2 instances and manage them independently you can manually detach them first then delete ASG.

We can attach one more Elastic Load Balancer (ELB) to our ASG.
The ELB must be in the same region as the ASG.
Once you do this any EC2 instance existing or added by ASG will be automatically registered with the ASG defined ELB.
Instances and the ELB must be in the same VPC.
Auto scaling classifies its EC2 instance health check or unhealthy.
By default, as uses EC2 status checks only to determine the health status of an instance.
When you have one or more ELB defined with the ASG you can configure auto scaling to use both the EC2 status check and SLB health check to determine the instances health check.
Health check grace period is 300sec by default.
If we set zero in grace period the instance health is checked once it is in service.
Until the grace period timer expires any unhealthy status reported by EC22 status check of the ELB attached to the ASG will not be acted upon.
After grace period expires ASG consider an instance unhealthy in any of the following cases:

- EC2 status check report to ASG an instance other than running.
- If ELB health check are configured to be used by the auto scaling then if the ELB report the instance as 'out of service'.

Unlike AZ rebalancing, termination of unhealthy instances happen first then auto scaling attempt to launch new instance to replace the ones terminated.
Elastic IP and EBS volumes gets detached from the terminated instances you need to manually attach there to the new instance.

Types of Auto Scaling Policies:

In four situations, ASG sends a SNS email notification:

i. an instance is launched
ii. an instance is terminated
iii. an instance fails to launch
iv. an instance fails to terminate

Merging ASG:

Can only be done form the CLI not form the AWS console.
You can merge multiple single AZ ASG into a single, one multi-AZ ASG.
Scale out means launching more EC2 instances.
Scale in means terminating one or more EC2 instances by scaling policy.
It is always recommended to create a scale-in event for each scale-out event you create.
AWS EC2 services sends EC2 metrics to CloudWatch about ASG instances.
Basic monitoring is every 300sec enabled by default and free of cost.
You can enabled detailed every 60sec which is chargeable.
When the launch configuration is done by AWS CLI, detailed monitoring for EC2 instances is enabled by default.

StandBy State:

You can manually move an EC2 instance form an ASG and put it in standby state.
Instances in standby state are still managed by auto scaling.
Instances in standby state are charged as normal in service instances.
They do not count towards available EC2 instances for workload/app use.
Auto scaling does not perform health check on instance in standby state.

Scaling Policies:

Generally scaling policy is of two types such as:

1. Manual
2. Dynamic

Again dynamic policy is divided into three categories as follows:

A. Target Tracking
B. Simple Scaling Policy
C. Step Scaling Policy

Define how much you want to scale based on defined conditions.
ASG uses alarms and policies to determine scaling.
For Simple or Step scaling a scaling adjustment can’t change the capacity of the group above the maximum group or below the minimum group.

Predictive/Scheduled/Cycle Scaling: it looks at historic pattern and forecast them into the future to schedule change in the number of EC2 instances. It uses machine learning model to forecast daily and weekly pattern.

Target Tracking Policies: increase or decrease the current capacity of the group based on a target value for specific metric. This is similar to the way that your thermostatic maintain the temperature of your home.

Step Scaling: increase or decrease the current capacity of the group based on a set of scaling adjustment known as step adjustment that vary based on the size of the alarm breach. It does not support/ wait for cool down times. It supports warm-up timer: time taken by newly launched instance to be ready and contribute to the watched metric.

Simple Scaling: single adjustment (up or down) in response to an alarm (cool down timer- 300sec by default)

Schedule Scaling: used for predictable load change. You need to configure a schedule action for a scale out at a specific date/time and to a required capacity. A scheduled action must have a unique data/time. You cannot configure two schedule activities at the same date/time.

ELASTIC LOAD BALANCER (ELB)

Load balancer distributes the web traffic to the available server.

Or

Load balancing refers to efficient distributing incoming traffic across a group of backend server.

Load Balancer is of 3 types:

1. Classic Load Balancer
2. Application Load Balancer
3. Network Load Balancer

An internet facing load balancer has a publicly resolvable DNS name.
Domain names for content on the EC2 instances served by the ELB, is resolved by the internet DNS server to the ELB DNS name (and hence IP address).
This is how traffic from the internet is directed to the ELB front-end.
Classic load balancer service support: http, https, TCP, SSL.
Protocols ports supported are 1-65535.
It supports IPV4, IPV6 and Dual Stack.
Application load balancer distributes incoming application traffic across multiple targets such as EC2 instances in multiple availability zone. This increases the availability of your application.
Network load balancer has ability to handle volatile workloads and scale to millions of request per seconds.
An ELB listener is the process that checks for connection request.
You can configure the protocol/ port number on which your ELB listener listen for connection request.
Fronted listeners check for traffic from client to the listener.
Backend listeners are configured with protocol/port to check for traffic from the ELB to the EC2 instances.
It may take some time for the registration of the EC2 instances under the ELB to complete.
Registered EC2 instances are those are defined under the ELB.
ELB has nothing to do with the outbound traffic that is initiated/generated from the registered EC2 instances destined to the internet or to any other instances within the VPC.
ELB only has to do with inbound traffic destined to the EC2 registered instances (as the destination) and the respective return traffic.
You start to be charged hourly (also for partial hours) once your ELB is active.
If you do not want to be charged as you so not need the ELB anymore, you can delete it.
Before you delete the ELB, it is recommended that you point the Route53 to somewhere else other than ELB.
Deleting the ELB does not affect or delete the EC2 instance registered with it.

ELB forwards traffic to “eth0” of your registered instances.
In case the EC2 registered instances has multiple IP address on eth0, ELB will route the traffic to its primary IP address.
Elastic load balancer supports IPV4 address only in VPC.

To ensure that the ELB service can scale ELB nodes in each AZ, ensure that the subnet defined for the load balancer is at least /27 in scale size and has at least 8 available IP address the ELB nodes can use to scale.
For fault tolerance it is recommended that you distribute your registered EC2 instances across multiple AZ with in the VPC region.
If possible, try to allocate same number of registered instances in each AZ.

The load balancer also monitors the health of its registered instances and ensures that it routes traffic only to healthy instances.
A healthy instance shows as healthy under the ELB.

When the ELB detects an unhealthy instance it stops routing traffic to that instance.
An unhealthy instance shows as unhealthy under the ELB.
By default AWS console uses ping http (port 80) for healthy check.
Registered instances must respond with an http “200 OK” message within the timeout period else it will be considered as unhealthy.

AWS API uses ping TCP (port-80) for health check.
Response time-out is 5 seconds (range is 2-60 sec).
Health check internet.
Period of time between health check (default 30 and range is 5 to 300 sec)

Unhealthy Threshold: number of consecutive failed health check that should occur before the instance is declared unhealthy.
Range is 2 to 10

Default is 2

Healthy Threshold: number of consecutive successful health checks that must occur before the instance considered unhealthy.
Range is 2 to 10

Default is 10

By default the ELB distributes traffic evenly between the AZ, it is defined in without consideration to the number of registered EC2 instances in each AZ.

Cross Zone Load Balancing:

Disabled by default.
When enabled, the ELB will distribute traffic evenly between registered EC2 instances.
If you have 7 EC2 instances in one AZ, and 3 in another AZ, and you enabled cross zone
Load balancing each registered EC2 instances will be getting the same amount of traffic load from the ELB.
ELB name you choose must be unique within the account.
ELB is region specific, so all registered EC2 instances must be in the same region, but can be in different AZs.
To define your ELB in an AZ you can select one subnet in that AZ. Subnet can be public or private.
Only one subnet can be defined for the ELB in an AZ.
If you try and select another one in the same AZ, it will replace the former one.
If you register instance in an AZ with ELB but do not define a subnet in that AZ for the ELB, these instances will not receive traffic form the ELB.
ELB should always be accessed using DNS and not IP.

An ELB can be internet facing or internal ELB:

Internet Facing:

- ELB nodes will have public IP address.
- DNS will resolve the ELB DNS name to these IP address.
- If routes traffic to the private IP address of your registered EC2 instances.
- You need one public subnet in each AZ where the internet facing ELB will be defined such that the ELB will be able to route internet traffic.

