# AWS Intro

**Cloud Computing**

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider.

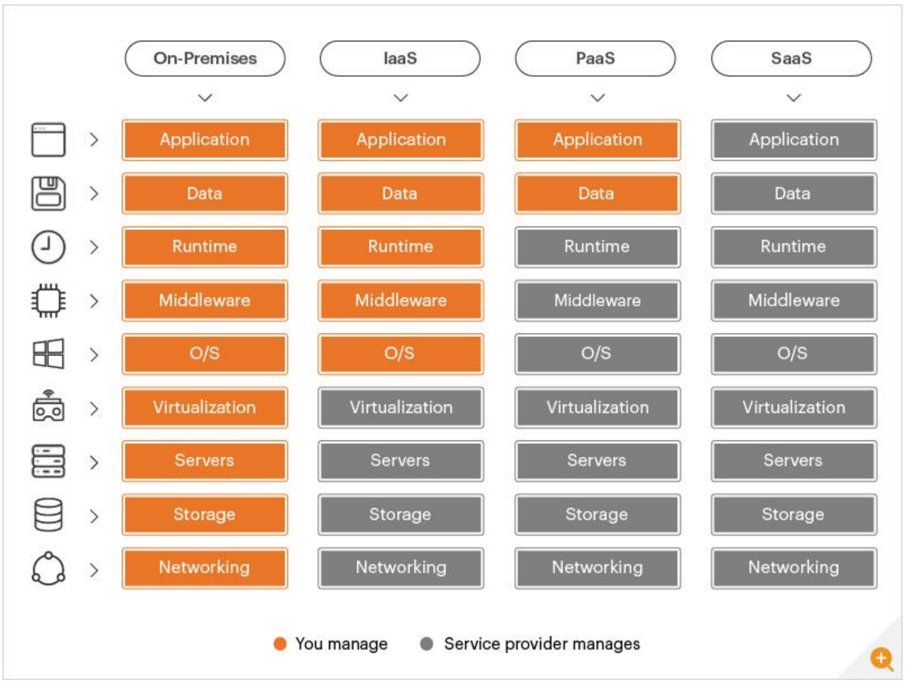
**Top Cloud Providers**

Amazon Web Services (AWS)
Google Cloud Platform (GCP)
Microsoft Azure

**On Prem**

An on-premises data center is a group of servers that you privately own and control.
**"On-prem"** refers to private data centers that companies house in their own facilities and maintain themselves.

**On Prem vs Cloud Models (IaaS, PaaS, SaaS)**



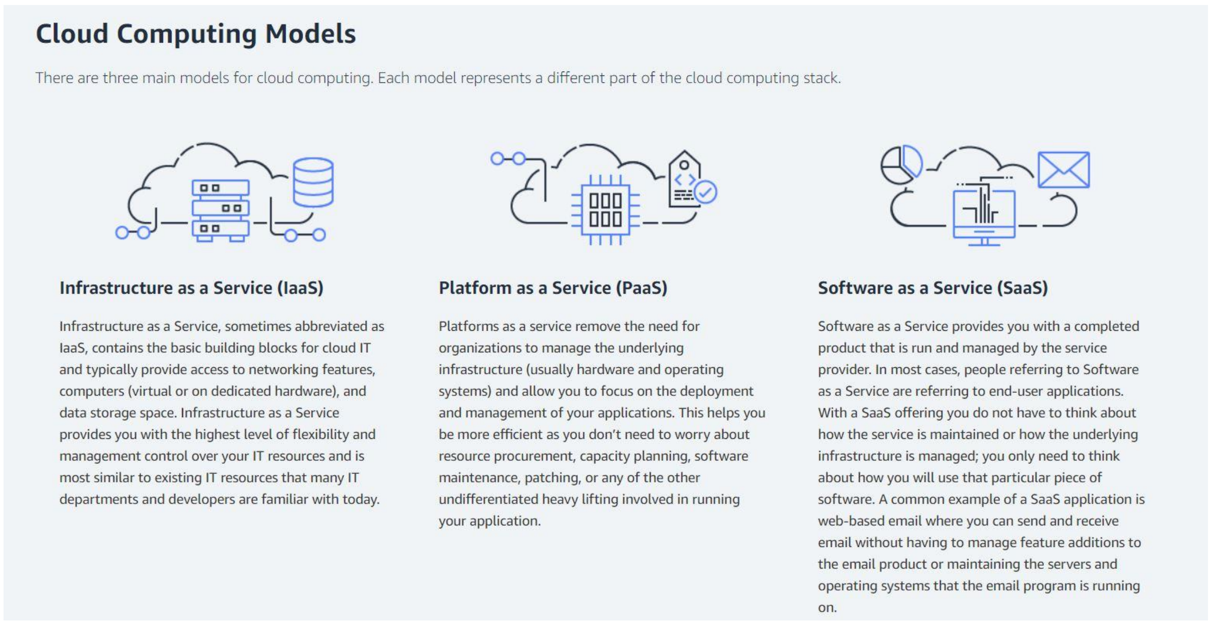
| On-Premises | IaaS | PaaS | SaaS |
|---|---|---|---|
| 🟧 Application | 🟧 Application | 🟧 Application | ▨ Application |
| 🟧 Data | 🟧 Data | 🟧 Data | ▨ Data |
| 🟧 Runtime | 🟧 Runtime | ▨ Runtime | ▨ Runtime |
| 🟧 Middleware | 🟧 Middleware | ▨ Middleware | ▨ Middleware |
| 🟧 O/S | 🟧 O/S | ▨ O/S | ▨ O/S |
| 🟧 Virtualization | ▨ Virtualization | ▨ Virtualization | ▨ Virtualization |
| 🟧 Servers | ▨ Servers | ▨ Servers | ▨ Servers |
| 🟧 Storage | ▨ Storage | ▨ Storage | ▨ Storage |
| 🟧 Networking | ▨ Networking | ▨ Networking | ▨ Networking |

🟧 You manage ▨ Service provider manages

**Infrastructure as a Service** provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.

**Platforms as a Service** remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems) and allow you to focus on the deployment and management of your applications

**Software as a Service** provides you with a completed product that is run and managed by the service provider.


## Cloud Computing Models

There are three main models for cloud computing. Each model represents a different part of the cloud computing stack.

### Infrastructure as a Service (IaaS)

Infrastructure as a Service, sometimes abbreviated as IaaS, contains the basic building blocks for cloud IT and typically provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. Infrastructure as a Service provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.

### Platform as a Service (PaaS)

Platforms as a service remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems) and allow you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

### Software as a Service (SaaS)

Software as a Service provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software. A common example of a SaaS application is web-based email where you can send and receive email without having to manage feature additions to the email product or maintaining the servers and operating systems that the email program is running on.

**Amazon Web Services – What is AWS ?**

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud, offering **over 200** fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.

**AWS Service:** is a collection of resources and tools that AWS provides to users to accomplish specific tasks or functions in the cloud.

**AWS Resources:** It refers to any computing or storage component that is used to build and deploy applications in the cloud. This can include virtual machines, databases, load balancers, network interfaces, and many other components that are managed by AWS.

**Example: EC2** is an AWS service that provides virtual machines as resources, and **S3** is an AWS service that provides object storage as resources.

We can Access AWS cloud using
a) AWS Console
b) AWS CLI
c) AWS SDKs

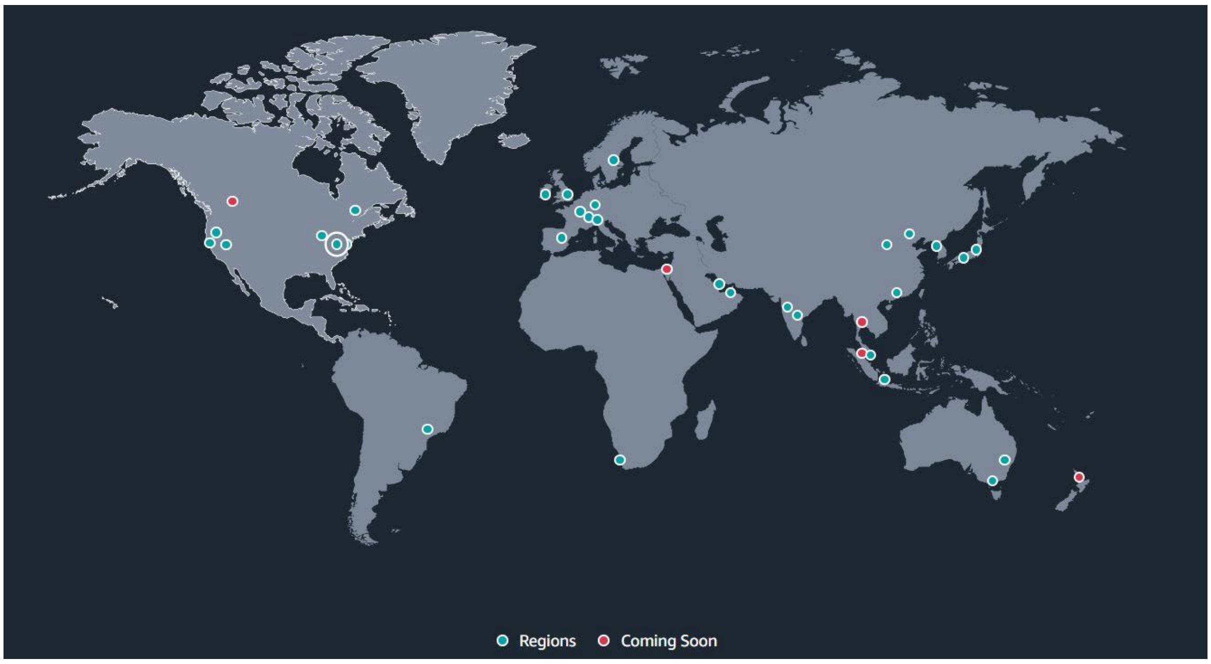
**AWS Region and Availability Zone**

**AWS Region** is a separate geographic area where we cluster data centers. Each Region is designed to be isolated from the other Regions. This achieves the greatest possible fault tolerance and stability.

When you view your resources, you see only the resources that are tied to the Region that you specified.

**Availability Zones** are multiple, isolated locations within each Region. The code for Availability Zone is its Region code followed by a letter identifier. For example, us-east-1a.

