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