Format of the public ELB DNS name of internet facing ELB:

name-1234567890.region.elb.amazonaws.com

Format of the internal ELB:

Internal-name.123456789.region.elb.amazonaws.com

An ELB listener is the process that checks for connection request.
Each network load balancer needs at least one listener to accept traffic.
You must assign a security group to your ELB. This will control traffic that can reach your ELB front end listeners.

Target Group:

Logical grouping of targets behind the load balancer.
Target groups can be exist independently from the load balancer.
Target group can be associated with an auto scaling group.
Target group can contain up to 200 targets.

IDENTITY AND ACCESS MANAGEMENT (IAM)

IAM refers to a framework or policy and technologies for ensuring that the proper people in an organization have the appropriate access to technology resources.

OR

AWS Identity and Access Management is a web service that you security control access to AWS resources. We use IAM to control who is authenticated (signed-in) and authorized (has permission) to use resources.

When you first create AWS account, you begin in a single sign-in identity that has completely access to all AWS services and resources in the account.
This identity is called the AWS account "Root-User" and is accessed by sighed-in with the email address and password that you used to create the account.
AWS strongly recommends that you do not use the root user for you everyday tasks, even the administrative ones.
Use other IAM user account to manage the administrative task of your account and securely lock away the root user credentials and use them to perform only a few account and service management task.
IAM user limit is 5000 per AWS account. You can add up to 10 users at one time.
You are also limit to 300 groups per AWS account.
Default limits of managed policies attached to an IAM role and IAM user is 10.
IAM user can be a member of maximum 10 groups.
We can assign maximum two access keys to an IAM user.

IAM Features:

1. **Shared access to your AWS account:**

   You can grant other people permission to administer and use resources in your AWS account without having to share your access credentials.

2. **Granular permission:**

   You can grant different permission to different people for different resources. For instance, you can allow some users complete access to EC2, S3, Dynamo DB, Redshift while for others, you can allow read only access to just some S3 buckets, or permission to administer just some EC2 instances or to access your billing information but nothing else.

3. **Secure access to AWS resources for application that run on Amazon EC2:**

   You can use IAM features to securely give application that run on EC2 instances the credentials that they need in order to access other AWS resources. For example, include S3 buckets and RDS or Dynamo DB databases.

4. **Multifactor Authentication (MFA):**

   You can add two factor authentication to your account and to individual users for extra security. You can use physical hardware or virtual MFA (for e.g: Google Authenticator)

5. **Identity federation:**

   You can allow users who already have passwords elsewhere. For e.g: in your corporate network of with an internet identity provider to get temporary access to your AWS account.

6. **Identity information for assurance:**

   If you use AWS Cloud Trail, you receive log records that include information about those who made request for resources in your account. That information is based on IAM Identities.

7. **PCI-DSS compliance:**

   IAM supports the processing, storage and transmission of credit cards by a merchant of service provider, and has been validated as being complaint with payment card industries (PCI) data security standards (DSS).

8. **Eventually consistent:**

   If a request to change some data is successful, the change is committed and safely stored. However the change must be replicated across IAM which can take some time.
   IAM achieves high availability by replicating data across multiple servers within AWS data centre around the world.

Fee to Use: AWS IAM is a feature of AWS account offered at no additional charge. You will be charged only for use of other AWS products by your IAM users.

IAM Terms:

Following are the major terms which are used in an IAM account.

1. Principal
2. Request
3. Authentication
4. Authorization
5. Action/Operation
6. Resources

1. **Principal:**

    A principal is a person or application that can make a result for an action or operation on an AWS resources.
    Your administrative IAM user is your first principal.
    You can allow users and services to assume a role.
    IAM users, roles, federated users and application are all AWS principals.
    You can support federated users of programmatic access to allow an application to access your AWS account.

2. **Request:**

    When a principal tries to use the AWS management console, the AWS API of the AWS CLI that principal sends a request to AWS. The request includes the following information:
    - Actions: That the principal wants to perform.
    - Resources: upon which the actions are performed.
    - Principal information: it's including the environment from which the request was made.
      Request context: before AWS can evaluate and authorize a request, AWS gathers the request information. Principal (the requester) which is determined based on the authorization data. This includes the aggregate permissions that the associated with that principal.
    - Environment data: such as IP address, user agent, SSL enabled status, or the time of the day.
    - Resource data: it is related to the resource that is being requested.

3. **Authentication:**

    A principal sending a request must be authenticated (sighed into AWS) to send a request to AWS.
    Some AWS services, like AWS S3 allow request from anonymous users, they are exception to the role.

To authenticate from the console as a root user, you must sign-in with your user name and password.
To authenticate from the API to CLI, you must provide your access key and secrete key.
You might also be required to provide additional security information like MFA (e.g: Google Authentication)

4. **Authorization:**

To authorize request, IAM uses value from the request context to check, for matching policies and determine whether to allow or deny the request.
IAM policies are stored in IAM as JSON documents and specify the permission that are allowed or denied.
User (identity) Based Policy specifies permission allowed/denied for principals.
Note: by default the AWS root user access to all the resources in that account.

- **Resource Based Policies:**

  - It specifies permission allowed/denied for resources. Popular for granting cross account permission.
  - IAM checks each policy that matches the context of your request.
  - If a single policy includes a denied actions, IAM denies the entire request and stop evaluating. This is called **explicit deny**.

  The evaluation logic follows these Rules:

  - By default, all request are denied.
  - An explicit allow overrides this default.
  - An explicit deny overrides any allows.

You can create a new IAM policy in the AWS management console using one of the following ways:

- JSON: you can create your own JSON syntax.
- Visual Editor: you can construct a new policy from scratch in the visual editor. If you can use the visual editor you do not have to understand JSON syntax.
- Import: you can import a managed policy with in your account and then edit the policy to customize it to your specific requirement.

5. **Actions:**

Actions are defined by a service, and more the things that you can do to a resource such as viewing, creating, editing, and deleting that resource.
IAM supports approx. 40 actions for a user resource including create user, delete user etc.

Any actions or resources that are not explicitly allowed are denied by default. After your request has been authenticated and authorized, AWS approves the actions in your request.

6. **Resource:**

   A resource is an entity that exists within a service.
   Examples are EC2 instances, S3 bucket, IAM users, and Dynamo DB table.
   Each AWS service defines a set of actions that can be performed on each resource.
   After AWS approves the actions in your request those actions can be performed on the related resources within your account.
   If you create a request to perform an unrelated action on a resource that request is denied.
   When you provide permissions using an identity based policy in IAM then you provide permissions to access resources only within the same account.

Identity Federation:

If your account users already have a way to be authenticated such as authentication through your corporate network, you can federated those user identities into AWS.
A user who has already logged to the corporate using their corporate identity, the corporate can replace their existing identity with a temporary identity in your AWS account.
The user can work in the AWS management console.
Similarly, an application that the user is working with can make programmatic request using permission that you make.

Federation is particularly useful in those cases:-

1. **If your corporate direct ory is compatible with Security Assertion Markup Language (2.0):**

   You can configure your corporate directory to provide Single Sign-On (SSO) access to the AWS management console for your users.
   If your corporate directory is not compatible with SAML 2.0, you can create identity broker application to provide single sign-on access to the AWS management console for your users.
   If your corporate directory is Microsoft active directory, you can use AWS directory service to establish trust between your corporate directory and your AWS account.

2. **Your users already have Internet Identities:**

   if you are creating a mobile app or web-based app that can let users identity themselves through an internet identity provider like login with amazon, facebook, google or any open ID connect (OIDC) compatible identity provider, the app can use web federation to access AWS.
   AWS recommends to use AWS Cognito for identity federation.

IAM Users and SSO:

IAM users in your account have access only to the AWS resources that you specify in the policy that is attached to the user of to an IAM group that use the user belongs to.
To work in the console user must have permissions to perform the actions that the console performs such as listing and creating AWS resources.

IAM Identities:

IAM identities is what you create under your AWS account to provide authentication for people, application and process in your AWS account.
Identities represents the user and can be authenticated and then authorized to perform actions in AWS.
Each of these can be associated with one or more policies to determine what actions a user, role or member of the group can do with which resources and under what conditions.
IAM group is a collection of IAM user.
IAM role is very limit IAM user.