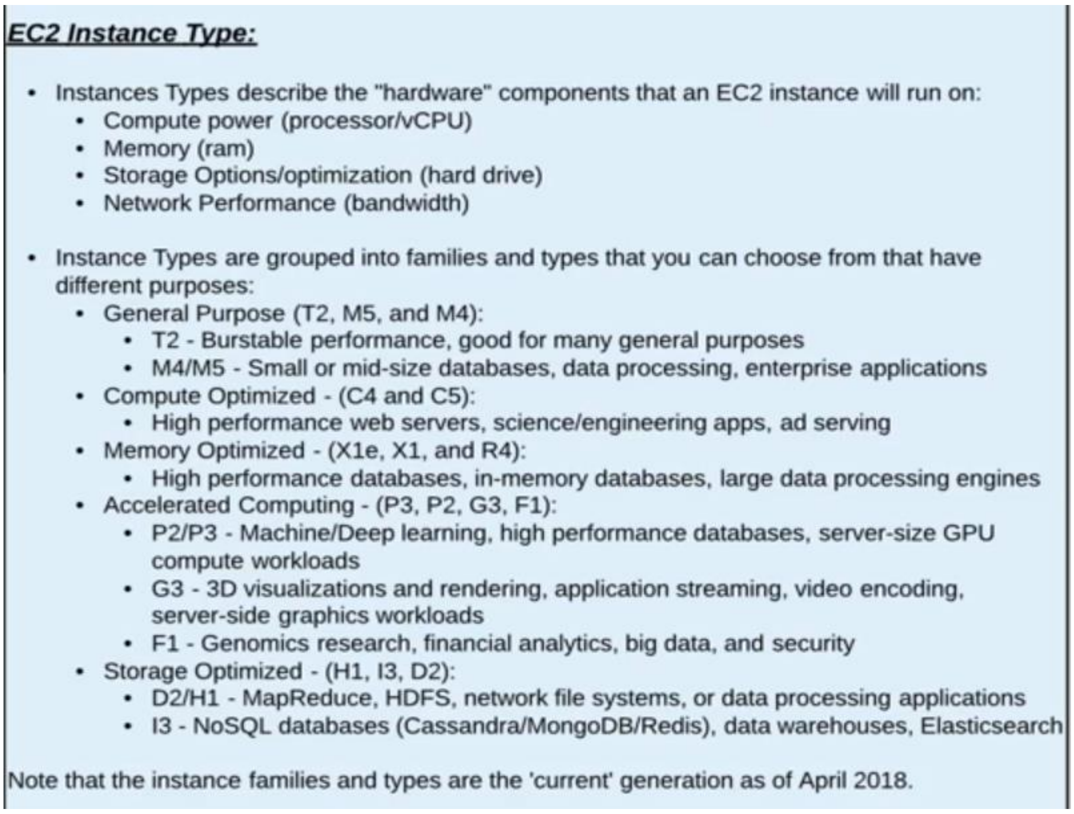
Each Region will have a minimum of 3 separate AZ.



## Amazon EC2 (Elastic Compute Cloud)

**EC2 Instance Types:**

> General Purpose > Compute Optimized > Memory Optimized > Storage Optimized



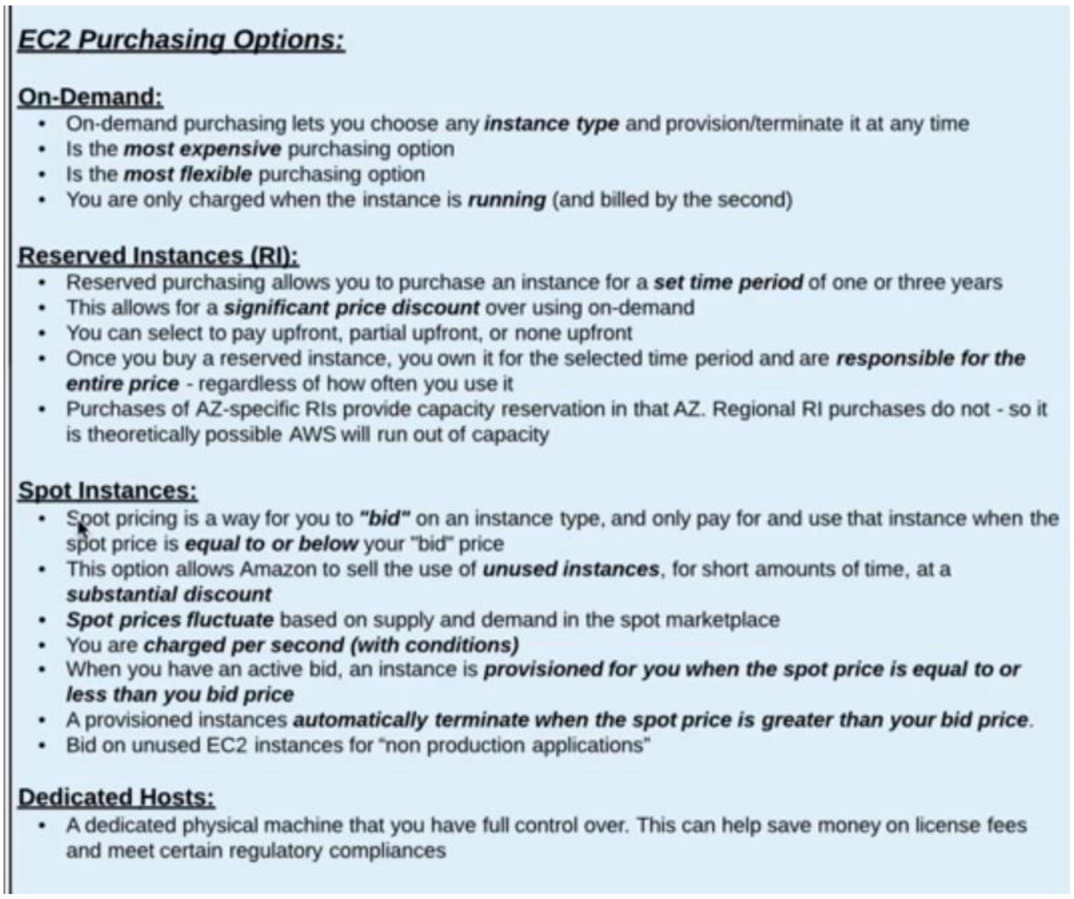
***EC2 Instance Type:***

- Instances Types describe the "hardware" components that an EC2 instance will run on:
  - Compute power (processor/vCPU)
  - Memory (ram)
  - Storage Options/optimization (hard drive)
  - Network Performance (bandwidth)

- Instance Types are grouped into families and types that you can choose from that have different purposes:
  - General Purpose (T2, M5, and M4):
    - T2 - Burstable performance, good for many general purposes
    - M4/M5 - Small or mid-size databases, data processing, enterprise applications
  - Compute Optimized - (C4 and C5):
    - High performance web servers, science/engineering apps, ad serving
  - Memory Optimized - (X1e, X1, and R4):
    - High performance databases, in-memory databases, large data processing engines
  - Accelerated Computing - (P3, P2, G3, F1):
    - P2/P3 - Machine/Deep learning, high performance databases, server-size GPU compute workloads
    - G3 - 3D visualizations and rendering, application streaming, video encoding, server-side graphics workloads
    - F1 - Genomics research, financial analytics, big data, and security
  - Storage Optimized - (H1, I3, D2):
    - D2/H1 - MapReduce, HDFS, network file systems, or data processing applications
    - I3 - NoSQL databases (Cassandra/MongoDB/Redis), data warehouses, Elasticsearch

Note that the instance families and types are the 'current' generation as of April 2018.

**EC2 Purchasing Options and Cost Saving:**



***EC2 Purchasing Options:***

**On-Demand:**

- On-demand purchasing lets you choose any ***instance type*** and provision/terminate it at any time
- Is the ***most expensive*** purchasing option
- Is the ***most flexible*** purchasing option
- You are only charged when the instance is ***running*** (and billed by the second)

**Reserved Instances (RI):**

- Reserved purchasing allows you to purchase an instance for a ***set time period*** of one or three years
- This allows for a ***significant price discount*** over using on-demand
- You can select to pay upfront, partial upfront, or none upfront
- Once you buy a reserved instance, you own it for the selected time period and are ***responsible for the entire price*** - regardless of how often you use it
- Purchases of AZ-specific RIs provide capacity reservation in that AZ. Regional RI purchases do not - so it is theoretically possible AWS will run out of capacity

**Spot Instances:**

- Spot pricing is a way for you to ***"bid"*** on an instance type, and only pay for and use that instance when the spot price is ***equal to or below*** your "bid" price
- This option allows Amazon to sell the use of ***unused instances***, for short amounts of time, at a ***substantial discount***
- ***Spot prices fluctuate*** based on supply and demand in the spot marketplace
- You are ***charged per second (with conditions)***
- When you have an active bid, an instance is ***provisioned for you when the spot price is equal to or less than you bid price***
- A provisioned instances ***automatically terminate when the spot price is greater than your bid price***.
- Bid on unused EC2 instances for "non production applications"

**Dedicated Hosts:**

- A dedicated physical machine that you have full control over. This can help save money on license fees and meet certain regulatory compliances

# Amazon IAM (Identity and Access Management)

IAM provides access to AWS accounts and services where we can manage Users, Groups, Roles and Policies.

It is a global service. Anything you create/change applies **globally** to all AWS regions.

**IAM Users** - We create users and assign necessary permissions to them in the form of policies.

**IAM Groups** - We can create groups for ex. Developers, Administrators, Testers/QA etc and attach policies at the group level.

**IAM Roles** - These are entities used to create and assign permissions to allow users/resources to perform actions

**IAM Policy** defines permissions for users/roles to perform an action/operation on AWS cloud

**Types of IAM Policy: (Reference: link)**

1. **Identity-based Policies:** Applicable on users, groups of users, and roles
    - AWS Managed policy
    - Custom Managed Policy
    - Inline Policy
2. **Resource-based policies:** Attach to a resource such as an Amazon S3 bucket
3. **Session Policies:** create a temporary session for a role or federated user
4. There are also Permissions Boundaries, Organizations SCPs and Access control lists (ACLs)

**Note:** More than one policy can be attached to a user or a group at the same time.
Policies can't be attached directly to resources like EC2 instance, S3 bucket etc.

## Amazon Machine Image (AMI):

- An Amazon Machine Image (AMI) provides the information required to launch an instance.
- You can launch multiple instances from a single AMI when you need multiple instances with the same configuration.
- You must specify an AMI whenever you launch an instance.

### Custom AMI

- You can launch an instance from an existing AMI, customize the instance by installing/uninstalling packages on the instance, and then save this updated configuration as a custom AMI. Instances launched from this new custom AMI include the customizations that you made when you created the AMI.
- AMI can be shared across different accounts without making it public, but in the same region. If the Custom AMI is required in another region, it needs to be copied and then shared again. (Reference)

### Difference between EBS Snapshot and AMI

An **EBS snapshot** is a backup of a single EBS volume. The EBS snapshot contains all the data stored on the EBS volume at the time the EBS snapshot was created.

An **AMI** image is a backup of an entire EC2 instance. Associated with an AMI image are EBS snapshots. Those EBS snapshots are the backups of the individual EBS volumes attached to the EC2 instance at the time the AMI image was created.

## Amazon EFS (Elastic File System):

- Amazon Elastic file system is a regional service storing data within and across multiple Availability Zones (AZs) for high availability and durability
- Amazon EFS is an NFS file system service offered by AWS.
- An Amazon EFS file system is excellent as a managed network file system that can be shared across different Amazon EC2 instances.
- Amazon EFS works like NAS devices and performs well for big data analytics, media processing workflows, and content management.

