**Amazon Web Services (AWS)**

Cloud computing:

It is a model in which computing resources are available as a service.

Three Important characteristics of cloud computing.

* On Demand & Self service (any time launch without manual intervention)
* Elasticity (can scale up and scale down any time)
* Measured services (Pay as you use)

Cloud computing Models:

There are 3 types of Cloud computing models:

SAAS -> Software as service [google docs, office 360, Gmail]

PAAS -> Platform as Service [google app engine]

IAAS -> infrastructure as Service [AWS, Digital Ocean]

AWS will provide all types of models in their service catalogue.

But if you depend on only AWS for all, you will lose lot of money. You should check same serves other cloud platforms and compare the costing.

Architecture of Cloud: Cloud is not in Cloud :)

Cloud is data centre secretly

Cloud from behind scenes is DC only



Virtualization: it will allow you to run multiple operating systems on single H/W

The most widely use virtualization platforms are

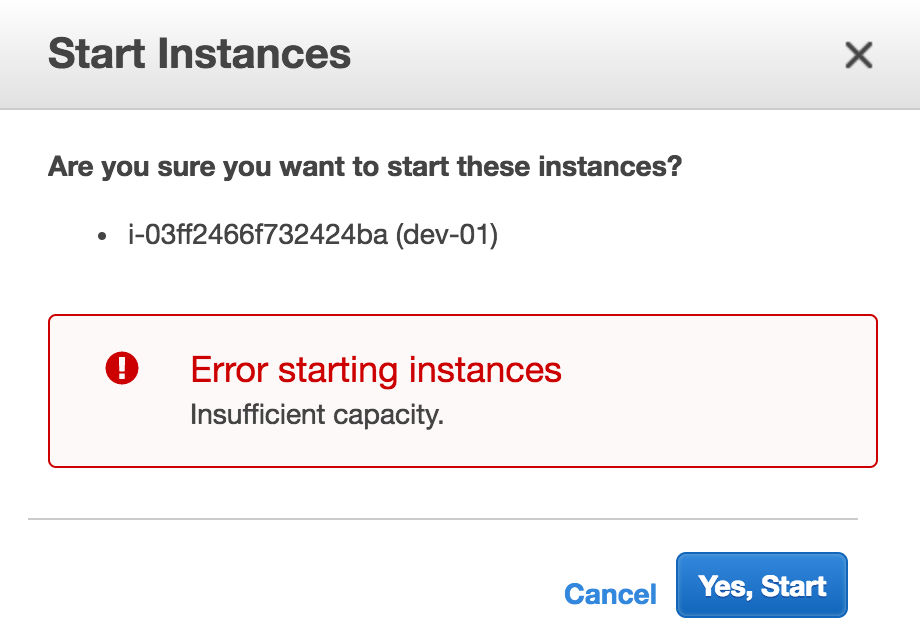
* VMware
* KVM
* XEN —> AWS used very huge
* Virtual Box

OnDemand & Self service:

OnDemand: We can provision resources whenever needed without any human interaction with service providers.

OnDemand makes self service with Automation possible in a seamless way.

Challenges: Even cloud is running on some DataCenter, We may get sometimes capacity issues like below (it is very rare)



Elasticity:

Adding/Removing the resources capacity whenever needed in the environment. Capacity generally means CPU and Memory most of the time.

Scalability:

Horizontal scaling: Add (or) remove instances from pool like a cloud form.

Vertical Scaling: Adding or removing resources to an existing server.



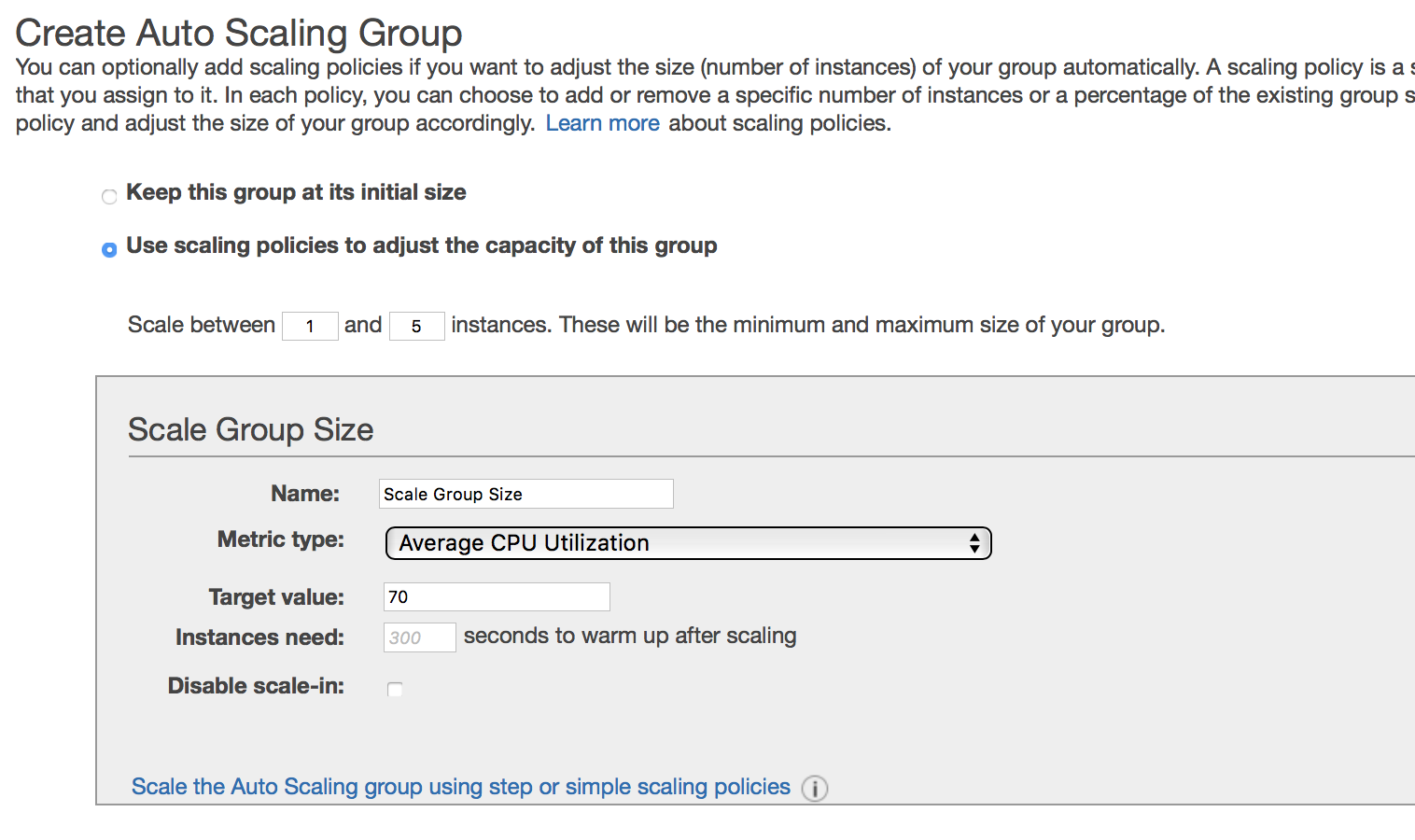
Auto Scaling:

The scaling should be automate which should not manual intervention, it can be acheved via Auto Scaling feature.

Use Case Scenario:

* Whenever CPU Load > 70%, scale up to two more servers
* Whenever CPU Load < 30%, scale down by two servers.

Here is the sample auto-scaling based configuration:



**AWS infrastructure:**

AWS is divided the global into different regions below are the recent regions which are deployed and in plan (orange colour)



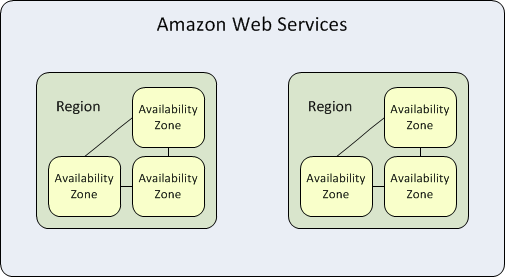
**AWS Region**

A region is a physical location around the world where we cluster data centers. We call each group of logical data centers an Availability Zone. Each AWS Region consists of multiple, isolated, and physically separate AZs within a geographic area.

AWS has 30 regions worldwide and the number keeps increasing.

**Availability Zone:**

An Availability Zone (AZ) is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. AZs give customers the ability to operate production applications and databases that are more highly available, fault tolerant, and scalable than would be possible from a single data center.



**Edge location/AWS Local Zones:**

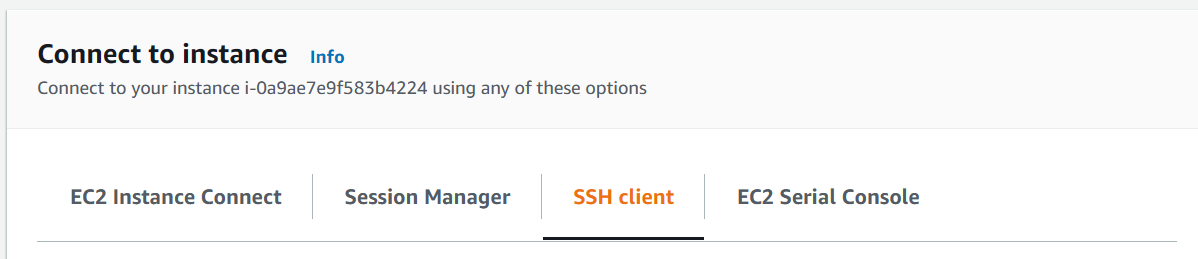
[AWS Local Zones](https://aws.amazon.com/about-aws/global-infrastructure/localzones/) place compute, storage, database, and other select AWS services closer to end-users. With AWS Local Zones, you can easily run highly-demanding applications that require single-digit millisecond latencies to your end-users such as media & entertainment content creation, real-time gaming, reservoir simulations, electronic design automation, and machine learning.

Create your AWS free tire account.

Setting up you ssh client.

* If its windows EC2 instances RDS (mstsc) need to use
* If its LINUX Ec2 instance ssh client.
* Gitbash
* Putty
* Mobaxterm

You can connect your EC2 instances in multiple ways as given below



The authentication can be happened two types:

1. Password based authentication
2. Key based authentication.

Password based authentication is not much secure since the password complexity it can be easily hack.

So Key based authentication will be more secure.

In SSH, when you generated any key for an user it will generate two types of keys.

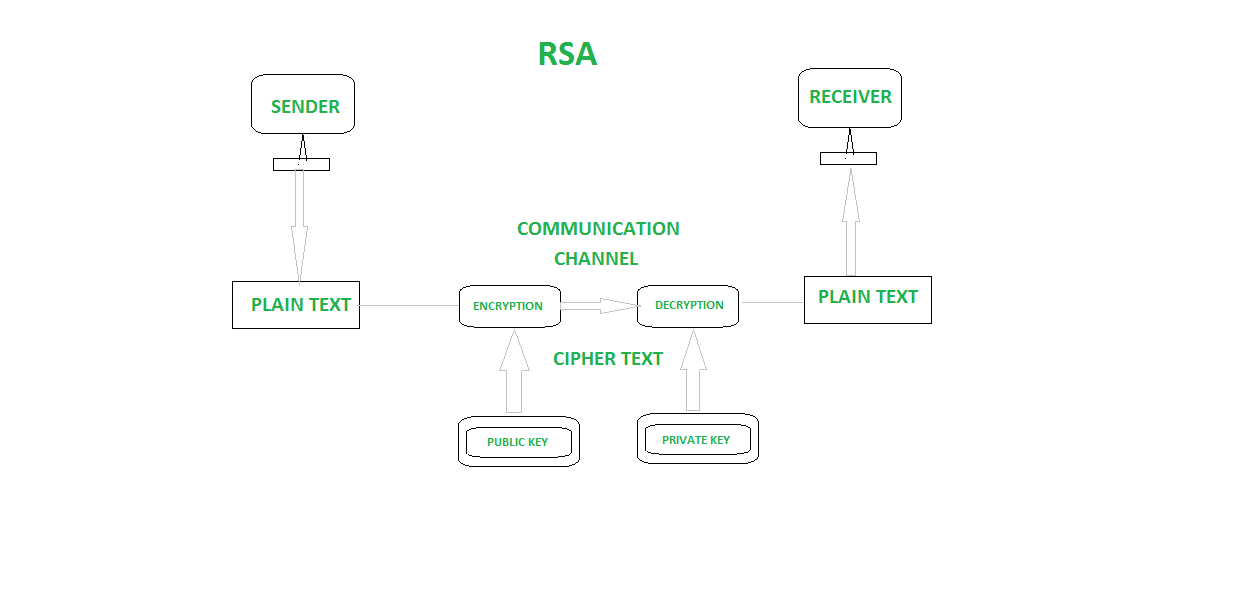
* Public key
* Private key

Public key: which is shared/copy to destination machine/remote machine to which you want to connect.

Private key: which is your private key you have to keep with you from where you want to connect. (jump server)

Key will have three types of extension.

* Ppk (putty private key)
* Pem(privacy enhanced mail)
* RSA(Rivest, Shamir and Adelman Algoritham)



-> Launch your first EC2 instance.

-> install web server

-> amazon-linux-extras install nginx1

-> Modify security groups and allow all traffic from anywhere.

-> Try to access the website using EC2 public IP.

-> CIDR understanding, Subnet Calculation, Network bits and Host bits understanding, check recording given and also follow below youtube video.

[Basics of Subnetting | How to find Subnet Mask, Network ID, Host IP Address from CIDR Value | 2018 - YouTube](https://www.youtube.com/watch?v=q7wNcYliJ1Q)

**VPC (Virtual Private cloud):**

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

To create VPC:

-> Choose Region (VPC is region Specific and span on across all you availability zones in your region)

Ex: ap-south-1 (mumbai region)

-> Choose you right CIDR notation

Ex: 10.10.0.0/16

->Login into your AWS console.

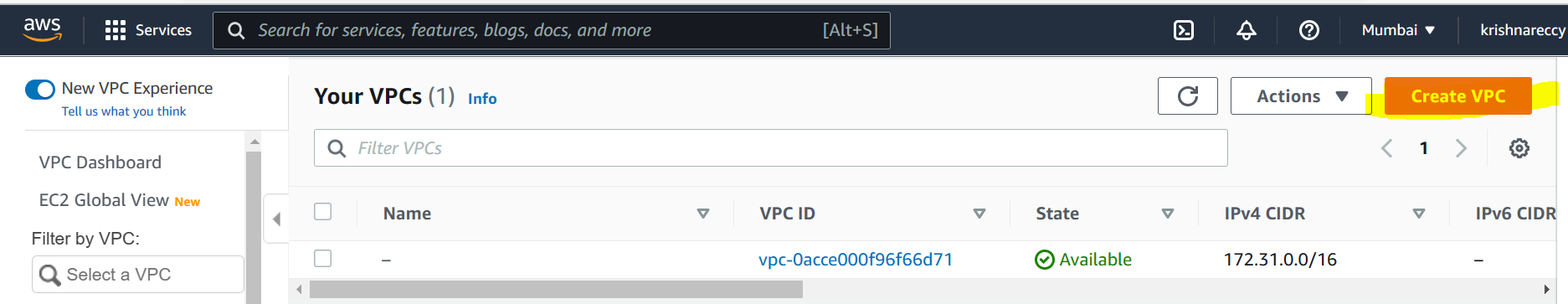
1. Change your region
2. Search for VCP



1. On VPC console you will find one default VPC which has already created by AWS for you.

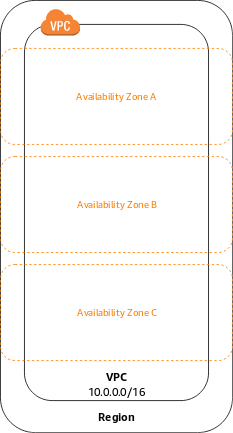
Note:- by default when you create your account in AWS, Amazon will create a default VPC for you in each region and also respective subnetes depending on the AZ count.

1. Click on create VPC



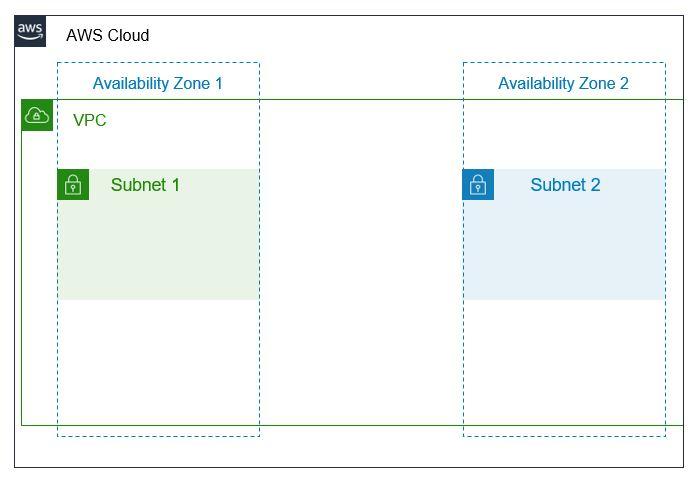
1. Enther you VPC name and CIDR block. It will create a VPC with your name and with CIDR which you gave.

Example VPC, as per our previous discussion VPC is a Region specific.

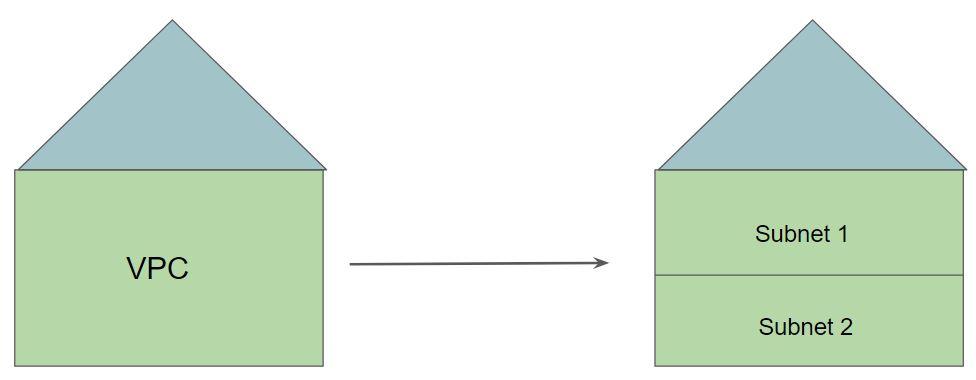


**Subnet:**

A subnet is a range of IP addresses in your VPC. You can launch AWS resources, such as EC2 instances, into a specific subnet. When you create a subnet, you specify the IPv4 CIDR block for the subnet, which is a subset of the VPC CIDR block.