A. **IAM Users:**

   An IAM user is an entity that you create in AWS. It represents the person or service who uses the IAM user to interact with AWS.
   You can create 5 users at time.
   An IAM user can represent an actual person or an application that requires AWS access to perform actions on AWS resources.
   A primary use of IAM users is to give people the ability to sign-in to the AWS management console for interactive task and to make programmatic request to AWS services using the API or CLI.
   For any user you can assign them:
   - A username and password to access the AWS console.
   - An access key ID and secrete key that can use for programmatic access.
   - The newly created IAM user have no password and no access key.
     You need to create the user password.
   - Each IAM user is associated with one and only one AWS account.

- Users are defined within your account, so user do not have to do payment. Bill would be pay by the parent account.

B. IAM Groups:

An IAM group is a collection of IAM users.
It is a way to assign permission/policies to multiple users at once.
Use groups to specify permissions for a collection of users, which can make those permissions easier to manage for those users.
For E.g: you could have a group called HR and give that group the types of permissions that HR department typically needs.
Any user in that group automatically has the permission that are assigned to the group.
If a new user joins your organization and should have administrator privileges, you can assign the appropriate permissions by adding the user to that group.
If a person changes job in your organization, instead of editing that user's permission, you can remove him or her from the old groups and add him or her to the appropriate new groups.

IAM Group Limitations:
A group is not truly an identity in IAM because it cannot be identified as a principal in a permission policy.
Group cannot be nested.
One have a limit of 300 groups in an AWS account.
A user can be a member of up to 10 IAM groups.

C. IAM Roles:

An IAM role is very similar to a user, in that it is an identity with permission policies that determine what the identity can and cannot do in AWS.
An IAM role does not have any credentials (password or access key) associated with it.
Instead of being associated with one person, a role is intended to be assumable by anyone who needs it.
An IAM user can assume a role to temporarily take on different permissions for a specific task.
An IAM role can be assigned to a federated user who sign-in by using an external identity provider instead of IAM.

IAM Temporary Credentials:

Temporary credential are primary used with IAM roles but there are also other uses.
You can request temporary credentials that have a more restricted set of permissions than your standard IAM users.
This prevent you from accidentally performing task that are permitted by the more restricted credentials.
A benefit of temporary credentials in that they expire automatically after a set period of time.

Permissions and Policies:

The access management portion of AWS Identity and Access Management (IAM) helps you to define what a user or other entity is allowed to do in an account, often referred to as authorization.
Permissions are granted through policies that are created then attached to user, groups of roles.

Policies and User:

By default, IAM users can’t access anything in your account.
You grant permissions to a user by creating a policy, which is a document that defines the effect, actions, resource and optional conditions.
Any actions or resources that are not explicitly allowed are denied by default.

IAM Multiple Policies:

Users of groups can have multiple policies attached to them that grant different permission.
In the case of multiple policies attached to a user or group, the user’s permission are calculated based on the combination of policies.

Federated Users and Roles:

Federated user don’t have permanent identities in your account the way that IAM users do.
To assign permissions to federated users you can create an entity referred to as a role and define permission for the role.
When a federated user sign-in to AWS the users is associated with the role and is granted the permission that are defined in the role.

Resource based Policy:

In some cases like S3 bucket, you can attach a policy to a resource in addition to attaching it to a group or user. This is called a resource based policy.
A resource based policy contains slightly different information than user-based policy.
In resource based policy you specify what actions are permitted and what resource is affected.
You also explicitly list who is allowed access to the resource (a principal).
Resource based policies include a principal element that specifies who is granted the permissions.

IAM User-The Root User:

When you first create an AWS account, you create an account (or root user) identity, which you use to sign-in to AWS.
The account root user credentials are the e-mail address to create the account and a password which can be used to sign-in to the AWS Management console as the root user.
When you sign-in as root user, you have complete unrestricted access to all resources in your account including access to your billing information and the ability to change your password.
The level of access is necessary when you initially set up the account.
It is not possible to restrict the permission that are granted to the AWS account.

AWS Recommends That:

AWS recommends that you don't use root user credentials for everyday access.
Also AWS recommends that you do not share your root user credentials with anyone because doing so gives them unrestricted access to your account.
Create an IAM user for yourself and then assign yourself administrative permission for your account.
You can then sign-in as that user to add more users as needed.
An IAM user with administrator permissions is not the same things as the AWS account root users.

IAM Users:

An IAM user is an entity that you create in AWS. It represents the person or service who uses the IAM user to interact with AWS.
An IAM can represent an actual person or an application that requires AWS access to perform action on AWS resources.
IAM users are global entities, like an AWS account is today. No region is required to be specified when you define user permissions. Users can use AWS services in any geographic region.

For Any User You can assign them:

A user name and password to access the AWS console.
An access key (access key and secrete key) that they can use for programmatic access (issuing request) to your AWS account.
You assign either or both based on the user activities and needs.
You can view and download your secret access key only when you create the access key.
You cannot view or recover a secret access key later.
If you lose your secrete access key, you can create a new access key.
Each IAM user is associated with one AWS account.

By Default a new IAM User:

A new IAM user has no permission to do anything.
Has no password and no access key (neither an access key ID nor a secret access key). It means no credentials of any kind.
You must create the type of credentials for an IAM users based on what the user will be doing.
You can grant user permissions by attaching IAM policies to them directly or making them members of IAM group where they inherit the group policies/permissions.
You can have up to 5000 user per account.

IAM Roles:

An IAM role is a set of permissions that grant access to the actions and resources in AWS.
These permissions area attached to the role, not to an IAM user or group. Instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.
A role does not have standard long-term credentials (password or access key) associated with it.
If a user assumes a role, temporary security credentials are created dynamically and provided to the user.
Following entities can used role:

- An IAM user in the same AWS account.
- An IAM user in a different AWS account.
- A webserver offered by AWS such as Amazon EC2.

There are Two ways to use a Role:

1. **Internally in the IAM Console:**

   IAM users in your account using the IAM console can switch to a role to temporarily use the permissions of the role in the console.
   The user give up their original permission and take on the permission assigned to the role.
   When the user exists the role, their original permissions are restored.

2. **Programmatically with the AWS CLI, tools for windows powershell or API:**

   An application or a service offered by AWS (like Amazon EC2) can assume a role by requesting temporary security credentials for a role with which to make programmatic request to AWS.
   You use a role this way so that you don't have to share or maintain long-term security credentials for each entity that requires access to a resource.

Difference between IAM Role and Resource Based Policy:

Unlike a user-based policy, a resource based policy specifies who can access that resource.
Cross account access with a resource based policy has an advantage over a role, with a resource that is accessed through a resource-based policy, the user still works in the trusted account and does not have to give up this or her user permissions in place of the role permissions.
In other words, the user continuous to have access to resources in the trusted account at the same time as he or she has access to the resource in the trusting account.
This is useful for task such as copying information to or form the shared resource in the other account.
Note that not all services support resource-based policy.

IAM Role Delegation:

Delegation is the granting of permission to someone to allow access to resource that you control.
Delegation involves setting up a trust between the account that owns the resource (the trusting account) and the account that contains the users that need to access the resource (the trusted account).
The trusted and trusting accounts can be of the following:

i. The same account
ii. Two accounts that are both under your organization's control.
iii. Two account owned by different organizations.

To delegate permission to access a resource you create an IAM role that has two policies attached.

i. The Trust Policy

ii. The Permission Policy

The trusted entity is included in the policy as the principal element in the document. When you create a trust policy, you cannot specify a wildcard (*) as a principal.

Cross Account Permissions:

You might need to allow user from another AWS account to access resources in your AWS account. If so, don't share security credentials, such as access keys between accounts. Instead use IAM roles.

You can define a role in the trusted account that specifies what permissions the IAM users in the other account are allowed.

You can also designate which AWS account have the IAM users that are allowed to assume the role. We do not define users here rather AWS account.

Role for Cross-Account Access:

Granting access to resources in one account to a trusted principal in a different account.

Roles are the primary way to grant cross-account access.

However with some of the web services offered by AWS, you can attach a policy directly to a resource. These are called resource-based policy. You can use them to grant principals in another AWS account access to the resource.

The following services support resource-based policy:

- Amazon S3.
- Amazon Simple Notification Service
- Amazon Simple Queue Service
- Amazon Glacier Vault

RELATIONAL DATABASE SERVICES (RDS)

What is DATA?