- EFS can be accessed by multiple instances at a time through NFS protocol. It is used as a clustered database and document sharing.

**Benefits of EFS:**

- With EFS you need not worry about managing file servers or storage, updating hardware, configuring software, or performing backups as EFS is a fully managed service
- The distributed architecture of Amazon EFS provides data protection from an AZ outage, system and component failures, and network connection errors.
- Network Access to the files can be controlled using Virtual Private Cloud security group rules and with Identity Access Management policies and EFS access points you can control the access to your files
- Amazon EFS is designed to provide the throughput, IOPS, and low latency needed for a broad range of workloads
- With Amazon EFS, storage capacity is elastic, growing and shrinking automatically as you add and remove files, dynamically providing the storage capacity to applications as needed.
- AWS EFS provides encryption of data both at rest and in transit so that your data is secure.

**EFS Access Points:** (Reference: Access Points)

- Access Points are application-specific entry points into an EFS file system that make it easier to manage application access to shared datasets.
- Access points can enforce a user identity, including the user's POSIX groups
- Access points can also enforce a different root directory for the file system so that clients can only access data in the specified directory or its subdirectories.

**NOTE:** We can mount and access EFS volumes across VPCs, i.e., EFS in VPC1 can be accessed from an ec2 instance running in VPC2, provided **both the VPCs are connected** using VPC peering or Transit Gateway. (Reference: Mount a File System from a Different VPC)

# Amazon S3 (Simple storage service)

- Amazon S3 is an object storage service. It is a scalable, high-speed, web-based cloud storage service.
- It has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web.
- S3 provides 99.999999999% durability for objects stored in the service and supports multiple security and compliance certifications.

**Single operation upload:**

- It’s a traditional upload where you will upload the object in one part
- A single operation upload can upload the file up to 5GB in size.

## Upload object in parts:

- Using multipart upload, you can upload the large objects up to 5TB.
- You can use multipart upload for the objects from 5MB to 5TB in size.

## Rules for bucket naming:

- Bucket names must be between 3 and 63 characters long.
- Bucket names can consist only of lowercase letters, numbers, dots (.) and hyphens (-)
- Bucket names must begin and end with a letter or number.
- Bucket names must not be formatted as an IP address (for example, 192.168.5.4).
- Bucket names can't begin with xn-- (for buckets created after February 2020).

## Limitation of S3 bucket:

- Only 100 buckets can be created per account.
- Can hold unlimited objects

## S3 Storage classes: (Reference: https://aws.amazon.com/s3/storage-classes/)

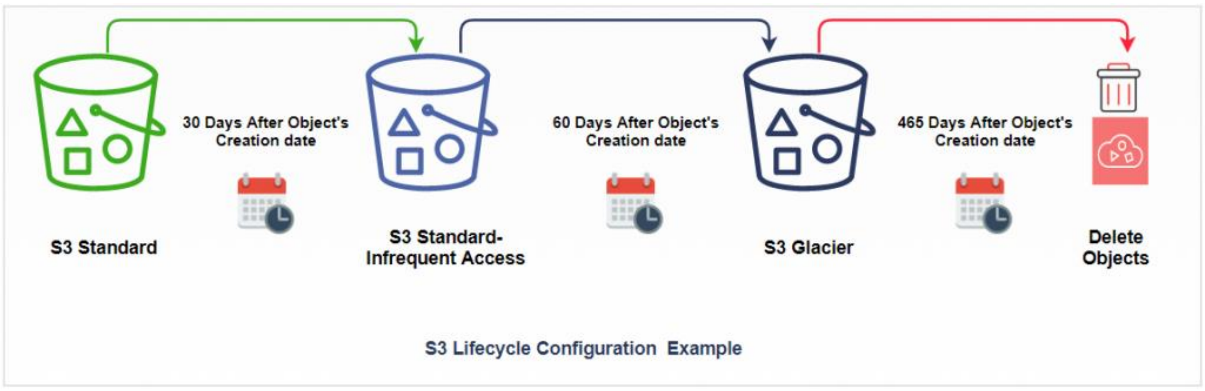
- Standard
- S3 Intelligent-Tiering
- S3 Standard-IA: Infrequent Access
- S3 One Zone-IA
- Glacier
  - Glacier Instant Retrieval
  - Glacier Flexible Retrieval
  - Glacier Deep Archive
- S3 Outposts

## S3 Lifecycle policy:

- An object lifecycle policy is a set of rules that automate the migration of the object storage class to different storage class
- By default, lifecycle policies are disabled for a bucket
- Lifecycle Management can be applied to both current and previous versions.
- It can be used either in conjunction with the versioning or without versioning.
- There are 2 types of actions:

**Transition actions:** Moving objects from one storage class to another storage class. Each storage class has a different cost associated with it.

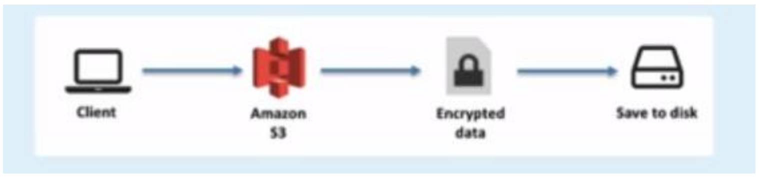
**Expiration actions:** When objects expire after a span of time (say 30 days,60 days, etc). Amazon S3 deletes expired objects on your behalf.


30 Days After Object's Creation date
60 Days After Object's Creation date
465 Days After Object's Creation date
S3 Standard
S3 Standard-Infrequent Access
S3 Glacier
Delete Objects
S3 Lifecycle Configuration Example

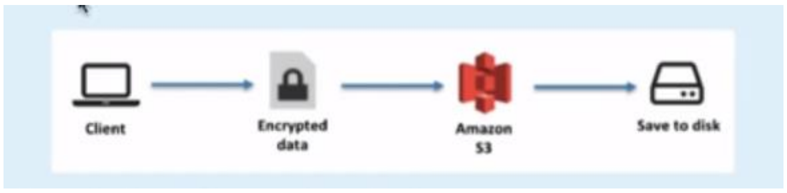
## <u>S3 Encryption:</u>

**Two ways of protecting information while transferring data with S3**

1. **Server side/ At rest:**



2. **In-transit/Client-side encryption:**
   (a) Using KMS or (b) Using Client side Master key



### S3 Bucket Versioning

Buckets can be in one of three states:

(a) Unversioned (the default)
(b) Versioning-enabled
(c) Versioning-suspended

- You can enable and suspend versioning at the bucket level. After you enable versioning on a bucket, it can never return to an unversioned state. But you can *suspend* versioning on that bucket.
- Objects that are stored in your bucket before you set the versioning state have a version ID of **null**. When you enable versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles the objects in future requests. (Reference: Working with objects in a versioning-enabled bucket)

- The bucket owner (or any user with appropriate permissions) can suspend versioning to stop accruing object versions.
  When you suspend versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles objects in future requests. (Reference: Working with objects in a versioning-suspended bucket)

**S3 Bucket Policy** (Reference: Bucket Policy)

- A bucket policy is a resource-based policy that you can use to grant access permissions to your Amazon S3 bucket and the objects in it.
- Only the bucket owner can associate a policy with a bucket.
- The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. These permissions do not apply to objects that are owned by other AWS accounts.
- Bucket policies use JSON-based IAM policy language.
- You can use bucket policies to add or deny permissions for the objects in a bucket. You can allow or deny requests based on the elements in the policy.

**Difference between EBS v/s EFS v/s S3**

| AMAZON S3 | AMAZON EBS | AMAZON EFS |
|---|---|---|
| Can be publicly accessible<br>Web interface<br>Object Storage<br>Scalable<br>Slower than EBS and EFS | Accessible only via the given EC2 Machine<br>File System interface<br>Block Storage<br>Hardly scalable<br>Faster than S3 and EFS | Accessible via several EC2 machines and AWS services<br>Web and file system interface<br>Object storage<br>Scalable<br>Faster than S3, slower than EBS |
| Good for storing backups and other static data | Is meant to be EC2 drive | Good for applications and shareable workloads |

## AWS CLI

The AWS Command Line Interface (**AWS CLI**) is an open source tool that enables you to interact with AWS services using commands in your command-line shell. With minimal configuration, the AWS CLI enables you to start running commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from the command prompt in your terminal program:

Linux shells – Use common shell programs such as bash, zsh, and tcsh to run commands in Linux or macOS.

Windows command line – On Windows, run commands at the Windows command prompt or in PowerShell.

Remotely – Run commands on Amazon Elastic Compute Cloud (Amazon EC2) instances through a remote terminal program such as PuTTY or SSH, or with AWS Systems Manager.

**Install** or **Update** AWS CLI by following the link here.

### STS – Getting Session Token for temporary access to AWS accounts

The AWS Security Token Service (**STS**) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).

Run the following commands to configure AWS CLI and get session token

```
aws configure
```

        <Enter Access Key Id and Secret Access Key, Region and Output>

```
aws sts get-session-token
```

```
{
   "Credentials": {
      "AccessKeyId": "ASIATFOKVBZQFMBZYVGJ",
      "SecretAccessKey": "mN7FUNC4RyGWs9sqYR1HBJ189VRgIU4D1lO6vSh8",
      "SessionToken":
"IQoJb3JpZ2luX2VjEIP//////////wEaCmFwLXNvdXRoLTEiSDBGAiEAlEemnNdZOFEOybdJOiCf8PhHffaV27H
mVPKmJr0hyjcCIQDSxW8lG3xUuGK0FZG+s+DB3SDrNx2sHTSFNohBNkCnfir0AQjs//////////8BEAAaDDIxN
zg1NzcyNDAwMCIMH6vajQD0VDC/NB3VKsgBznws8M43A3Dq/cRtmruDKNY6rO/oNcz5fTnBQt7o8RcDA
DNjSFHsCMXg4lk+iGvSqDjdAhR7a4T21ifMUquYNjszu81olhpOORuIxyRWLc1TMskhdKm68wkjWXalbIUW
y4mVD54nQ7Ww4mbkgsNHkHlLw0sHZCAKFKAa12dBQUz8AgbCjoajuSx8cXbvPaZHNEgsgv6+rIL713iodb
4j2Vb0Cw1XvL85BD715kxU5kBIAjM1a4+pu+B89hORaxz/UDlUZqOwvzkwrteKpQY6lwE2NCOqIKsuxlhhaio
5UPCY+JpdMIyyNvjgIrcsldHIw+jYfQ5NFwPxzv/76+2NrnoLQtknQfiFbgSBABpHUFo6qoAf6Ty6SBzScihyHu
exZOa5Sb5ntPbc5ZVwIr+kplFrYhnCkjatGoz542JF9UzB00N82gkGzySZit4baihzgXRwV6g7nnsoWE1f5+Npnhl
3JocOHVln",
      "Expiration": "2023-07-03T23:06:22+00:00"
   }
}
```

**Assume Role:**

```
aws sts assume-role
```

Returns a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to. These temporary credentials consist of an access key ID, a secret access key, and a security token.

