Experiment no 5.3

## Aim

To deploy a full stack web application on AWS using EC2 instances and configure an Elastic Load Balancer (ELB) to distribute traffic evenly across multiple backend servers, ensuring high availability, scalability, and fault tolerance.

## Theory

### What is a Full Stack App Deployment?

* A full stack app usually has:
  + Frontend (React/Angular/Vue) → Runs in browser or deployed via S3/CloudFront.
  + Backend (Node.js/Express, Django, Spring Boot, etc.) → Runs on EC2 instances or containers.
  + Database (RDS, DynamoDB, MongoDB Atlas, etc.).

### Why AWS?

* Provides on-demand cloud infrastructure.
* Offers Elastic Load Balancer (ELB) to distribute incoming traffic across multiple instances.
* Ensures fault tolerance, auto-scaling, and high availability.

### Load Balancing

* Load balancer sits between clients and servers.
* Distributes incoming requests across healthy EC2 instances.
* Types of AWS Load Balancers:
  1. Application Load Balancer (ALB) – for HTTP/HTTPS traffic.
  2. Network Load Balancer (NLB) – for TCP/UDP high performance.
  3. Classic Load Balancer (CLB) – older, basic option.

## Code / Steps

Here’s a practical outline to deploy:

### 1. Backend Setup (Node.js Example)

server.js

const express = require("express");

const app = express();

const PORT = process.env.PORT || 3000;

app.get("/", (req, res) => {

res.send("Hello from AWS Backend!");

});

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

package.json

{

"name": "aws-backend",

"version": "1.0.0",

"main": "server.js",

"scripts": {

"start": "node server.js"

},

"dependencies": {

"express": "^4.18.2"

}

}

### 2. Create EC2 Instances

* Launch 2 EC2 instances (Ubuntu).
* SSH into each instance and install:

sudo apt update -y

sudo apt install -y nodejs npm

git clone <your-repo-url>

cd aws-backend

npm install

npm start

* Ensure security group allows HTTP (port 80).

### 3. Setup Application Load Balancer (ALB)

* Go to AWS Console → EC2 → Load Balancers → Create Load Balancer.
* Choose Application Load Balancer.
* Configure listener on Port 80.
* Create a Target Group and register both EC2 instances.
* After provisioning, AWS will give you a DNS name for your Load Balancer.

### 4. Frontend Setup (React Example)

Inside src/App.js:

function App() {

return (

<div>

<h1>Full Stack App on AWS</h1>

<p>Backend API URL: http://<LOAD\_BALANCER\_DNS>/</p>

</div>

);

}

export default App;

Build frontend:

npm run build

Deploy options:

* Upload build/ folder to AWS S3 + CloudFront (static hosting).
* Or serve via Nginx on another EC2 instance behind the same ALB.

## Learning Outcomes

By the end of this practical, you will be able to:

1. Understand the process of deploying full stack applications on AWS.
2. Configure EC2 instances for backend and frontend deployment.
3. Set up and manage an Application Load Balancer (ALB).
4. Distribute traffic across multiple servers to achieve scalability.
5. Ensure fault tolerance and high availability using AWS infrastructure.
6. Gain hands-on experience with cloud networking, load balancing, and deployment automation.