```
======================
AWS Introduction
======================

-> AWS stands for Amazon Webservices

-> AWS cloud managing by Amazon company

-> AWS is one of the leading Cloud Provider in the market

-> AWS started providing IT resources over internet from 2006 onwards

-> 190+ countries using AWS Cloud

-> AWS providing 200+ Services

-> AWS providing Cloud Services based on 'Pay As You Go' model

======================
AWS Services Names
======================

EC2 : Elastic Compute Cloud : To create virtual machines

EBS : Elastic Block Store (External HD)

EFS : Elastic File System

S3 : Simple Storage Service  : Unlimited Storage

RDS : Relational Database System : To create SQL Databases (Oracle, MySQL, Postgres, MS SQL etc..)

VPC : Virtual Private Cloud : Isolated Network

Route 53 : Domain Name Mapping ( URL Mapping )

BeanStalk : For Paas Model

IAM : Identity & Access Management (who can access which service in AWS)

ECS : Elastic Container service (To run containers)

ELB : Elastic Load Balancer ( Load Balancing )

Lambda : Serverless Computing (run the code without thinking about servers)

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

-> To use AWS provided cloud services we need to create one account in AWS

Note: It will ask debit / credit card for account creation

-> AWS will charge 2 rs for account creation and they will send 2 rs back to our account after
account verified

-> In AWS few services are free and few services are paid

-> As part of our training we will use both free and paid services

Note: When we use paid services, after practise completion we need to delete those service to avoid
billing

-> If bill got generated we can request AWS Support team to waveoff our bill because we are AWS
learners and we are exploring AWS Cloud services.
```

-> AWS will not deduct bill amount from our card directley. We need to pay that bill manually.

-> If we don't pay AWS bill amount then our AWS account will be terminated.

++

-> AWS providing global infrastructure

-> 190+ contries are using AWS Cloud through internet

-> To provide Global Infrastructure AWS using Regions & Availability Zones concept

-> Region means one geograhical location

-> Availability Zone means data center

-> Data Center means a big building which contains servers with network

-> One Region can have multiple Availability Zones (AZ)

-> AWS Having 26 Regions and 84 Availability Zones in the world

-> In india AWS having 1 region (Mumbai) ---- ap-south-1

-> Mumbai region having 3 availability zones

- ap-south-1a
- ap-south-1b
- ap-south-1c

Note: Hyd Region Coming Soon

Note: It is recommended to use Nearest Region in AWS to setup our infrastructure

Note: In AWS few services are global (S3, Route 53 etc...) and few services regional (EC2, VPC, RDS etc...)

========
EC2
========

-> EC2 stands for " Elastic Compute Cloud "

-> EC2 is the most demanded service in AWS

-> EC2 is used to create virtual machines

-> EC2 instances are re-sizable

-> EC2 is regional service

-> EC2 instance minimum billing period is 1 hour

Ex: If you run for 5 minutes also it will be considered as 1 hour

-> The Virtual Machine which we create using EC2 service is called as EC2 instance

Alias Names : EC2 Instance / Virtual Machine / VM / Server

-> EC2 is a paid service

Note: EC2 providing t2.micro as free tier eligible for 1 year with monthly 750 hours

-> When we create EC2 instance, AWS will provide default storage and default network

-> Storage will be provided using EBS (Elastic Block Store)

-> Network will be provided using VPC (Virtual Private Cloud)

==========
EC2 instances types
==========

1) On-Demanded Instances

2) Reserved Instances

3) Spot Instances

4) Dedicated Host

==========
OnDemanded Instance
==========

-> Whenever we want then we can create it

-> Fixed Price (Hourly)

-> Pay For Use

-> No Prior Payments

-> No Committment

==========
Reserved Instances
==========

-> It is like advanced booking

-> Long Term Commitment

-> Prior Payment option available (paritial payment / full payment)

-> Commitment for 1 year or 3 years

-> AWS will provide discount on price

==========
Spot Instances
==========

-> It is like bidding or Auction

-> AWS will offer high capacity systems for low price

-> 72% savings on price

==========
Dedicated Host Instance
==========

-> This is a physical machine

-> Licensed Softwares

-> It is very costly when compared with other instance types

==========
EC2 instance states
==========

Running --- Billing

Stopped -- No Bill

Terminated -- No Bill

==========
EC2 instance families
==========

t2. t3, t4, c2, c3, i2, i3, m2, m3, m4 etc....