As part of Demo pls create you subnets in above created VPC



**IP Address Pointer - Analogy**

All the resources inside VPC will have an IP address from the VPC CIDR.

Let’s understand this with an example:

We assign one hundred number to our VPC (0-100)

Every resource in VPC will be assigned one of these numbers.

* EC2 Instance 1 - Number 4
* EC2 Instance 2 - Number 10

**IP Address Pointer - Technical**

Our VPC has a CIDR of 10.77.0.0/16

Total IP Addresses: 65,536

Every resource in VPC will be assigned IP addresses from the given pool.

* EC2 Instance 1 - 10.77.0.5
* EC2 Instance 2 - 10.77.0.10

**Subnet CIDR - Analogy**

Every subnet has its own IPv4 CIDR block that is a subset of the VPC’s CIDR block.

We assign one hundred number to our VPC (0-100)

We subdivide it based on subnets.

* Subnet 1 - Will have a range of 1-10
* Subnet 2 - Range of 11-50
* Subnet 3 - Range of 51-90

**Subnet CIDR - Technical**

Every subnet has its own IPv4 CIDR block that is a subset of the VPC’s CIDR block.

Our VPC has a CIDR of 10.77.0.0/16

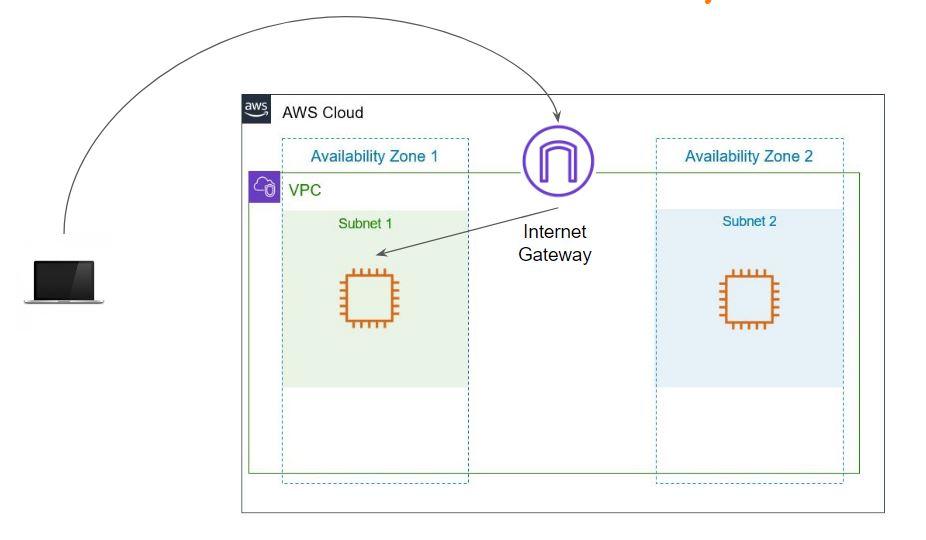
Total IP Addresses: 65,536

We subdivide it based on subnets.

* Subnet 1 - 10.77.1.0/24 (256 IP Addresses)
* Subnet 2 - 10.77.2.0/24 (256 IP Addresses)
* Subnet 3 - 10.77.3.0/24 (256 IP Addresses)

**Internet Gateway:**

Internet Gateway is a highly available VPC component that allows communication between resources in the VPC and the Internet.



* Create an Internet Gateway
* Attach the Internet Gateway to the VPC
* Ensure that subnet’s route table points to the Internet Gateway
* Ensure that the instances have Public / EIP attached.

**Overview of Route Table:**

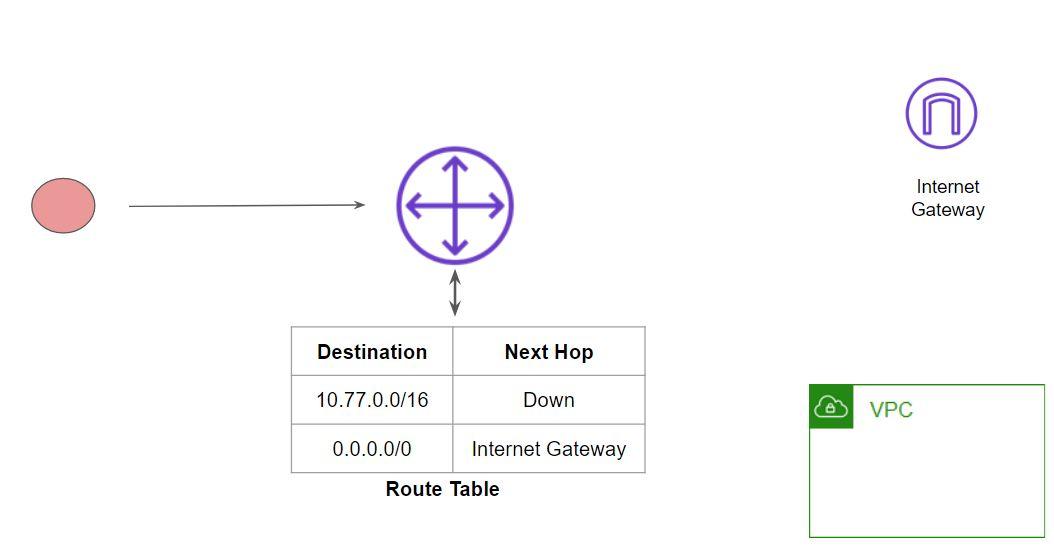
Normally routes are very important to reach from source to destination.

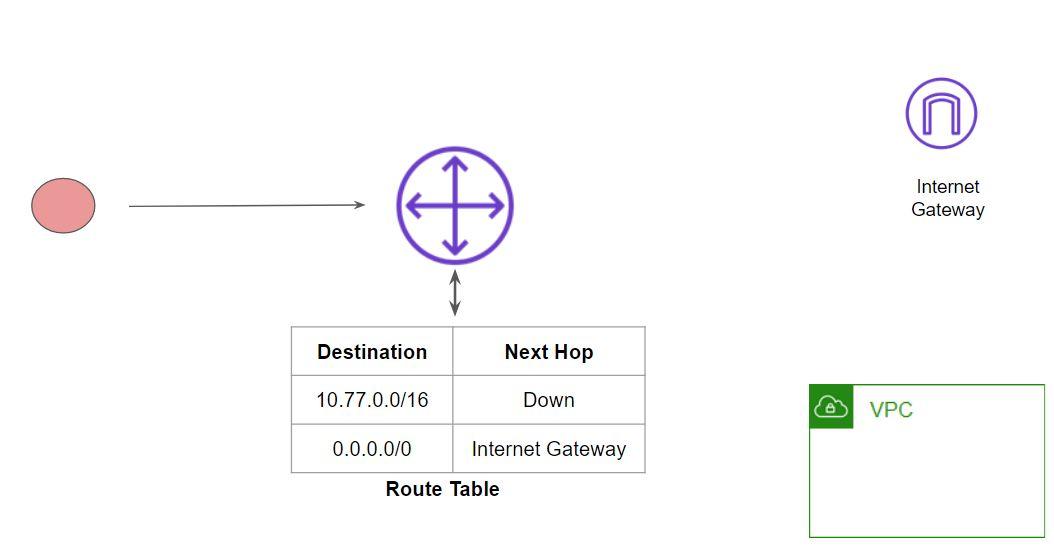


Just consider above is an example for the routs which direct you for different directions if u want to go Madurai got straight, Tuticorian go right etc…

Route:

A router is a networking device that forwards data packets between networks.





Route Table:

A route table contains a set of rules, called routes, that are used to determine where network traffic from your subnet or gateway is directed.

By default, whenever a VPC is created, the route table is also created we called it as a Main routing table.

We can create custom routing tables for your subnets as per our requirements.

The following diagram depicts High-Level Working - Internet Route

**IP Addressing in AWS:**

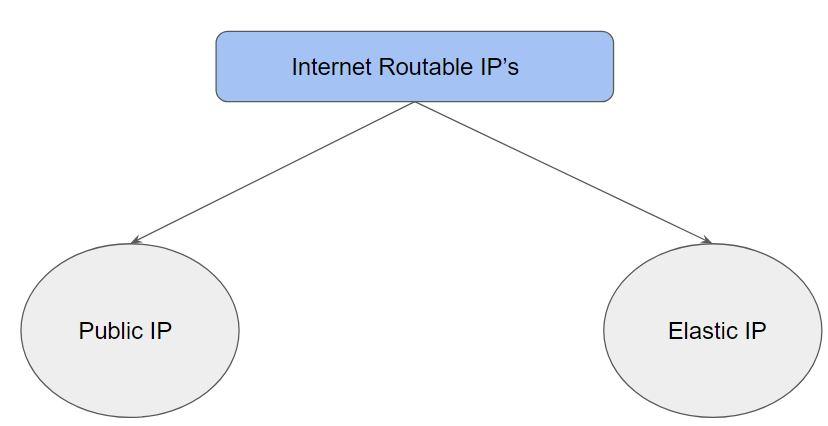
We have three types of IPs for an EC2 instance will have.

1. Private IP
2. Public IP
3. Elastic IP

**Private IP:**

Which is used for internal communication (within the AWS level) and you will get Private IP from your subnet CIDR range.

Public IP and Elastic IPs are used for outside (Internet communication)

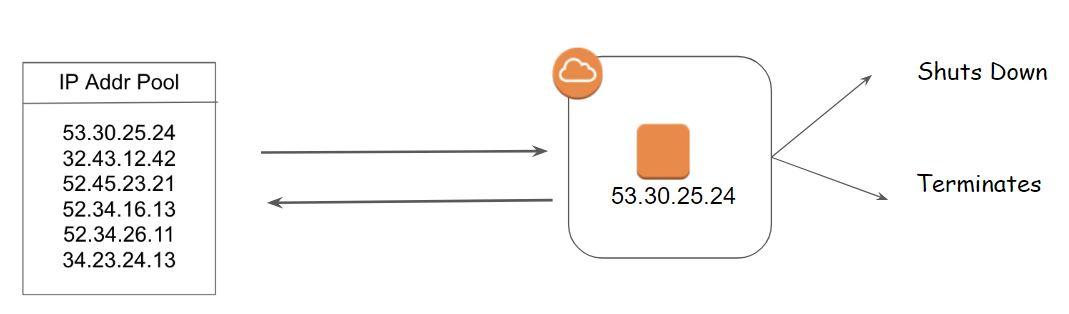


**Public IP**

Public IP is an IP address that is reachable from the Internet.

Every instance that is given a public IP address is also given a DNS hostname, something like ec2-52-34-16-123.us-west-2.compute.amazonaws.com

These are dynamic IP’s.



**Elastic IP**

Elastic IP is an static IPV4 address and it is associated with your AWS account.

To use Elastic IP, we first allocate it for our AWS account and then associate it with a instance that we need.

The IP is associated with your AWS account until you release it.

Important Note:

There is a small hourly charge if the Elastic IP is not associated or if it’s associated with a stopped instance.

**Public Subnet:**

Public subnet is a subnet which is having Internet GW in its routing table to connect internet directly.

This subnet is recommended if you want to run a public-facing web application.

Overall Security Risk: High

**Private Subnet:**

Private subnets are the ones that do not have an Internet Gateway attached to it.

New connections from the Internet cannot reach to the EC2 instances within the private subnet.

Instances in private subnet do not have a Public IP / Elastic IP attached.

Example Use-Case:

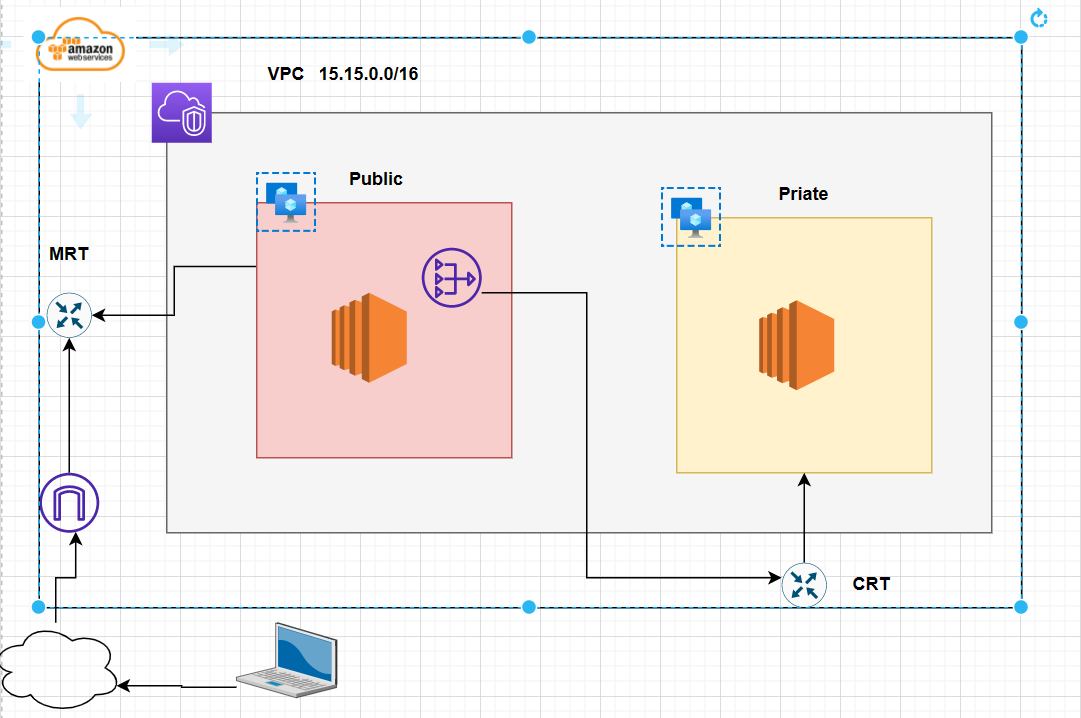
Database Servers

Servers who do not directly interact with Internet resources.

**NAT Gateway/NAT Instance:**

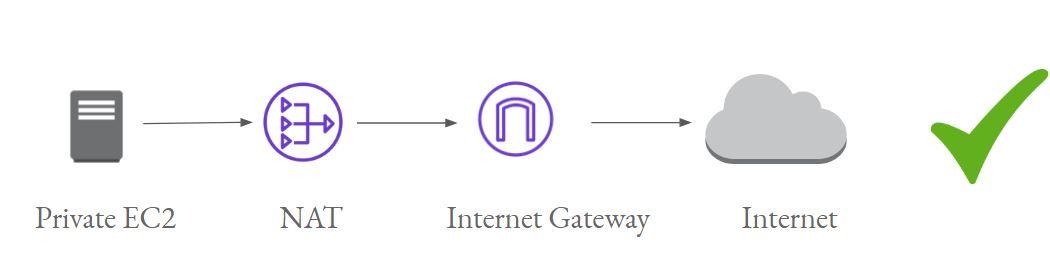
EC2 Instances from a private subnet should be able to connect to the internet to perform various activities like patch updates, downloading software, and others.

To achieve this use-case, NAT Gateways are introduced.



NAT instance also same use case but its not its launch as a instance with NAT AMI.

Using NAT GW its possible to connect internet from the private subnet, But its not possible to connect to from outside. So only Egress Traffic to the internet is possible.



Thumb Rule: To use NAT instance or NAT GW you have to follow below rules.

1. Those should be launch in Public subnet
2. Should have Elastic IP or Public IP (in case of NAT instance)

Differences between NAT GW and NAT Instance, below is the link from AWS official document to know exact differences.

[Compare NAT gateways and NAT instances - Amazon Virtual Private Cloud](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html)

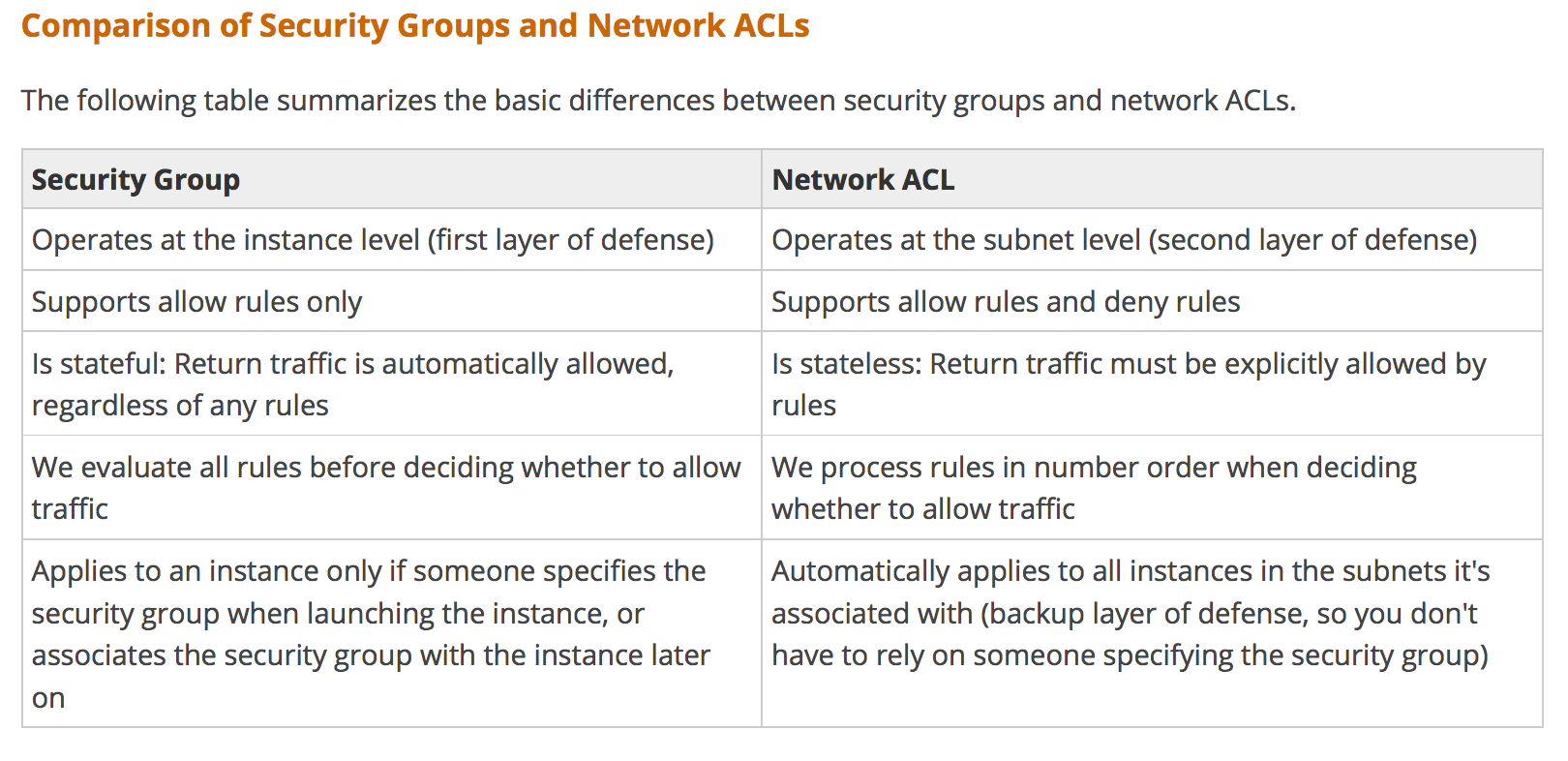
Security:

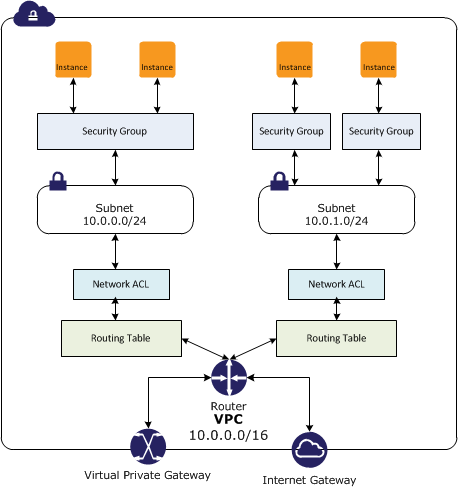
We can provide two our VPC security in two ways:

1. Security Groups
2. NACL

**Security groups** – Act as a virtual firewall for associated instances, controlling both inbound and outbound traffic at the instance level

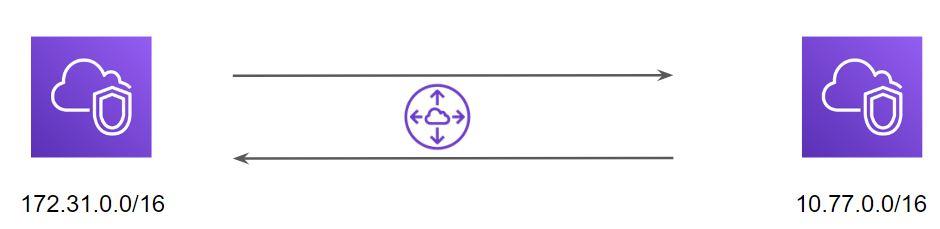
**Network access control lists (NACLs)** – Act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level





**VPC Peering:**

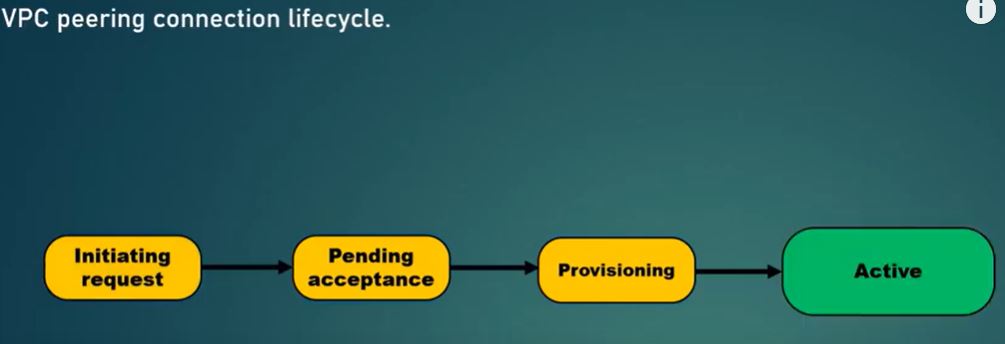
VPC peering is a network connection between two VPC that enables the communication between instances of both the VPC.

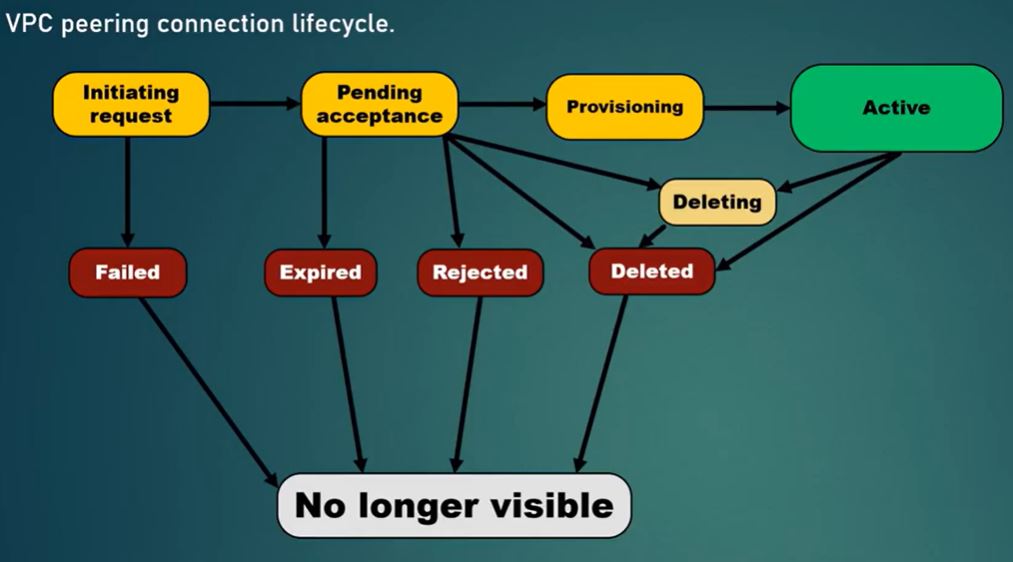


Peering can be done between VPS:

1. VPCs are With in the same Region
2. VPCs are Different region
3. VPCs are different accounts.

Below is the VPC Peering life cycle:

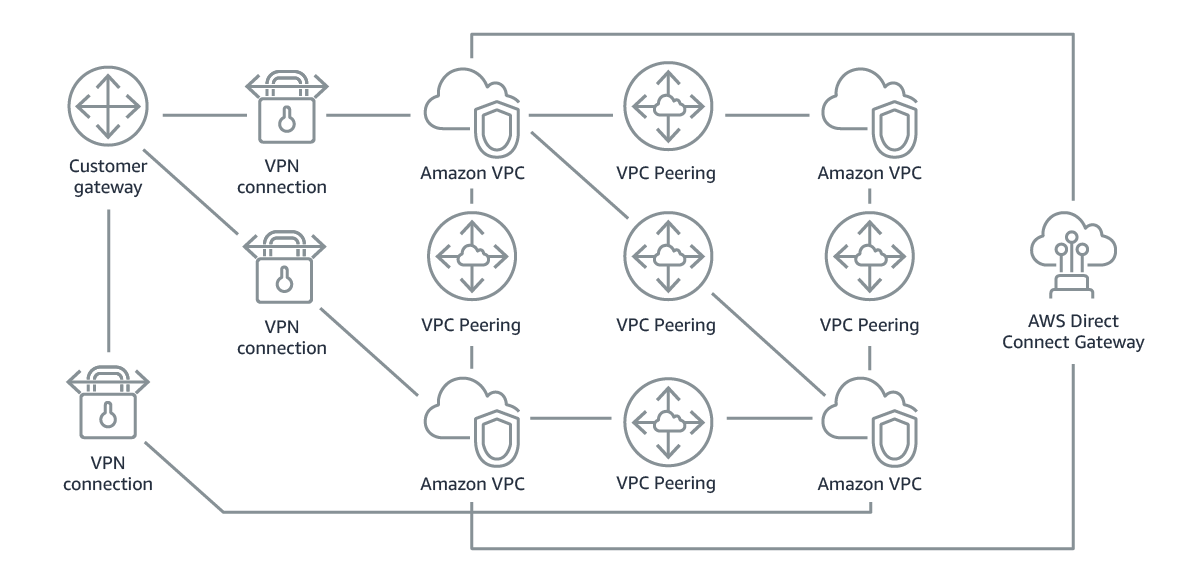




**VPC Transit Gateways:**

AWS Transit GW will allow you to connect different VPCs with Transitive Peering connection.

Below is an example without Transit GW if u want to connect between multiple VPCs.



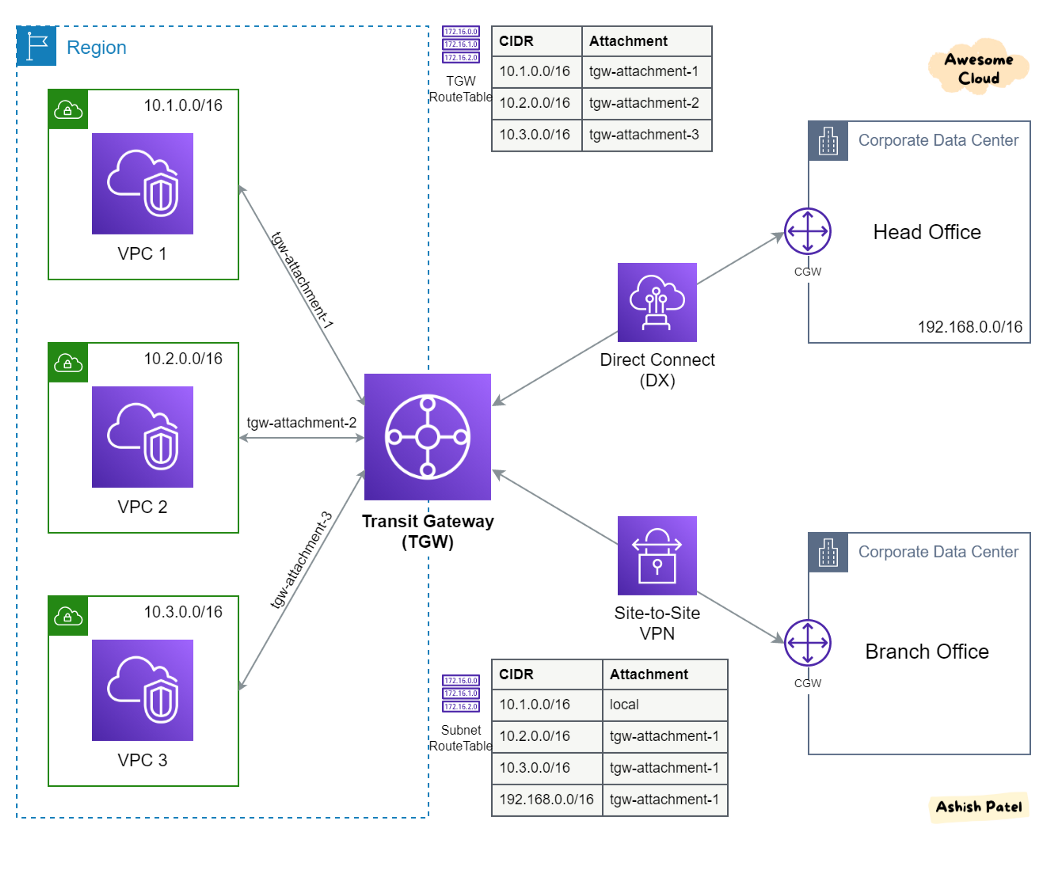
From the above the design will be very complex and difficult. Where as if you use Transit GW it will be simple.

The above Architecture can be achieved like below with Transit GW.



AWS Amazon Virtual Private Clouds (VPCs) and on-premises networks through a central hub. This simplifies your network and puts an end to complex peering relationships. It acts as a cloud router – each new connection is only made once.

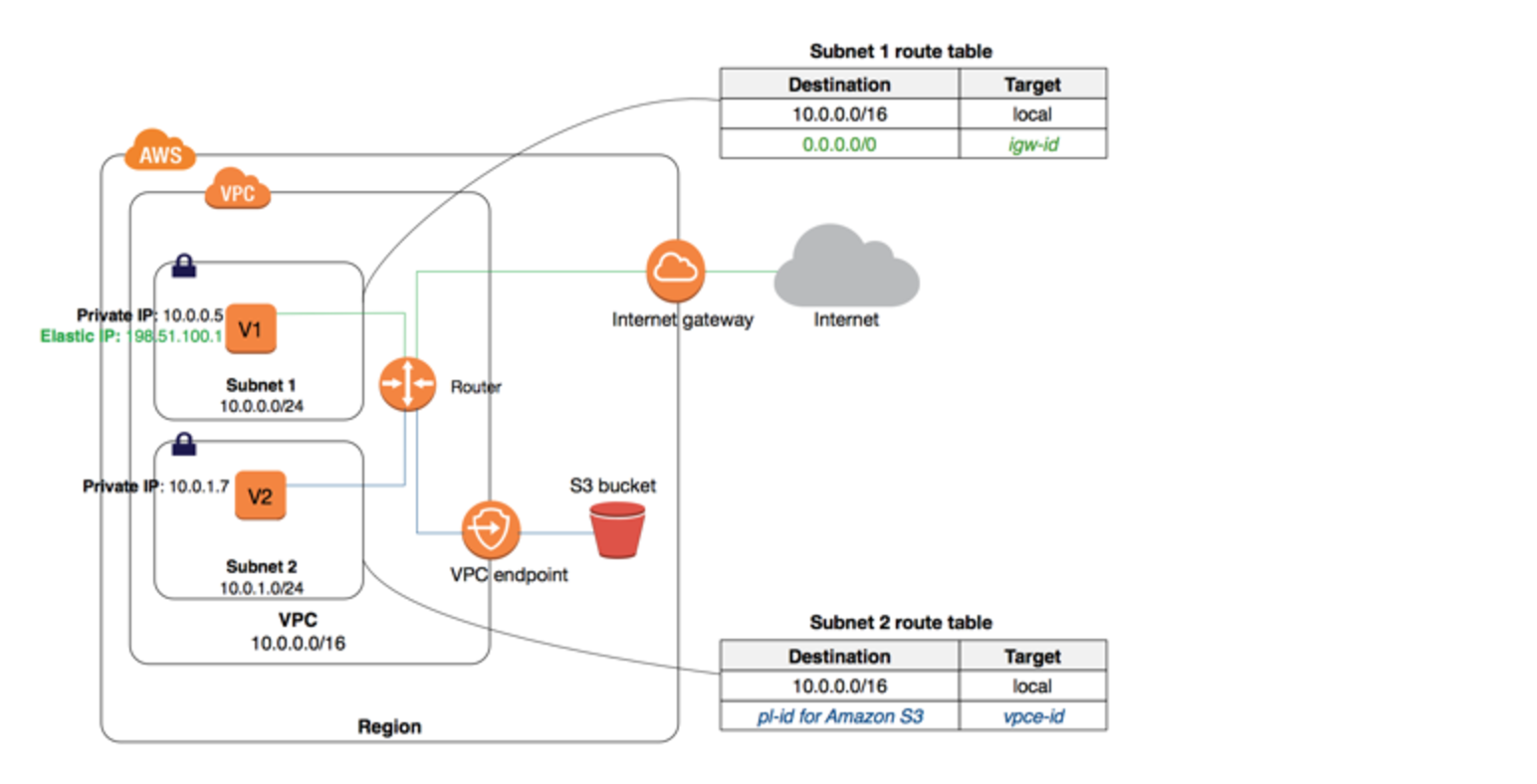
As you expand globally, inter-Region peering connects AWS Transit Gateways together using the [AWS global network](https://aws.amazon.com/about-aws/global-infrastructure/). Your data is automatically encrypted, and never travels over the public internet. And, because of its central position, [AWS Transit Gateway Network Manager](https://aws.amazon.com/transit-gateway/network-manager/) has a unique view over your entire network, even connecting to Software-Defined Wide Area Network (SD-WAN) devices.

As per the Demo you can create TGW and TGW attachments for each VPC and exchange the CIDR notations of your routing table of all VPCs.

**VPC End Points:**

VPC endpoint enables creation of a private connection between VPC to supported AWS services and VPC endpoint services powered by PrivateLink using its private IP address

VPC Endpoint does not require a public IP address, access over the Internet, NAT device, a VPN connection or AWS Direct Connect



As per the demo create End points.

A route is automatically added to the Route table with a destination that specifies the prefix list of service and the target with the endpoint id. for e.g. A rule with destination pl-68a54001 (com.amazonaws.us-west-2.s3) and a target with this endpoints’ ID (e.g. vpce-12345678) will be added to the route tables

Create EC2 instance in private region and also attach proper IAM role to access other AWS services (in above diagram example is S3 bucket) and try to access S3 bucket from your EC2 created on private region.

**VPC Flow Logs:**

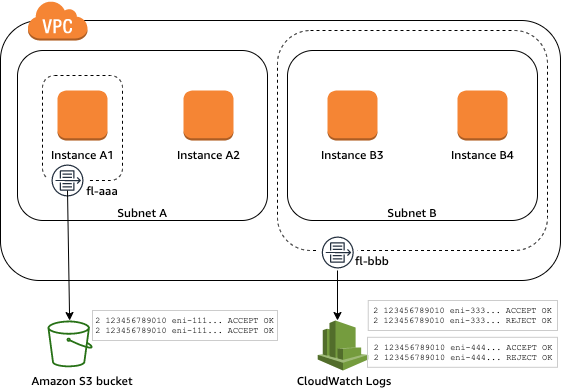
VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data can be published to Amazon CloudWatch Logs or Amazon S3. After you create a flow log, you can retrieve and view its data in the chosen destination.

Flow logs can help you with a number of tasks, such as:

* Diagnosing overly restrictive security group rules
* Monitoring the traffic that is reaching your instance
* Determining the direction of the traffic to and from the network interfaces

To create a flow log, you specify:

* The resource for which to create the flow log
* The type of traffic to capture (accepted traffic, rejected traffic, or all traffic)
* The destinations to which you want to publish the flow log data



**EC2 Pricing**

There are 4 ways to pay for an EC2 instance :

1. On-demand instance
2. Reserved instance
3. Spot instances
4. Dedicated hosts

**1.On-Demand Instance**

With On-demand instances, we pay for compute capacity per hour or per second depending on the instances which are being run.

No upfront payments are needed and we can increase or decrease the capacity whenever it is needed.

Challenges with On-Demand:

Monday: 500 customers using 16GB RAM on-demand servers individually.

Wednesday: 10 customers using 16GB RAM on-demand servers individually.

A “Cloud Service Provider” will not have a clear picture on how many servers should the provision. Too high → resources might unused and too low → money loss

**2.Reserved Instance**

Reserved Instance provides us with significant discount (upto 70%) compared to on-demand instance pricing.

Reserved instance are assigned to a specific availability zone and provides capacity reservation for AWS EC2 instances.

Example :

You know you will always be running 20 servers of m4.2xlarge type, then buy reserved instances for them.