In simple words, data can be facts related to any object. For e.g: your age, job, house no, contact no., name, places are some data related to you.

What is DATABASE?

Database is a systematic collection of data. Databases supports storage and manipulation of data.

e.g : facebook, telecom companies, amazon.com

What is DBMS?

DBMS is a collection of programs which enable its users to access database, manipulate data, reporting/ representation of data.

Types of DBMS

1. Hierarchical
2. Network
3. Relational
4. Object oriented 5.

Relational Database:

A relational database is a data structure that allows you to link information from different tables of different types of data bucket.
Tables are related to each other.
All fields must be filled.
Best suited for OLTP (online transaction processing)
Relational DB: MySQL, Oracle, DBMS, IBM DB2
A row of a table is also called records. It contains the specific information of each individual entry in the table.
Each table has its own primary key.
A schema (design of database) is used to strictly define tables, columns, indexes and relation between tables.
Relational DB are usually used in enterprises application/scenario. Exception in MySQL which is used for web application.
Common application for MySQL include php and java based web applications that requires a database storage backend. E.g: JOOMLA
Cannot scale out horizontally.
Virtually all relational DB uses SQL.

Non-Relational DB/NO-SQL DB:

Non-relational databases store data without a structured mechanism to link data from different tables to one another.
Required low cost hardware.
Much faster performance (read/write) compared to relational DBs.
Horizontal scaling is possible.
Never provide tables with flat fixed column records. It means schema-free.
Best suited for online analytical processing (OLAP).
E.g: of NoSQL databases: MongoDB, Casssandra, DynamoDB, Postegre, Raven, Redis.

Types of No-SQL Databases:

1. Columnar Database (cassandra, HBase)
2. Document Databse (MongoDB, CouchDB, RavenDB)
3. Key Value Database (Redis, Riak, DynamoDB, Tokyo Cabinet)
4. Graph Based Database (Neo4J, FlockDB)

1. **Columnar Database:**

A columnar database is a DBMS that stores data in columns instead of Rows. In a columnar DB all the column-1 values are physically together followed by all the column-2 values.

In a row oriented DBMS the data would be stored like this: (1, bob, 30, 8000: 2, arun, 26, 4000: 3, vian, 39, 2000 ;)

In a column based DBMS the database would be stored like this: (1, 2, 3: bob, arun, vian; 30, 26, 39; 8000, 4000, 2000 ;)

Benefit is that because a column based DBMS is self-indexing, it uses less disk space that a RDBMS containing the same data. It easily perform operation like min, max, average.

2. **Document Database:**

Document DB make it easier for developer to store and querying data in a DB by using the same document model format they use in their application code.
Document DBs are efficient for storing catalogue.
Store semi-structure data as document typically in JSON or XML format. (example)
A document database is a great choice for contain management application such as blogs and video platform.

3. **Key-Value Database:**

   A key-value DB is a simple DB that uses an associative array (like dictionary) as a fundamental model where each key is associated with one and only one value in a collection.
   It allows horizontal scaling.
   Used cases: shopping cart, and session store in app like fb and twitter.
   They improve application performance by storing critical pieces of data in memory for low latency access.
   Amazon elasticache as an in-memory key-value stores.

4. **Graph Based Database:**

   A graph DB is basically a collection of nodes and edges.
   Each node represent an entity and each edges represent a connection or relationship between two nodes.

   In an AWS fully managed relational DB engines service where AWS is responsible for:
   - Security and patching.
   - Automated backup.
   - Software updates for DB engine.
   - If selected multi AZ with synchronous replication between the active and stand by DB instances.
   - Automatic failover if multi AZ option was selected.
   - By default, every DB has weekly maintenance window. (max 35 days.)

   Settings managed by the users:

   - Managing DB settings.
   - Creating relational database schema.
   - Database performance tuning.

Relational Database Engine Options:

1. MS SQL Sever
2. My SQL: supports 64TB of DB
3. Oracle
4. AWS Aurora: high throughput

5. Postgre SQL: highly reliable and stable
6. Maria DB: MySQL compatible, 64TB DB

There are two Licensing Options:

1. BYOL (Bring Your Own License)
2. License from AWS on hourly basis

RDS Limits:

Up to 40DB instances per account.
10 of this 40 can be Oracle or MS-SQL server under license included

model. Or

Under BYOL model, all 40 can be any DB engine you need.

RDS Instance Storage:

Amazon RDS use EBS volumes (not instance store) for DB and logs storage.

1. General Purpose: use for DB workloads with moderate I/O requirement. Limits: min: 20 GB
   Max: 16384 GB

2. Provisional IOPS RDS Storage: use for high performance OLTP workloads. Limits: min: 100GB
   Max: 16384GB

Templates available in RDS:

a. Production
b. Dev/Test
c. Free-Tier

DB Instance Size:

a. Standard class
b. Memory-Optimized class
c. Burstable class

What is Multi-AZ in RDS:

You can select multi AZ option during RDS DB instance launch.
RDS service creates a standby instances in a different AZ in the same region and configure “synchronous replication” between the primary and standby.
You cannot read/write to the standby RDS DB instances.
You cannot select which AZ in the region will be chosen to create the standby DB instance.
You can however view, which AZ is selected after the standby is created.
Depending on the instance class it may take 1 to few minutes to failover to the standby instance.
AWS recommends the use of provisioned IOPS instances for multi-AZ RDS instance.

When Multi-AZ RDS Failover Triggers:

- In case of failure of primary DB instance failure.
- In case of AZ failure.
- Loss of network connectivity to primary DB.
- Loss of primary EC2 instance failure.
- EBS failure of primary DB instance.
- The primary DB instance is changed.
- Patching the O.S of the DB instance.
- Manual failover. (in case of rebooting.)

Multi-AZ RDS Failover Consequences:

- During failover the CNAME of the RDS DB instance is updated to map to the standby IP address.
- It is recommended to use the end point to reference your DB instances and not its IP address.
- The CNAME doesn’t change because the RDS endpoint doesn’t change.
- RDS end point doesn’t change by selecting multi-AZ option, however the primary and standby instances will have different IP addresses, as they are in different AZ.
- It is always recommended that you do not use the IP address to point RDS instances, always use endpoint. By using endpoint there will be no change whenever a failover happens.

When we do manual failover?

In case of rebooting.
This is by selecting the “reboot with failover” reboot options on the primary RDS DB instances.

A DB instance reboot is required for changes to take effect when you change the DB parameter group on when you change a static DB parameter.

Whenever failover occurs AWS RDS sends SNS notification.
You can use API calls to find out the RDS events occurred in the last 14 days.
Even you can use CLI to view last 14 days events.
Using AWS console you can only last one day events
In case of OS patching, system upgrades and DB scaling, these things happens on standby first then on primary to avoid outage.
In multi-AZ, snapshots and automated backups are done on standby instance to avoid I/O suspension on primary.

RDS Multi-AZ Deployment Maintenance:

Firstly perform maintenance of standby.
Now convert standby into primary so that maintenance can be done on primary.(currently)
You can manually upgrade a DB instance to a supported DB engine version from AWS console as follows:- RDS->DB instance->modify DB->set DB engine version.
By default change will take effect during the next maintenance window.
Or you can force a immediate upgrade if you want.
In multi-AZ version upgrade will be conducted by both primary and standby at the same time which will cause an outage.
Do it during maintenance window.

There are two methods to backup and restore your RDS DB instances:

1. AWS RDS automated backup
2. User initiated manual backup

Either you can take backup of entire DB instance or just the DB.
You can create a restore volume snapshots of your entire DB instances.
Automated backups by AWS, backup your DB data to multiple AZ to provide for data durability.
Select-automated backup in AWS console.
Stored in Amazon S3.
Multi-AZ automated backups will be taken from the standby instance.
The DB instance must be in “ACTIVE” state for automated backup.
RDS automatically backups the DB instances daily by creating a storage volume snapshots of your DB instance (fully daily snapshots) including the DB transaction logs.
You can decide when you would like to take backup (window)
No additional charge for RDS backing up your DB instance.
For multi-AZ deployment, backups are taken from the standby DB instance (true for Maria DB, MySQL, Oracle, Postgre SQL).
Automated backups are deleted when you delete your RDS DB instance.