**Example using STS and Assuming Role to query AWS resources:**

1. create a IAM user

2. Add ec2 full access policy

3. Try to list the bucket (aws s3 list-buckets) - you can't list the bucket because the user is not having the permission to list bucket (access denied)

4. Grant the user to assume a role

   Create a role

   Attach a policy s3 full access policy

   Edit the role under  Trust Relationship, modify the policy to user instead of ec2

```
"Principal": {
  "Service": "ec2.amazonaws.com"
}
```

   to

```
"Principal": {
  "AWS": "arn:aws:iam::217857724000:user/test-user-cli"
}
```

5. aws sts assume-role --role-arn <enter_role_arn> --role-session-name s3-access --duration-seconds 3600

6. copy the Accesskey, Secretkey and Session token (Reference: Linux or Windows)

7 export AWS_ACCESS_KEY_ID=<enter the copied access key>

   export AWS_SECRET_ACCESS_KEY=<enter the copied session key>

   export AWS_SESSION_TOKEN=<sessiontoken>

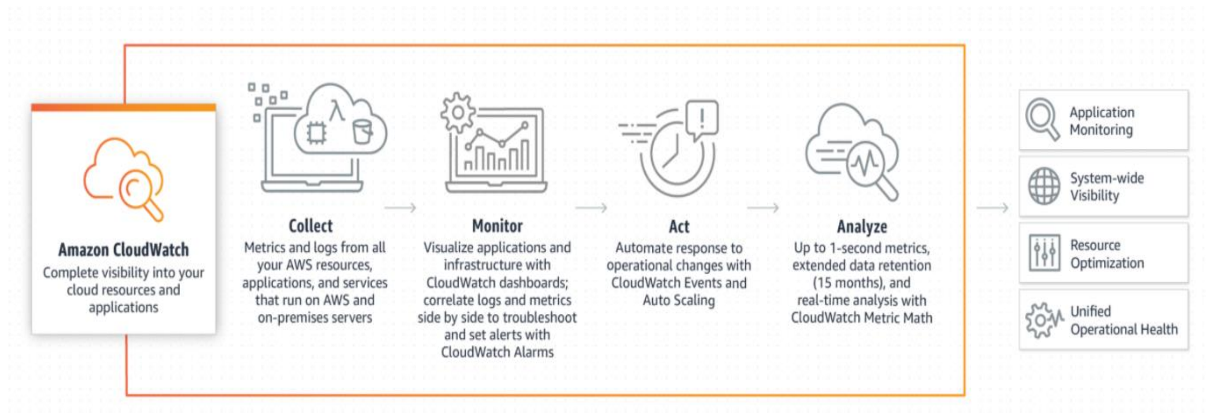
8. aws s3 ls **OR** aws s3api list-objects --bucket <bucket_name>

## AWS CloudWatch

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time.

Default metrics of EC2 instance: Network usage, CPU Usage, Storage

We can install CloudWatch agent (installation guide) on the servers in case we need system level data like Storage, Memory, Application data etc.


Amazon CloudWatch
Complete visibility into your cloud resources and applications
Collect
Metrics and logs from all your AWS resources, applications, and services that run on AWS and on-premises servers
Monitor
Visualize applications and infrastructure with CloudWatch dashboards; correlate logs and metrics side by side to troubleshoot and set alerts with CloudWatch Alarms
Act
Automate response to operational changes with CloudWatch Events and Auto Scaling
Analyze
Up to 1-second metrics, extended data retention (15 months), and real-time analysis with CloudWatch Metric Math
Application Monitoring
System-wide Visibility
Resource Optimization
Unified Operational Health

**CW Metrics:**

Metrics are data about the performance of your systems/resources.

Metric data is kept for **15 months**

Basic monitoring: polls for every 5 minutes

Detailed monitoring: polls for every 1 minute.

**CW Alarm:**

CloudWatch Alarms feature allows you to watch CloudWatch metrics and to receive notifications when the metrics fall outside of the levels (high or low thresholds) that you configure and take action accordingly.

**Example:** If CPU utilization goes beyond the static threshold alarm goes to alarm state and triggers **Alarm Notification**. We can also configure actions **Autoscaling action, EC2 action or Systems Manager action**.

**Steps to create the alarm**

- Go to create alarm
- Select the metric
- Set the condition
- Select the type of alarm
  - **In alarm** - The metric or expression is outside of the defined threshold.
  - **Ok** - The metric or expression is within the defined threshold.
  - **Insufficient data** - The alarm has just started or not enough data is available.
- Select SNS topic if already created. Or create a new topic.
- If you have chosen, EC2 metric, then you can select Autoscaling action or EC2 action
- Then give a name to the alarm and create alarm.

**CW Event:**

A CloudWatch Event indicates a change in AWS environment or A change in the state of AWS resource.
**Steps to create an event**

- Create rule – Give a name to the rule and choose event pattern
- Choose Event source (ex: AWS service)
- Choose event to watch for that service (ex: EC2 instance state-change notification)
- Add target – SNS topic or Lambda function or CodePipeline
- Add relevant tags (optional) and Create

**CW Logs:**

Amazon CloudWatch Logs service allows you to collect and store logs from your resources, applications, and services in near real-time.
**Steps to create logs**

- Give a log group name
- Choose retention setting

# AWS CloudTrail

AWS CloudTrail tracks the activities of users and APIs. It continuously logs your AWS account activity

- Auditing tool that records all account activity.
- Any action taken by users, roles and AWS services are recorded to cloud trial.
- Cloud trial events are kept for **90 days** in event history.
- You can create a trail of your own to store the event history in s3 bucket.

**Working of Cloud Trail**

- **Capture** - Record activity in AWS services as AWS CloudTrail events
- **Store** - AWS CloudTrail delivers events to the AWS CloudTrail console, Amazon S3 buckets, and optionally Amazon CloudWatch Logs
- **Act** - Use Amazon CloudWatch Alarms and Events to take action when important events are detected
- **Review** - View recent events in the AWS CloudTrail console, or analyze log files with Amazon Athena

**Steps to create trails**

- Create trial
- Specify the trial name
- Choose storage location - s3 bucket (either create new bucket or use existing)
- Choose the encryption method - Server side or custom managed encryption
- If you want to enable the cloudwatch alarm, you can select that.
- Choose the types of events.
- Review and create.

There are three types of events –

**1. Data events** - Data events provide visibility into the resource operations performed on or within a resource.
Example – s3, lambda, dynamodb.

- Amazon S3 object-level API activity (for example, GetObject, DeleteObject, and PutObject API operations)
- AWS Lambda function execution activity (the Invoke API)
- Amazon DynamoDB object-level API activity on tables (for example, PutItem, DeleteItem, and UpdateItem API operations).

**2. Management events** - Management events provide visibility into management operations that are performed on resources in your AWS account.

- Configuring security (for example, IAM AttachRolePolicy API operations)
- Registering devices (for example, Amazon EC2 CreateDefaultVpc API operations)
- Configuring rules for routing data (for example, Amazon EC2 CreateSubnet API operations)
- Setting up logging (for example, AWS CloudTrail CreateTrail API operations)

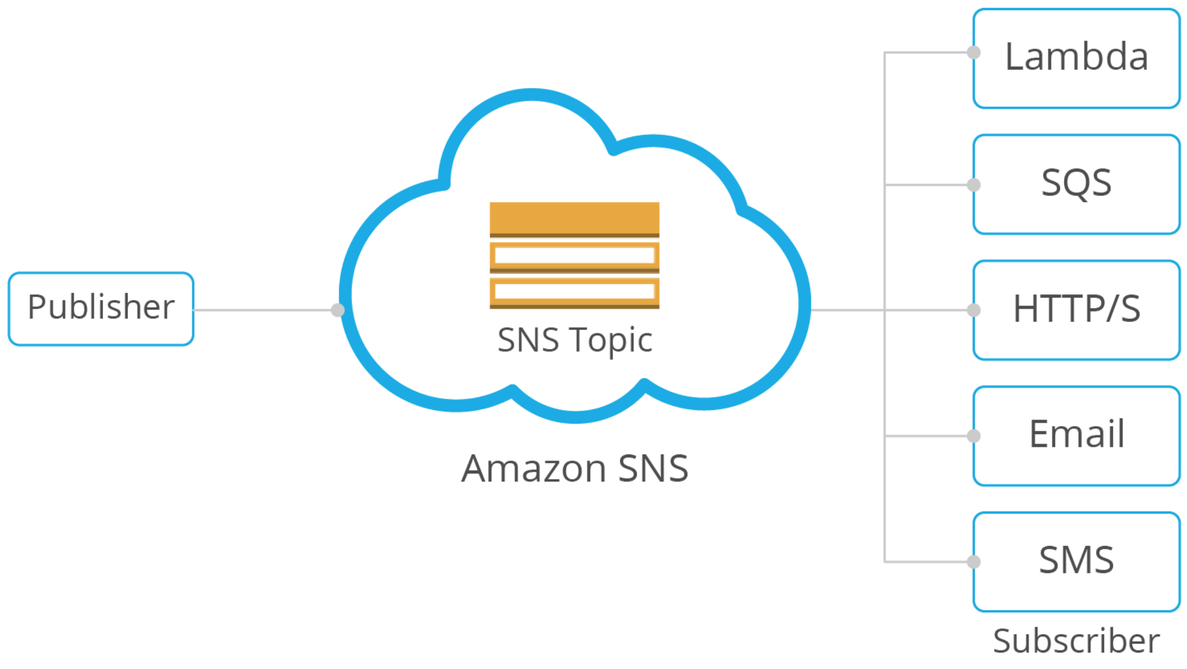
**3. Insight events** - AWS users identify and respond to unusual activity associated with write API calls.

CloudTrail Insights measures your normal patterns of API call volume, also called the baseline, and generates Insights events when the volume is outside normal patterns. Insights events are generated for write management APIs.

## AWS Simple Notification Service (SNS)

Amazon Simple Notification Service is a notification service provided as part of Amazon Web Service.