**3.Spot Instances**

Spot instances allows us to bind on spare Amazon EC2 computing capacity for up to 90% of the on-demand cost.

Such instances are recommended for applications which can have flexible start and end times

**4.Dedicated Hosts**

A dedicated host is a physical EC2 server dedicated for your use.

It can be purchased on-demand as well as reserved instances.

**Types of EC2 Instances:**

Depending on the Performance, EC2 instances has divided into below types.

**General Purpose Instances:** A1 instance, M5, T3, T3a

**Compute Optimised Instances:** c5, C5n

**Memory-Optimised Instances:** R5/ R5a/ R5n instance R6g/ R6gd instance X1/ X1e instance

**Accelerated Computing Instances:** Accelerated Computing instances use additional hardware accelerators like Graphics Processing Units (or GPUs) and Field Programmable Gate Arrays (or FPGAs) that enable higher throughput in compute-intensive applications with more parallelism. For example, with GPU-powered instances, applications can access NVIDIA GPUs that have thousands of computing cores.

P2,P3,NF1,G3,G4,F1

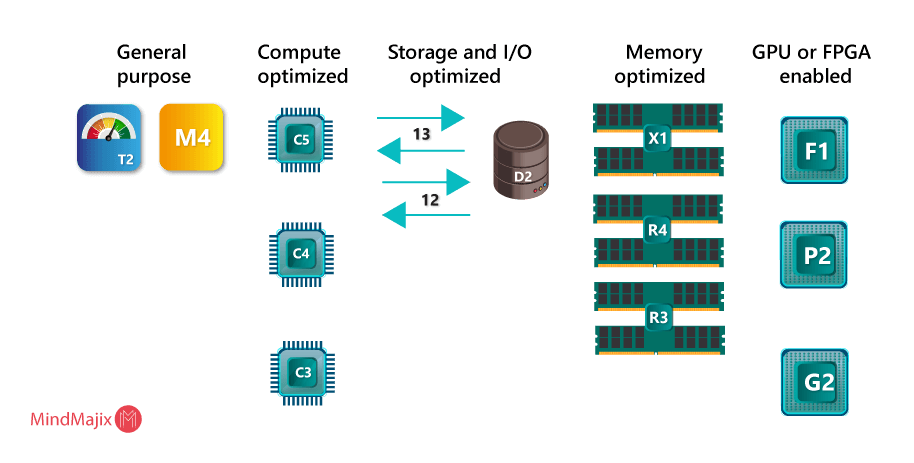
**Storage Optimised Instances:** D2, H1,I3/I3en

For more information about EC2 types please go through below link.

[Amazon EC2 Instance Types - Amazon Web Services](https://aws.amazon.com/ec2/instance-types/)

and

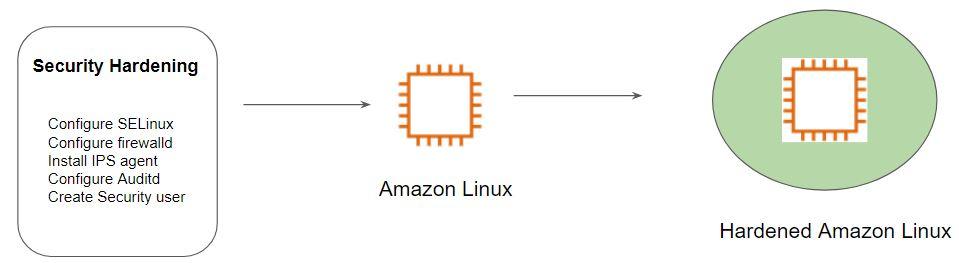
[EC2 Instance Types \* 2022 | A Complete AWS EC2 Instances info. (mindmajix.com)](https://mindmajix.com/aws-ec2-instance-types)



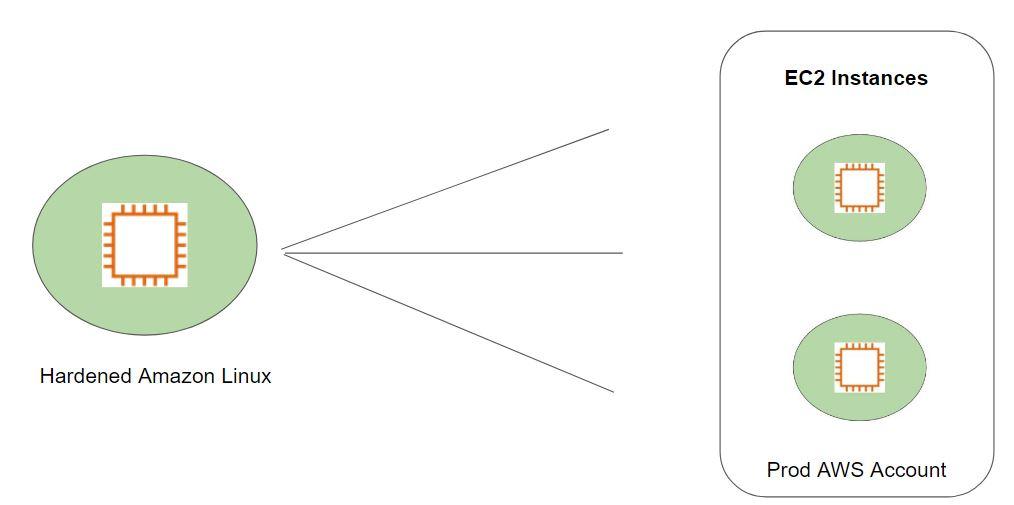
**Amazon Machine Image (AMI)**

Amazon Machine Image (AMI) is the master image from which new EC2 instances can be launched.

Let’s understand with an example:



The architecture of Hardened AMI Deployment:

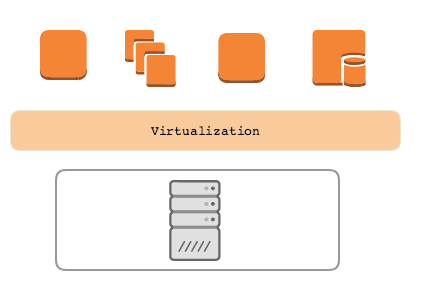


## **Placement Groups**

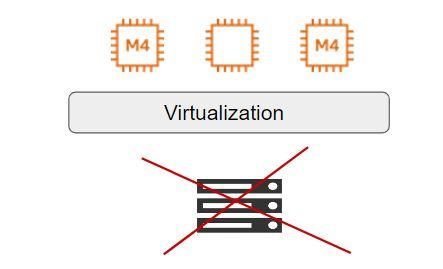
Point 1: Placement group are recommended for applications that require low latency, high network throughput.

Point 2: Placement groups can also be used to influence placement of a group of EC2 instances.

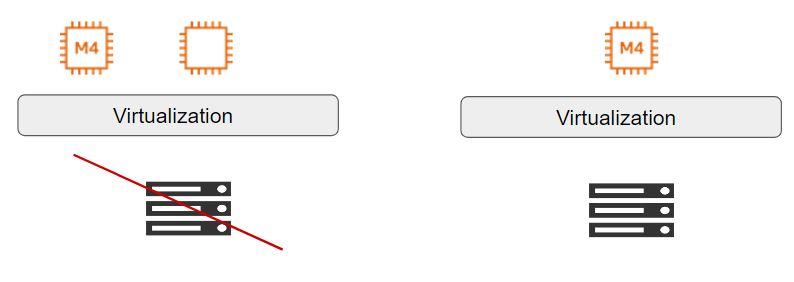
Point 2 can be better understood if you have worked with virtualization. In the below diagram, we see that there are multiple virtual machines running on a single hardware. The same can be the scenario where there are multiple EC2 instances running on same hardware.



When the VM architecture is in place, there is always risk specifically when two servers of the same cluster are running on the same underlying host. If the underlying server goes down then all VM running on top of it are down.



With Point 2, we can explicitly place two EC2 instances in a different hardware even when it is running in the same Availability Zone.



**Types of Placement Groups:**

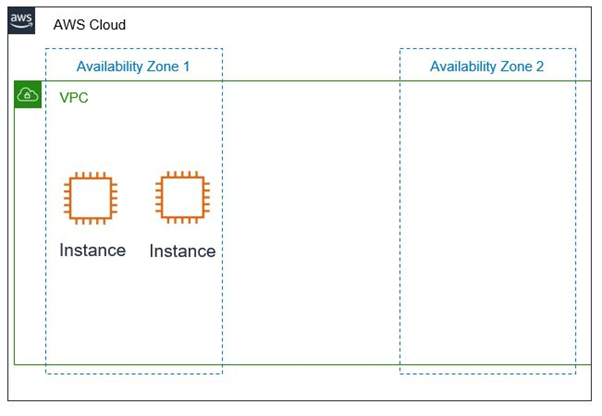
There are three types of placement groups available:

|  |  |  |
| --- | --- | --- |
| **Sr No** | **Type** | **Description** |
| 1 | Cluster | Packs instances close to each other in an Availability Zone. |
| 2 | Partition | Spreads instances in logical partition such that group of instances in one partition do not share underlying hardware. |
| 3 | Spread | Strictly places group of instances across distinct hardware to reduce failures. |

## **Cluster Placement Group**

## Logical grouping of instances within a single Availability Zone.

## Intended for applications that require low network latency and high network throughput.

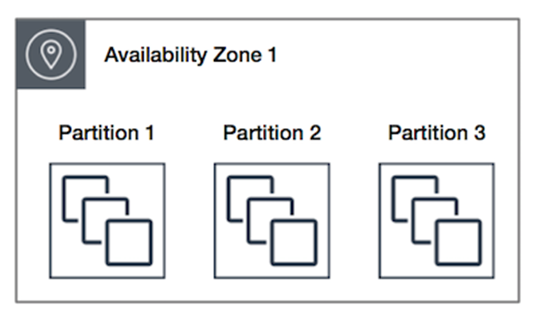


**Partition Placement Group**

AWS ensures that each partition within a placement group has its own set of racks.

## In the below diagram, there are 3 partitions and each partition has multiple EC2 instances.

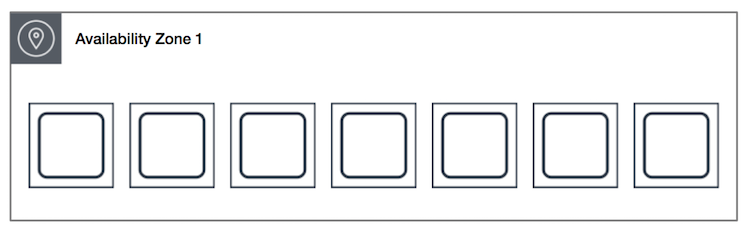
## Each of these partitions resides in a different rack inside the Datacenter.



## **Spread Placement Group**

## A spread placement group is a group of instances that are each placed on distinct racks, with each rack having its own network and power source.

## In the following diagram, there are 7 EC2 instances and each instance is in a separate rack.



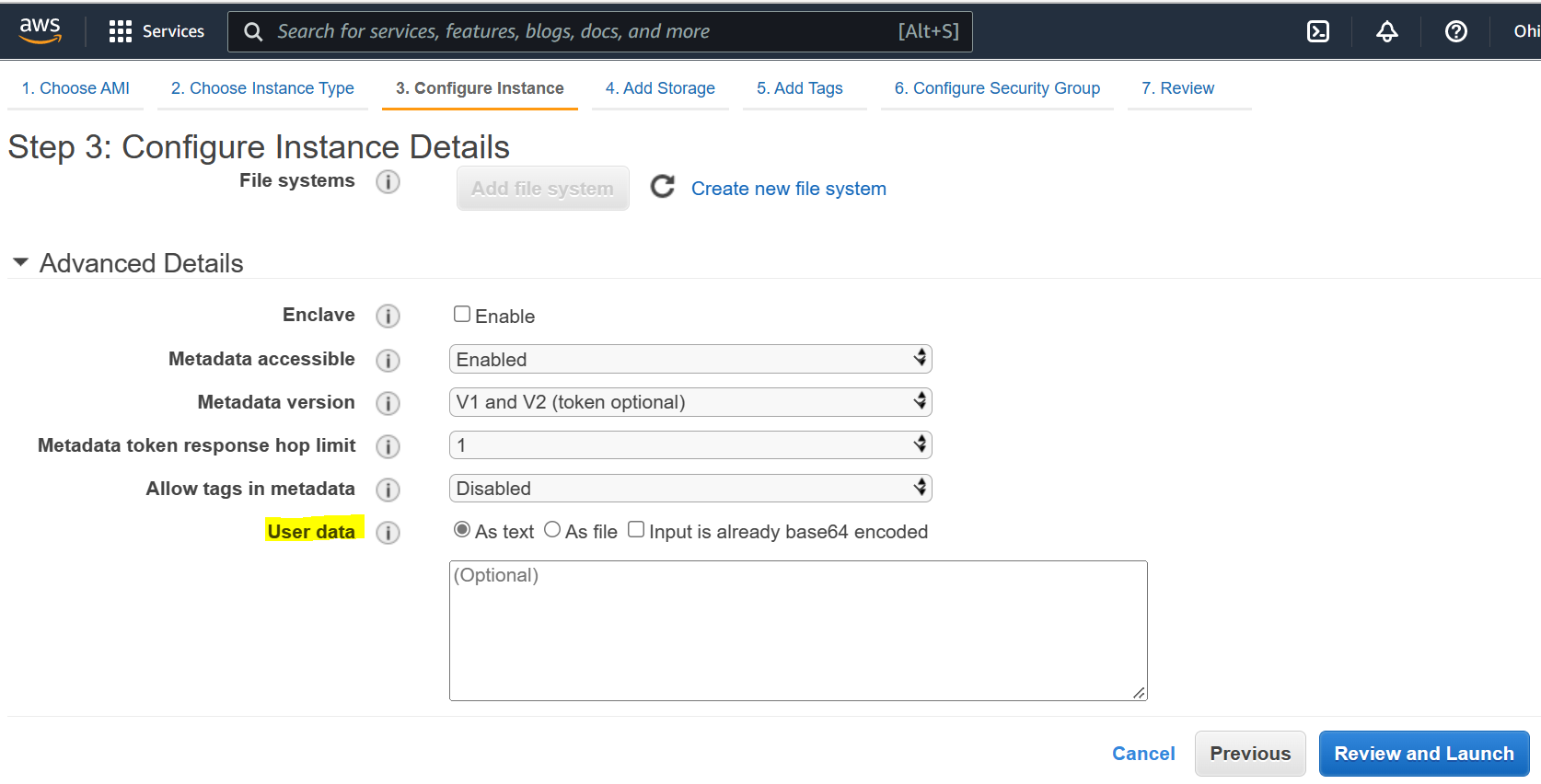
For more information about placement groups please go through below link:

[Placement groups - Amazon Elastic Compute Cloud](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html)

**Run commands on your Linux instance at launch:**

When you launch an instance in Amazon EC2, you have the option of passing user data to the instance that can be used to perform common automated configuration tasks and even run scripts after the instance starts. You can pass two types of user data to Amazon EC2: shell scripts and cloud-init directives. You can also pass this data into the launch wizard as plain text, as a file (this is useful for launching instances using the command line tools), or as base64-encoded text (for API calls).

To configure any script at the time of launching the instance, when your configuring the instance properties you can put your script in user Data as show below.



## **Launch Template**

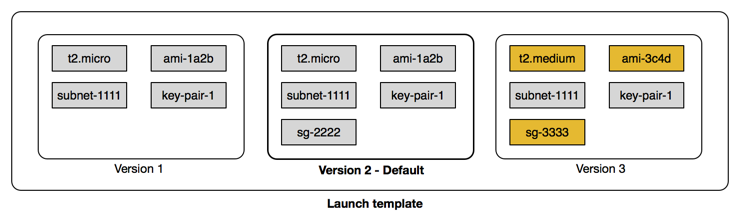
When you launch an EC2 instance, there are various configurations that needs to be set.

Some of the common configuration includes:

* AMI ID
* Instance Type
* Security Group
* Key Pair
* Storage
* IAM Role
* VPC

Everytime when you intend to launch instance, going through process is time consuming,

Launch templates enable you to store launch parameters so that you do not have to specify them every time you launch an instance.



## 

## 

**AWS Storage**

There are 3 types of Storage services provided by AWS

1. Block Storage
2. Object Storage
3. File Storage.

**Block Storage vs Object Storage:**

Overview of Block Storage

In block storage, the data is stored in terms of blocks

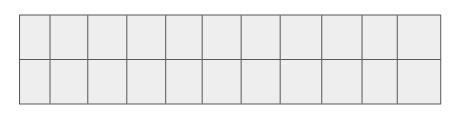
Data stored in blocks are normally read or written entirely a whole block at the same time

Most of the file systems are based on block devices.

Every block has an address and the application can be called via SCSI call via its address

There is no storage side meta-data associated with the block except the address.

Thus block has no description, no owner

****

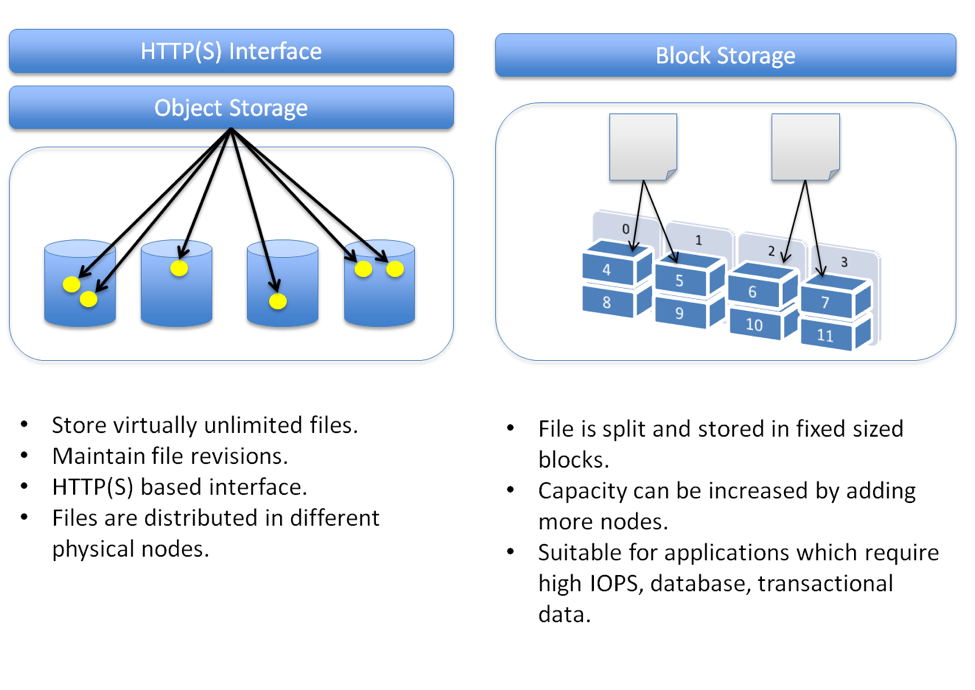
Overview of Object Storage

Object storage is a data storage architecture that manages data as objects as opposed to blocks of storage.