An outage occurs if you change the backup retention period from zero to non-zero value or the other way around.
Retention period of automate backup is 7 days (by default) via AWS console.
AWS Aurora is an exception. Its default is 1 day.
Via CLI or API 1 day by default.
You can increase it up to 35 days.
If you don't want backup, put zero "0" in the retention period.
In case of manual snapshot, point-in-time recovery is not possible.
Manual snapshot is also stored in S3.
They are not deleted automatically, if you delete RDS instance.
Take a final snapshot before deleting your RDS DB instance.
You can share manual snapshot directly with other AWS Account.
When you restore a DB instance only the default DB parameters and security groups are associated with the restored instance.
You cannot restore a DB snapshot into an existing DB instance rather it has to create a new DB instance it has new endpoint.
Restoring from the backup or a DB snapshot changes the RDS instance endpoint.
At the time of restoring, you can change the storage type (general purpose or provisioned.)
You cannot encrypt an existing unencrypted DB instance.
To do that you need to: create a new encrypted instance and migrate your data to it (from unencrypted to encrypted) or you can restore from a backup/snapshot into a new encrypted RDS instance.
RDS supports encryption-at-rest for all DB engines using KMS.
What actually encrypted when data-at-rest:

a. All its snapshots.
b. Backups of DB (S3 storage.)
c. Data on EBS volume.
d. Read replica created from the snapshots.

Some points related to RDS Billings:

No upfront cost.
You have to pay only for:

a. DB instance hours (partial hour charged as full hours)
b. Storage GB/month.
c. Internet data transfer.
d. Backup storage (i.e S3)
   This is increases by increasing DB backup's retention period.
   Also charged for:
   a. Multi-AZ DB hours.
   b. Provisioned stage (multi-AZ)
   c. Double write I/O
   d. You are nor charged for DB data transfer during replication from primary to standby.

DYNAMO DB

Database Types:

Generally there are three types of data are available such as:

1. Unstructured Data
2. Semi-structured Data
3. Structured Data

1. **Unstructured Data:**

   It is the information that either doesn't have a predefined data model or it is not organized in a predefined manner.
   Unstructured information is text-heavy but may contains data such as dates, numbers and facts as well as examples include email messages, word processing documents, videos, photos, audio files, presentations webpages.

2. **Semi-structured Data:**

   Semi-structured data is information that doesn't reside in a relational database but that does have some organizational properties that make it easier to analyze.
   E.g: XML, JSON

3. **Structured Data:**

   It refers to information with a high degree of organization, such that inclusion in a relational database is seamless and readily searchable by simple, straight forward search engine algorithms or other search operation.
   All data which can be stored in a database SQL in table with rows and columns. They have relational key and can be easily mapped into pre-defined fields.

DynamoDB Table:

A table is a collection of data items.
Like all other DB, dynamoDB stores data in tables.

Items:

Each table contains multiple items.
An item is a group of attributes that is uniquely identifiable among all of the other items.
Am item consists of a primary or composite key and a flexible number of attributes.

Items in DynamoDB are similar into rows, records in other DB.

Attributes:

Each item is composed one or more attributes.
An attribute consists of the attribute name and a value or a set of values.
An attribute is a fundamental data element, something that does not need to be broken down any further.

Note: aggregate size of an item cannot exceed 400kb including key and all attributes.

DynamoDB allows low latency read/write access to items ranging from 1 byte to 400kb.
DynamoDB can be used to store pointers to S3 stored objects, or items of sizes larger than 400kb too if needed.
DynamoDB stores data indexed by a primary key, you can specify the primary key when you create the table.
Each item in the table has a unique identifier. Or primary key, that distinguished the item from all of the others in the table.
The primary key in the only required attributes for items in a table.
DynamoDB tables are schema less, which means that neither the attributes not their data types need to be defined before head.
Each item can have its own distinct attributes.

DynamoDB-Read Capacity Unit:

One read capacity unit represents one strongly consistent read per second, or two eventually consistent reads per seconds for an item up to 4kb in size.
It you need to read an item that is larger than 4kb, DynamoDB will need to consume additional read capacity units.
The total number of read capacity units required depends on the item size and whether you want an eventually consistent or strongly consistent read.

DynamoDB-Write Capacity Unit:

One write capacity unit represents one write per second for an item up to 1kb in size.
If you need to write an item that is larger than 1kb, DynamoDB will need to consume additional write capacity units.
The total number of write capacity units required depends on the item size.

DynamoDB- Pricing:

Reads are cheaper than writes when using DynamoDB. We are only pay for:

Each table’s provisioned read/write throughput (hourly rates).
You are charged for provisioned throughput regardless whether you use it or not.
Indexed data storage.
Internet data transfer (if crosses a region).
Free tier per account (access all tables) of 25 read capacity unit and 25 write capacity units per month.
DynamoDB can do 10,000 writes capacity units or 10,000 read capacity units per second per table.

DynamoDB Limits:

256 tables per account per region.
No limits on the size of any table.

ROUTE 53

You can use Amazon Route53 to register new domains, transfer existing domains, route traffic for your domains to your AWS and external resources and monitor the health of your resources.

Route 53 Functions:

1. DNS Management
2. Traffic Management
3. Availability Monitoring
4. Domain Registration

Route 53 performs three main functions:

I. Register your domain.
II. As a DNS, it routes internet traffic to the resources for your domain.
III. Check the health of your resources.

Route 53 sends automated requests over the internet to a resource (can be a web server) to verify that the server is reachable, functional or available.
Also you can choose to receive notifications when a resource becomes unavailable and choose to route internet traffic away from unhealthy resources.

You can use Route 53 for any combination of these functions: for e.g: you can use Route 53 both to register your domain name and to route internet traffic for the domain.
Or you can use Route 53 to route internet traffic for a domain that you register with another domain register.

When you register a domain with Route 53 the service automatically makes itself the DNS service for the domain by doing the following:

- It creates a hosted zone that has the same name as your domain.
- It assign a set of four name servers to the hosted zone, unique to the account.
- When someone uses a browser to access your website, these name servers inform the browser where to find your resources such as a web server of an Amazon S3 bucket.
- It gets the name servers form the hosted zone and adds them to the domain.

AWS supports:

1. Generic top level domains
2. Geographic top level domain

Registering a domain with Route 53:

You can register a domain with Route 53 if the TLD is included on the supported TLD list.
If the TLD is not included you can't register the domain with Route 53.

Using Route 53 as your service: you can use Route 53 as the DNS service for any domain, even if the TLD for the domain is not included on the supported TLD list.

NOTE: Each Amazon Route 53 account is limited to a maximum of 500 hosted zones and 10,000 resource record sets per hosted zone. You can increase this limit by requesting to AWS.

Steps to Configure Route 53:

You need to register a domain, this can be Route 53, or another DNS registrar but then you connect your domain name in that registrar to Route 53.
Create hosted zone on Route 53, this is done automatically if you registered your domain using Route 53, inside the hosted zone, you need to create record sets.

Deleting to Route 53:

This steps connects everything and make it works.
Connect the domain name to the Route 53 hosted zone: this is called delegation.
Update your domain registrar with the correct name servers for your Route 53 hosted zone.
No other customer hosted zone will share this delegation set with you.
Doing this means Route 53 DNS service will be serving DNS traffic for the domain of the hosted zone.
If you registered your domain with a different registrar you need to configure the Route 53 name servers list in your registrar DNS database for your domain.

If you are using another domain provider and you did all the changes:

- When you migrate from one DNS provider to another for an existing domain, this change can take up to 48 hours to be effective.
- This is because name server DNS records are typically cached across the DNS system globally on the internet for up to 48 hours (TTL) periods.

Transferring a domain to Route 53:

You can transfer a domain to Route 53 if the TLD is included on the following list:
- If the TLD is not included, you can't transfer the domain to Route 53.

- For most TLD you need to get an authorization code from the current registrar to transfer a domain.

