**AWS For DevOps**

What is Cloud Computing

[**https://aws.amazon.com/what-is-cloud-computing/**](https://aws.amazon.com/what-is-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 like Amazon Web Services (AWS).

Cloud Computing video from Azure

<https://www.youtube.com/watch?v=txZrgdehLaw>

Who is using cloud computing?

Organizations of every type, size, and industry are using the cloud for a wide variety of use cases, such as data backup, disaster recovery, email, virtual desktops, software development and testing, big data analytics, and customer-facing web applications. For example, healthcare companies are using the cloud to develop more personalized treatments for patients. Financial services companies are using the cloud to power real-time fraud detection and prevention. And video game makers are using the cloud to deliver online games to millions of players around the world.

# Benefits of cloud computing

* Agility
* Elasticity
* Cost saving
* Deploy globally in minutes

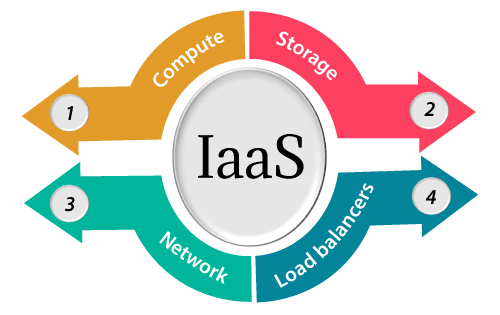
# Types of cloud computing

* Infrastructure as a Service (IAAS)
* Platform as a Service (PAAS)
* Software as a Service (SAAS)

**Infrastructure as a Service (Iaas)**

Iaas is also known as Hardware as a Service (HaaS). It is one of the layers of the cloud computing platform. It allows customers to outsource their IT infrastructures such as servers, networking, processing, storage, virtual machines, and other resources. Customers access these resources on the Internet using a pay-as-per use model.

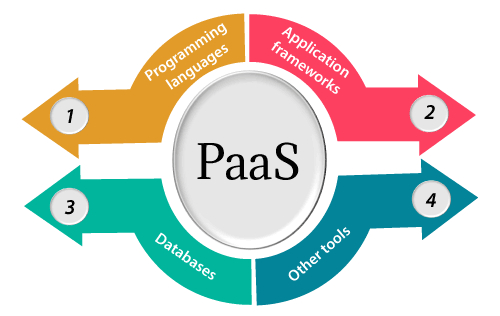
IaaS is offered in three models: public, private, and hybrid cloud. The private cloud implies that the infrastructure resides at the customer-premise. In the case of public cloud, it is located at the cloud computing platform vendor's data center, and the hybrid cloud is a combination of the two in which the customer selects the best of both public cloud or private cloud.



**Platform as a Service (PaaS)**

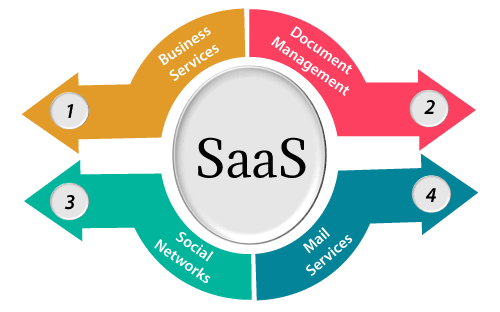
Platform as a service (PaaS) is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications. You purchase the resources you need from a [cloud service provider](https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/choosing-a-cloud-service-provider/) on a pay-as-you-go basis and access them over a secure Internet connection.

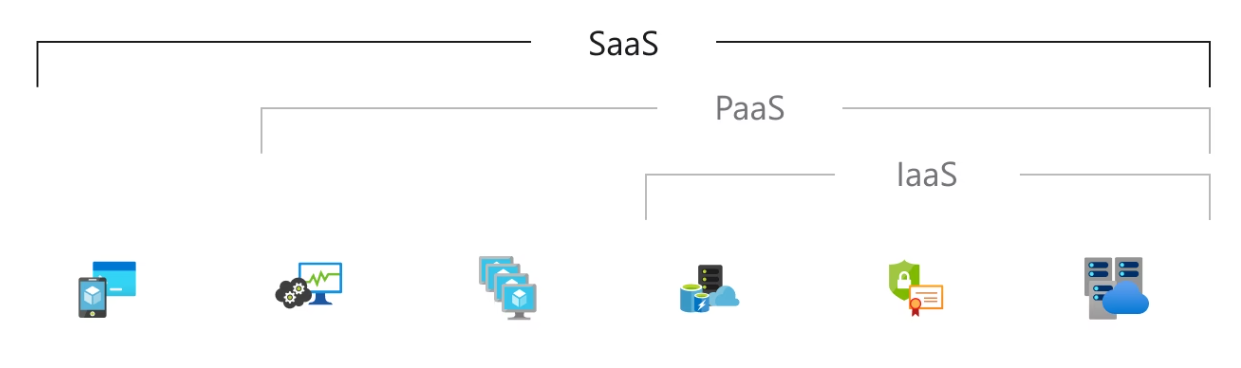
Like [IaaS](https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-iaas/), PaaS includes infrastructure—servers, storage, and networking—but also middleware, development tools, business intelligence (BI) services, database management systems, and more. PaaS is designed to support the complete web application lifecycle: building, testing, deploying, managing, and updating.



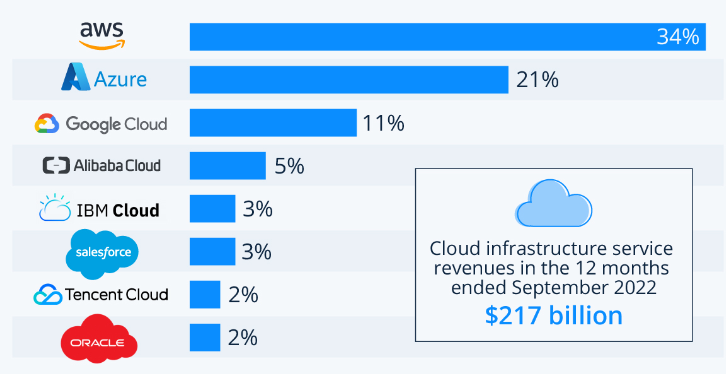
**Software as a Service (SaaS)**

Software as a service (SaaS) allows users to connect to and use cloud-based apps over the Internet. Common examples are email, calendaring, and office tools (such as Microsoft Office 365).





**TOP Cloud Providers**



**AWS Global Infrastructure**

The AWS Cloud spans **96** Availability Zones within **30** geographic regions around the world, with announced plans for **15** more Availability Zones and **5** more AWS Regions in Australia, Canada, Israel, New Zealand, and Thailand.