It provides a low-cost infrastructure for the mass delivery of messages, predominantly to mobile users

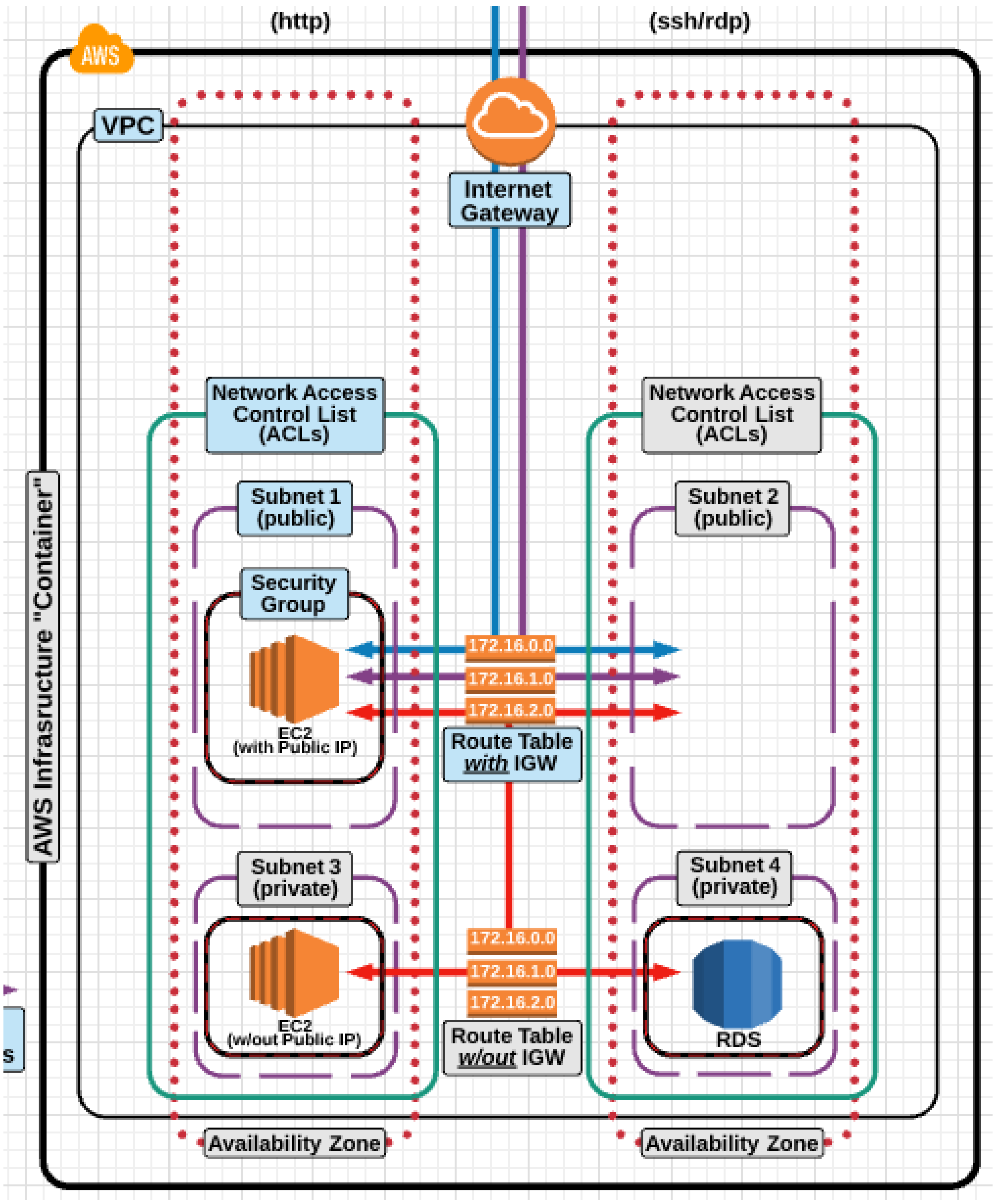


**Topic:**
An Amazon SNS topic is a logical access point that acts as a communication channel.
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

## Virtual Private Cloud (VPC)

Amazon Virtual Private Cloud is a commercial cloud computing service that provides users a virtual private cloud, by provision[ing] a logically isolated section of Amazon Web Services Cloud.

Amazon VPC (Reference: VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define like EC2 instance, Databases, lambda function.



**CIDR**
Classless Inter-Domain Routing - When you create a VPC, you must specify a range of IPv4 addresses for the VPC in the form of CIDR. For example, 10.0.10.0/22 (consists of 1024 ips). **This is the primary CIDR block for your VPC**.

Classless Inter-Domain Routing is a method for allocating IP addresses and for IP routing.

**Subnet**
Subnet is a logical subdivision of an IP network. It is a range of IP-address.

- A subnet is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet.
- Some IP addresses are reserved for a subnet with CIDR block 10.0.10.0/24 are
  - 10.0.10.0 Network address
  - 10.0.10.1 VPC Router

    - 10.0.10.2 DNS server (DNS. (Domain Name System) The Internet's system for converting alphabetic names into numeric IP addresses)
    - 10.0.10.3 Future use
    - 10.0.10.255 N/W Broadcast address
- VPC spans multiple Availability zones.
- Subnets must be associated with route table
    - A public subnet has a route to internet
    - A private subnet doesn't have route to internet. It creates higher level of security.
- You can use a network address translation (NAT) 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.

**Route Table**
It is a interface which connects IGW, NAT, subnets in the VPC. It helps to configure route by specifying gateway and associate specific subnets to the table.

**VPC Quota or VPC limitations:**
- 5 VPC per region
- 5 IGW per region
- Subnet per VPC 200
- IPv4 CIDR blocks per VPC 4
- Elastic IP addresses per Region 5
- Internet gateways per Region 5
- NAT gateways per Availability Zone 5
- Network ACLs per VPC 200
- Rules per network ACL 200

**Internet Gateway** – It is a gateway which is connected to the VPC to route traffic to and from the internet.

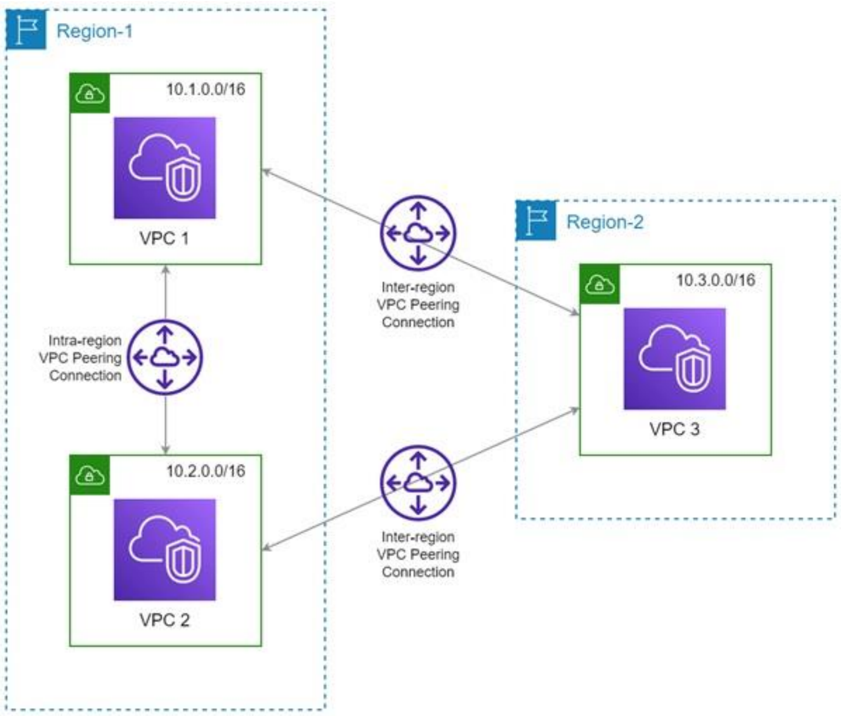
**NAT Gateway –** It is a gateway through which resources in the private subnet can get access to the internet. A NAT Gateway is a device used to enable instances in a private subnet to connect to the internet or other AWS services.

**Egress only Internet Gateway**
It allows outbound communication over IPv6 from instances in your VPC to the internet, and prevents the internet from initiating an IPv6 connection with your instances.

## VPC Peering

A VPC peering connection is a networking connection between two VPCs 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 VPCs, or with a VPC in another AWS account.
The VPCs can be in different regions (also known as an inter-region VPC peering connection).


Region-1
10.1.0.0/16
VPC 1
Intra-region VPC Peering Connection
10.2.0.0/16
VPC 2
Inter-region VPC Peering Connection
Inter-region VPC Peering Connection
Region-2
10.3.0.0/16
VPC 3

**Conditions:**

- CIDR block shouldn't overlap
- Transitive peering relationships are not supported. i.e here VPC B cannot connect with VPC C.
- If the VPCs are in different regions, inter-region data transfer costs apply.
- You cannot have more than one VPC peering connection between the same two VPCs at the same time.

**Security Groups**
It is a stateful service which acts like a firewall at the instance to route inbound and outbound traffic by defining inbound and outbound rules. Security groups are stateful — if you send a request from your instance, the response traffic for that request is allowed to flow in regardless of inbound security group rules.

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic

**Network ACL**
Network ACL is a stateless service work as firewall to the subnet which route inbound and outbound traffic to the subnet by configuring inbound and outbound rules. Network ACLs are stateless, which means that responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

It 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. (Firewall at subnet level)

• Inbound means – incoming (ingress)
• Outbound means – outgoing (egress)
• Always explicit deny take precedence over allow

**Elastic IP - (Permanent IP address)**
An Elastic IP address is a static, public IPv4 address designed for dynamic cloud computing. You can associate an Elastic IP address with any instance or network interface in any VPC in your account.

With an Elastic IP address, you can mask the failure of an instance by rapidly remapping the address to another instance in your VPC.

**Bastion Host or Jump Server**
BASTION host are EC2 instances present in the public subnet and connected to the private instance in the private subnet inorder to route traffic from external network to the private instance. It is used to communicate instances in the private subnet.

**VPN – Virtual Private Network**
It is a tunnel established to the private network to access resources using private IP address.

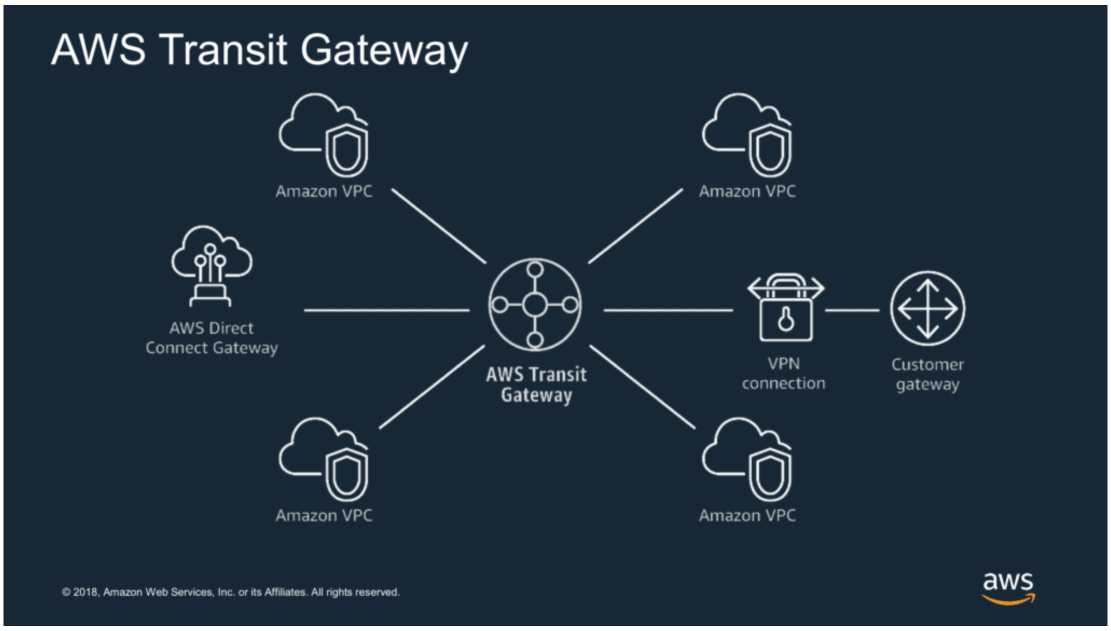
There are two types of VPN
1. **Site-to-site VPN** - A secure connection between your on-premises equipment and your VPCs.
2. **Client VPN** - is a managed client-based VPN service that enables you to securely access AWS resources and resources in your on-premises network.