Route 53 Hosted Zone:

A Route 53 hosted zone is a collection of records for a specific domain.
You create a hosted zone for a domain, and then you create records to tell the domain name system how you want traffic to be routed for that domain.
Basically a hosted zone is a container that holds information about the how you want to route traffic for a domain and its subdomains.
You can create public (internet) hosted zones, or private (internal DNS) hosted zones.
For each public hosted zone that you create Amazon Route 53 automatically creates a name server record and a start of authority (SOA) record. Don't change these records.
Route 53 automatically creates a name server record with the same name as your hosted zone.
It list the four name servers that are authoritative name servers for your hosted zone.
Do not add, change, or delete name servers in this record.
When you create a hosted zone Amazon Route 53 automatically creates a name server records and a start of authority (SOA) record for the zone.
The name server record identifies the four name server that you give to your registrar of your DNS service so that DNS queries are routed to Route 53 name server.
By default Route 53 assigns a unique set of four name servers (known collectively as a delegation set.) to each hosted zone that you create.
e.g:

ns-1337.awsdns-39.co
m
ns-895.awsdns-47.net
ns-428.awsdns-53.org
ns-1597.awsdns-07.co.uk

Route 53 as your Authoritative DNS:

Once you update the Route 53 name server settings with your domain registrar to include the Route 53 name servers, Route 53 will be responsible to respond to DNS queries for the hosted zone.
This is true whether you do have a functioning website or not.
Route 53 will respond with information about the hosted zone whenever someone types the associated domain name in a web server.

You can create more than one hosted zone with the same name and add different records to each hosted zone.
Route 53 assigns four name servers to every hosted zone.
The name servers are different from each other.

When you update your registrar's name server records be careful to use the Route 53 name servers for the correct hosted zone- the one that contains the records that you want Route 53 to use when responding to queries for your domain.
Route 53 never returns values for records in other hosted zones that have the same name.

Route 53 hosted zone default entries: inside the hosted zone by default you have two entries:

a. **Name server entry:** it contains the unique sets of name servers for this hosted zone.
b. **SOA entry:** it contains information about the hosted zone.

If you are currently using another DNS service and you want to migrate to Amazon Route 53:

- Start by creating a hosted zone.
- Route 53 automatically assigns the delegation sets the four name servers to your hosted zone.
- To ensure that DNS routes queries for your domain to the Route 53 name servers.
- Update your registrar's or your own DNS service's name server records for the domain to replace the current name servers with the names of the four Route 53 name servers for your hosted zone.
- The method that you use to update the name server records depends on which registrar or DNS service you are using.
- Some registrar only allow you to specify name servers using IP addresses they don't allow you to specify fully qualified domain names.
- If your registrar requires using IP addresses,, you can get the IP addresses for your name servers using the dig utility (for mac and linux) and nslookup (for windows).[cmd- nslookup].

Transferring a domain between accounts within AWS:

- Transferring a domain to a different AWS account: if you registered a domain using one AWS account and you want to transfer the domain to another AWS account you can do so by contacting the AWS support center and requesting the transfer.

Migrating a hosted zone to a different AWS account:

- If you are using Route 53 as the DNS service for the domain, Route 53 doesn't transfer the hosted zone when you transfer a domain to a different AWS account.
- If domain registration is associated with one account and the corresponding hosted zone is associated with another account, neither domain registration not DNS functionality is affected.

- The only affect is that you will need to sign into the Route 53 console using one account to see the domain and sign-in using the another account to see the hosted zone.

Supported DNS record types by Route 53:

1. **A record:** Address record- maps domain name to IP address. e.g: www.cns.com IN A 5.5.5.5

2. **AAAA record-** IPv6 address record: maps domain name to an IPv6 address. e.g: www.cns.com IN AAAA 2002:b786::1

3. **CNAME**- maps an alias to a hostname. E.g: web IN CNAME www.cns.com

4. **NS record**- name server record used for delegating zone to a name server. E.g: cns.com IN NS nsi.cns.com

5. **SOA record**- start of authority record.

6. **MX record**- mail exchange: defines where to deliver mail for user@ a domain name. e.g: cns.com IN MX 10 mail01.cns.com

   Name server records defines which name server is an authoritative to a particular zone of domain name and point you to other DNS servers.
   A/AAAA are called host records, like business cards.
   CNAME is an alternative record or an alias for another record.
   Helpful in redirection or if you want to hide details about your actual server from the users.

Start of Authoritative (SOA) Record:

Every single zone has one and only one SOA resource record at the beginning of the zone.
It is not an actual record, it includes the following information:

- Who is the owner (email for the domain)
- The authoritative server
- The serial number which is incremental with changes to the zone data.
- The refreshing time/cycle into and the TTL.

CNAME Record Type:

A CNAME value element is the same format as a domain name.
The DNS protocol doesn't allow you to create a CNAME record for the top node of a DNS namespace, also known as the zone apex. (root domain)
For e.g: if you register the DNS name cns.com, the zone apex is cns.com.
You cannot create a CNAME record for cns.com.
However you can create CNAME records for www.cns.com, supports.cns.com and so on.
In addition it you create a CNAME record for a subdomain, you cannot create any other records for that subdomain.
For e.g: if you create a CNAME for www.cns.com, you cannot create any other records for which the value of the name field is www.cns.com.

AWS Route 53 Routing Policies:

1. Simple Routing (default)
2. Failover Routing
3. Geo Location Routing
4. Multi Value Routing
5. Latency based Routing
6. Weighted Routing
7. Geo-Proximity

Failover Routing:

Failover routing lets you route traffic to a resource when the resource is healthy. If the main resource is not healthy then route traffic to a different resource.
The primary and secondary records can route traffic to anything form an Amazon S3 bucket that is configured as a website to a complex tree of records.
Failover routing policy is applicable for public hosted zone only.

Geo Location Routing:

Geo location routing lets you choose the resources that serves your traffic based on the geographic location of the user's i.e the location that DNS queries originate from.
For e.g: you may have presence in Europe and Asia. Now you want users in the Asia to be served in the Asia and those in Europe to be served by servers in Europe.

Benefits:

You can localize your content and present some of all of your website in the language of your user's.

You can also use geo location routing to restrict distribution of content to only the locations in which you have distribution rights.
You can specify geographic locations by continent, by country or by state in the United States.
If you create separate records for overlapping geographic regions, for e.g: one record for north America and one for Canada- priority goes to the smallest geographic location.
Geo location works by mapping IP address to locations. However some IP address are not mapped to geographic location.

Latency Based Routing:

If your application is hosted in multiple Amazon EC2 regions, you can improve performance for your users by serving their requests form the Amazon EC2 region that provide the lowest latency.
To use latency-based routing, you create latency records for your resources in multiple EC2 regions.
When Amazon Route 53 receives a DNS query for your domain of subdomain:

- It determines which Amazon EC2 region you have created latency record for it.
- Determine which region gives lowest latency to users.
- Then select a latency records for that region.
- For e.g: suppose you have ELB in US-East and in Asia Pacific (Mumbai) region:
- You can create a latency based record for each load balancer.
- Here's what happens when a user in London enters the name of your domain in a browser, DNS routes the request to its data on latency between London and the Mumbai region and between London and the N.Virginia.
- Id latency is lower between London and N.Virginia, Route 53 respond to the query with the IP address for the N.Virginia load balancer.

Weighted Based Policy:

Weighted routing policy lets you associate multiple resources with a single domain name or subdomain name and choose how much traffic is routed to each resource.
This can be useful for a variety of purposes, including load balancing and testing new versions of software.
Weights can be assign any number from 1 to 255.
Weighted routing policy can be applied when there are multiple resource that perform the same function for e.g: webserver serving the same website.
To configure weighted routing, you can create records that have the same name and type for each of your resource.
Amazon Route 53 send traffic to a resource based on the weight that you assign to the record as a proportion of the total weight for all records in the group.
For e.g: suppose for www.cns.com has three resource record sets with weights of 1 (20%), 1 (20%), and 3 (60%), (sum=5)

On average Route 53 selects each of the first two resource record set one-fifth of the time, and returns the third resource record set three-fifth of the time.

Geo Proximity Routing Policy:

Use when you want to route traffic based on the location of your resources and optionally shift traffic form resources in one location to resources in another.
You can also optionally choose to route more traffic of less to a given resource by specifying a value known as “BIAS”, it expands or shrinks the size of the geographic region from which traffic is routed to a resource.