<https://aws.amazon.com/about-aws/global-infrastructure/>

**Regions, Availability Zones, Edge Locations**

**Regions**

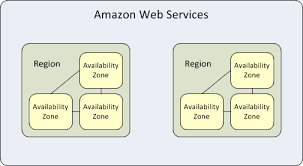
* **Regions are Large geographic areas**

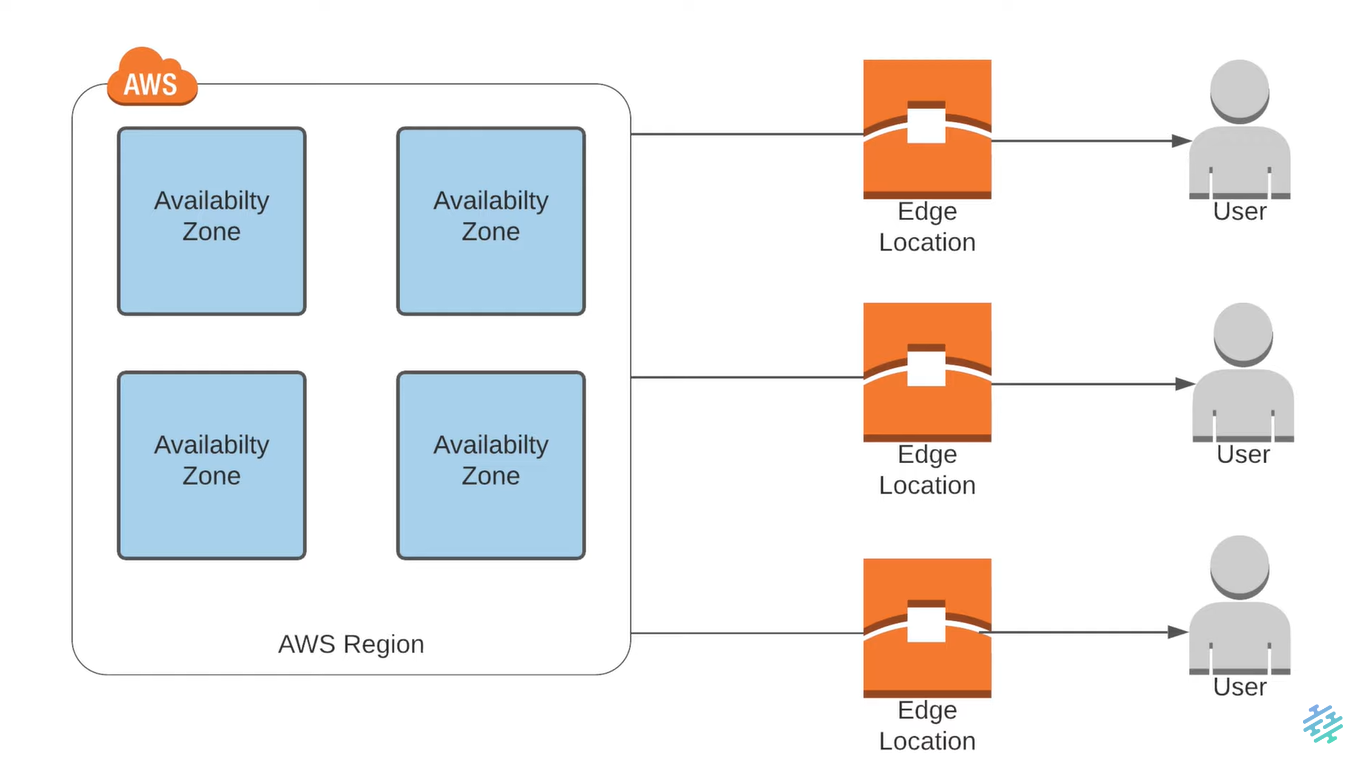
**Availability Zones**

* **Datacenters**

**Edge Locations**

* **Edge locations allows users to access content with lower latency**
* **A site that CloudFront uses to cache copies of your content for faster delivery to users at any location**





EC2 Instance

**Best Practices:**

* Gather the requirement
* Create the Key pairs
* Create the Security Groups
* Launch Instance

**Gather Requirement:**

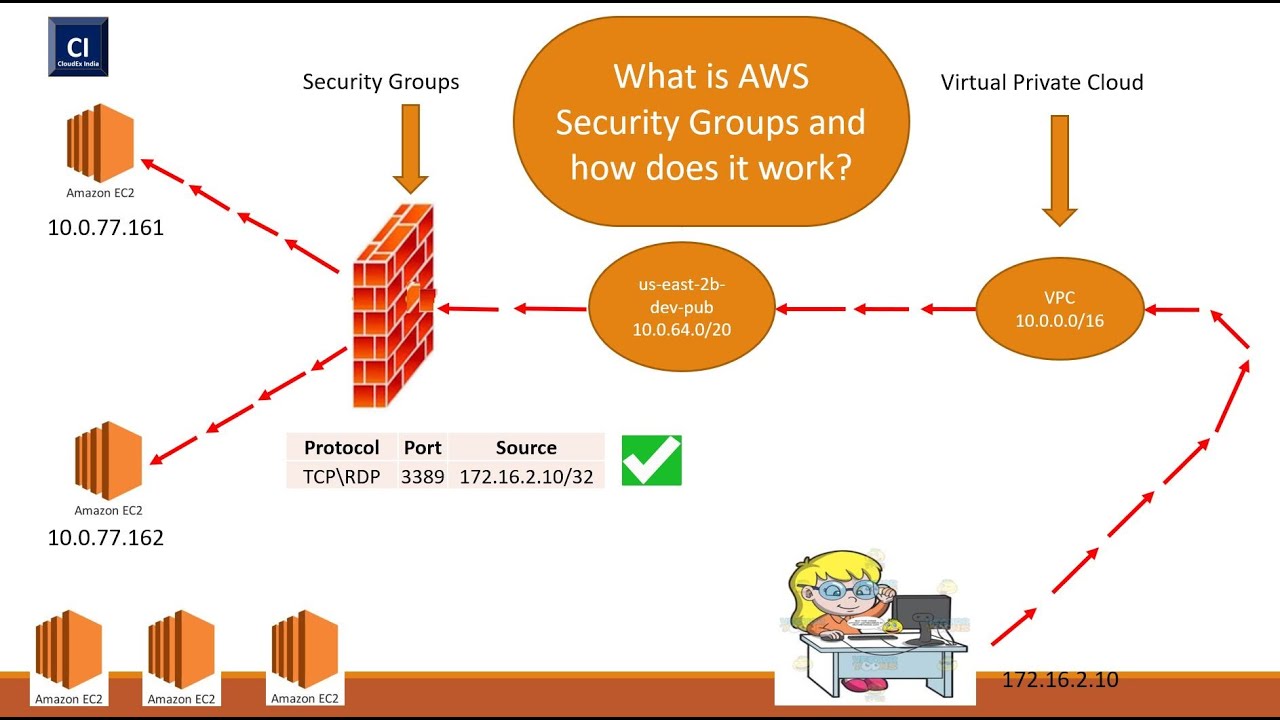
1. Operating System
   1. Ubuntu
   2. Centos
   3. Etc
2. Size of RAM, CPU, Network Etc
3. Storage size
   1. 10 Gig
4. Project Tag
5. Services/Apps will be Running
   1. SSH, HTTP, MySQL etc
6. Environment
   1. DEV, QA, Staging, Pre-Prod, Prod
7. Login user/owner

**Setup a Website on EC2 Instance**

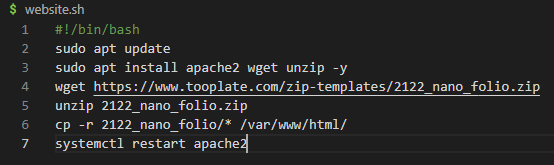
* Go to tooplate.com
* Create a Key Pair for this project – Give name as “Nano-Dev”
* Select .pem format & Click Create Key pair
* Private key will be downloaded into your system and public key will be stored in the AWS Key Pairs section
* Go to “Security Groups” 🡪 Create Security Group 🡪 Give name as “Nano-Dev-SG”