An object is defined as a data (file) along with all its meta-data which is combined together as an object.

This object is given an ID which is calculated from the content of the object (from the data and metadata ). The application can then call the object with the unique object ID.

Difference between Block and Object Storage

****

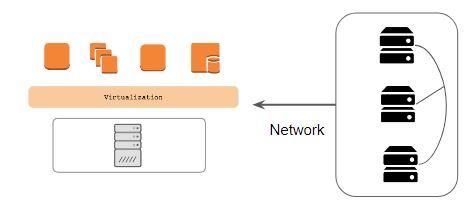
## **Elastic Block Store (EBS)**

Overview of EBS

AWS Elastic Block Store is a persistent block storage volume for use with EC2.

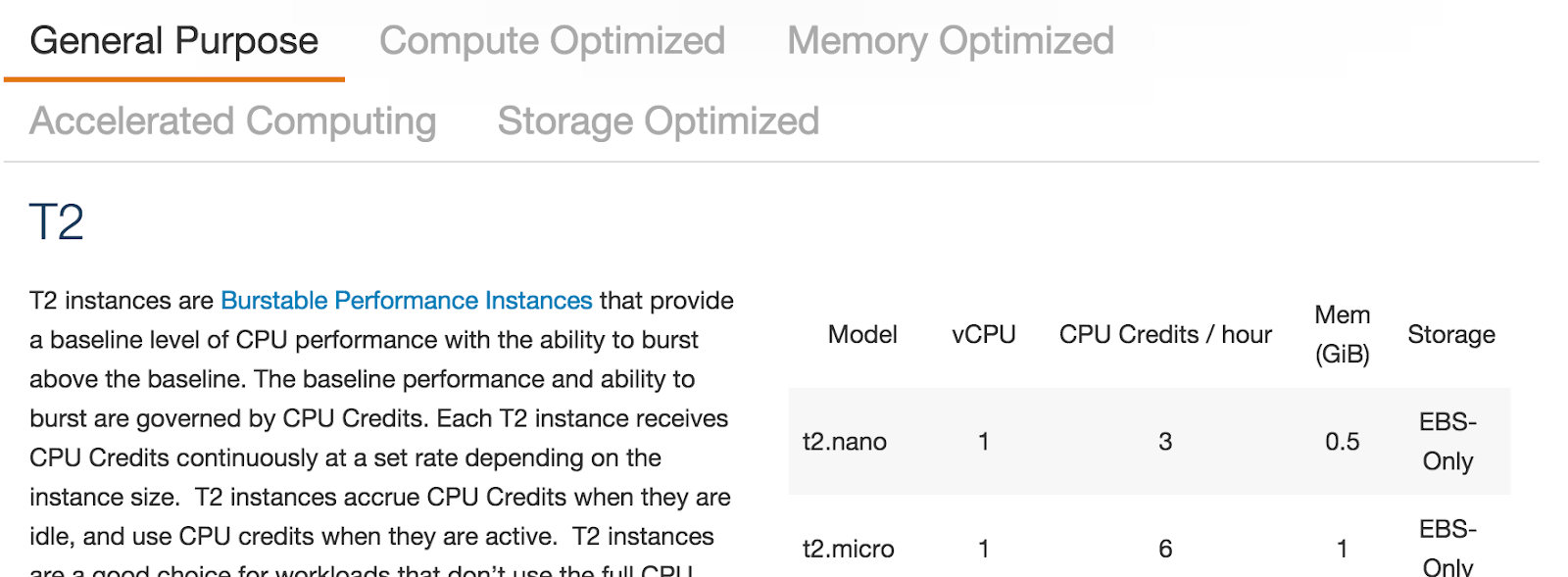
Each EBS volume is designed for 99.9999% availability & is automatically replicated within its availability zone.

EBS can be elastic in nature, thus supports the dynamic increase in capacity, performance and can change the instance type of live volumes.

****

AWS EC2 is regarded as a compute based service.

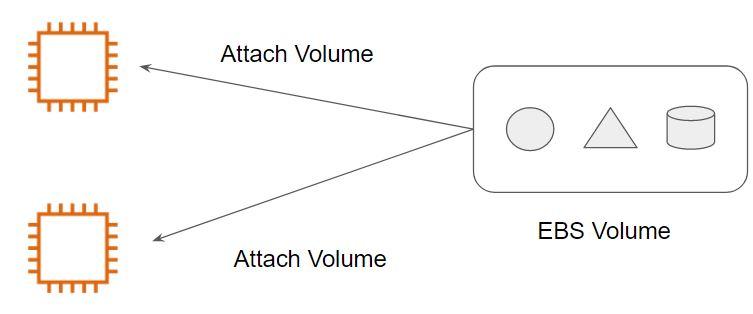
Compute generally refers to Memory & CPU

****

## **EBS Portability**

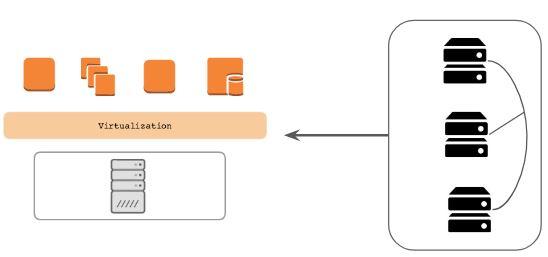
AWS Elastic Block Store is based on Network Attached Storage.

This allows customers to attach and detach EBS volumes easily and across EC2 instances.



AWS Elastic Block Store is based on Network Attached Storage.

Since the storage is attached via a network, they can be easily detached as well hence providing the portability based feature.



**EBS Volume Types**

Storage Device is a piece of equipment on which information can be stored

Common disk performance metrics are :

Input / Output operations per second ( IOPS )

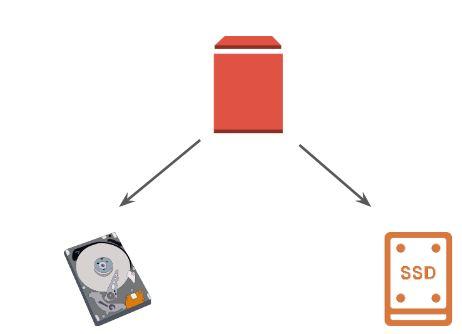
Throughput ( MB/s or MiB/s )

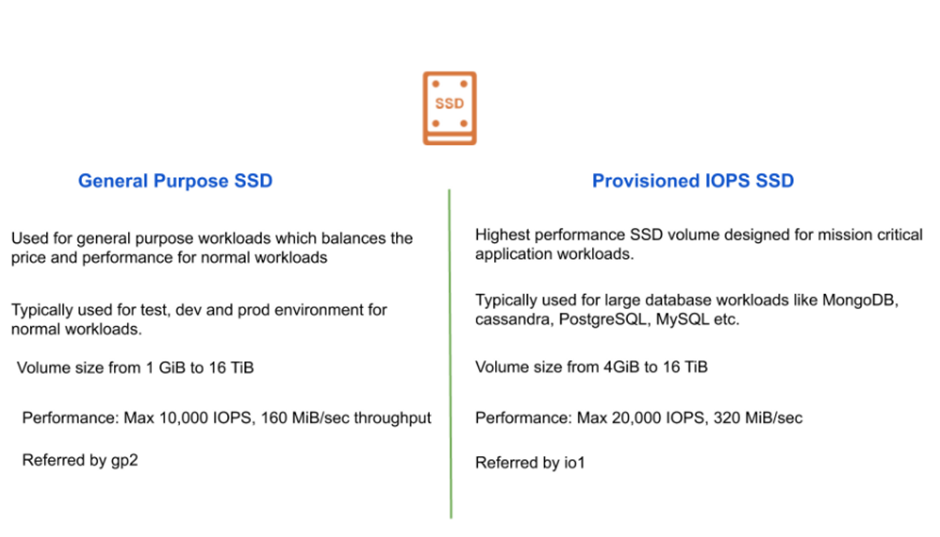
Response time ( ms )

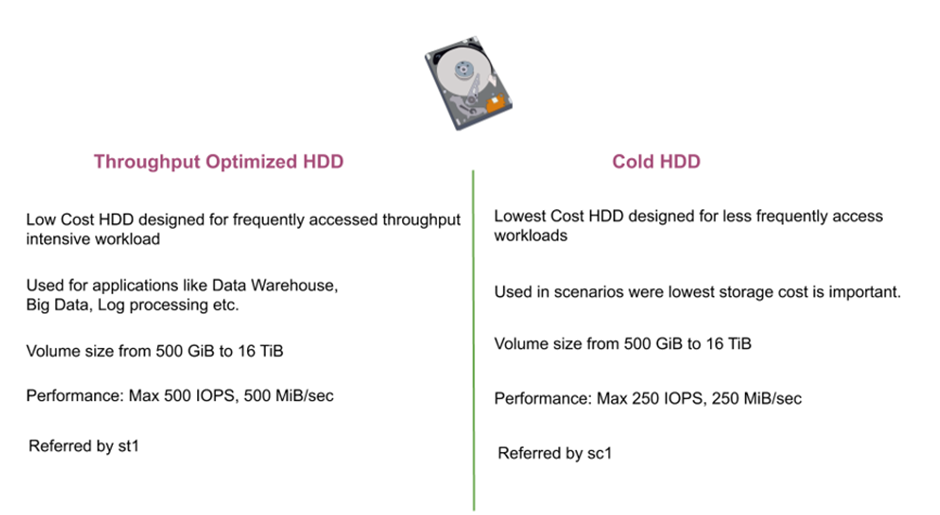


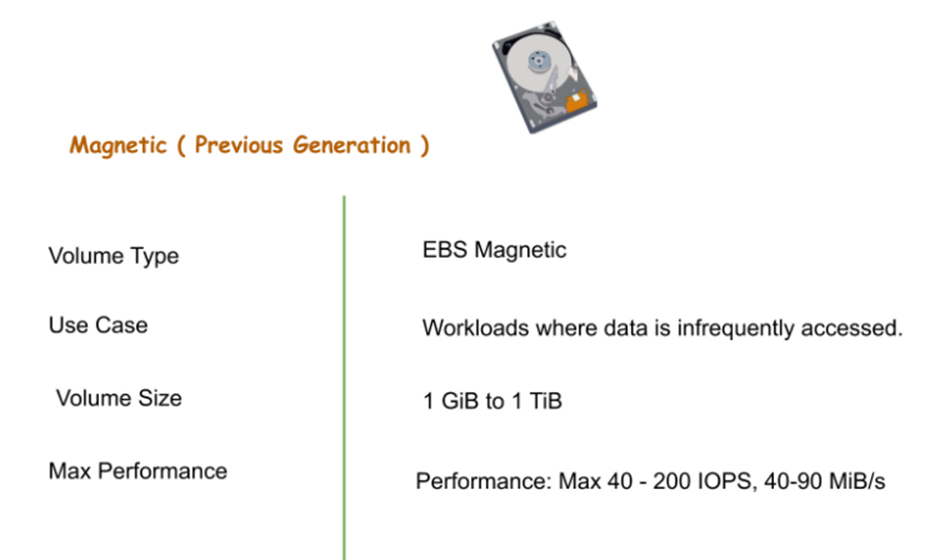
EBS provides different volume types that differ in performance and price.

This gives flexibility for a user to design and implement optimal EBS solutions.







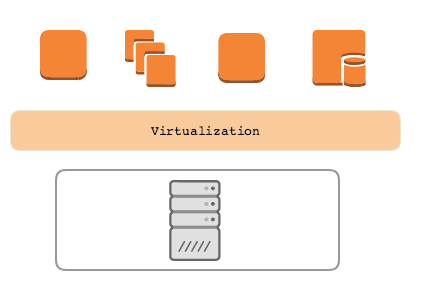
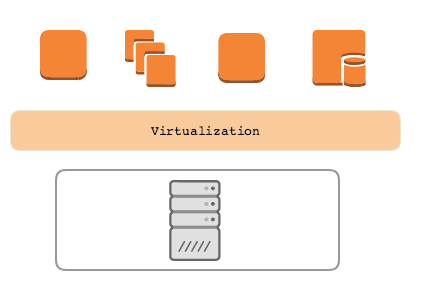


## **Instance Store:**

AWS Instance Store provides temporary block storage volumes for use with EC2.

This storage is located on the disks that are physically attached to the host computer.

The size of instance store varies depending on your instance type.

****

Important Pointers for Instance Store:

Data in instance store is lost in the following situation:

- The underlying disk drive fails.

- The instance stops.

- The instance terminates.

The instance store is included in the cost of EC2 instances, so they are quite cost-effective.

If planning to use an instance store, make sure you backup your data to central storage places like S3.

## **Simple Storage Service (S3)**

AWS S3 is object storage designed to store and retrieve any amount of data from anywhere

It is designed for 99.999999999% durability and 99.99% availability.

The thing that makes AWS S3 so powerful is the features that it comes preloaded with which are simply the best.

Let’s understand this with a use-case:

Large Corp is a payments organization and has more than 1000 servers. As being PCI DSS compliant, they must retain their logs for 1 year. It has been found that every day, the payment server generates logs of 200 GB. How to achieve this use case pertaining to the storage capacity in a cost-effective manner?

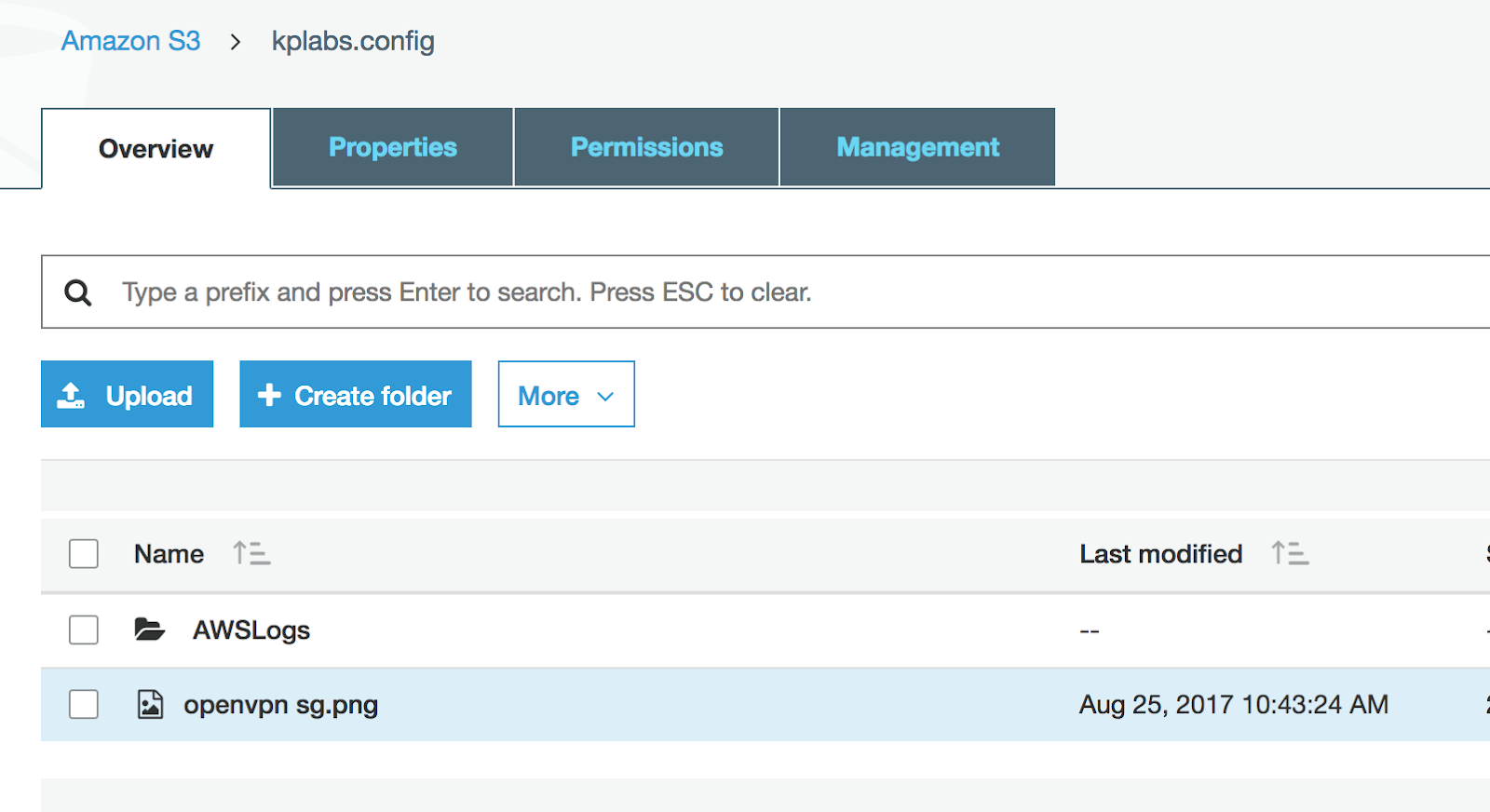


**S3 Terminology**

There are two important terminologies in AWS S3 :

* Buckets
* Objects

Buckets are like “Folders” where you can store multiple files (objects)



## **S3 Storage Classe**

S3 offers various kinds of storage classes for different use cases:-

* Standard
* Intelligent-Tiering
* Standard-IA
* One Zone-IA
* Glacier
* Glacier Deep Archive
* Reduced Redundancy

**Durability vs Availability**

Durability is percent ( % ) over one year period of time that the file which is stored in S3

will not be lost.

Availability is percent (%) over one year period of time that the file stored in S3 will

not be available.

Example:-

For Servers, Availability is one of the key metrics and any minute of downtime is a loss.

However, what happens if the component of the server itself fails and the server goes down?

**Understanding Every S3 Storage Class**

S3 Standard:

Amazon S3 Standard offers high durability, availability and performance for objects stored.

Designed for durability of 99.999999999% of objects ( eleven nines )

Designed for 99.99% availability over a given year

Example:-

If we have 10,000 files stored in S3 ( 11 nines durability ) then you can expect to lose one file every ten million years.

S3 Standard In-Frequent Access:

Amazon S3 Standard - Infrequent Access is for data that is accessed less frequently but requires rapid access when needed.

