Name: Nikhil Sharma

UID:23BCC70030

23BCC1-A

Deploy Full Stack App on AWS with Load Balancing

Aim

To deploy a full-stack web application on Amazon Web Services (AWS) using EC2 instances and configure load balancing to ensure high availability and scalability.

Theory

A full-stack application usually consists of a frontend, backend, and database. Deploying such applications on the cloud ensures better performance, reliability, and availability.

Key AWS Components:

1. **EC2 (Elastic Compute Cloud):** Provides virtual servers to host the backend and frontend.
2. **RDS (Relational Database Service):** Managed database service for storing data.
3. **Elastic Load Balancer (ELB):** Distributes incoming traffic among multiple EC2 instances to prevent overloading a single server.
4. **Security Groups:** Acts as a virtual firewall controlling inbound and outbound traffic.
5. **Route 53 (Optional):** DNS service to route domain traffic to your load balancer.

Benefits of Load Balancing:

- Even distribution of traffic
- High availability and redundancy
- Automatic scaling

- Fault tolerance

Learning Outcomes

After completing this experiment, students will be able to:

1. Understand how to deploy frontend and backend applications on AWS EC2 instances.
2. Configure Elastic Load Balancing for high availability.
3. Connect EC2 instances with RDS for database operations.
4. Configure security groups and IAM roles for secure deployment.
5. Access and monitor deployed applications via a public URL.

Step-by-Step Procedure

Step 1: Set Up AWS EC2 Instances

1. Login to your AWS Management Console.
2. Go to **EC2** > **Launch Instances**.
3. Choose an **Amazon Linux 2** or **Ubuntu Server** AMI.
4. Select instance type (e.g., t2.micro for free tier).
5. Configure network and security group to allow **HTTP (80)**, **HTTPS (443)**, and **SSH (22)**.
6. Launch the instance and download the **key pair (.pem)** for SSH access.

Step 2: Connect to EC2 Instances

1. Open a terminal.
2. Change permissions for your key:
3. chmod 400 keypair.pem
4. Connect using SSH:

5. ssh -i "keypair.pem" ec2-user@<EC2_PUBLIC_IP>

Step 3: Install Required Software

On the EC2 instance:

1. Update packages:
2. sudo yum update -y # For Amazon Linux
3. sudo apt update -y # For Ubuntu
4. Install **Node.js**, **npm**, **git**, **pm2** (for backend process management):
5. curl -fsSL https://rpm.nodesource.com/setup_18.x | sudo bash -
6. sudo yum install -y nodejs git
7. npm install -g pm2
8. Install **Nginx** to serve frontend:
9. sudo yum install nginx -y
10. sudo systemctl start nginx
11. sudo systemctl enable nginx

Step 4: Deploy the Backend

1. Clone the backend repository:
2. git clone <backend_repo_url>
3. cd backend
4. npm install
5. Configure environment variables (e.g., DB connection string, JWT keys):
6. nano .env
7. Start the backend server using **pm2**:
8. pm2 start server.js --name backend
9. pm2 save

10. pm2 startup

Step 5: Deploy the Frontend

1. Clone the frontend repository:
2. git clone <frontend_repo_url>
3. cd frontend
4. npm install
5. npm run build
6. Configure Nginx to serve frontend from /var/www/html:
7. sudo cp -r build/* /usr/share/nginx/html/
8. sudo systemctl restart nginx

Step 6: Set Up RDS (Optional if using a DB)

1. Go to **RDS > Create Database**.
2. Choose database engine (MySQL, PostgreSQL, etc.).
3. Configure instance type, username, and password.
4. Connect EC2 backend app to RDS using the endpoint URL.