**Security Group**

* Security Group is nothing but a virtual Firewall which will allow / deny the incoming and outgoing traffic

****

* Launch Instance 🡪 Add Tags for Application & for Project
* Select the AMI
* Select the Instance-type
* Select the Key-pair which we created
* Select the Security Group which we created
* Login to EC2 instance from GIT Bash
* Copy & paste the website.sh content into a shell file like – website.sh



* Give it executable permission using **chmod +x website.sh** command
* Execute the shell script using “**sh website.sh or ./website.sh**” command
* Check the apache2 service using “**systemctl status apache2**”
* Allow port 80 from security group
* Now access the application using Browser

**More about EC2 Instance:**

* Stop and start the EC2 instance and check the public IP
* When you do this operation the Public IP will change. This is because AWS is assigning the Public IP from the pool. When you stop the Instance the IP will be released to the pool
* When you start the instance again, the public IP will be assigned again from the pool, but this time the IP will not be the same
* If you want to have fixed IP/Static IP then you have to use Elastic IP in AWS
* You will get 5 EIP by default by AWS, if you want more then you have to send request to the AWS support team and purchase more IP’s.
* Create a public IP, keep the same region name and click on Allocate.
  + Click on Actions
  + Click on “Associate Elastic IP address.
  + Choose the Instance and click on Associate
* Now Go to Instances
  + Select our Instance and Go to Actions tab
  + Settings
  + Change Instance Type
  + Before that you have to stop the Instance
  + Now you can change the instance type
* Network Interface
  + An elastic network interface is **a logical networking component in a VPC that represents a virtual network card.**
* Now go to Actions tab by selecting the Instance.
  + Select the Networking option
  + You will have options to Attach and Dettach to Network Interface
* Select Security in Actions tab
  + You can select different security group if you want to change the Security group

**AWS CLI**

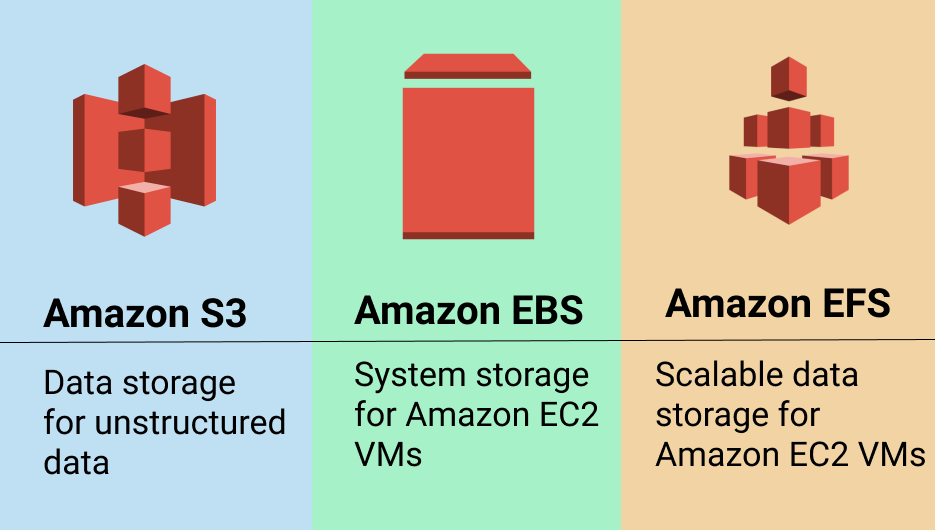
<https://aws.amazon.com/cli/>

* Before using the AWS CLI, you should be installing the AWS CLI in your laptop
* You can below command to install AWS CLI
  + Open Powershell as Administrator
  + $ choco install awscli -y
  + If you have not installed Chocolaty in your windows laptop, you can install it using below link
  + <https://chocolatey.org/install>
  + You can check which version of aws cli has been installed
    - $ aws –version
  + Configure the AWS CLI on the laptop
  + Create an IAM User in AWS
    - Give name as “awscli”
    - Select Programmatic Access
    - Click Next
    - Attach policy – select “AdministratorAccess”
    - Next and create it
    - Download the .csv file
    - Now go to GIT Bash on your laptop
    - Run below command to configure using create user
      * $ aws configure
      * Copy & Paste the Access key and secret access key
      * Type the region name: “us-east-1”
      * Output format: json
      * Now it will create a hidden directory like : /.aws/
        + ls ~/.aws/
        + cat ~/.aws/config
        + cat ~/.aws/credentials
      * Run below command to check the configuration details
        + $ aws sts get-caller-identity
      * Run some commands like
        + $ aws ec2 describe-instances

<https://docs.aws.amazon.com/cli/latest/reference/>

**EBS (Elastic Block Storage)**

* Block based storage
* Run EC2 instance OS, Stores data from DB, File Data etc.
* Placed in the same AZ of EC2 instance. Automatically replicated withing the AZ to protect from failure.
* Snapshot is backup of a volume



**Types of EBS**

* General Purpose (SSD)
  + Most work loads
* Provisioned IOPS
  + Large Databases
* Throughput Optimized HD
  + Big data and Data warehouses
* Cold HDD
  + File servers
* Magnetic
  + Backups & archives

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html>

**Create an EBS Volume and Attach it to EC2 instance**

* Go to Volumes section in EC2 Service
* Select Volumes
* Click on **Create volume**
* Select the Volume type
* Give the Volume size
* Choose the Availability zone for the EBS volume (Make sure to keep in the same AZ where the EC2 instance is running which you are going to attach)
* Give a Tag for the volume and click on Create volume
* Now to go to the volumes and select the newly created volume and click on Actions tab 🡪 Select “Attach volume”
* Select the Instance to which you want to attach the volume and click on Attach volume
* Now login to EC2 instance using SSH connection
* Go to /var/www/html
* There you will be having the img folder.

**Linux commands for Partition format and mount the new volume attached**

* $ fdisk -l
  + - This command will list all the disks available in the server
* $ df -h
  + - This show the disk space usage in Human readable format and where they are mounted
* $ fdisk /dev/xvdf
  + - This will open the fdisk utility for new volume called **“/dev/xvdf”**
    - Type **“m”** for all the options which you can use on this volume
    - We will use **“n add a new partition”**
    - Type **“n”**
    - Now Type **“p”** for primary partition
    - Now type the partition number like **“1”**
    - First sector 🡪 hit Enter button
    - Last sector 🡪 You can type like **“+3G”** it will take 3Gb from 5Gb or if you don’t type any number it will take entire 5Gb
    - Type **“p”** to print
    - Type **“w”** to write and partition will be created
* $ fdisk -l
* $ mkfs
  + - The **mkfs** command stands for**“make file system”** is utilized to make a file system
    - Hit the tab button two times it will display the available mkfs utilities
* $ mkfs.ext4 /dev/xvdf1
* Go to /var/www/html
* Create a backup directory 🡪 mkdir /tmp/img\_backup
* $ mv img/\* /tmp/img\_backup
* $ mount /dev/xvdf1 /var/www/html/img/ 🡪 This the temporary mount, if you reboot the server it be gone
* $ df -h
* $ umount /var/www/html/img/ 🡪 It will unmount the directory
* For Permanent mount you have to open a file called /etc/fstab
* $ vi /etc/fstab