Designed for durability of 99.999999999% of objects

Designed for 99.90% availability over a given year

S3 Reduced Redundancy Storage (RRS)

AWS S3 Reduced Redundancy storage enables customers to reduce their costs by storing non-critical, reproducible data at lower levels of redundancy than Amazon S3’s standard storage

Designed for durability of 99.99% of objects

Designed for 99.99% availability over a given year

Glacier

* AWS Glacier is meant to be for archiving and for storing long-term backups.
* It may take several hours for the object to get restored.
* 99.999999999% durability of object.
* It is much cheaper than S3 ( very low cost )

Example Use Case:-

Backup of Application logs more than 1 year older can be moved to Glacier.

AWS S3 Intelligent Tiering

The S3 Intelligent Tiering is primarily designed to optimize cost by automatically moving data to the most cost-effective tier.

* General Purpose - Standard S3
* Infrequent Access - Standard IA

1TB of data stored in Standard S3 = 22.88$

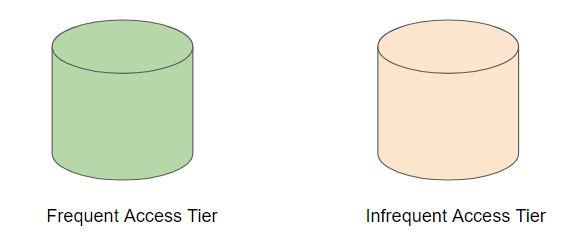
1TB of data stored in Standard IA = 12.50$

Organization stores terabytes of data in S3.

It will be great if a solution automatically moves infrequent data to Standard IA.

The S3 Intelligent Tiering works by storing data in one of the two access tiers:

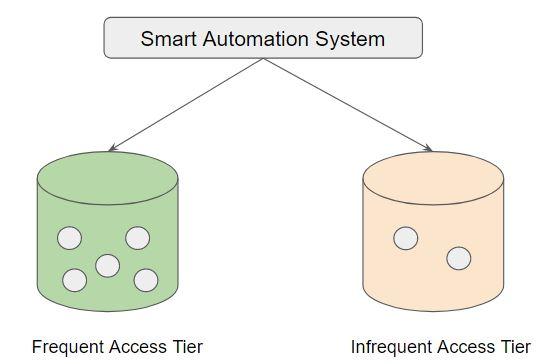
* Frequent Access Tier (Costly)
* Infrequent Access Tier (Much cheaper)



In this tier, the objects are automatically moved to frequent or infrequent access tier based on the access patterns.

Amazon S3 monitors access patterns of the objects in S3 Intelligent-Tiering and moves the ones that have not been accessed for 30 consecutive days to the infrequent access tier.

If an object in the infrequent access tier is accessed, it is automatically moved back to the frequent access tier.



This type of storage class is preferable for long-lived data with access patterns that are unknown or unpredictable.

S3 Intelligent-Tiering like other storage classes is configured at the object level.

1TB of data stored in Standard S3 = 22.88$

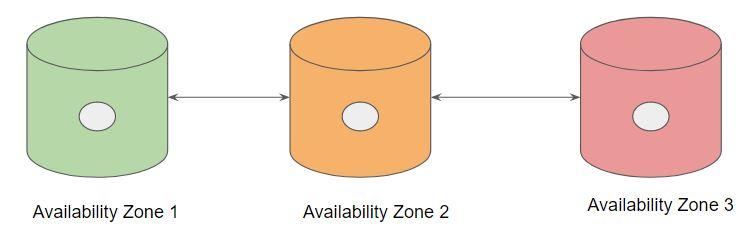
1TB of data stored in Standard IA = 12.50$

1 TB of data stored with Standard-Intelligent = 23$

One Zone Infrequent Access (One Zone IA)

Storage classes like Standard S3, Standard IA stores the data in a minimum of 3 availability zones.

Due to this, the overall cost per of storage is increased with such architecture.



S3 One Zone-IA stores data in a single AZ and costs 20% less than S3 Standard-IA.

It’s a good choice for storing secondary backup copies of on-premises data or easily recreatable data.

Data will be lost in-case of availability zone destruction.



Overview of Pricing comparison between storage classes:

* 1TB of data stored in Standard S3 = 22.88$
* 1TB of data stored in Standard IA = 12.50$
* 1 TB of data stored in One Zone IA = 10$

Glacier Deep Archive

S3 Glacier Deep Archive is Amazon S3’s lowest-cost storage class and supports long-term retention and digital preservation for data that may be accessed once or twice in a year.

All data stored in S3 Glacier Deep Archive can be restored within 12 hours.

On the contrary, Glacier is ideal for archives where data is regularly retrieved and some of the data may be needed in minutes.

Pricing Comparison:

1 TB of data stored in Glacier: 14$

1 TB of data stored in Glacier Deep Archive: 10.99$

## **S3 Lifecycle Policies**

We now understand that there are various S3 Storage classes offered by S3.

We need to make that storage durable + affordable for long term storage.

* We can store 1 months of logs in Amazon S3 Standard.
* Move the logs older than 1 months to S3 Standard IA
* Move the logs older than 6 year to Glacier

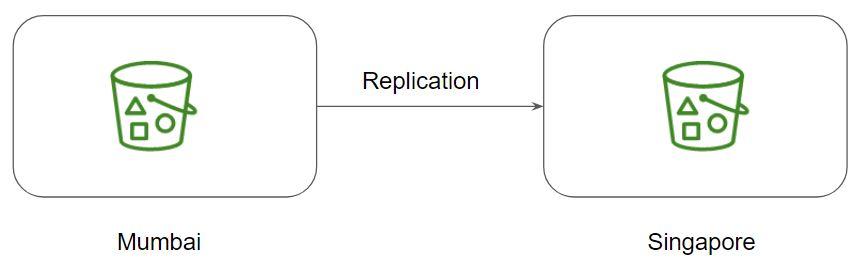
These automatic moving of data to S3 storage classes can be achieved with the help of S3 Lifecycle policies.



**S3 Cross-Region Replication**

Many compliances has a requirement that the data must be replicated across greater distances.

Cross-Region Replication allows data from S3 buckets to be replicated across regions.



Both source and destination buckets must have versioning enabled.

**S3 - Static Website Hosting**

AWS S3 allows us to host static websites.

The static website includes individual webpages which might include static content.

Note:

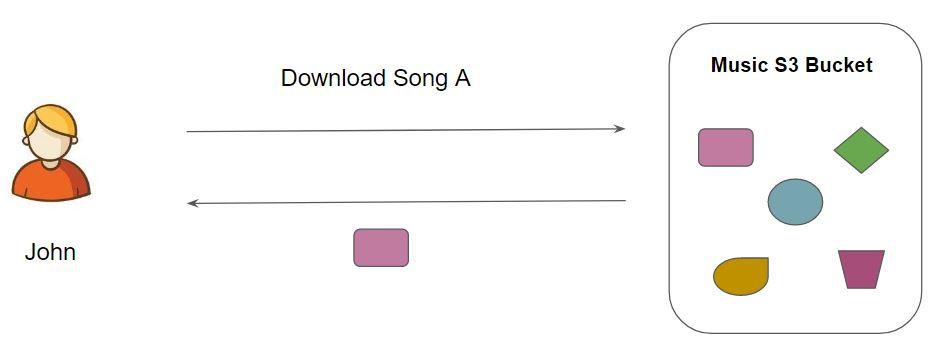
AWS S3 does not support dynamic websites that rely on server-side scripts like PHP, JSP, ASP.NET, and many more.

**S3 - Presigned URLs**

Use-Case of Music Company

Company ABC is an online Music selling company. Once the user purchases a song, he should be able to download the song. Your company has decided to store all of its song data in S3 due to its highly durable option.

How will you go ahead with this scenario?



Understanding Presigned URL

All objects in S3 are ‘Private’ by default.

However, Object owner can optionally share objects with others by creating a pre-signed URL to grant time-limited permission to download the object.

Achieving the Use Case:-

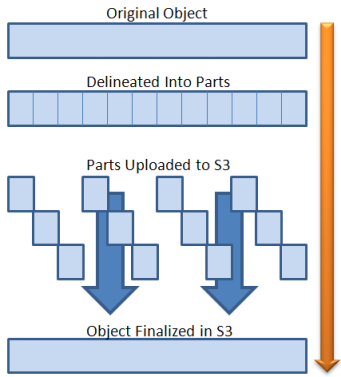
After a user purchases a song and requests to Download, the application should generate a pre-signed URL that will allow the ‘MP3’ file stored in S3 to be downloaded by the user.

[presign — AWS CLI 2.4.23 Command Reference (amazonaws.com)](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/s3/presign.html)

**Multi-Part Uploads**

Multi-Part upload is a way in which we upload an entire file in the form of small individual chunks to the storage device.

While uploading data via multi-part, we need to specify the part number and its position in the uploaded object. This will help AWS reconstruct data.



For more information to do practicals please go through below link

[Use the AWS CLI for a multipart upload to Amazon S3](https://aws.amazon.com/premiumsupport/knowledge-center/s3-multipart-upload-cli/)

**High Availability**

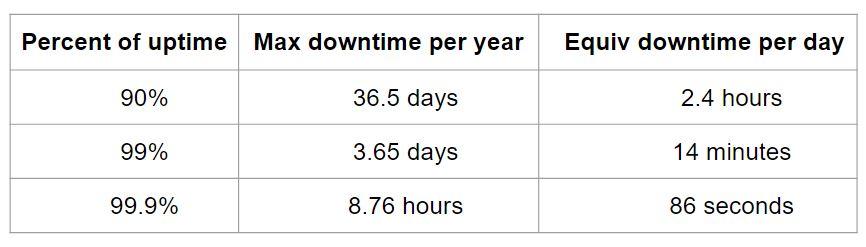
## **Overview of High-Availability**

High Availability is a characteristic of the system design which makes resources available even in the case of any component failure in a computer system.

Availability refers to the amount of time that the system is in functioning condition

.

General Availability: 100% minus system downtime



Events that might disrupt the system’s availability can occur at any time, we need to make sure that the application is designed to be highly available.

Below is the website to do the calculator of above.

[SLA & Uptime calculator: How much downtime corresponds to 99.9 % uptime](https://uptime.is/)



## **RTO/RPO**

**Recovery Time Objective:**

Recovery Time Objective (RTO) is the amount of time frame it takes for you to recover your infrastructure and business operations after a disaster has struck.

Sample Example:

If RTO is 3 hours, then one needs to invest quite a good amount of money to make sure the DR region is always ready in-case the main region goes down due to disaster.

**Recovery Point Objective:**

Recovery Point Objective (RPO) is concerned with data and the maximum tolerance period to which data can be lost.

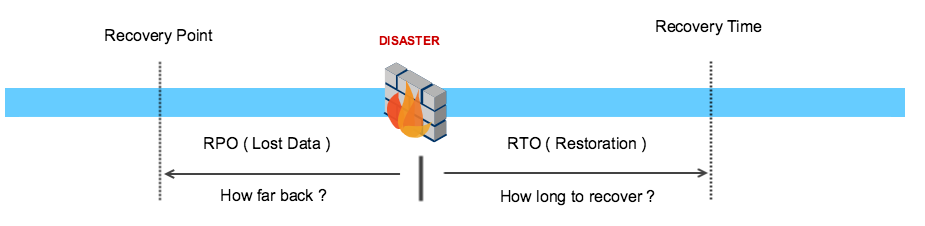
It helps in determining how well we should be designing the infrastructure.

Sample Example:

If RPO is 5 hours for the database, then we should be taking backup of the database every five hours.

**RTO vs RPO**

RTO is a more broader scope and covers the whole business and systems involved while RPO is more directly related to the interval of backup to take to avoid data loss.



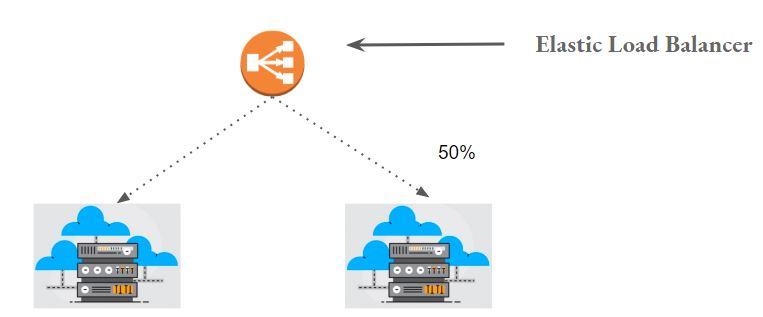
## [Website Performance and Availability Monitoring | Pingdom](https://www.pingdom.com/)

## **Elastic Load Balancers (ELB)**

Elastic Load Balancer allows us to distribute traffic to multiple EC2 instances thus avoiding a single point of failure.

ELB is capable of handling rapid changes in network traffic patterns.

Since it’s a managed service, the client does not have to worry about the high availability related aspects.

****

## **Load Balancers Types**

AWS currently offers 4 major types of Load Balancers:

1. Classic Load Balancers.
2. Network Load Balancers.
3. Application Load Balancers
4. Gateway Load Balancer

[Network Traffic Distribution – Elastic Load Balancing – Amazon Web Services](https://aws.amazon.com/elasticloadbalancing/features/)

Classic Load Balancers were the old generations and are recommended if you still have instances in EC2 classic.

**Classic Load Balancer**

These are the older generation of load balancers which can be used for instances both under the VPC and the EC2-Classic network.

Provides basic set of features for HTTP, HTTPS, TCP, and SSL protocols.

The newer updates and configurations no longer come to classic load balancers.

Important Limitation of Classic Load Balancer:

* Does not support native HTTP/2 protocol.
* IP address as targets are not supported.
* Path based routing not supported. (eg: /images directed to server 1 & /php to server 02)
* Server Name Indication (SNI) is not supported.
* Load Balancing to multiple ports on the same instance.

**Application Load Balancers**

Application Load Balancers are the next generation load balancer by AWS.

They support the HTTP and HTTPS protocol.

New Features:

* Path and Host-Based Routing
* Register IP as targets.
* SNI support.
* Load Balancing to multiple ports on the same instance.

There are two type of conditions which can be set:

i) Path-Based Conditions

ii) Host-Header based condition

Path-based conditions allow to forward traffic to an appropriate target group based on the path mentioned within the request URI.

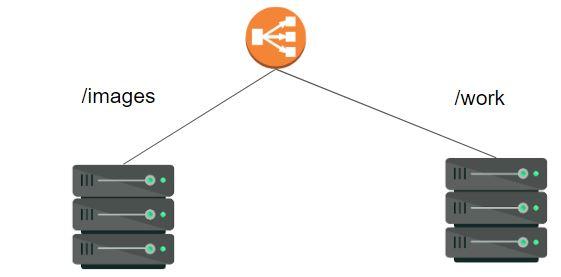
The host-based condition allows forwarding traffic to the appropriate target group based on the host header received from the client’s request.

Path-Based Routing in ALB

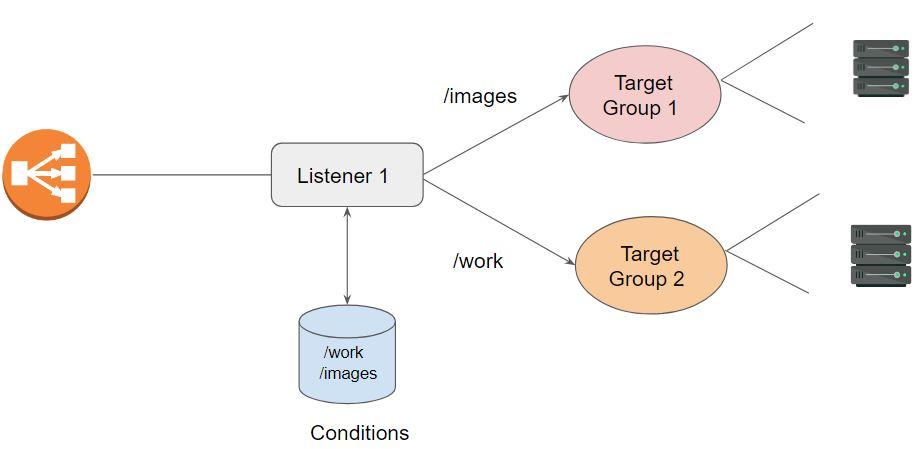
The request is routed based on the URI path.

example.com/images/ → server 01

example.com/work/ → server 02

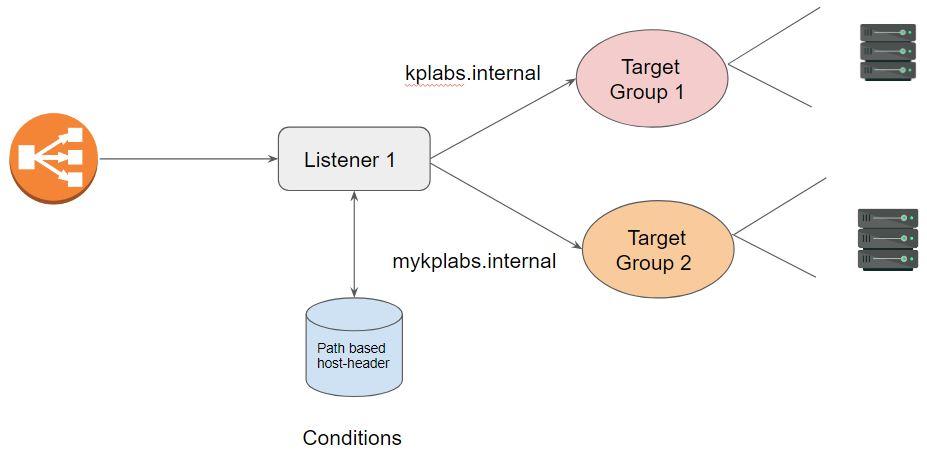


A low-level diagram based on path-based routing would look like the following:



Host Based Routing

A request is redirected based on the host-header field.

****

**Network Load Balancers**

Network Load Balancer works on the four-layer of the OSI model.

NLB supports TCP protocol.

NLB selects target based on flow-hash algorithm which is based on:

* IP address, destination port and TCP sequence number.

Each individual TCP connection is routed to a single target for the lifetime of the connection.