-> We are using t2.micro instance for practise

Note : t2.micro instances are Free for 1 year (from aws account creation date)

Note: Monthly 750 hours usage is free for 1 year when we go for t2.micro

Note: We can find EC2 instance types service rates in dashboard

==========
Amazon Machine Image (AMI)
==========

-> AMI is a template to launch our instances

-> Image represents pre-configured system with softwares installed

-> To create EC2 instance with Windows OS we can use Windows OS AMI

-> To create EC2 instance with Linux OS we can use Linux OS AMI

-> In AWS, we can create our own AMI also

==========
What is Key Pair in EC2 ?
==========

-> Key Pair is the combination of Public Key and Private key

-> AWS will store public key and it will provide private key for us (We have to save that)

-> Private Key file extension will be .pem

-> Public key & Private Key is used to connect with EC2 instance securley

-> If we want to connect with EC2 VM we need to provide private key for AWS then AWS will compare our private key with its public key. If both keys are matched then only AWS will allow to connect with EC2.

Note: One key pair we can use to create multiple instances

Note: Key-Pairs are free of cost (No Bill)

Note: Key pair we can use for any OS Based VM

==========
Security Groups
==========

-> Security Group is like a virtual firewall for our EC2 instance

-> Security Group will control Incoming and Outgoing traffic of our EC2 instance

-> To allow the incoming traffic we will configure Inbound Rules

-> To allow the outgoing traffic we will configure Outbound Rules

Windows => RDP : 3389

Linux => SSH : 22

Webserver => HTTP : 80

HTTPS => HTTPS : 443

Note: Security Groups are free

Note: One security group we can use for Multiple Instances

==========
Creating Windows Virtual Machine
==========

1) Goto EC2 service

2) Launch instance

3) Select AMI (Windows - Free tier eligible)

4) Select Instance Type (t2.micro - Free tier eligible)

5) Storage (Default 30 GB - Free tier)

6) Network (Default VPC)

7) Create New Pair

8) Create New security group with RDP protocol

=> Once EC2 VM created then click on 'Connect' button and get below details

DNS : ec2-15-207-89-254.ap-south-1.compute.amazonaws.com
Username : Administrator
Password : rcAF2=D3s%g&%O98o?)*xUYd&!vdw?dp

-> Connect to Windows VM using RDP

==========

Launching Linux Virtual Machine in AWS
==========

1) Goto EC2 service

2) Launch instance

3) Select AMI (Amazon Linux - Free tier eligible)

4) Select Instance Type (t2.micro - Free tier eligible)

5) Storage (Default 8 GB - Free tier)

6) Network (Default VPC)

7) Create / Use Key Pair

8) Create New security group with SSH protocol

=> Once EC2 VM created then click on 'Connect' button and get details

=> Connec to Linux VM using MobaXterm

==========
Types of IP's in AWS
==========

-> IP stands for Internet Protocol

-> Every Machine should have one IP address

-> IP is like an address for computer

-> AWS providing 3 types of IPs

1) private ip

2) public ip

3) elastic ip (static ip)

-> When we launch EC2 instance then AWS will provide one private ip and one public ip for our instance

-> Private IP is a fixed IP and it is used by AWS for internal purpose. It will not change when we re-start our EC2 instance.

-> Public IP is a dynamic IP. When we re-start our EC2 instance new Public IP will be generated.

Note: To connect with EC2 instance from outside we will use Public IP.

-> Elastic IP means fixed public IP address.

-> We can create Elastic IP and we can associate that elastic ip for our EC2 instance

-> Elastic IP address will not change when we re-start our ec2 instance

Note: Elastic IPs are commercial (paid)

Working process

++++++++++++++++
1) Create Elastic IP
2) Associate Elastic IP for EC2 instance
3) If we don't want to use elastic ip then De-Associate Elastic IP from EC2 instance
4) Release Elastic IP to AWS (Its mandatory)

***** Note: If we create Elastic IP then we have to associate that Elastic IP otherwise bill will be generated for that******
(We shouldn't keep Elastic IP as un-used ip)

================
Load Balancing
=================

-> If we deploy our application in one server then burden will increase on that server

-> If burden increased on server then below are the problems we are going face

1) Request processing will become slow

2) Responses will be delayed for customers

3) Server might get crash

4) Brand / Trust issues on our business

5) Revenue Loss

6) Single point of failure

Note: Good Business needs Good website

-> To overcome all these problems we will run our application in Multiple Servers

-> The process of running our application in Multiple Servers is called as Load Balancing.