Step 7: Configure Load Balancer

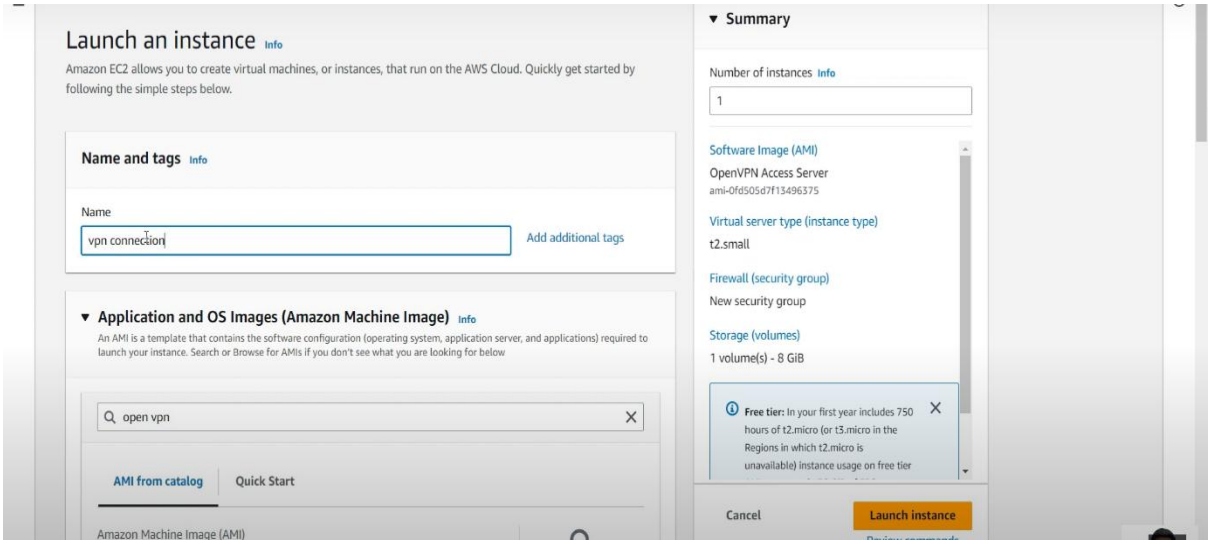
1. Go to **EC2 > Load Balancers > Create Load Balancer**.
2. Choose **Application Load Balancer (ALB)**.
3. Configure listeners (HTTP:80 / HTTPS:443).
4. Select the target group and register EC2 instances.
5. Configure health checks (e.g., /health endpoint).
6. Access app via the Load Balancer DNS.

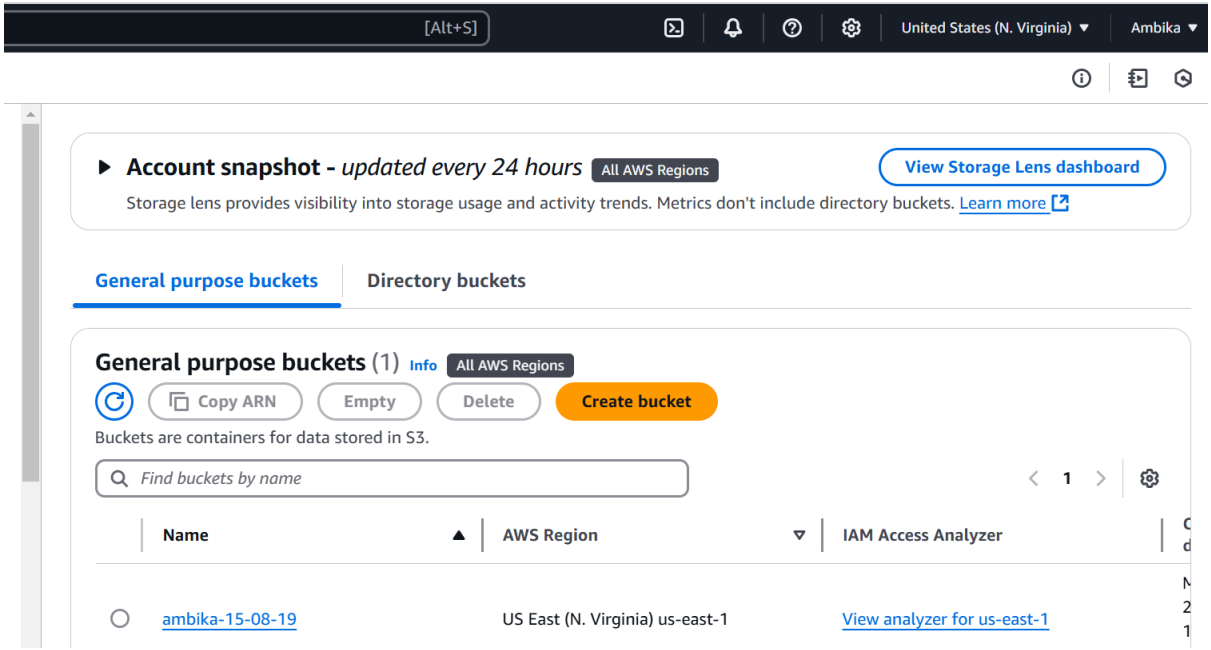
Step 8: Security and Monitoring

1. Adjust **Security Groups** to allow only necessary traffic.
2. Enable **CloudWatch** for monitoring logs and metrics.
3. Ensure auto-scaling is set up if needed.

Step 9: Testing

1. Open Load Balancer DNS in a browser.
2. Test frontend and backend functionalities.
3. Verify database connectivity and API endpoints.





Welcome to AWS Tour

Enter your details to explore AWS features

Ambika Kashyap

ambikakashyap736@gmail.com

Start Tour

AWS Key Features:

- **EC2:** Virtual servers in the cloud
- **S3:** Scalable storage service
- **Lambda:** Serverless computing
- **RDS:** Managed databases
- **IAM:** Secure identity management
- **CloudFront:** Fast content delivery