**Firewall**
A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules.

## Transit Gateway

The Transit Gateway (reference) is a managed service from AWS that acts as a hub interconnecting VPCs and VPN connections within a single region.

Transit Gateway is Highly Scalable. It can support bandwidths up to 50 Gbps between it and each VPC attachment. And, each Transit Gateway supports up to 5,000 VPCs and 10,000 routes



**Advantages of TGW**

- Simplified management of VPC connections. Each spoke VPC only needs to connect to the TGW to gain access to other connected VPCs.
- Supports more VPCs compared to VPC peering.

- TGW Route Tables per attachment allow for fine-grained routing.

**Disadvantages**

- Additional hop introduces some latency.
- Extra cost of hourly charge per attachment in addition to data fees.

**Limitations**

- You can connect to a maximum of three Transit Gateways over a single Direct Connect Connection for hybrid connectivity.
- Transit Gateway doesn't support routing between VPCs with overlapping CIDRs.

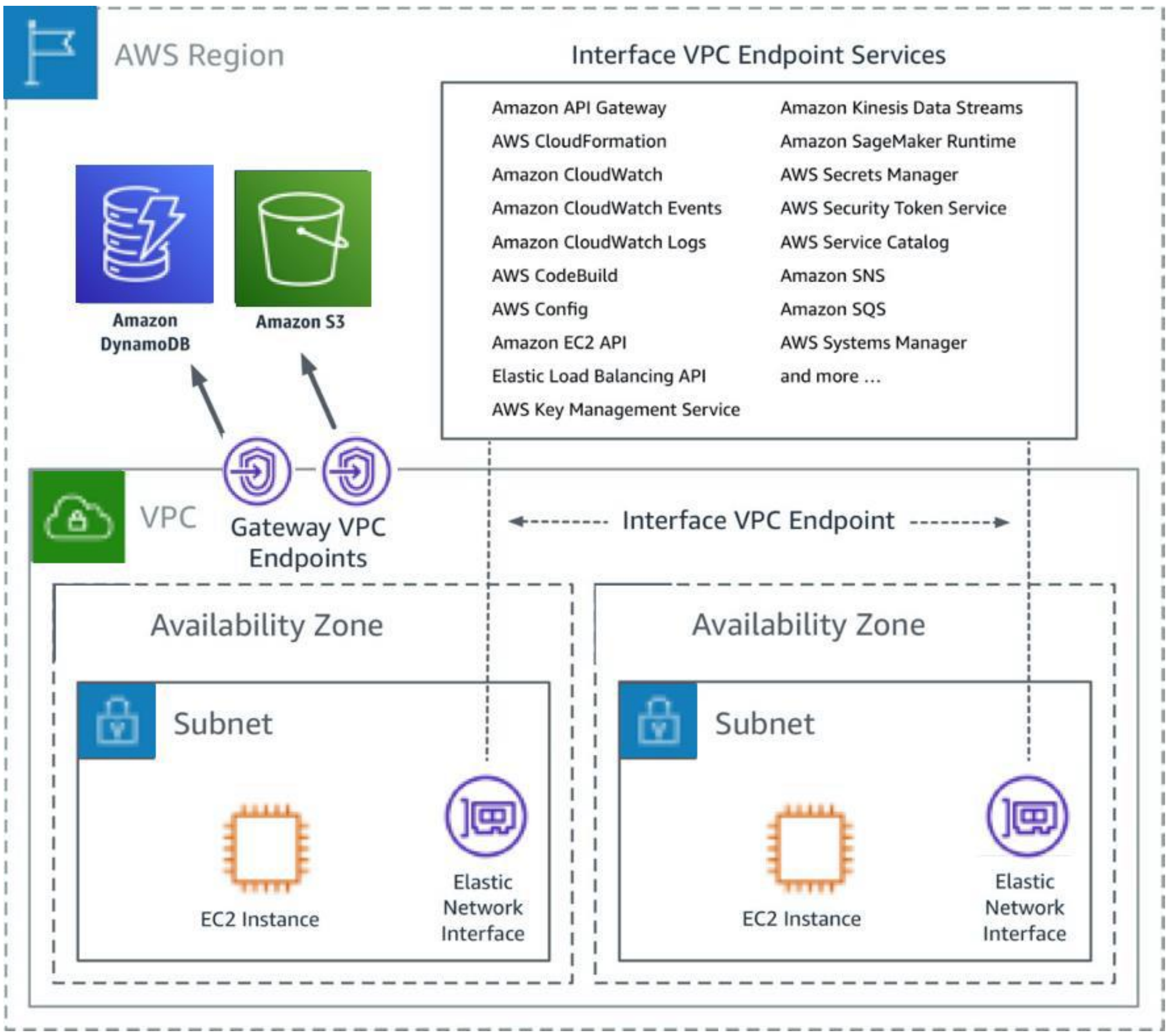
**Some important points to note on TGW:**

- It can be connected across different regions and accounts
- Static routes are required
- Data is Encrypted
- Choose unique ASN (The autonomous system number) for BGP (Border Gateway Protocol)
- VPCs with Overlapping CIDR Ranges cannot be Attached to Same Transit Gateway
- TGW Attachments can only be associated with 1 Route Table
- 5000 attachments per transit gateway and 10000 static routes per transit gateway
- Up to 50 peering attachments (inter-region) per TGW

**VPC Endpoints**
A VPC endpoint enables customers to privately connect to supported AWS services and VPC endpoint services powered by AWS PrivateLink. (Reference)

Amazon VPC instances do not require public IP addresses to communicate with resources of the service. Traffic between an Amazon VPC and a service does not leave the Amazon network.

2 types of endpoints
Gateway endpoints – for Amazon S3 and Dynamo DB
Interface endpoints – Many services listed below



## Auto Scaling Groups

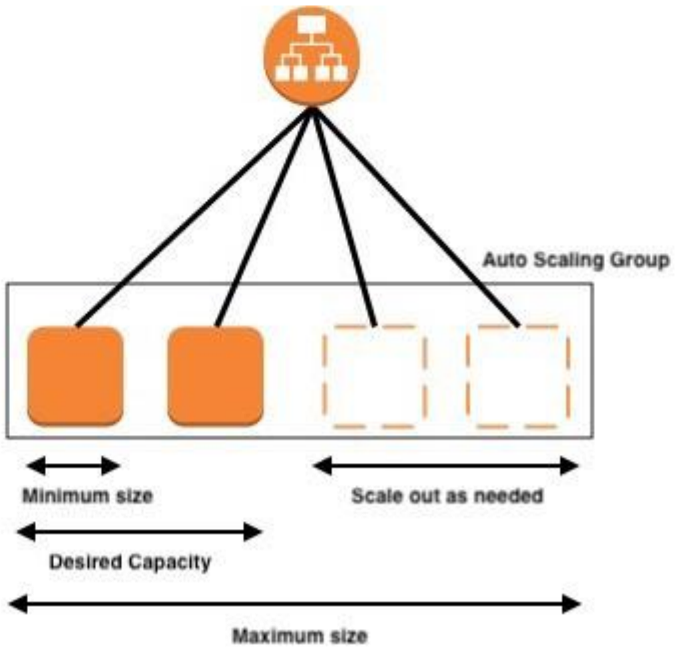
Amazon EC2 Auto Scaling (Reference) helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups.

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

**What is Autoscaling?**
AWS Auto Scaling lets you build scaling plans that automate how groups of different resources

respond to changes in demand. AWS Auto Scaling monitors your application and automatically adds or removes capacity from your resource groups in real-time as demands change.

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



Auto Scaling components: The following describes the key components of Amazon EC2 Auto Scaling.

**Groups**
Your EC2 instances are organized into groups so that they can be treated as a logical unit for the purposes of scaling and management. When you create a group, you can specify its minimum, maximum, and desired number of EC2 instances.

**Configuration templates**
Your group uses a **launch template**, or a launch configuration (not recommended, offers fewer features), as a configuration template for its EC2 instances. You can specify information such as the AMI id, instance type, key pair, security groups, and block device mapping for your instances.

**Scaling options**
Amazon EC2 Auto Scaling provides several ways for you to scale your Auto Scaling groups. For example, you can configure a group to scale based on the occurrence of specified conditions (dynamic scaling) or on a schedule.

**Launch template**
Launch Templates is a new capability that enables a new way to templatize your launch requests. Launch Templates streamline and simplify the launch process for Auto Scaling, Spot Fleet, Spot, and On-Demand instances.

# Elastic Load Balancing

Elastic Load Balancing (Reference) automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones.

It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer as your incoming traffic changes over time. It can automatically scale to the vast majority of workloads.

It will assign the requests to the multiple target or server in a round-robin manner by evenly distributing the requests.

**What is Application Load Balancer?**

The Application Load Balancer is a feature of Elastic Load Balancing that allows a developer to configure and route incoming end-user traffic to applications based in the AWS public cloud. Security group rules are used to define the inbound traffic to the load balancer.

**What is network load balancer?**

A Network Load Balancer functions at the fourth layer (transport layer) of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second. It manages TCP/UDP connection requests.

