SOFTWARE ENGINEERING LAB

EXERCISE – 8

TOPIC – 3

MAVEN WEB PROJECT DEPLOYMENT IN THE AWS CLOUD USING EC2 INSTANCE

In this exercise, we will be:

- Launch an EC2 instance on AWS.
- Install Docker, Git, and Nano.
- Deploy a Maven web project using Docker and expose it on port 9090.
- Access the application online.
- Clean up resources to avoid unnecessary charges.

Note:

1. At every step take screenshots and save in a document.

2. This guide uses JDK 11, which was used for developing the project. If your project was developed with JDK 21, adjust the Dockerfile as instructed in Step 5.

Step 1: Launch an EC2 Instance

In this step, we will set up a virtual server to host our Maven web project.

1. **Log in to AWS**: Access your AWS account and navigate to **Services** > **Compute** > **EC2**.
2. **Launch the instance**:
   - **Name**: Enter a descriptive name, e.g., `MavenWebProjectServer`.
   - **AMI**: Select **Ubuntu Server (Free Tier Eligible)**.
   - **Instance Type**: Choose **t2.micro**.
   - **Key Pair**: Create a key pair or use an existing one. Save the `.pem` file securely.
   - **Network Settings**: Enable **Allow HTTP/HTTPS traffic**.

- **Storage**: Use the default size (8 GB).

3. Click **Launch Instance** and wait for the status to change to "Running."
4. **Note down the Public IP Address** from the EC2 dashboard.

Step 2: Connect to the EC2 Instance

In this step, we will connect to the server.

1. Open **PowerShell** (Windows) or **Terminal** (Mac/Linux) and navigate to the folder with the `.pem` file using the `cd` command.
2. Use SSH to connect to the instance:

   ```
   ssh -i "<KeyFile>.pem" ubuntu@<Public_IP>
   ```

   Replace `<KeyFile>` with the `.pem` file name and `<Public_IP>` with your instance's public IP.

3. If prompted, type "yes" to confirm the connection.

Step 3: Prepare the EC2 Server

Now, we will install the necessary tools.

1. **Update the system**:

   ```
   sudo apt update
   ```

2. **Install Docker**:

   ```
   sudo apt-get install docker.io -y
   ```

3. **Install Git**:

   ```
   sudo apt install git -y
   ```

4. **Install Nano** (text editor):

   ```
   sudo apt install nano -y
   ```

Step 4: Clone Your Maven Web Project

In this step, we will download the Maven project from GitHub.

1. Go to your GitHub repository, click **Code** > **HTTPS**, and copy the URL.
2. Clone the repository:

   ```
   git clone <Your_Repo_URL>
   ```

   Replace `<Your_Repo_URL>` with the copied URL.

Step 5: Create a Dockerfile

We will create a Dockerfile to containerize the Maven project.

1. Navigate to the project folder:

   ```
   cd <Your_Project_Folder>
   ```

   Replace `<Your_Project_Folder>` with the folder name.

2. Open Nano to create the Dockerfile:

   ```
   nano Dockerfile
   ```

3. Add the following content based on the JDK version used during development:
   - For **JDK 11** (used in this guide):

     ```
     FROM tomcat:9-jdk11
     COPY target/*.war /usr/local/tomcat/webapps/
     ```

   - For **JDK 21**:

     ```
     FROM tomcat:9-jdk21
     COPY target/*.war /usr/local/tomcat/webapps/
     ```

4. Save and exit Nano: Press `Ctrl + O`, then `Enter`, and `Ctrl + X`.

Explanation:

- `FROM tomcat:9-jdk11` or `FROM tomcat:9-jdk21` specifies the Tomcat base image with the appropriate JDK version.
- `COPY target/*.war /usr/local/tomcat/webapps/` copies the `.war` file into the webapps directory of Tomcat for deployment.

Step 6: Build and Run the Docker Container

1. **Build the Docker image**:

   ```
   sudo docker build -t maven-web-project .
   ```

2. **Run the container**:

   ```
   sudo docker run -d -p 9090:8080 maven-web-project
   ```

   - `-d:` Runs the container in the background.
   - `-p 9090:8080:` Maps port 9090 on your instance to port 8080 in the container.

Step 7: Configure Security Group for Port 9090

We will ensure the EC2 instance allows traffic on port 9090.

1. In the AWS EC2 dashboard, go to **Security** and click the **Security Group ID**.
2. Add an inbound rule:
   - **Type**: Custom TCP
   - **Port Range**: 9090
   - **Source**: Anywhere (0.0.0.0/0) or your IP.
3. Save the changes.

Step 8: Access the Web Application

We will now test the deployment.

1. Open a browser and navigate to:

```
http://<Public_IP>:9090/<Your_Project_Name>
```

Replace **<Public_IP>** with the instance's public IP and **<Your_Project_Name>** with your Maven project name.

Step 9: Clean Up

Finally, we will stop the container and terminate the instance.

1. **Stop the Docker container**:

```
sudo docker ps
sudo docker stop <Container_ID>
```

Replace **<Container_ID>** with the container ID.

2. **Terminate the EC2 instance** In the EC2 dashboard, go to **Instance State** and select **Terminate Instance**.