EXPERIMENT-5.3

Name: Jayanaath S | Subject: Full Stack

Section: 23BCC-1 | Subject Code:23CSP-339

UID: 23BCC70022 | Date: 27-10-2025

- **Aim:**
  To Deploy Full Stack App on AWS with Load Balancing

- **Objective:**

  To deploy a full stack web application (React frontend + Node.js/Express backend + optional MongoDB database) on AWS EC2 instances and configure an **Application Load Balancer (ALB)** to ensure scalability and high availability.

- **Theory:**

  A Full Stack Application typically has:

  - Frontend (React): Handles the user interface and runs in the browser.
  - Backend (Node.js/Express): Handles API requests, authentication, and database communication.
  - Database (MongoDB/MySQL): Stores data persistently.

  Amazon Web Services (AWS) provides the infrastructure to host such apps using:

  - EC2 (Elastic Compute Cloud): Virtual servers used to run your frontend and backend applications.
  - Elastic Load Balancer (ELB): Distributes incoming traffic evenly across multiple EC2 instances for high availability.
  - VPC (Virtual Private Cloud): Provides a private, secure network environment for your EC2 instances.
  - Route 53 (optional): Used to assign a custom domain name to your application.
  - Load balancing ensures:
    - Better scalability
    - Fault tolerance — if one server goes down, others still serve requests
    - Optimized performance under heavy traffic

- **Procedure:**

1. Create EC2 Instances

- Go to AWS Management Console → EC2 → Launch Instance.
- Create two EC2 instances for your backend (Node.js) and one instance for your frontend (React).
- Choose Ubuntu or Amazon Linux 2 as the OS.
- Configure Security Groups to allow:
  - Port 22 (SSH)
  - Port 80 (HTTP)
  - Port 3000 (if running locally during setup)
  - Port 5000 (for backend API, if needed)

2. Deploy Backend on EC2

- SSH into each backend instance:

  ssh -i your-key.pem ubuntu@<ec2-backend-public-ip>

- Install Node.js and Git:

  sudo apt update

  sudo apt install -y nodejs npm git

- Clone your backend repo:

  git clone <your-backend-repo-url>

  cd backend

  npm install

  npm start

- Make sure it runs on port 5000 or 80 (adjust in your Express app if needed).

3. Deploy Frontend on EC2

- SSH into the frontend instance:
  - ssh -i your-key.pem ubuntu@<ec2-frontend-public-ip>
- Install Node.js and build the React app:
  - sudo apt update
  - sudo apt install -y nodejs npm git
- git clone <your-frontend-repo-url>
  - cd frontend
  - npm install
  - npm run build

- Install and configure Nginx to serve the build:
  - sudo apt install nginx -y
  - sudo rm -rf /var/www/html/*
  - sudo cp -r build/* /var/www/html/
  - sudo systemctl restart nginx

4. Set Up Application Load Balancer (ALB)

- Go to AWS Console → EC2 → Load Balancers → Create Load Balancer.
- Select Application Load Balancer.
- Configure:
  - Listener: HTTP on port 80
  - Availability Zones: Select multiple for redundancy
  - Target Group: Create a new one and add your backend EC2 instances as targets
- Once created, note down the DNS name of your ALB (it will look like my-app-alb-123456.elb.amazonaws.com).

5. Configure Frontend to Use the Load Balancer

- In your React app, change your backend API URL from:

  const API_BASE = "http://<backend-ec2-ip>:5000";

to:

  const API_BASE = "http://<your-alb-dns-name>";

- Rebuild and redeploy the frontend build files to your Nginx instance:

  npm run build

  sudo cp -r build/* /var/www/html/

  sudo systemctl restart nginx

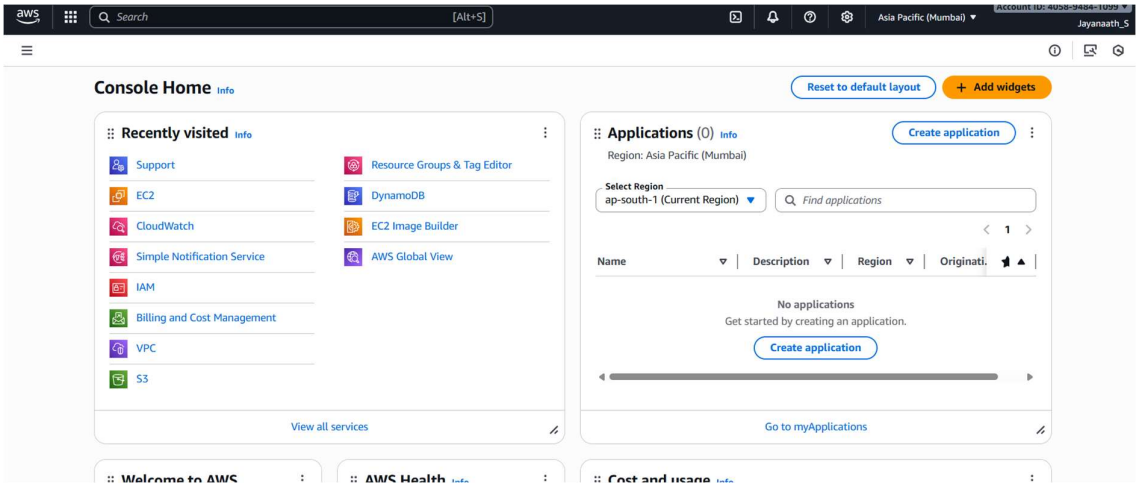
6. Test the Application

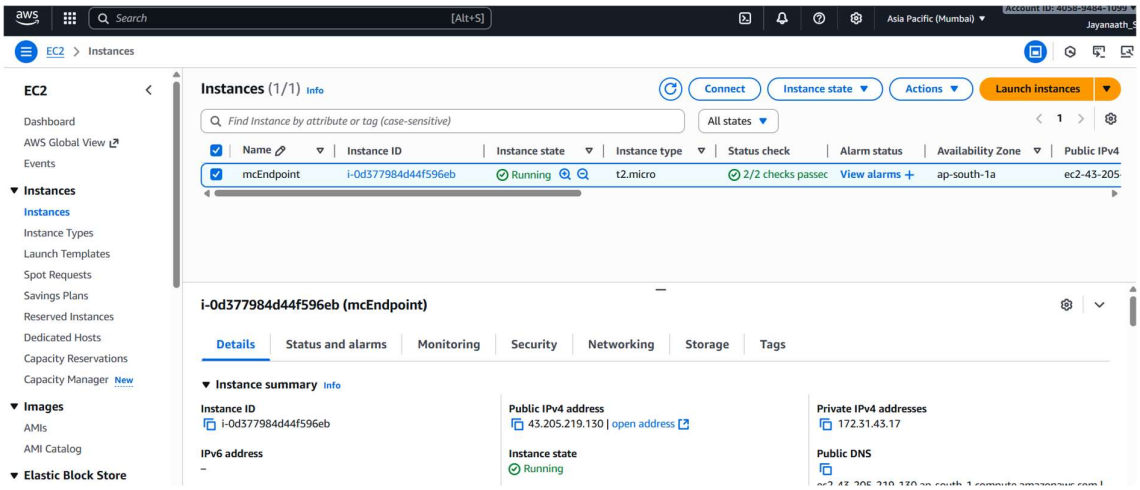
- Visit your frontend EC2's public IP or domain in a browser.
- The app should load, and API calls will go through the Load Balancer, distributing traffic between backend servers.
- Try stopping one backend EC2 instance — the load balancer will still serve requests using the other.

• Result

- A full stack React + Node.js app is deployed successfully on AWS.
- The **Application Load Balancer** evenly distributes traffic between backend servers.
- The application remains accessible even if one backend instance fails.
- Basic scalability and fault tolerance are achieved through load balancing.

• Output:





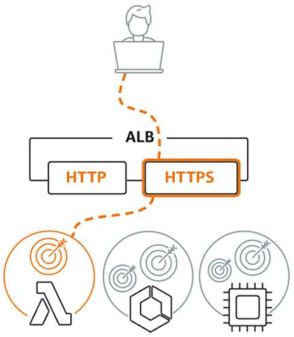
EC2 > Load balancers > Compare and select load balancer type

Compare and select load balancer type

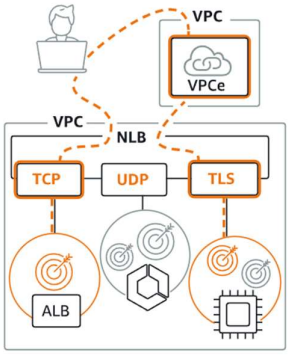
A complete feature-by-feature comparison along with detailed highlights is also available. Learn more

Load balancer types

Application Load Balancer Info



Network Load Balancer Info



Gateway Load Balancer Info