-> To implement Load Balacing we will use Load Balancer (ELB) in AWS

-> LBR will recieve the request and it will distribute the requests to servers in round robbin fashion

==============================
Types of Load Balancers in AWS
==============================

1) Application Load Balancer (ALB)

2) Network Load Balancer (NLB)

3) Gateway Load Balancer (GLB)

4) Classic Load Balancer (old, getting retried on 15-Aug-2022)

-> To implement load balancing for HTTP & HTTPS requests we will go for Application Load Balancer (ALB)
-> By using Application Load Balancer we can implement Path Based Routing

```
==========================
Implemenging Load Balancer
==========================

1) Create 1st EC2 intance with below user data script

#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT :: Server 1</h1></html>" > index.html
service httpd start


2) Create 2nd EC2 intance with below user data script

#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT :: Server 2</h1></html>" > index.html
service httpd start


3) Create 3rd EC2 intance with below user data script

#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT :: Server 3</h1></html>" > index.html
service httpd start


4) Create Target Group with above 3 EC2 instances
(Target Group means group of servers which are running our aplication)

5) Create Application Load Balancer using Target Group

6) Access the application Load Balancer DNS URL

Note: When request comes to Load Balancer it will distribute the requests to servers which are part
of given target group.



========================
Monolith Vs Microservices
========================

-> Application can be in 2 ways

        1) Monolith Architecture

        2) Microservices Architecture


-> Monolith Architecture means all the functionalities will be developed in single project
-> A big war file will be created
-> Monolith Architecture based project is difficult to maintain
-> For any small change in the code then we have to re-deploy entire application
-> Single Point Of failure

Note: To overcome the problems of Monolith Architecture we are using Microservices Architecture
```

-> Microservices is an architectural design pattern
-> Micservices architecture means project functionalities will be developed in multiple apis
-> Every API is called as one project
-> Every API contains limited functionality
-> Making changes to the functionality is easy in microservices
-> Maintenence of the project will become easy
-> For any code changes we no need re-deploy all the apis (deploy only changed api)

Note: For Monolith app load balancing one target group will be created and application will be deployed in all the servers who are belong that target group.

==========
Microservices Based Load Balancing
==========

-> Multiple APIs will be available In Microservices Architecure

-> Every API is called as one microservice

-> For every microservice one target group will be created

Hotels API ===> Hotels Target Group with 3 servers

Flights API ===> Flights Target Group with 3 servers

Trains API ===> Trains Target Group with 3 servers

==========
How to implement LBR for Microservices based application
==========

1) Create EC2 Instance with below user-data (Name it as HotelServer-1)

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Hotel Server - 1</h1></html>" > index.html
service httpd start
```

2) Create EC2 instance with below user-data (Name it as HotelServer-2)

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Hotels Server - 2</h1></html>" > index.html
service httpd start
```

3) Create HotelServers Target group with above 2 instances

4) Create EC2 instance with below user-data (Name it as FlightServer-1)

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Flights Server - 1</h1></html>" > index.html
```

```
service httpd start

4) Create EC2 instance with below user-data (Name it as FlightServer-2)

#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Flights Server - 2</h1></html>" > index.html
service httpd start


5) Create FlightsServers Target group with above 2 instances


6) Create Load Balancer by select HotelServers Target Group

7) Goto LBR Listeners and configure Route Based Routing for Flights Target Group

8) Test it with DNS name


Note: Once practise completed, delete LBR, Target Groups and EC2 instances

================
Auto Scaling
================

=> AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady,
predictable performance at the lowest possible cost.

=> Using AWS Auto Scaling, itâ€™s easy to setup application scaling for multiple resources across
multiple services in minutes.

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


============================
Advantages with Auto Scaling
============================

1) Fault Tolerenece

2) Availability

3) Cost Management

============================
How to setup Auto Scaling Group
============================

1) Create Launch Template

2) Create AutoScaling Group with Launch Template

3) Configure Desired, Min and Max Capacity

4) Attach AutoScaling Group to particular Target Group

5) Configure Scaling Policy
```