Multi Value Answer Routing Policy:

Use when you want Route 53 to respond to DNS queries with up to eight healthy record selected at random.
Multi value routing lets you configure Amazon Route 53 to return multiple values such as IP address for your webservers, in response to DNS queries. You can specify multiple values for almost any record nut multi value answer routing also lets you check the health of each resource, so Route 53 returns only values for healthy resources, it’s not a substitute for a load balancer.
But the ability to return multiple health checkable IP addresses is a way to use DNS to improve availability and load balancing.

CLOUD FRONT

Amazon cloud front is a web service that gives business and web application developers an easy and cost effective way to distribute content with low latency and high data transfer speed.

Cloud front is a global service.
Amazon cloud front is a web service that speeds up distribution of your static and dynamic web contents such as .html, .css, .js and image files to your users.
Cloud front delivers your content through a worldwide network of data centers called Edge Locations.
When a user request content that you are serving with cloud front, the user is routed (via DNS resolution) to the edge location that provides the lowest latency, so that content is delivered with the best possible performance.
If the content is already in the edge location with the low latency, cloud front delivers it immediately.
This dramatically reduces the numbers of networks that your user's request must pass through which improves performance.
If not, cloud front retrieves it from an Amazon S3 bucket or an HTTP/ web server that you have identifies as the source for the definitive version of your content (origin server).
Cloud front also keeps persistent connection with origin servers so files are fetched from the origins as quickly as possible.

You can access Amazon cloud front in the following ways:

a. AWS management console
b. AWS SDKs
c. Cloud front API
d. AWS CLI

Cloud front Edge Locations:

Edge locations are not tied to availability zones or regions.
Amazon cloud front has 216 points of presence (205 edge locations and 11 regional edge caches in 84 cities across 42 countries)

Cloud front Regional Edge Cache:

Amazon cloud front has added several regional edge cache locations globally at close proximity to your viewers.
They are located between your origin web server and the global edge locations that serve content directly to your viewers.

As objects become less popular, individual edge locations may remove those objects to make room for more popular content.
Regional edge cache working as an alternative of origin to reduce the burden of origin.
Regional edge cache have a large cache width than any individual edge location, so object remain in the cache longer at the nearest regional edge caches.

Cloud front Regional Edge Cache- working:

Whenever a viewer makes a request on your website or through your application, DNS routes the request to the cloud front edge location that can best serve the user's request.
This location is typically the nearest cloud front edge location in terms of latency.
In the edge location, cloud front checks its cache for the requested files.
If the files are in the cache, cloud front returns then to the user.
If the files are nor in the cache, the edge servers go to the nearest regional edge cache to fetch the object.
Regional edge caches have features parity with edge locations. For e.g: a cache invalidation request removes an object form both edge cache and regional edge caches before it expires.
The next time a viewer request the object, cloud front returns to the origin to fetch the latest version of the object.
Proxy method PUT/ POST/ PATCH/ OPTIONS/ DELETE go directly to the origin from the edge locations and do not proxy through the regional edge caches.
Dynamic content as determined at request time, doesn't flow through regional edge cache but goes directly to the origin.

SIMPLE QUEUE SERVICE (SQS)

SQS is a fast, reliable, fully managed message queue service.
It is a web service that gives you access to message queue that stores messages waiting to be processed.
It offers a reliable highly scalable, hosted queue for storing messages between servers.
It allows the decoupling of application components such that a failure in one components doesn't cause a bigger problem to application functionality. (like in coupled application)
Using SQS, you no longer need a highly available message cluster or the burden of running it.
You can delete all the messages in an SQS queue without deleting the SQS Queue itself.
You can use applications on EC2 instances to read and process the SQS queue message.
You can use auto scaling to scale the EC2 fleet processing the SQS messages, as the queue size increases.
These applications on EC2 instances can process the SQS message/ jobs then post the SQS results to other SQS queues or other AWS service.

Types of Amazon Queue Services:

It is of two types such as:

1. Standard Queue
2. FIFO Queue

1. **Standard Queue:**
   High Throughput
   At Least One Delivery
   Duplicacy is possible
   Best effort ordering

2. **FIFO Queue:**
   Limited throughput (300 TPS)
   Exactly one processing
   Duplicacy not possible
   Strict ordering: first-in-first-out
   FIFO queue are limited to 300 transactions per second (TPS), but have all the capabilities of standard queue.

SQS Pricing:

The first 1 million monthly requests are free, after that pricing is according to regions.
For e.g: in Mumbai region:

- Standard Queue- $0.40/ million request
- FIFO Queue- $0.50/ million request

How Amazon SQS charges:

1. API action: every Amazon SQS actions count as request.
2. FIFO request: API actions for sending, receiving, deleting and changing visibility of messages from FIFO queues are charged at FIFO rates.
3. Contents of Request: a single request can have from 1 to 10 messages, up to a maximum total payload of 256kb.
4. Size of Payload: each 64kn chunk of a payload is billed as 1 request. (for e.g: API action with a 256kb payload is billed as 4 request)
5. Interaction with Amazon S3.
6. Interaction with AWS KMS.

Short Polling:

A request is returned immediately even if the queue is empty.
It doesn't wait for message to appear in the queue.
It queries only a subset of the available servers for messages (based on weighted random distribution)
Default by SQS
Receive message wait time is set to 0.
More request are used which implies higher cost.

Long Polling:

It is preferred to regular/ short polling. It uses fewer requests and request cost by: eliminating false empty responses by querying all the servers.
Reduce the number of empty responses by allowing Amazon SQS to wait until a message is available in the queue before sending a response, Unless the connection timeout (20sec)
Receive message wait time is set to a non-zero value (max 20sec)
Billing is same for both pollings.

SQS Retention Period:

SQS messages can remain in the queue for up to 14 days. (SQS retention period)
Range is 1 min to 14 days. (default is 4days)
Once the maximum retention period of a message id reached, it will be deleted automatically from the queue.
Messages can be sent to the queue and read from the queue simultaneously.
SQS can be used with Dynamo DB, EC2, ECS, redshift, RDS, lambda, S3 to make distributed/ decoupled application.
You can have multiple queues with different properties.

SQS Visibility Timeout:

It is the duration of time a message is locked for read by other servers.
Maximum is 12 hours and default is 30 sec.
A server that read a message process it, can change the message visibility timeout if it needs more time to process the message.
After a message is read, there are the following possibilities:

a. An ACK is received that a message is processed, so it must be deleted from the queue to avoid duplicates.
b. If a fail is received of the visibility timeout expires, the message will then be locked for read such that it can be read and processed by another servers.

Delivery Delay: AWS SQS provides delivery delay options to postpone the delivery of new messages to a queue. If delivery delay is defined for a queue, any new messages will not be visible to the server for the duration of delay. The default (min) delay for a queue is 0 seconds. The maximum is 15 minutes.

Receive Message Wait Time: the default time is 0 seconds. This is maximum amount of time that a long polling receive call will wait for a message to become available before returning an empty response (maximum value is 20sec).

Dead Letter Queue:

The main task of a dead letter queue is handling message failure. A dead letter queue lets you to set aside and isolated message that can't be processed correctly to determine why their processing didn't success.
Don't use a dead letter queue with a FIFO queue, if you don't want to break the exact order of messages or operations.
DLQ must be of the same type as the source queue. (standard or FIFO)

SIMPLE NOTIFICATION SERVICE (SNS)

SNS is a fast, flexible, fully managed PUSH notification service.
It is a web service that delivery or sending of messages to subscribing endpoints or clients.
It allows for sending individual messages of fan-out messages to a large number of recipients or to other distributed AWS services.
Messages published to an SNS topics will be delivered to the subscriber immediately.
Inexpensive, pay-as-you-go model with no upfront cost.
Reliable: at least three copies of the data are store across multiple AZ in same region.
It is a way of sending messages. When you are using auto scaling, it triggers an SNS service which will email you that "your EC2 instance is growing".

Publisher: publishers are also known as produce and send the message to the SNS which is a logical access point.

Subscriber: subscribers such as web servers, email addresses, Amazon SQS queues, AWS lambda receive the message of notification from the SNS over one of the supported protocols (Amazon SQS, email, lambda, HTTPS, SMS).