* $ mount -a
* $ df -a

**EBS Snapshots**

* Firstly unmount the volumes which we have mounted
  + $ umount /var/www/html/img
  + $ df -h
* Detach the volume from the EC2 instance by going into volume section in AWS console
* Delete the volume
* Create a Volume 🡪 5gb
* Correct availability zone (AZ)
* Give name “db01-volume”
* Create volume
* Attach it to EC2 instance again
* Go to EC2 instance format and repeat the steps
* Create folder for mysql 🡪 mdkir /var/lib/mysql
* Go to fstab file and edit the mount path
* $ mount -a
* $ df -h
* $ apt install mariadb-server -y
* $ systemctl start mariadb
* $ systemctl status mariadb
* $ ls /var/lib/mysql/

**Snapshot Backup & Restore**

* Unmount partition
* Detach volume
* Create new volume from snapshot
* Attach the new volume created from snapshot
* Mount it back

**How to take the snapshot**

* Go to Volumes section in AWS console
* Select the volume and click on Actions
* Click on “Create snapshot”
* Give some description like “db01-volume-snapshot”
* Give a Tag
* Click on Create snapshot
* Check snapshot has been created

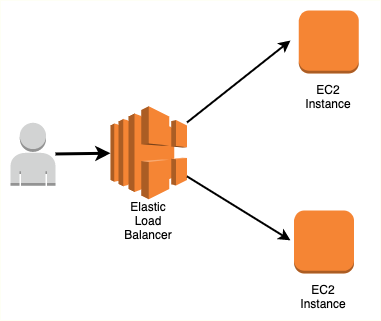
**How to recover the data from snapshot**

* Login to EC2 instance and delete the data from the directory which we have mounted like “/var/lib/mysql” using command
  + $ rm -rf \*
  + $ systemctl stop mariadb
  + $ umount /var/lib/mysql/
  + Now we have to detach the volume
    - Go to AWS console and go to Volumes
    - Select the volume and detach it
    - Now to Go to Snapshot in AWS
    - Select the snapshot and click on Actions
    - Select Create Volume
    - Give Tag name and click on Create Volume
    - Now go to Volumes section & you should see the recovered volume
    - Now go to Actions and attach the volume
    - Type command $mount -a
    - Now you will see all the data

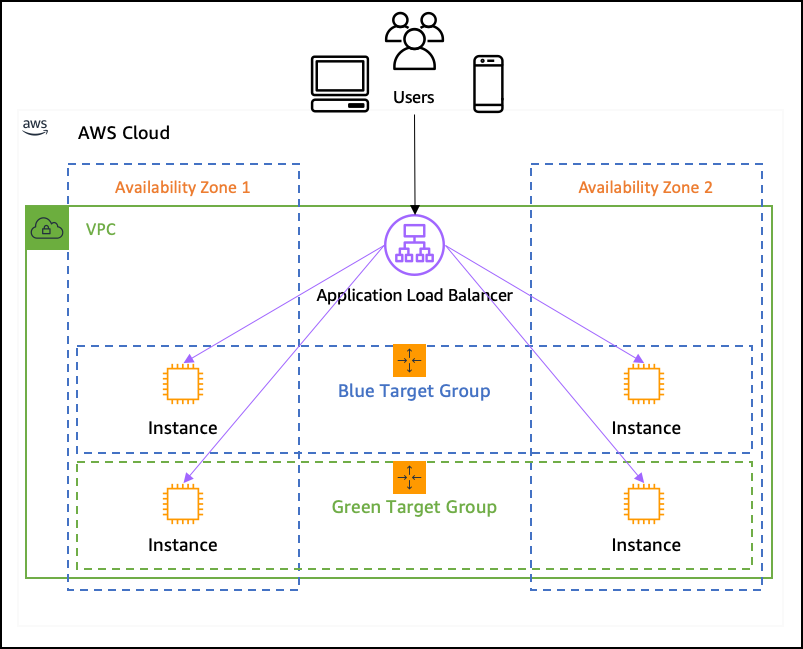
**ELASTIC LOAD BALANCER (ELB)**

* Frontend Port: Listens from the User Requests on this port AKA listeners.
  + E.g., 80, 443, 25 etc.
* Backend Port: Services running on OS listening on this port
  + E.g., 80, 443, 8080 etc.
* Elastic Load Balancing distributes incoming application or network traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses in multiple Availability Zones.

**Example 1:**



**Example 2:**



**Types**:

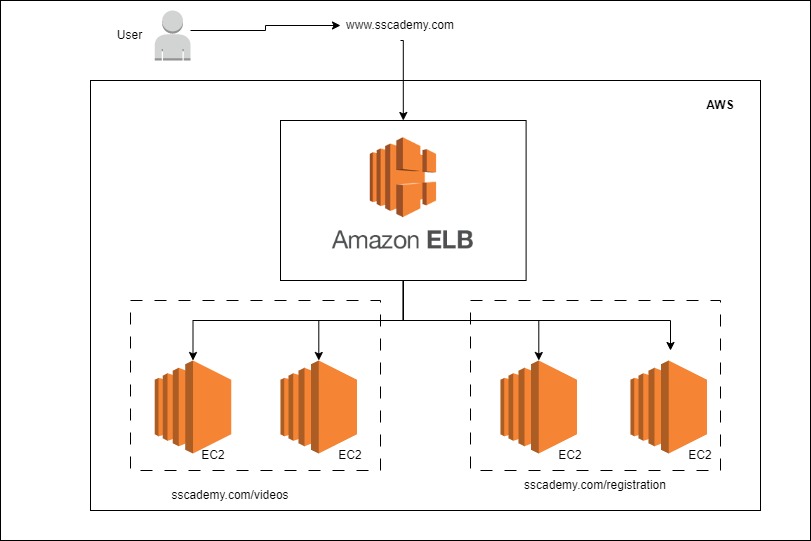
* Application Load Balancer
* Network Load Balancer
* Classic Load Balancer

**CLASSIC LOAD BALANCER**

* The Classic Load Balancer is the simplest Load Balancer that routes traffic based on either application or network level information
* The Classic Load Balancer is ideal for simple load balancing of traffic across multiple EC2 instances

**APPLICATION LOAD BALANCER**

* Application Load Balancer that routes traffic based on advanced application-level information that includes content of the request.
* It will be used for HTTP & HTTPS.
* It works on OSI Layer 7(Application Layer). It is not only forward the request to the port numbers but also it will route on the basis of application.



**NETWORK LOAD BALANCER**

* A Network Load Balancer functions at the fourth layer of OSI model
* It can handle millions of requests per second
* Static IP

**Key Differences: ALB vs NLB**

* **OSI Layer:**Application Load Balancer (as the name implies) works at the Application Layer (Layer 7 of the OSI model, Request level). Network Load Balancer works at the Transport layer (Layer 4 of the OSI model, Connection level).
* **Routing:** NLB just forward requests whereas ALB examines the contents of the HTTP request header to determine where to route the request. So, an ALB support advanced request (content-based) routing.
* **Static IP:** ALB doesn’t provide support for static IP address whereas NLB provides support for zonal static IP addresses (in each AZ).