**Two Important Advantages of NLB:**

* Ability to handle volatile workloads and scale to millions of requests per second.
* Supports static IP addresses. We can use Elastic IP now

## **ELB Health Checks**

ELB will continuously poll the server to check if the app is running

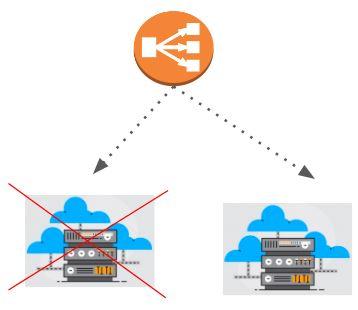
If it gets 200 response from the server, it will consider server UP

Timeout Interval: 5 seconds

Unhealthy Threshold: 2

Healthy Threshold: 10

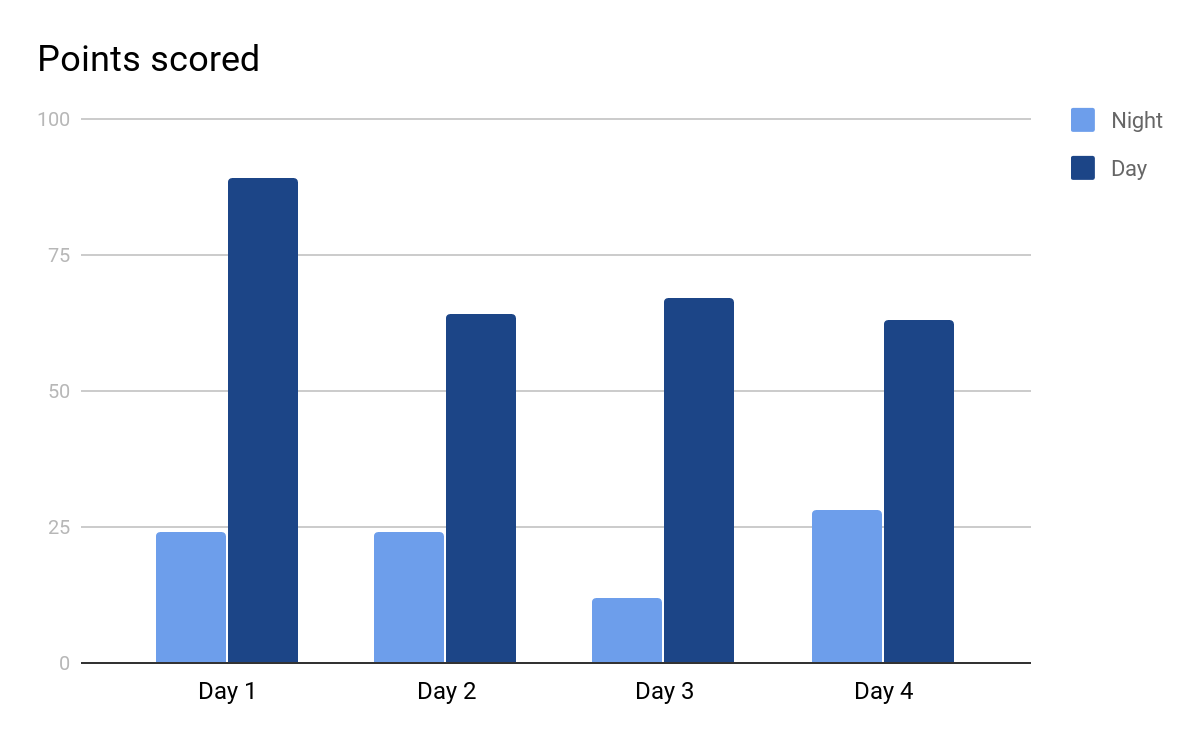
Interval: 30 seconds



## **Introduction to Auto Scaling**

Scalability is the ability of a system to change in size depending on the needs.

Infrastructure should scale to support changes in traffic patterns.



What if new servers automatically get launched on high load?

Simple Scaling Policy:

Base: 2 servers

Scalable :

If average CPU utilization > 60%; add two more instance

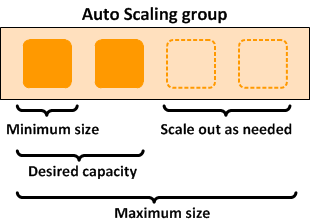
If average CPU utilization < 30%; remove two instance

1.1 Understanding AWS EC2 Auto Scaling

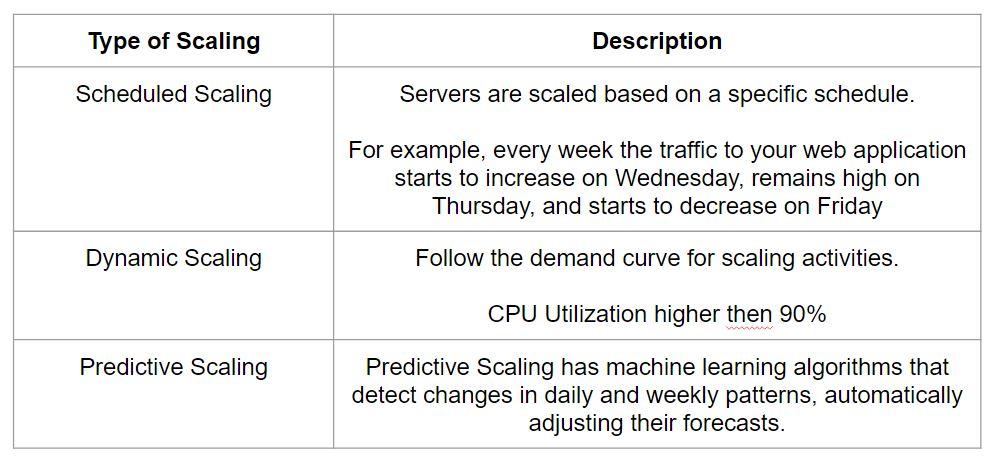
Amazon EC2 Auto Scaling helps you maintain application availability and allows you to automatically add or remove EC2 instances according to conditions you define.

Example Scenario:

* Minimum: 2 EC2 instance
* Maximum: 10 EC2 instance
* Threshold: 50% of CPU



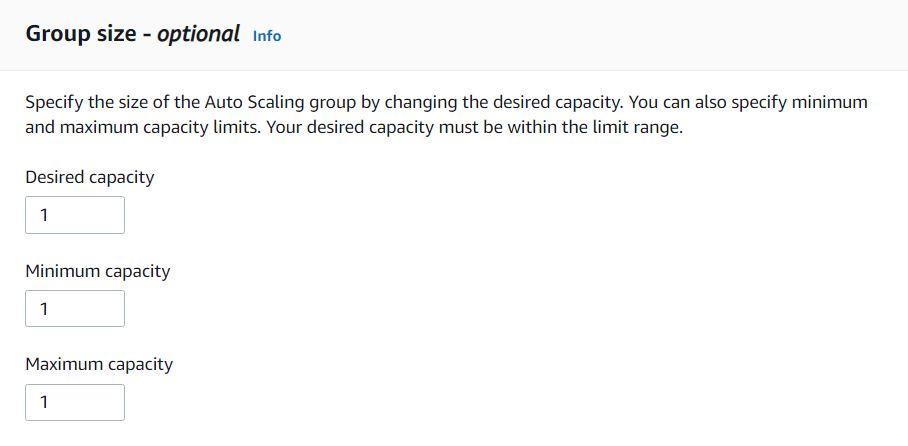
Multiple Types of Scaling



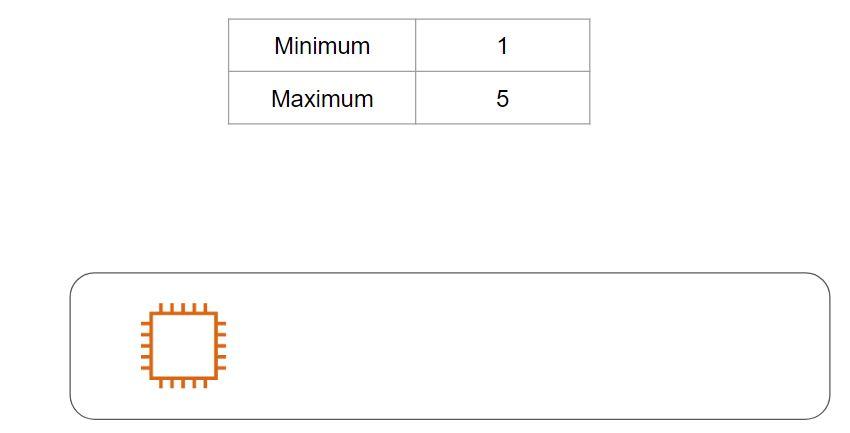
## 

## **Implementing Auto-Scaling**

While configuring auto-scaling, you set the minimum, maximum, and desired capacity.



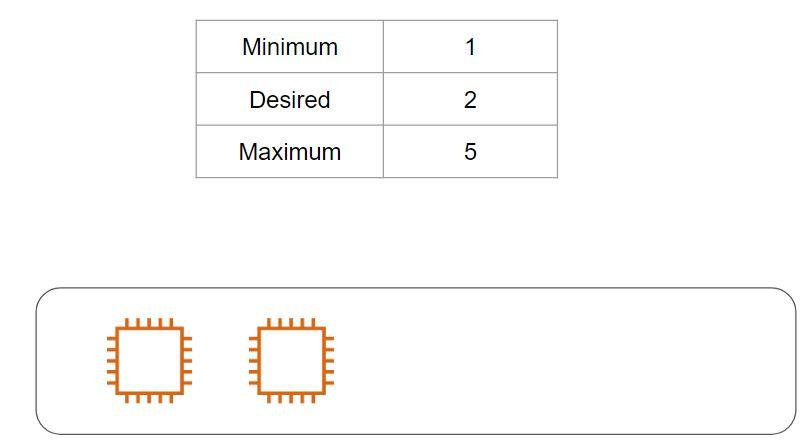
Minimum Capacity



Maximum Capacity



Desired Capacity

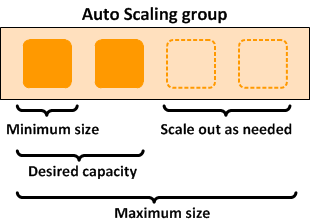


Auto Scaling Group Size

Minimum denotes the minimum number of EC2 instances that will run.

Maximum denotes the maximum number of EC2 instances that can be scaled.

Desired capacity denotes the number of Amazon EC2 instances that should be running in the group.



Launch [Configuration/Template]

Contains instance configuration information, some of these include:

* AMI ID
* Security Groups
* Instance Types
* Key Pair

It is recommended to use the Launch Template instead of Launch configuration.

Health Checks

Amazon EC2 Auto Scaling can determine the health status of an instance using one or more of the following:



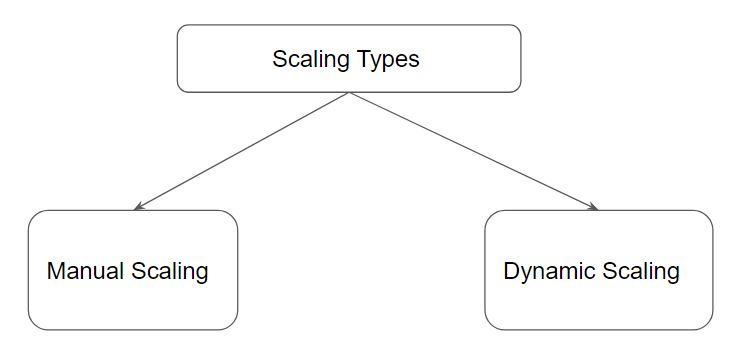
Manual Scaling

Manual scaling is the most basic way to scale your resources, where you specify only the change in the maximum, minimum, or desired capacity of your Auto Scaling group.

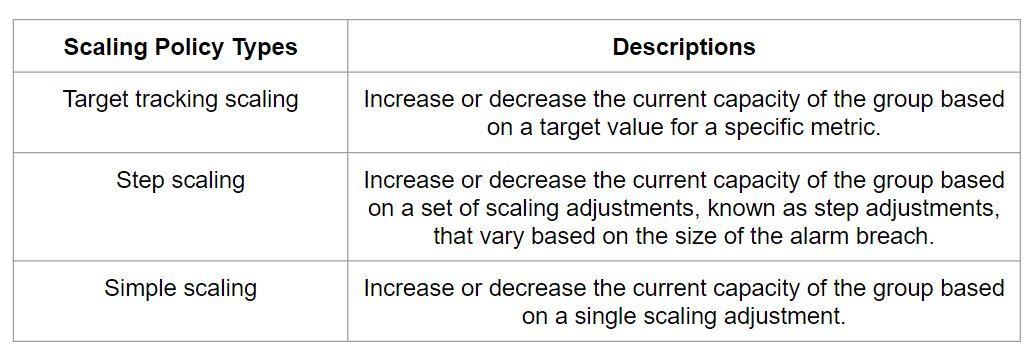
Amazon EC2 Auto Scaling manages the process of creating or terminating instances to maintain the updated capacity.

## **Dynamic Scaling**

There are two primary types of scaling approaches that are available:

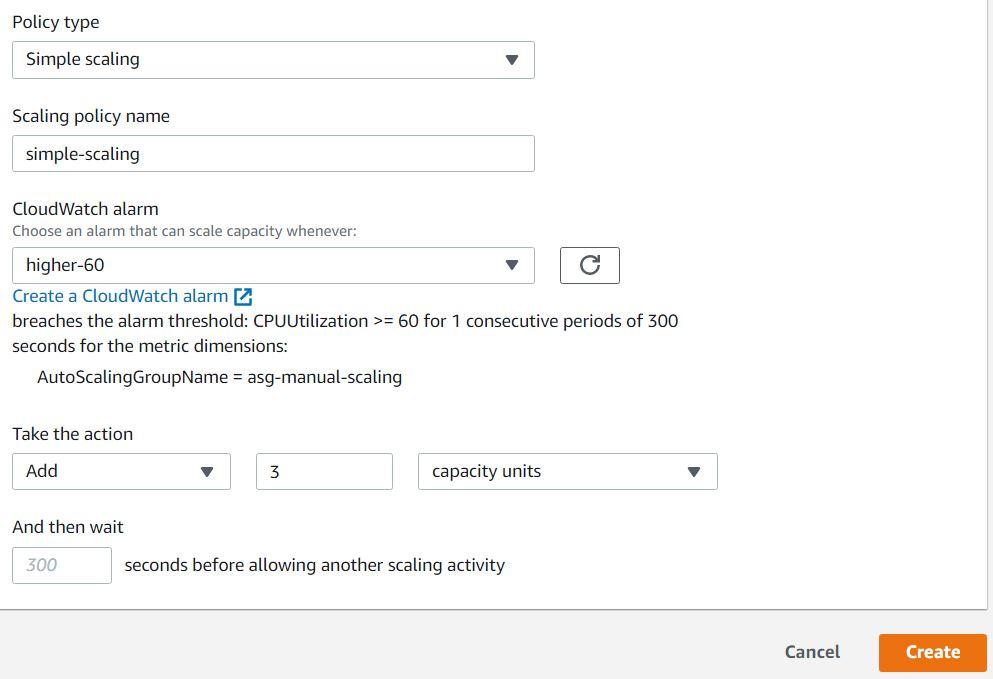


When you configure dynamic scaling, you define how to scale the capacity of your Auto Scaling group in response to changing demand.



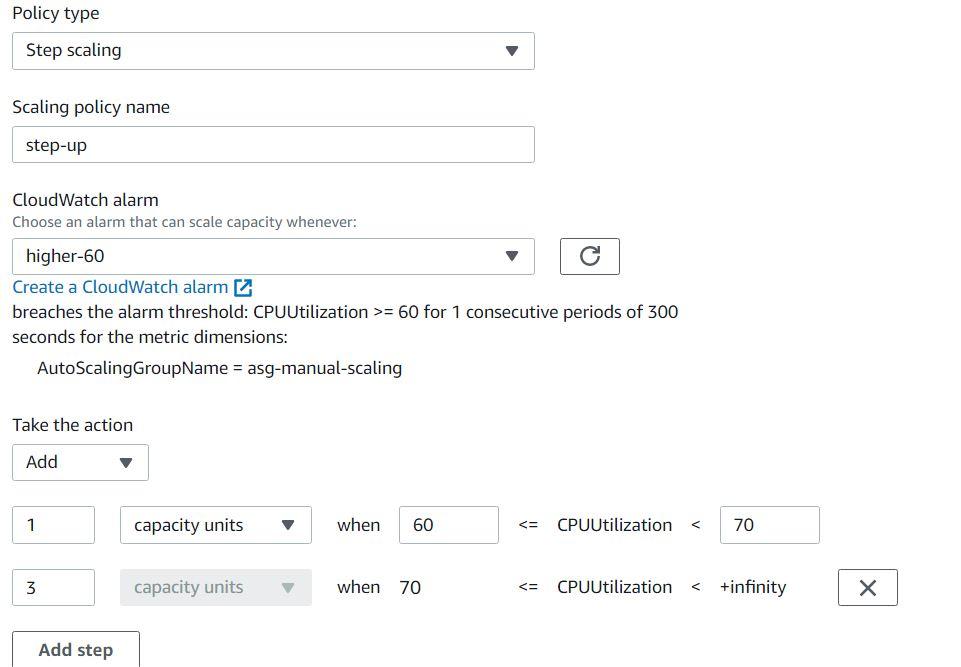
Simple Scaling Policy

With a simple scaling policy, you can configure a specific number of instances to be added when a threshold reaches a certain value.



Step Scaling Policy

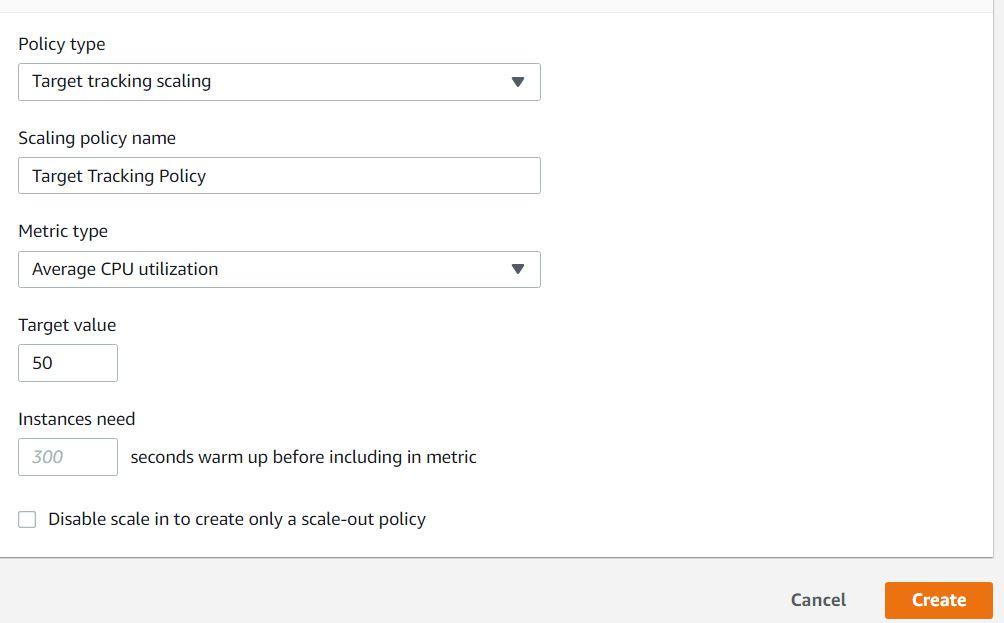
In step scaling, the adjustment of the current capacity of instances varies based on the size of the alarm breach.