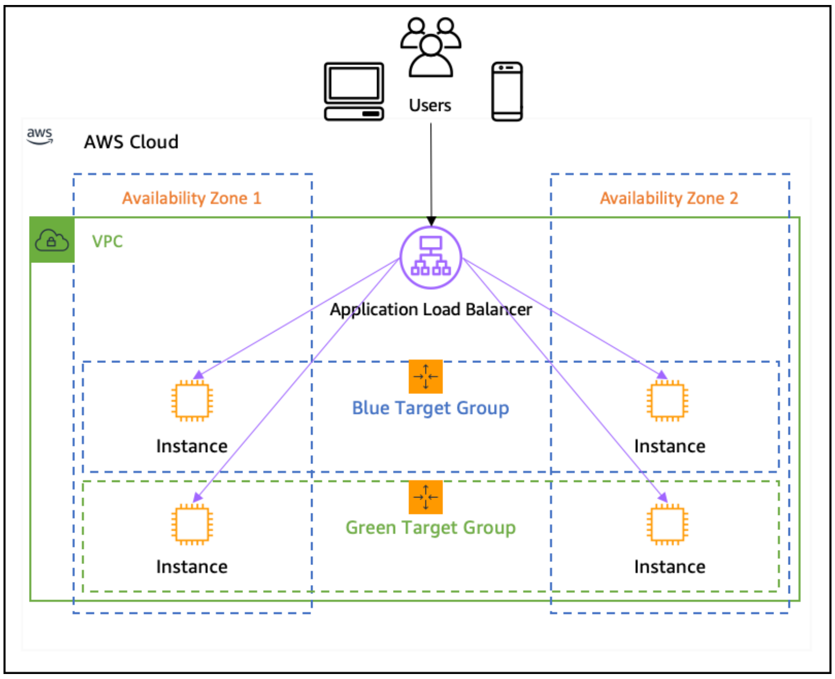
**What is classic load balancer?**

Classic Load Balancer provides basic load balancing across multiple Amazon EC2 instances and operates at both the request level and connection level. Classic Load Balancer is intended for applications that are built within the EC2-Classic network.

**Gateway Load Balancers**

Gateway Load Balancers enable you to deploy, scale, and manage virtual appliances, such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems.

**Application Load Balancer**



The application load balancer serves as the single point of contact for clients and distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. This increases the availability of your application.

We can monitor the load balancer traffic by enabling Access logs.

**Listener**
A listener is a component that checks for connection requests, using the protocol and port that you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

**Target Groups** for Application Load Balancers:
Target groups route requests to one or more registered targets, such as EC2 instances, using the protocol and port number that you specify. You can register a target with multiple target groups. You can configure health checks on a per target group basis.

Target types can be instance, IP address(excluding publicly routable IP addresses) or Lambda function and Target groups support the following protocols and ports:

- **Protocols**: HTTP, HTTPS
- **Ports**: 1-65535

**Health Check**
The Application Load Balancer periodically sends requests to its registered targets to test their status. These tests are called health checks. The requests are routed only to the healthy targets in the enabled Availability Zones for the load balancer.

## Route53

Amazon **Route 53** is a highly available and scalable cloud Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost effective way to route end users to Internet applications by translating names like www.example.com into the numeric IP addresses like 192.0.2.1 that computers use to connect to each other.

Route 53 takes the DNS names and maps to its ip address of the web server. Route 53 to configure DNS health checks to route traffic to healthy endpoints or to independently monitor the health of your application and its endpoints.

**DNS**

- DNS stands for Domain Name System.
- DNS is a directory service that provides a mapping between the name of a host on the network and its numerical address.

DNS is a TCP/IP protocol used on different platforms. The domain name space is divided into three different sections: generic domains, country domains, and inverse domain.

**DNS Resolution**
The process of DNS resolution involves converting a hostname (such as www.example.com) into a computer-friendly IP address (such as 192.168.1.1).

**Hostname**
a hostname is the name of a computer or device (host) on a network.

**Host id**
A Host ID is a a specific piece of information which uniquely identifies a computer.

**Network id**
Network ID is the portion of an IP address that identifies the TCP/IP network on which a host resides.

**Types of records in AWS Route53**

- **A record -** Maps domain name to ip address (IPV4 and AAAA is for IPV6)
- **Cname -** Maps hostname to hostname
- **Aliases -** Maps domain name to aws resources.

**NOTE: DNS** name is **unique** across the world**.**

**Time to live**
Time to live (**TTL**) or hop limit is a mechanism which limits the lifespan or lifetime of data in a computer or network.

To avoid burden on route 53, TTL is used. It will attach the IP address attaching the TTL.

TTL is the time stamp attached to the ip address to specify for how long data should be saved in the user cache. If user requests of DNS resolution, within this time, route 53 will not respond.

It helps to regulate traffic at the route 53. Higher the TTL, less traffic.

**Latency**
Latency is the time that passes between a user action and the resulting response.

**Types of Routing Polices in Route53 (reference)**

1. **Simple routing policy**
   Used for a single resource that performs a given function for your domain, for example, a web server that serves content for the example.com website
2. **Weighted routing policy**
   Weighted routing lets you associate multiple resources with a single domain name (example.com) or subdomain name (acme.example.com) and choose how much traffic is routed to each resource.
3. **Latency routing policy**
   Used when you have resources in multiple AWS Regions and you want to route traffic to the region that provides the best latency
4. **Failover routing policy**
   Use when you want to configure active-passive failover.
5. **Geolocation routing policy**
   Used when you want to route traffic based on the location of your users.

**Health Checks**
Health checks are a way of asking a service on a particular server whether or not it is capable of performing work successfully.

**Unhealthy threshold**
The number of consecutive failed health checks that must occur before declaring an EC2 instance unhealthy.

**Healthy Threshold**
The number of consecutive successful health checks that must occur before declaring an EC2 instance healthy.

**Response timeout**
The amount of time to wait when receiving a response from the health check, in seconds.

**Types of Encryption**
Server-side Encryption – Done by AWS – Aws managed encryption
Client-side Encryption – Done by user – Customer managed encryption

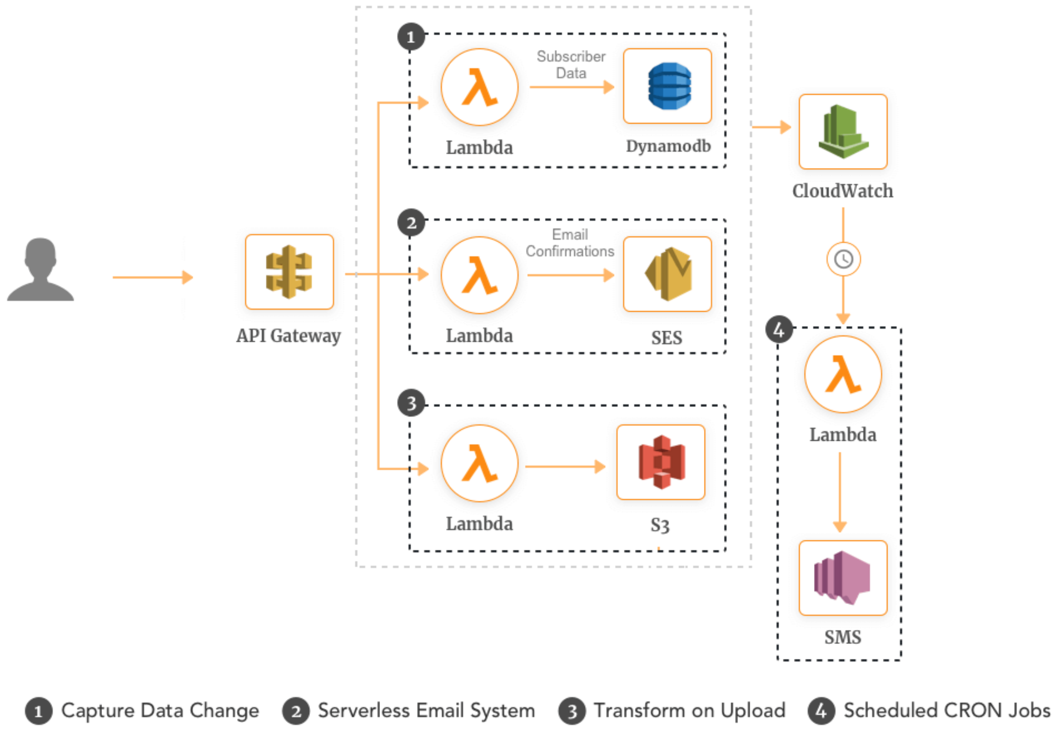
In server side encryption, master key is managed by AWS.
In client side encryption, we need to provide the master keys.

Server side encryption is also called as encryption at rest. (after storing the data in s3)
In client side encryption, encryption will be done during transit (first encrypt the data and then store it in s3).

## Lambda

AWS **Lambda** lets you run code without provisioning or managing servers. You pay only for the compute time you consume.

Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. It automatically manages the compute resources across multiple availability zones and scales them when new actions are triggered.



- Manage your virtual functions not really caring about the servers
- Run on demand
- Scaling is automated
- It can run from 1sec to 15 minutes

**Trigger**: A trigger is a resource you configure to allow another AWS service to invoke your Lambda function when certain events or conditions occur.

**Destination**: Lambda Destinations allow you to specify what to do if a lambda function succeeds or fails. You can pass on the information to either another Lambda Function, an SNS Topic or to an EventBridge Event Bus.

**Billing:**
Pay per request, first one million requests is free $0.20 per one million request.
Compute time 0.00001667 for every GB-seconds used.

**AWS Lambda Languages:**
NodeJS, Python, Python3, Gr00vy, java, csharp, Scala and GO

**AWS Lambda Integration**
Kinesis, API Gateway, DynamoDB, AWS S3, CloudWatch Events, CloudWatch logs, SNS and Cognito

## CloudFormation

AWS CloudFormation is a service that is used to create and manage resources on AWS cloud. You can create complete infrastructure on AWS by writing templates in either **JSON or YAML** format without having to perform actions manually. It is an IAC tool native to AWS and cannot be used in other cloud providers like GCP or Azure.

- Create stacks using these CloudFormation templates
- Can create a template using designer or write a text file in JSON or YAML language
- Store the template locally or in an S3 bucket

**Infrastructure as Code (IaC):** Infrastructure as code is a practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control and continuous integration. The cloud's API-driven model enables developers and system administrators to interact with infrastructure programmatically, and at scale, instead of needing to manually set up and configure resources. Thus, engineers can interface with infrastructure using code-based tools and treat infrastructure in a manner similar to how they treat application code. Because they are defined by code, infrastructure and servers can quickly be deployed using standardized patterns, updated with the latest patches and versions, or duplicated in repeatable ways.

**Template Structure**:
**Format version** (optional): Format version defines the capability of a template.
**Description** (optional): A text string that describes the template
**Metadata** (optional): Can be used to provide additional information about the template
**Parameters** (optional): Values to pass to your template at runtime (when you create or update a stack). You can refer to parameters from the Resources and Outputs sections of the template.
**Rules** (optional): Validates a parameter or a combination of parameters passed to a template during a stack creation or stack update.
**Mappings** (optional): A mapping of keys and associated values that you can use to specify conditional parameter values
**Conditions** (optional): Conditions that control whether certain resources are created or not or assign certain properties to resources during stack creation or update. Example: Enable termination protection on the EC2 instance if the stack is for production environment.
**Transform** (optional): Transform builds a simple declarative language for AWS CloudFormation and

enables reuse of template components.
**Resources (*required*)**: Specifies the stack resources and their properties
**Outputs** (optional): Describe the values that we need that can be returned as output whenever a stack is created/updated.

**Intrinsic Functions** (Reference)**:** AWS CloudFormation provides several built-in functions that help us manage the stacks. Use intrinsic functions in your templates to assign values to properties that are not available until runtime.