**Use Cases**

*Choose****ALB****if:*

* Applications need advanced routing (host-based, URL-based, query string based).
* Run multiple services (microservices) behind a single load balancer.

*Choose****NLB****if:*

* Applications need to handle millions of requests per second while maintaining ultra-low latencies.
* You need a static IP address that can be used by applications as the front-end IP of the load balancer.

**Setup ELB for an Application**

**Launch an EC2 Instance**

* Go to EC2 service and Launch an EC2 instance
* Give name as “webapp01”
* Select “Amazon Linux” which is RPM (Red-Hat Package Manager) based OS
* Select “Amazon Linux 2 AMI (HVM) – Kernel 5.10, SSD Volume Type”
* Instance type- t2.micro
* Create a Key Pair – wavecafe-prod-key
* Edit Security Group 🡪
  + Allow port 22 from anywhere
  + Allow port 80 from anywhere
* Go to Advanced details
* Copy the “multios\_websetup.sh” & paste it

**Create custom AMI from the running instance**

* Select the running instance which we just created, go to Actions Tab
* Click on “Image and templates”, click on “Create Image”
* Give a name to the AMI
* Create Image
* Now in AMI section you should see your AMI
* Select AMI, click on Actions Tab 🡪 Copy AMI 🡪 you can copy your AMI to other regions
* By Clicking on the Launch Instance from AMI 🡪 it will launch the Instance from your customized AMI 🡪 it will be having all the data which we had in EC2 instance

**Create EC2 Launch Template**

* Go to Launch Templates Section
* Click on “Create launch template”
* Select “My AMIs” 🡪 Owned by Me
* Select the AMI which we created
* Select all the information’s
* Create Launch Template
* Now we can Launch instance from the template
* Select the template and click on Launch Instance
* Launch new Instance as “webapp02”

**Target Group Setup**

* Click on “Create Target Group”
* Select “Instances”
* Give a name “webapp-tg”
* Select port “http-80”
* Keep all other values as default
* Click Next
* Select the Instances
* Click on “Include as pending below”
* Click on Create Target Group

**Load Balancer Setup**

* Go to Load Balancer section
* Click on “Create Load Balancer”
* Select “Application Load Balancer”
* Click on Create
* Give name “webapp-alb”
* Keep “Internet Facing”
* In Network mapping select the zones, you have to select minimum two zones
* Click on “Create security group” for load balancer
* Give name as “webapp-alb-sg”
* Add “Inbound rule” 🡪 give port 80 allowed from anywhere
* Click on Create security group
* Select the Security group in the Load Balancer creation
* In “Listeners and routing” 🡪 select the target group which we created
* Click on Create Load Balancer

Now we have to Allow security group of Load Balancer to send traffic to the Security group of application instances

* Allow port number 80 from load balancer security group in the application security group

**ELASTIC FILE SYSTEM (EFS)**

<https://aws.amazon.com/efs/>

Setup EFS:

* Launch an EC2 instance with one tooplate.com template in it.
  + You can use multios\_websetup.sh file as a userdata while creating the EC2 instance. You can get this file from aws branch
* We will mount the EFS to /var/www/html/img folder to the EFS so that we can share the images across the different EC2 services using EFS
* Create Security group for EFS
  + Go to Security group section from EC2 services
  + Give name as “wave-efs-sg”
  + Allow NFS type from security group of Webserver
  + Create Security Group
* Search “EFS” from the AWS console search bar
* Click on “Create file system”
* Give name as “wave-web-image”
* Click on “Customize” option
* Keep all values as default and click Next
* In Network section 🡪 Change the security group and select the security group which we created for EFS
* Delete the default security group
* Do it for all the AZ’s
* Click Next
* Click on Create
* Click on Access Points from the left panel
* Click Create Access point 🡪 Select the ‘File system” EFS which we created
* Click on Create access point

**Now we have to mount the EFS volume into the EC2 instance**

<https://docs.aws.amazon.com/efs/latest/ug/mounting-fs.html>

* First we have to install “amazon-efs-utils”

**For Installing on Amazon Linux use below link and command**

* <https://docs.aws.amazon.com/efs/latest/ug/installing-amazon-efs-utils.html#installing-efs-utils-amzn-linux>
* sudo yum install -y amazon-efs-utils

For Installing on Ubuntu or Debian based Linux then follow the instructions given in the Link

<https://docs.aws.amazon.com/efs/latest/ug/installing-amazon-efs-utils.html#installing-other-distro>

We are using Amazon Linux so we will login and install using just one command

sudo yum install -y amazon-efs-utils

**Now let’s mount the EFS storage in EC2 instance**

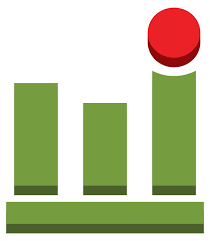
* file-system-id efs-mount-point efs \_netdev,tls,accesspoint=access-point-id 0 0
* This entry we have add it in fstab file
* Copy the file-system-id from EFS which we created
* It will be something like this “fs-47b7cca2”
* Go to the access points and copy the access point ID as well and paste it in the command
* Also give the **efs-mount-point** as **/var/www/html/img**
* Copy the full command and go to instance using SSH
* Before mount take a backup of IMG directory
  + $ mkdir /tmp/image-backup
  + $ mv img/\* /tmp/image-backup
* Now open the fstab file
  + $ vi /etc/fstab
  + Copy and paste the command here
  + Save and quit
* Now let’s test it
  + $ mount -fav
  + Now you should see Successfully mounted
* Now bring back our data from /tmp/image-backup
  + $ mv /tmp/image-backup/\* /img
  + $ ls img/
  + $ df -h
* Now create an AMI for this Instance

**AWS CLOUD WATCH**

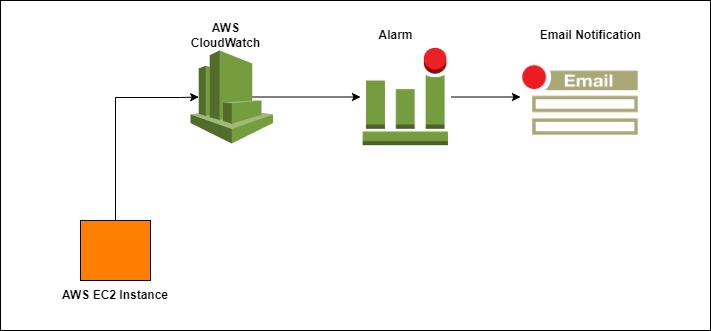
Cloud watch is a monitoring service.

* Cloud watch – Monitor performance of AWS environment – standard infrastructure metrics.
* Metrics:
  + AWS cloud watch allows you to record metrics for services such as EBS, EC2, ELB, Route53 health checks, RDS, Amazon S3, CloudFront etc.
* Events:
  + AWS events delivers a near real-time stream of system events that describe changes in Amazon Web Service (AWS) resources.
* Logs:
  + You can use Amazon CloudWatch logs to monitor, store and access your log files from Amazon Elastic Compute Cloud (Amazon EC2) instances, AWS CloudTrail, Route 53, and other sources.