SNS Topic:

It is a logical access point and communication channel.
Each topic has a unique name.
A topic name is limited to 256 alphanumeric character.
The topic name must be unique with in the AWS account.
Each topic is assigned an AWS ARN once it gets created.
A topic can support subscribers and notification deliveries over multiple protocols.
Message/ request published to a single topic can be delivered over multiple protocols as configured when creating each subscriber.
Delivery format/ transport protocols (endpoints.), SMS, e-mail, email: JSON –for applications, HTTP/ HTTPS, SQS, AWS lambda.
When using Amazon SNS, you (as the owner) create a topic and control access to it by defining access policies that determine which publishers and subscribers can communicate with the topic.
Instead of including a specific destination address in each message to topics that they have created or to topics they have permission to publish to.
Amazon SNS matches the topic to a list of subscribers who have subscribed to that topic and delivers the message to each of these subscriber.
Each topic has a unique name that identifies the Amazon SNS endpoint for publishers to past message and subscribers to register for notification.
Subscriber receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same message.

By default only the topic owner (who created it) can publish to the SNS topic.
The owner can set/ change permission to one of more users (with valid AWS ID) to publish to his topic.
Only the owner of the topic can grant/ change permission for the topic.
Subscriber can be those with/ without AWS ID, only subscriber with AWS ID can request subscription.
Both publishers and subscriber can use SSL to help secure the channel to send and receive message.
Supported push notification platforms:

- Amazon Device Messaging
- Apple Push Notification Service.
- Google Cloud Messaging
- Windows Push Notification Service
- Baidu Cloud Push for Android

SNS topic can have subscribers from any supported push notification platforms as well as any other endpoint type such as SMS or Email.
When you publish a notification to a topic SNS will send identical copies of that, message to each endpoint subscribed to the topic.

Amazon SNS Alternatives:

a. Amazon Kinesis Data Stream
b. Amazon Managed Queue Service (AWS MQS)
c. Apache Kafka
d. Twilio
e. Pusher

Amazon SNS Pricing:

a. Publish action: each 64kb of request payload count as one request. So, 256kb payload will charged as four request.
b. Mobile push notification: for e.g: $0.50/ million request
c. SMS: price depends on country
d. E-mail: $2/100,000
e. HTTP/s notification : $ 0.60/ million
f. SQS and lambda calls are free. These are charged at SQS and lambda rates.
g. Data Transfer

LAMBDA

AWS lambda is a compute service that lets you run the code without provisioning or managing servers.
With AWS lambda, you can run code for virtually any type of application or backend service- all with zero administration.
AWS lambda manages all the administration it manages:

- Provisioning and capacity of the compute fleet that offers a balance of memory, CPU, network and other resources.
- Server and O.S maintenance
- High availability and automatic scaling
- Monitoring fleet health
- Applying security patches
- Deploying your code
- Monitoring and logging your lambda functions.
- AWS lambda runs your code on a high-availability compute infrastructure.

AWS lambda runs your code on a high availability compute infrastructure.
AWS lambda executes your code only when needed and scales automatically form a few requests per day to thousands per seconds.
You pay only for the compute time, you consume no charge when your code is not running.
All you need to do is supply your code in the form of one or more lambda functions to AWS lambda, in one of the languages that AWS supports (currently Node.js, java, powershell, C#, Ruby, Python and Go) and the service can run the code on your behalf.
Typically the lifecycle for an AWS lambda based application includes authoring code, deploying code to AWS lambda and then monitoring and troubleshooting.
This is in exchange for flexibility, which means you cannot log into compute instances or customize the operating system of language runtime.
If you do want to manage your own compute, you can use EC2 of Elastic Beanstalk.

How Lambda Works:

First you upload your code to lambda in one or more lambda function.
AWS lambda will then execute the code in your behalf.
After the code is invoked, lambda automatically take care of provisioning and managing the required servers.

Difference between AWS Lambda and EC2:

AWS Lambda:

AWS lambda is a platform-as-a-service.
It supports only limited languages like Node.js, python, java, C#, Ruby, Go and powershell.
Write your code and push the code into AWS lambda.
You cannot log into compute instances, choose customized O.S or language platform.

AWS EC2:

AWS EC2 is an infrastructure—as-a-service.
No environment restrictions, you can run any code or language.
For the first time in EC2, you have to choose the O.S and install all the software required and then push your code in EC2.
You can select variety of O.S, instance types, network and security patches, RAM and CPU etc.

Important Terms Used in Lambda:

a. **Function:** a function is a resource that you can invoke to run your code in AWS lambda. A function has code that processes events and a runtime that passes request and responses between lambda and the function code.

b. **Runtime:** lambda runtimes allows functions in different languages to run in the same base execution environment. The runtime sits in between the lambda service and your function code relying invocation events, context information and responses between the two.

c. **Event:** it is a JSON formatted document that contains data for a function to process.

d. **Event Source/ Trigger:** an AWS service such as Amazon SNS or a custom service that triggers your function and executes its logic.

e. **Downstream Resource:** an AWS service, such as Dynamo DB tables or S3 buckets that your lambda function call once it is triggered.

f. **Concurrency:** number of request that your function is serving in any given time.

When Lambda Triggers:

You can use AWS lambda to run your code in response to :

- Events such as changes to data in an Amazon S3 or an Amazon Dynamo DB table.
- To run your code in response to HTTP request using Amazon API gateway.
- With the capabilities you can use lambda to easily build data processing triggers for AWS services like Amazon S3, and Amazon Dynamo DB process streaming data stored in kinesis or create your own backend that operates at AWS scale, performance and security.

Example of S3:

The user create an object in a bucket.
Amazon S3 invokes your lambda functions using the permission provided by the execution role.
Amazon S3 knows which lambda function to invoke based on the event source mapping that is stored in the bucket notification configuration.

AWS Lambda Function Configuration:

A lambda function consists of code and any associated dependencies.
In addition a lambda function also has configuration information associated with it.
Initially you specify the configuration information when you create a lambda function.
Lambda provides an API for you to update some of the configuration data.

Lambda function configuration information includes the following key elements:

Compute resources that you need you only specify the amount of memory you want to allocate from your lambda function.
AWS lambda allocates CPU power proportional to the memory by using the same ratio as a general purpose Amazon EC2 instance type, such as an M3 type.
You can update the configuration and request additional memory in 64mb increments from 128mb to 3008mb.
Functions larger than 1536mb are allocated multiple CPU threads.

Maximum Execution Timeout:

You pay for the AWS resources that are used to run your lambda function.
To prevent your lambda function from running indefinitely, you specify a timeout.
When the specified timeout is reached, AWS lambda terminates your lambda function.
Default is 3sec and maximum is 9000sec (15min).

IAM ROLE: this is the role that AWS lambda assume when it executes the lambda function on your behalf.

AWS Lambda Function- Services it can access:

Lambda function can access:

- AWS services and non-AWS services.
- AWS services running in AWS VPC (e.g: redshift, elastic cache, RDS instance)
- Non-AWS services running on instances in an AWS VPC.
- AWS lambda run your function code securely with in a VPC by default.
- However to enable your lambda function to access resources inside your private VPC you must provide additional VPC specific configuration information that includes VPC subnet ID and Security group IDs

Different way to invoke Lambda Function:

1. Synchronous invoke (push)
2. Asynchronous invoke (event)
3. Poll-based invoke (pull based)

1. Synchronous Invoke:

Synchronous invoke are the most straight forward way to invoke your lambda function. In this model your functions execute immediately when you perform the lambda invoke API call.
Invocation flag specifies a value of "request-response".
You wait for the function to process the event and return a response.

Here is a list of services that invoke lambda function synchronously:

- Elastic Load Balancer
- Amazon Cognito
- Cloud front
- API Gateway
- Amazon Lex
- Kinesis data firehose

2. Asynchronous Invoke:

For asynchronous invocation, lambda places the event in a queue and returns a success response without additional information.
Lambda queues the event for processing and returns a response immediately.

You can configure lambda to send an invocation record to another service like SQS, SNS, lambda and eventbridge.

Here is a list of services that invoke lambda function asynchronously:

- Amazon S3
- Amazon SNS
- SES
- Cloud watch logs
- Cloud watch events
- AWS code commit
- AWS config

3. **Poll-Based Invoke:**
   The invocation model is designed to allow you to integrate with AWS stream and queue based service with no code or server management lambda will poll the following service on your behalf, retrieve records and invoke your function. The following are supported service:
   - Amazon Kinesis
   - Amazon SQS
   - Amazon Dynamo DB Streams