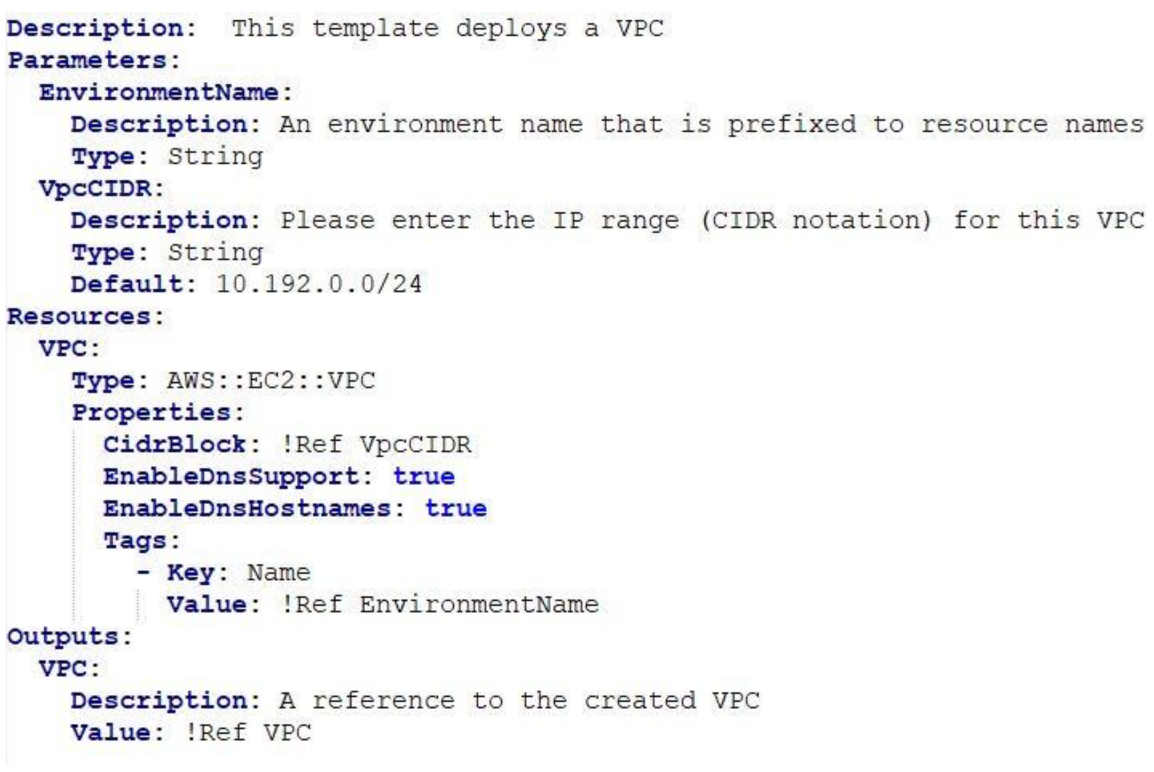
Some of the Intrinsic functions are *Ref, Sub, Select, Join, Split.* Intrinsic functions are currently supported only in resource properties, outputs, metadata attributes, and update policy attributes

**Stacks**: A collection of AWS resources is called a stack, and it can be managed in a single unit

**Drift**: A CloudFormation stack resource is considered to have drifted if the actual property value is different from the values defined in the template. This usually occurs when we change some configuration manually instead of updating the stack.

**Link to CloudFormation Resource and Property Reference**

**Sample CloudFormation Template:**



```
Description:  This template deploys a VPC
Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String
  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/24
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC
```

# Databases – RDS & DynamoDB

## RDS

Amazon Relational Database Service (**Amazon RDS**) makes it easy to set up, operate, and scale a relational database (SQL database) in the cloud.

**Amazon RDS supports:**

- Amazon Aurora

- PostgreSQL
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server

Encryption at rest is supported by RDS. However, encrypting existing DBs is not supported. To do this, you'll need to create a new encrypted instance and migrate data to it. The encryption key can be stored in KMS (Key Management Service).

*Features of RDS* – Multi-AZ for high availability, Read Replicas (Scale for heavy workloads), Automatic Backups (AWS takes automated snapshots), Patching (AWS patches servers during maintenance window), Monitoring (using CloudWatch), Supports different Storage (General-purpose SSD, Provisioned IOPS SSD, Magnetic) ,Encryption (using KMS)

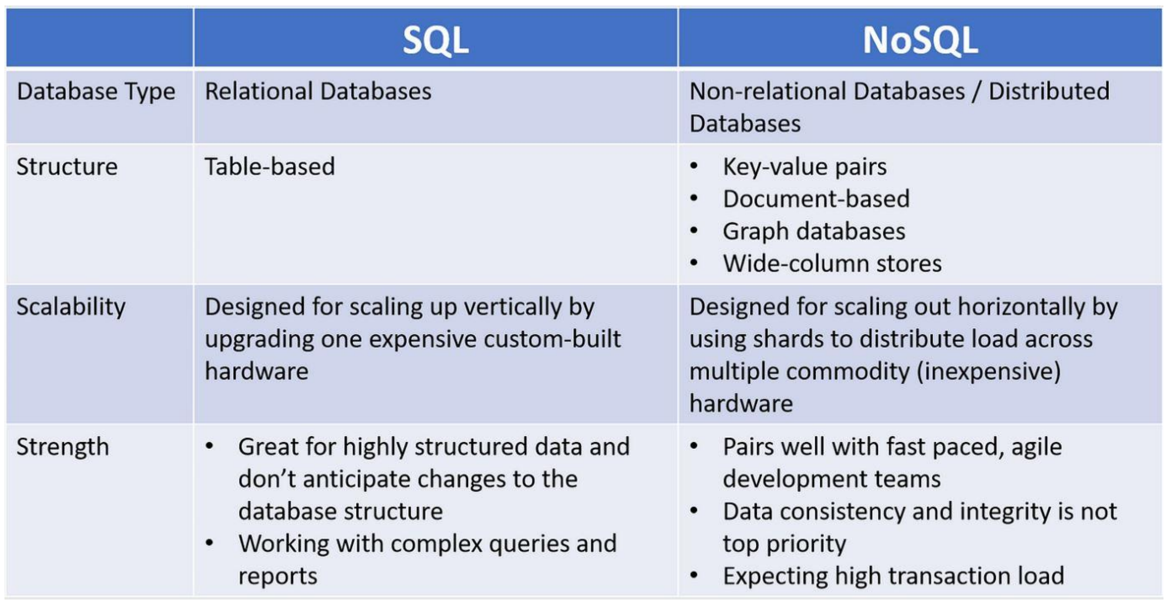
## DynamoDB

Amazon **DynamoDB** is one of the NoSQL database service on AWS. It is a fully managed, serverless, key-value database designed to run high-performance applications at any scale.

DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools.

*Features of DynamoDB* – High Availability (Automatically replicates across AZ), Scalability (Automatically scales), Backups (full backup and Point in time recovery option), Encryption (at rest) and Monitoring (using CloudWatch)

Difference between SQL and NoSQL Databases (Reference)



| | SQL | NoSQL |
|---|---|---|
| Database Type | Relational Databases | Non-relational Databases / Distributed Databases |
| Structure | Table-based | • Key-value pairs<br>• Document-based<br>• Graph databases<br>• Wide-column stores |
| Scalability | Designed for scaling up vertically by upgrading one expensive custom-built hardware | Designed for scaling out horizontally by using shards to distribute load across multiple commodity (inexpensive) hardware |
| Strength | • Great for highly structured data and don't anticipate changes to the database structure<br>• Working with complex queries and reports | • Pairs well with fast paced, agile development teams<br>• Data consistency and integrity is not top priority<br>• Expecting high transaction load |

# Cognito

- It Mainly provides authentication authorization and user management for your application
- It provides a managed user pool to mange identity for the application

Cognito provides user flows:

- Signup
- Signin
- Forgot or change password
- Multifactor authentication
- Email and phone verification

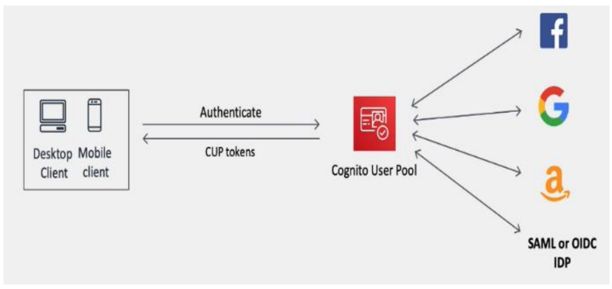
It also provides software development kit to your mobile or web application, and also provide lambda triggers in order to customize any of these user flows with you own business logic

It also provides a built-in hosted UI for these user flows

Social identity can be integrated
Facebook
Google
Amazon
SAML

After authentication the user the Cognito provides the best practice way of accessing the AWS resources securely from the app by providing temporary credentials

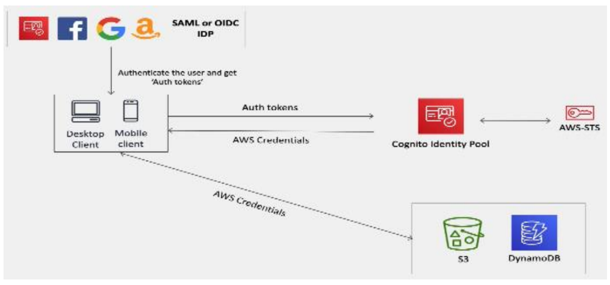
**User Pool:**



User pools acts as mediator between your app and external social identity providers

- You can add multiple identity providers as you need.
- The user pool manages the token exchange with each of the providers and gives your app standard user pool tokens of same format

**Identity poll:**



- Where you exchange the authentication token to get temporary aws credentials which you can use to access the resources directly from the app
- These can be used independently of each other or used together

Difference between user pool and identity pool

**AWS Cognito User Pool** is there to authenticate users for your applications.

Say you were creating a new web or mobile app and you were thinking about how to handle user registration, authentication, and account recovery, you don't need to implement user authentication inside your application, rather you can integrate AWS Cognito User Pools, which will manage user sign-up, sign-in, password policies.

**AWS Cognito Identity pool**:

This is a service which was designed to authorize your users to use the various AWS services. The source of these users could be a Cognito User Pool or even Facebook or Google.

In other words, Identity Pools are used to assign IAM roles to users (who had been authenticated through a separate Identity Provider which could be Cognito User Pools or Social logins (e.g; Gmail, Facebook & etc.)). Because these users are assigned an IAM role, they each have their own set of IAM permissions, allowing them to access AWS resources directly.

So, the difference is

- AWS Cognito User Pools: Granting access to a application
- AWS Cognito Identity Pools: Granting access to amazon service

Difference between IAM and Cognito

**AWS IAM** gives securely and control access to AWS services and resources for your users

**AWS Cognito** It Mainly provides authentication authorization and user management for your application

Link for Cognito user pool creation: https://www.youtube.com/watch?v=jTu--LpjA18