**CloudWatch Alarm**



**Alarm**

* Alarm monitors cloudwatch metrics for Instances.
* ****Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.

**Cloud-Watch Setup**

* Click on our EC2 running instance
* Go to Monitoring section
* Click on “Manage detailed monitoring” option
* Click on “enable”
* Login to the EC2 instance using SSH
* We will install now stress on this EC2 instance
  + $ sudo amazon-linux-extras install epel -y
  + $ sudo yum install stress -y
  + $ stress
  + $ nohup stress -c 4 -t 300 &
  + $ top
  + $ nohup ./stress.sh &
* Now to go CloudWatch service in AWS console by searching name as cloudwatch
  + Click on All Alarms
  + Click on Create Alarm
  + Select Metric
  + Click on “EC2” 🡪 “per-instance Metrics” 🡪 Find your EC2 instance
  + Select “CPUUtilization” metric
  + Click on “Select metric”
  + Keep period for 5minutes
  + Select “Greater/Equal” 🡪 type “50”
  + Click Next
  + Select “In alarm”
  + Click on “Create new topic” 🡪 give topic a name
  + Type your email address
  + Click on Create topic
  + Click Next
  + Give a name for the alarm – “Warning | High CPU web01”
  + Next and create alarm

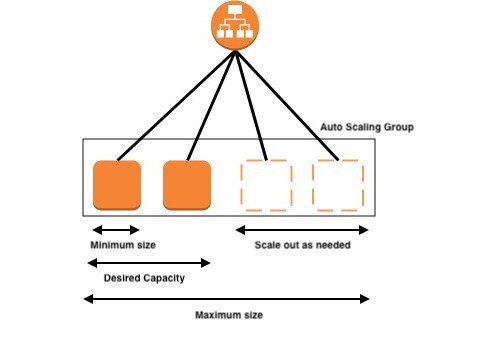
**AUTO SCALING**

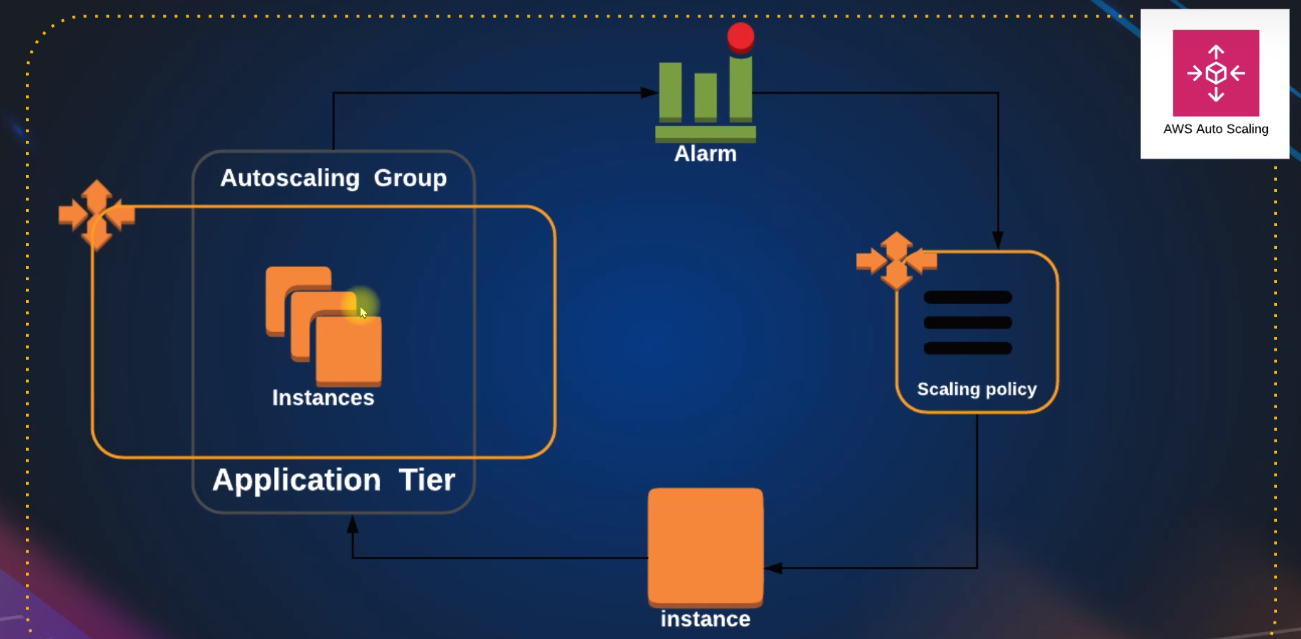
Auto scaling is a service that automatically monitors and adjust the compute resources to maintain performance for applications hosted in AWS

****

**Auto Scaling**

* Auto scaling group will be using A Launch configuration / launch template – it is an instance configuration template that an Auto Scaling group uses to Launch EC2 instances.
* Scaling policy is used to increase and decrease the number of running instances in the group dynamically to meet changing conditions.



**Setup:**

* Go to Load Balancer section
* Go to Target Group
* Create a Target group
* Give a name as “health-tg”
* Click Next
* Click on Create
* Click on Load Balancer
* Create a Load Balancer
  + Click on Application Load Balancer – name as “HealthELB”
  + Select all the AZ
  + Select the Security group which we have used before
  + Select the target group
  + Create Load balancer
* Go to Launch Template
* Create Launch template same way how we created before
* Now go to Auto-scaling group
* Click on Create Auto Scaling Group
* Give a name “wave-ASG”
* Select our Launch Template
* Click Next
* Select all the zones
* Click on “Attach to a existing load balancer.
* Select the Target group
* Select “ELB” health check along with EC2
* Click Next
* In Group Size option select below options
  + Desired capacity 🡪 2
  + Minimum capacity 🡪 1
  + Maximum capacity 🡪 4
* Scaling policies
  + Select “Target tracking scaling policy”
* Click Next
* Add notification
  + Select the SNS topic which we created
* Click Next
* Add Tags
  + Name – Webserver
* Click Next
* Create Auto scaling group

**AWS S3**

**Simple Storage Service**

**What is S3**

* Amazon Simple Storage Service (Amazon S3) is storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, anywhere from the web.