Target Tracking Policy

With target tracking scaling policies, you select a scaling metric and set a target value.

The scaling policy adds or removes capacity as required to keep the metric at, or close to, the specified target value.

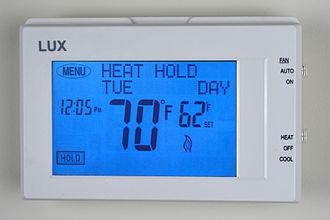


Use Case of Thermostat

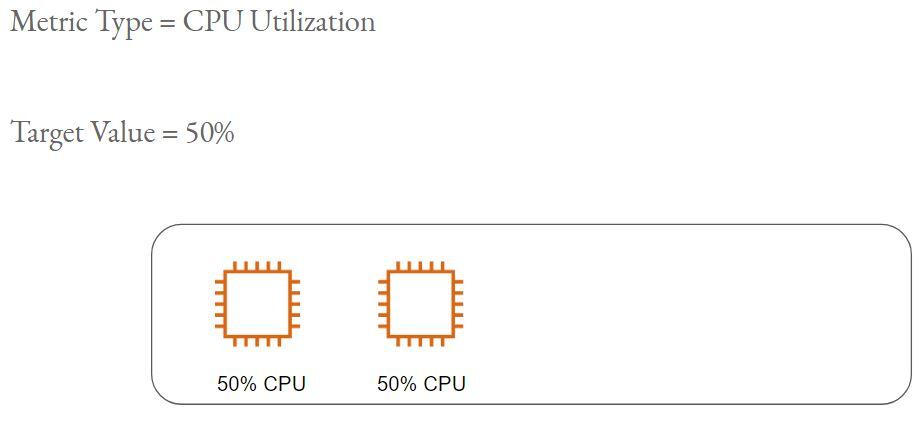
A thermostat is a component that senses the temperature of a physical system and performs actions so that the system's temperature is maintained near a desired setpoint.

Example:

* Desired = 24
* Current = 18



Example Target Tracking Policy

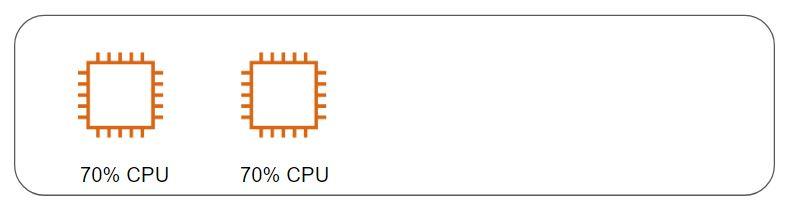


Example Target Tracking Policy - 2

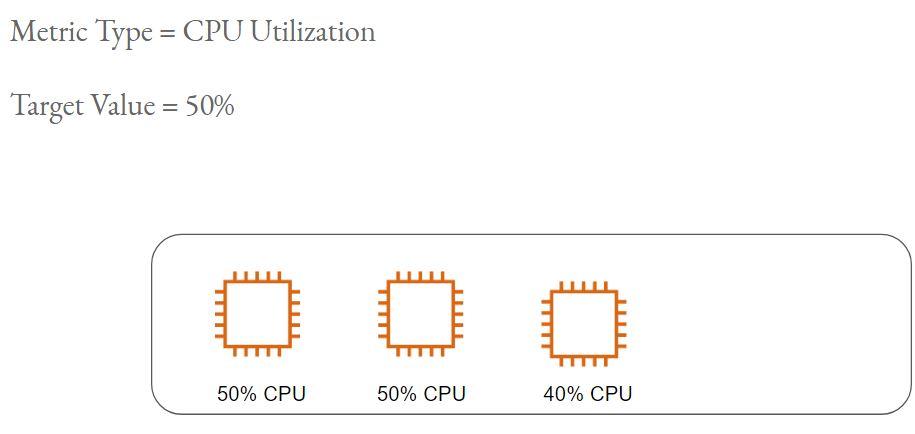
Metric Type = CPU Utilization

Target Value = 50%

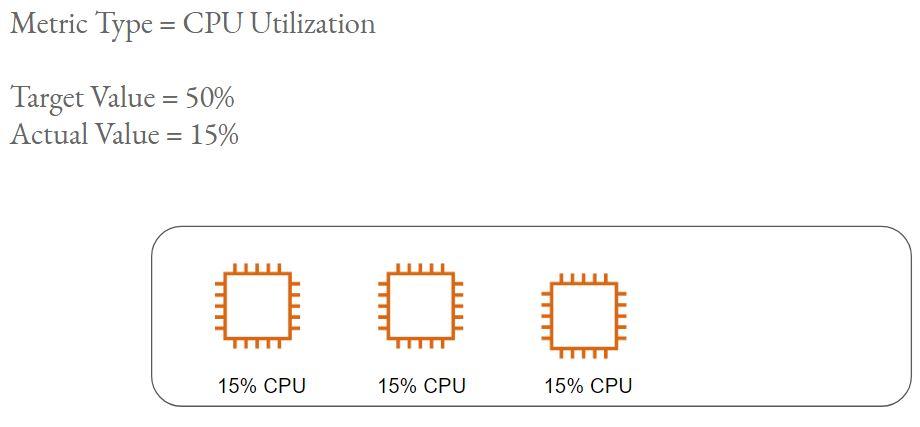
Actual Value = 70%



Example Target Tracking Policy - 3



Example Target Tracking Policy - 4



## **Scheduled Scaling**

Scheduled scaling allows you to set your own scaling schedule.

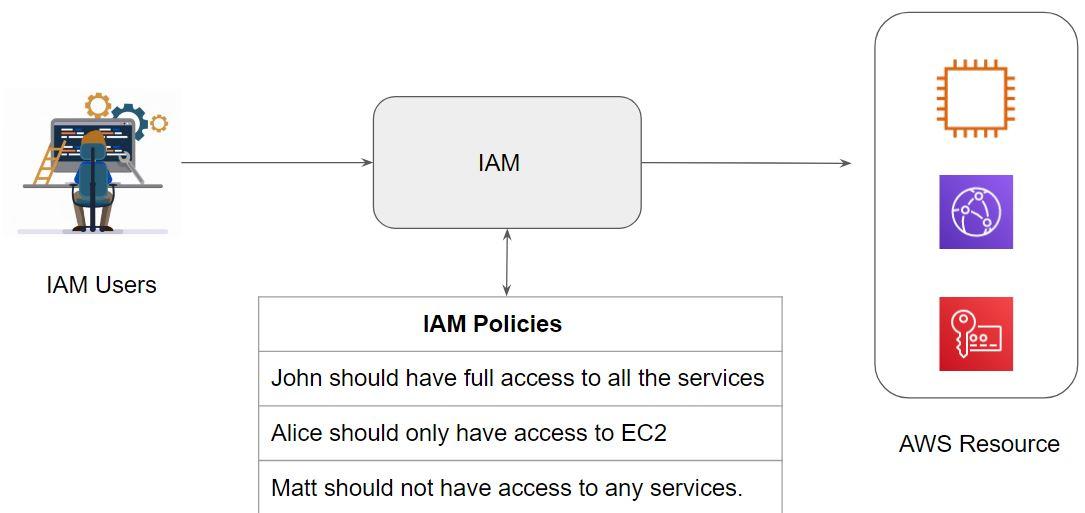
For example, let's say that every week the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday.

Scaling actions are performed automatically as a function of time and date.

**IAM (Identity and Access Management)**

## **Introduction to IAM**

Identity and Access Management is a framework of policies for ensuring that the proper people in an enterprise have the appropriate access to technology resources.



Let us understand some of the key IAM related terminologies.

IAM User

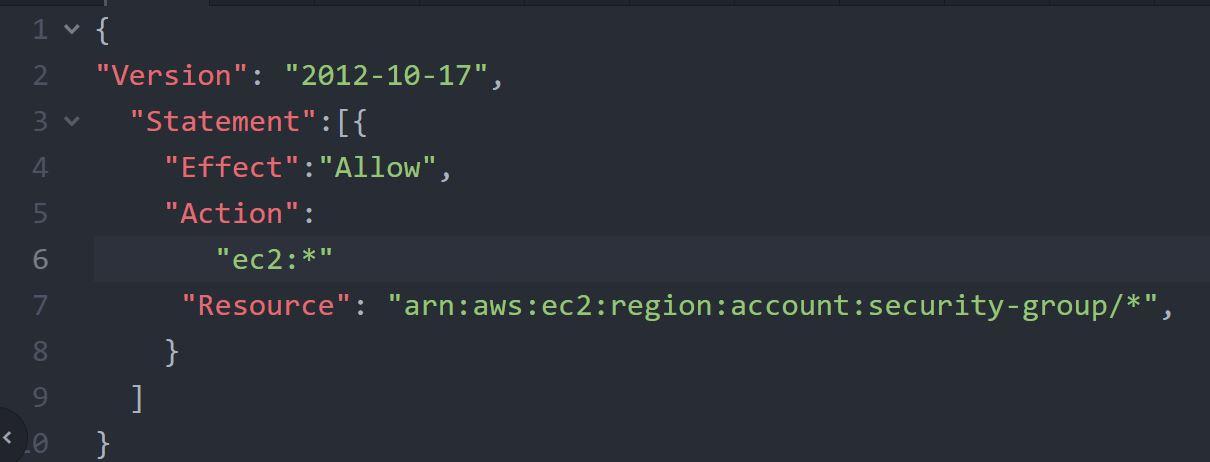
IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

By default, the IAM user does not have any permission associated with it.

IAM Policy

IAM Policy is an object in AWS that defines the permission of a specific object.

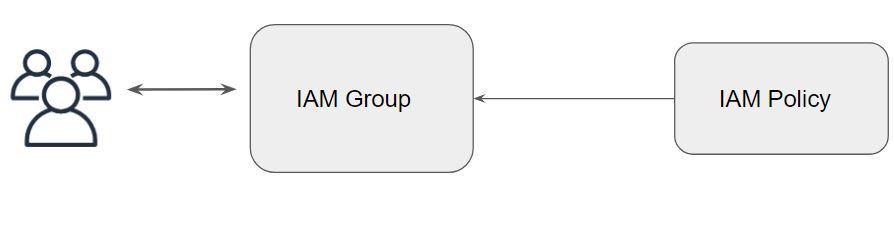
AWS evaluates the policies, and depending on that, the permission is granted or denied.



IAM Groups

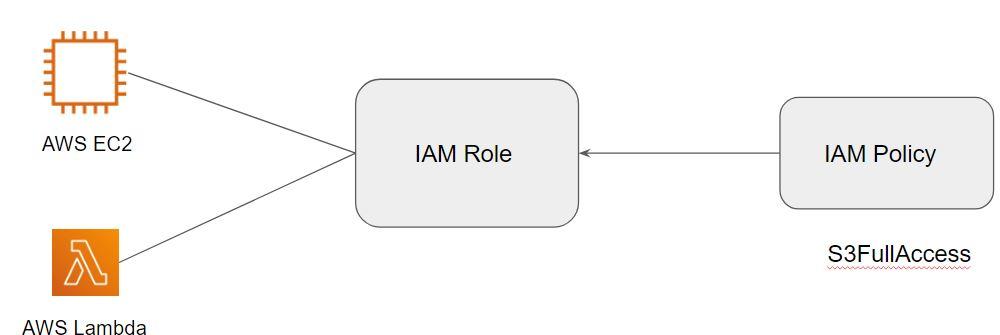
An IAM group is a collection of IAM users.

IAM policy attached to the group is associated with all the users that are part of the group.



IAM Role

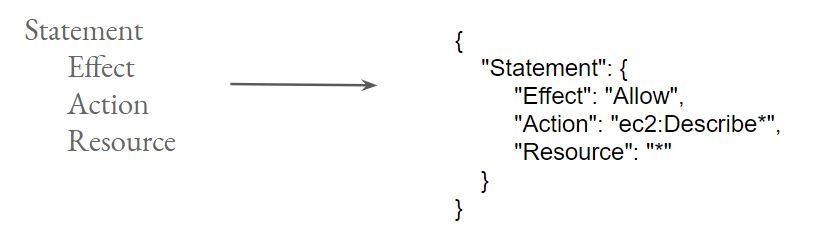
An IAM role is similar to an IAM user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS.



## **IAM Policies**

IAM Policies allows us to define at granular level access on what permissions need to be given to access a particular AWS resource.

There are 4 important elements of IAM Policy :

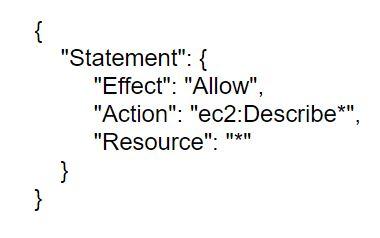


Statement Element

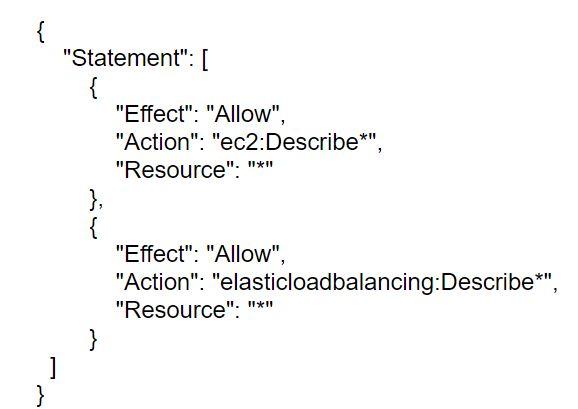
The statement element is the main element for the IAM policy.

This element is a must.

The statement element can contain multiple individual statements in order of [ { .. } {...} . Each of the individual statement is enclosed in blocks of { }



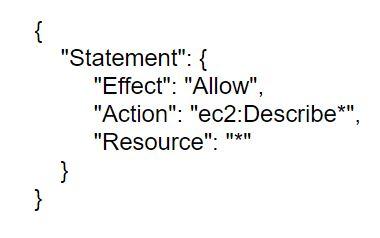
A policy can also consist of multiple statements.



Effect Element

Effect basically specified whether the statement is allowed or explicitly denied.

There are two possible values: Allow or Deny



Action Element

Action element defines a list of actions that will be allowed or denied.

Each AWS service has its own set of actions.

Example :

* ec2:CreateKeyPair
* ec2:CreateVpc
* sqs:ListQueues
* sqs:SendMessage
* s3:CreteBucket
* s3:DeleteBucket

Resource Element

The resource element defines the object that the statement covers.

Amazon Resource Name (ARN) uniquely identified AWS resources.

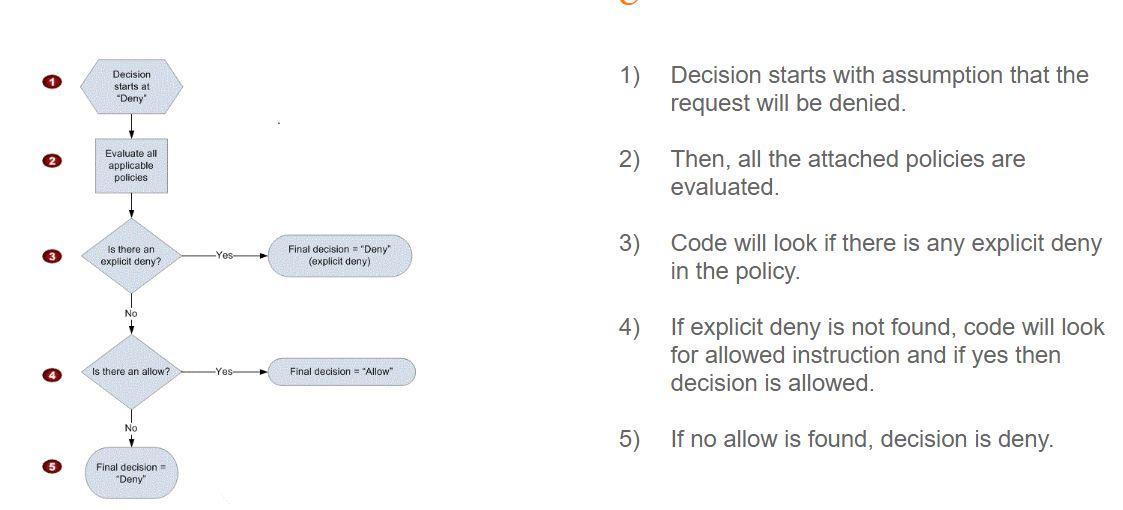
Example :

arn:aws:ec2:us-west-2:836802967410:instance/i-0067dce1525f98ab2

The above ARN is based on the following format:

arn:aws:ec2:region:account-id:instance/instance-id

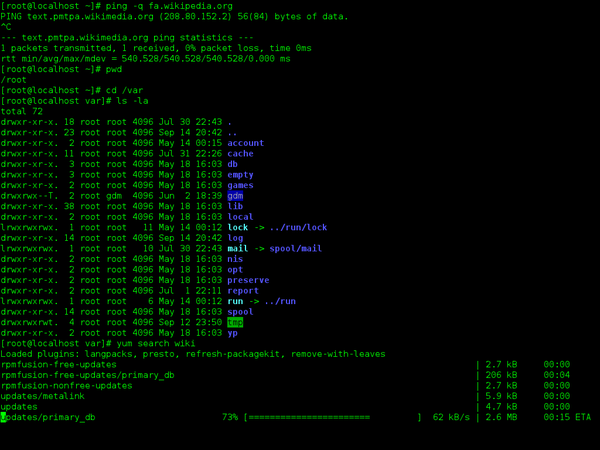
## **IAM Policy Evaluation Logic**



## **AWS CLI**

Command Line Interface ( CLI ) is a way of interacting with the system in the form of commands

It is considered as the fastest way of doing things in a repeated, automated fashion.



GUI vs CLI -

* Create a directory called a TEST
* Inside it, create three text file named one.txt, second.txt and third.txt
* The contents in each one of them would be “this is kplabs demo”
* The permission of all these files should be 600

The above use-case can be automated via CLI and can be repeated hundreds of times with just a click of a button. Doing things GUI way will take more time.

AWS CLI is used for managing AWS resources from the terminal.

It makes room for automation & makes things much faster.

To start interacting with AWS resources using CLI, you will need the following things:

* Access / Secret Key of ROOT or an IAM user.
* AWS CLI Package installed.
* Configure AWS CLI package with “aws configure” command.

## 

## **IAM Role**

IAM Role contains a set of policies and any entity assuming that role will be able to have permissions mentioned in the role.

A role can be used by :

* An IAM user
* A web service offered by the AWS such as EC2
* An external user authenticated by external IdP service compatible with SAML etc.

