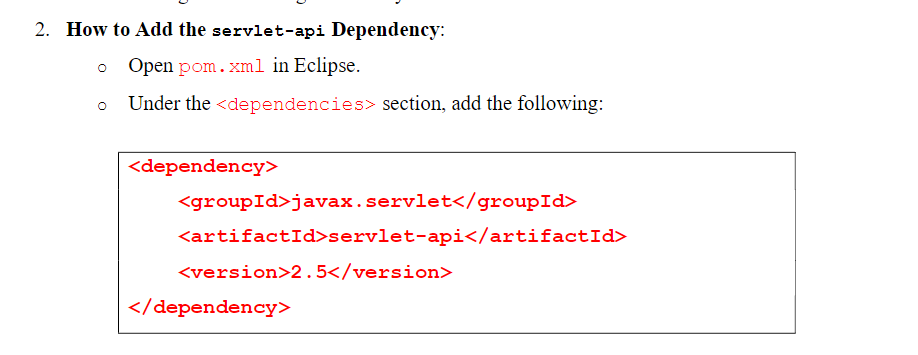
  
tomcat configure

  
servelet addition to pom.xml<html>

<head><title>My Webpage</title></head>

<body><h1>Hello from AWS!</h1></body>

</html>

2. Initialize Git in the file's folder:

git init

git add .

git commit -m "First commit"

3. Create a GitHub repository, copy its HTTPS URL, and upload your file:

git remote add origin <Your\_Repo\_URL>

git push -u origin main

Step 6: Deploy the Web Application Using Docker

Here, we will deploy the web application to the EC2 instance.

1. On the EC2 instance, clone your GitHub repository:

git clone <Your\_Repo\_URL>

2. Create a Dockerfile in the project folder using Nano:

nano Dockerfile

Add the following content:

FROM nginx:alpine

COPY . /usr/share/nginx/html

RKR21 SOFTWARE ENGINEERING LAB CSE/IT/CSM/CSD III/I

4

Save the file by pressing Ctrl + O, then Enter, and exit Nano with Ctrl + X.

3. Build and run the Docker container to serve the web application:

sudo docker build -t my-web-app .

sudo docker run -d -p 80:80 my-web-app

Step 7: Access Your Web Application

In this step, we will view the deployed web page online.

1. Copy the Public IP Address of your EC2 instance from the AWS console.

2. Paste it into your browser (e.g., http://<Public\_IP>).

3. You’ll see your web page with the message "Hello from AWS!" displayed.

Step 8: Clean Up

Finally, we will clean up resources to avoid any charges.

1. Stop the running Docker container:

sudo docker ps

sudo docker stop <Container\_ID>

2. Terminate the EC2 instance in the AWS console by selecting it, clicking Instance

State, and choosing Terminate Instance.  
  
8C………………….  
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

RKR21 SOFTWARE ENGINEERING LAB CSE/IT/CSM/CSD III/I

3

Step 4: Clone Your Maven Web Project

git clone <Your\_Repo\_URL>

Replace <Your\_Repo\_URL> with the copied URL.

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:

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:

o For JDK 11 (used in this guide):

FROM tomcat:9-jdk11

COPY target/\*.war /usr/local/tomcat/webapps/

o 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.

RKR21 SOFTWARE ENGINEERING LAB CSE/IT/CSM/CSD III/I

4

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

o -d: Runs the container in the background.

o -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:

o Type: Custom TCP

o Port Range: 9090

o 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:

RKR21 SOFTWARE ENGINEERING LAB CSE/IT/CSM/CSD III/I

5

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.