**S3 Basics**

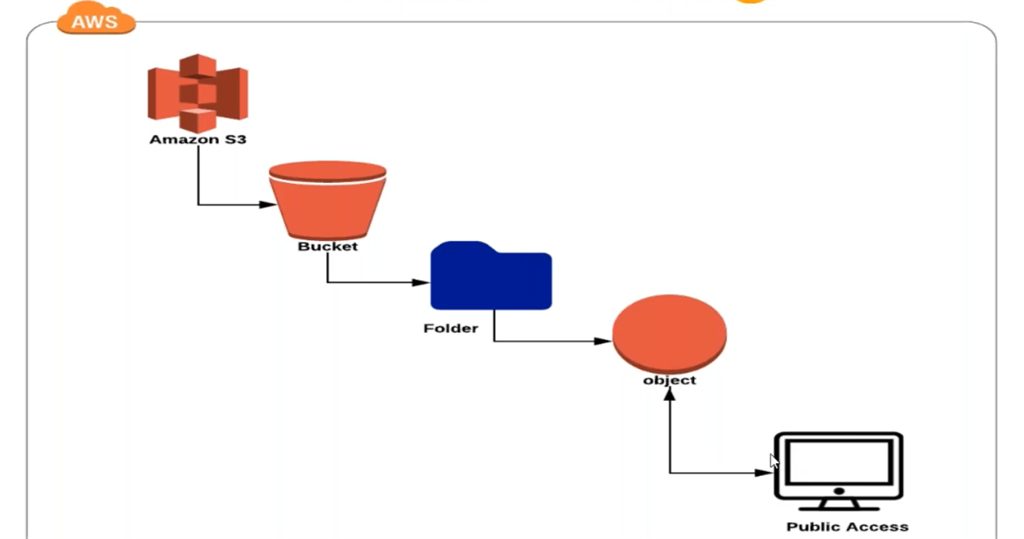
* Its object-based storage
* Data replicated across multiple facilities
* Unlimited data
* Amazon S3 stores data as **objects** within **buckets**
* Bucket name has to be unique

**Bucket**

* A Bucket is a logical unit of storage in Amazon Web Services (AWS)

**Object Storage**

* Object storage is a computer data storage architecture that manages data as object

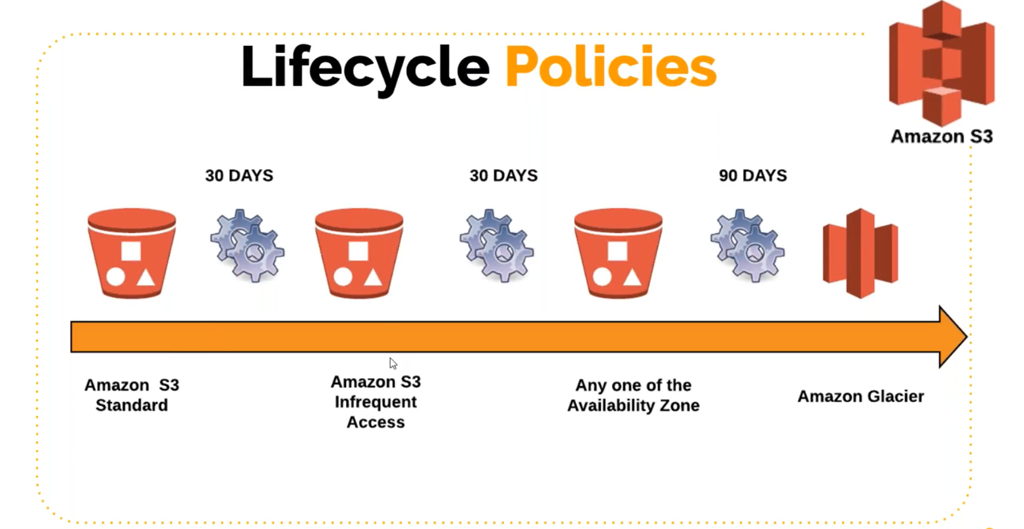
**S3 Creation**

**S3 Storage Classes**

1. S3 Standard
   * + General-purpose storage of frequently accessed data. Fast access & object replication in multi-AZ
2. S3 IA – Infrequent Access

* Long-lived, but less frequently accessed data. Slow access, object replication in multi-AZ

1. S3 One Zone – IA
   * Is for data that is accessed less frequently, but requires rapid access when needed. Slow access, no object replication.
2. S3 Intelligent Tiering
   * Automatically moves data to most cost-effective tier.
3. S3 Glacier
   * Low cost storage class for data archiving.
4. S3 Glacier Deep Archive
   * Lowest cost storage, retrieval time of 12 Hrs.



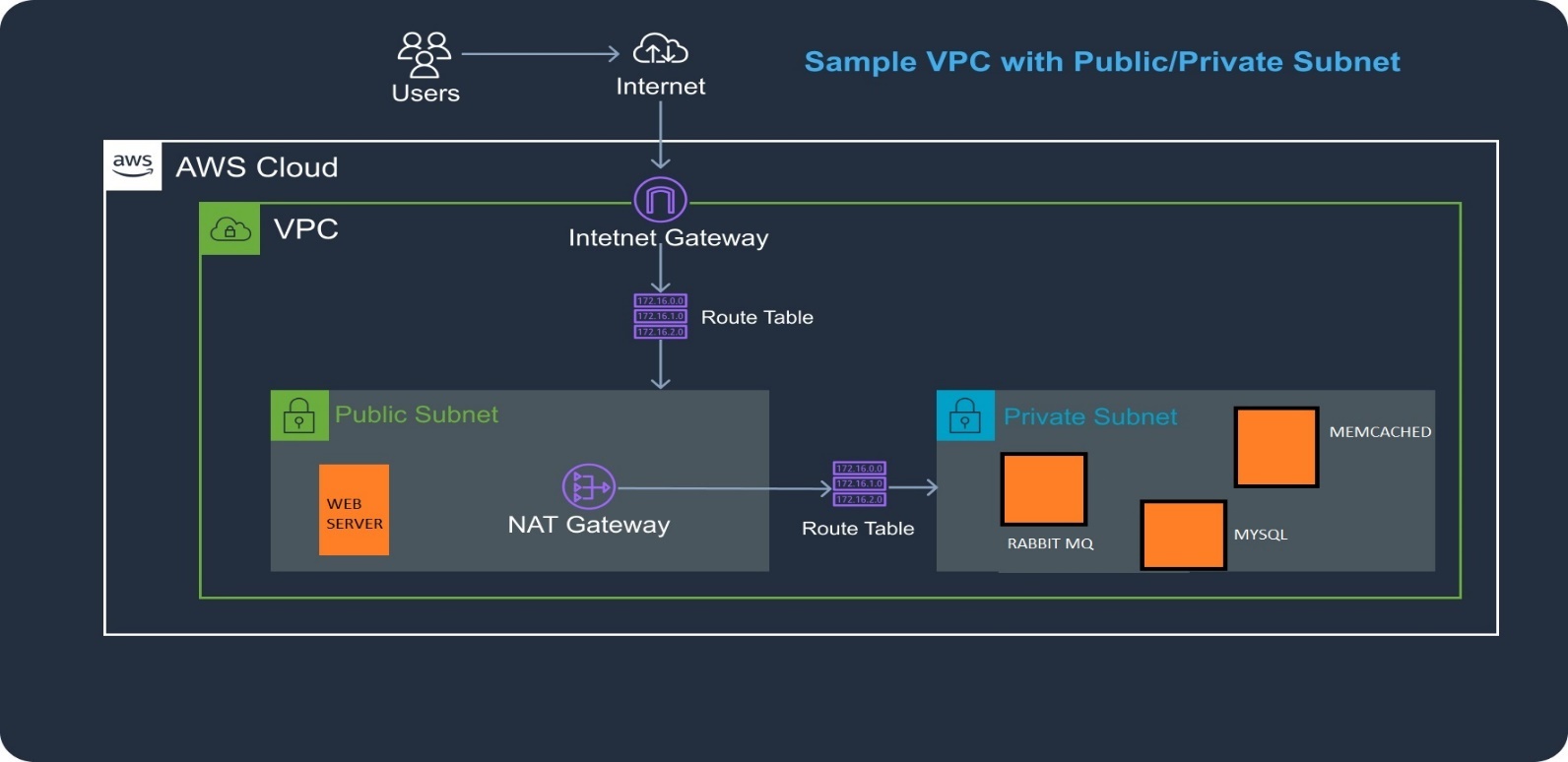
**S3 Charges**

* Storage
* Requests
* Tiers
* Data Transfer
* Region Replication

**AWS VPC**

Virtual Private Cloud

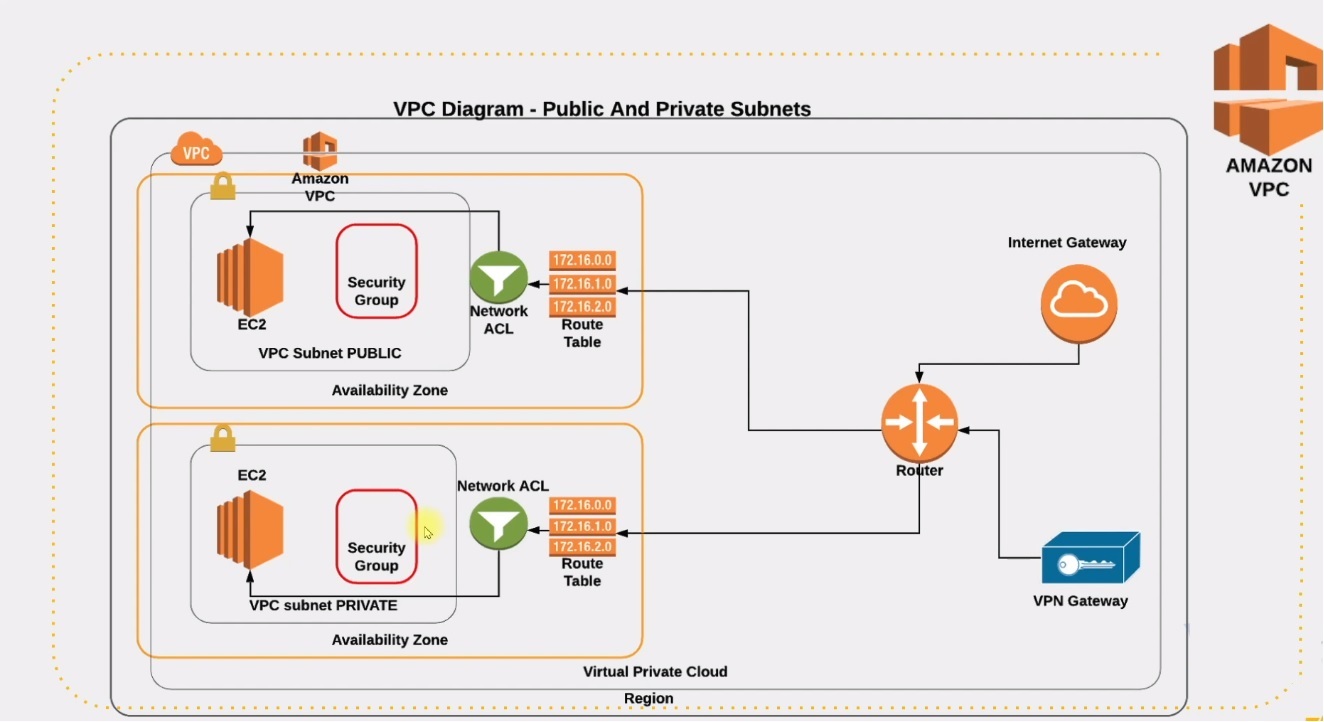
**What is VPC**

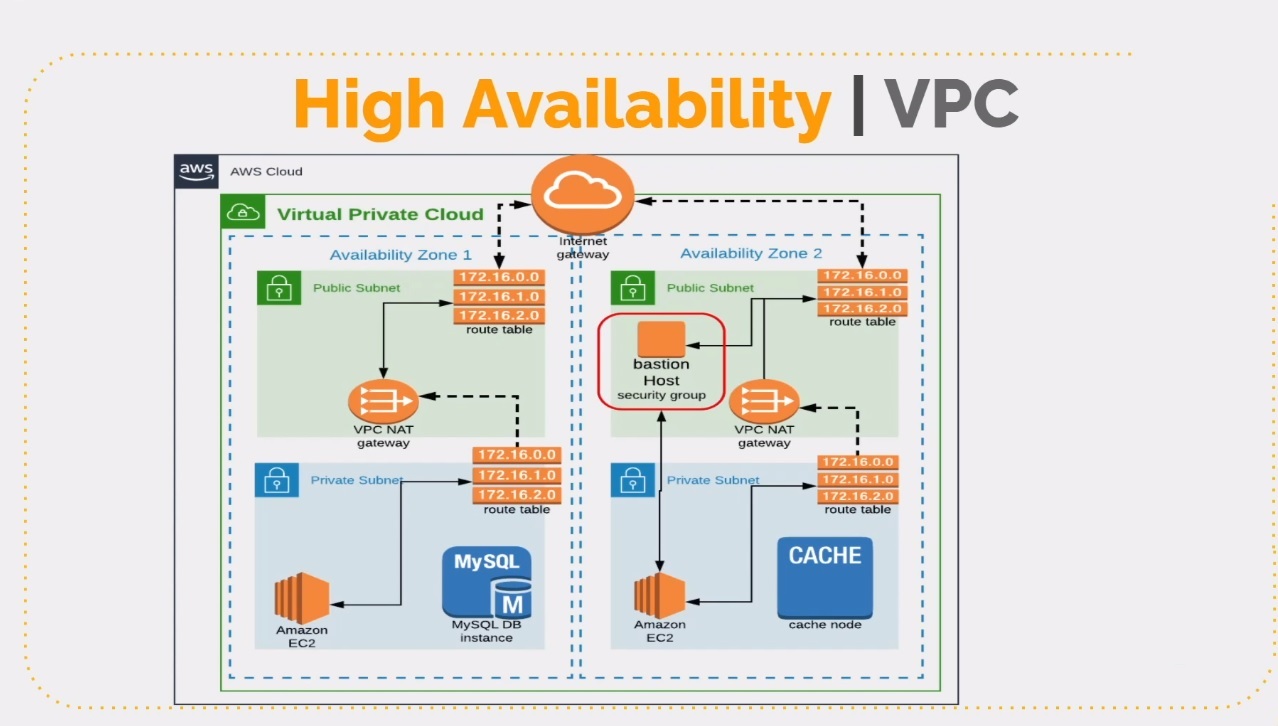
* Amazon Virtual Private Cloud (Amazon VPC) provides a logically isolated area of the AWS cloud where you can launch AWS resources in a virtual network that you define.
* You have complete control over your virtual networking environment, including a selection of your IP address range, the creation of subnets, and configuration of route tables and network gateways.
* You can easily customize the network configuration for your Amazon Virtual Private Cloud. For example, you can create a public-facing subnet for web servers that can access to the internet and can also place your backend system such as databases or application servers to a private-facing subnet.
* You can provide multiple layers of security, including security groups and network access control lists, to help control access to Amazon EC2 instances in each subnet.

**NAT Gateway**

* NAT (Network Address Translation) Gateway which enables instances in private subnet to connect to the internet or other AWS services

**INTERNET Gateway**

* Internet Gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the internet

****