Building and Deploying a Scalable Java Web Application

Beginner Assessment

Contents

Title: Building and Deploying a Scalable Java Web Application

Difficulty Level

Beginner

Duration

90 minutes

What you will learn

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

- Manage Git repositories and initialize a project structure.
- Build and package a Java application with Maven.
- Write a Dockerfile to containerize applications.
- Deploy a containerized application to a server using Docker.

What you will be provided

- A Linux Virtual Machine with the necessary software, including Visual Studio Code, Docker, Maven, Jenkins, and Java libraries, is available in the lab.
- The project folder, containing the required files, is located at Desktop > Project.

What you need to know

- Familiarity with Docker, including building and running containers.
- Basic knowledge of Jenkins and pipelines for continuous integration.
- Some experience with Java, Maven, and Git.

Skill Tags

- Docker
- Jenkins Pipeline Basics
- Creating a Pipeline Job
- Maven
- Docker Hub

What you will do

You are tasked with creating and deploying a Java-based web application for Innovatech, a company looking to modernize its deployment workflows. The goal is to ensure that the web application can be developed, containerized, and deployed using automated tools and practices.

Note:

The user must log in to their GitHub and Docker Hub accounts using their credentials.

Activities

1. Setting Up Docker

1. Install Maven using the commands below:
   sudo apt update
   sudo apt install maven

2. Use the following credentials to log in to Jenkins:

   Username: jenkinsuser
   Password: Jenkinsuser@123

3. Run the following commands to add Jenkins to the Docker group and restart Jenkins:

   sudo usermod -aG docker jenkins
   sudo systemctl restart jenkins

2. Building the Docker Image

1. Navigate to the Project folder on the Desktop. Open the Dockerfile using Visual Studio Code in the VM lab.
2. Build the Docker image using the configurations below:

   Stage 1: Build the project using "maven:3.9.9-eclipse-temurin-17."
   - Set the working directory inside the container.
   - Copy the ***pom.xml*** and source code.
   - Build the Maven project.

   Stage 2: Run the project using openjdk:17-slim
   - Copy the JAR file from the build stage.
   - Expose the application port on ***port 8081***.
   - Run the JAR file.
3. Build the Docker image with the name my-web-app and tag 1.0.
4. Once the image is built, run the container on port 8082 with the container name "webapp_container."
5. Verify if the output is visible at http://localhost:8082.
6. Keep the Docker container in a running state.
7. Create a repository with the name "my-web-app" in Docker Hub using this link.

8. Navigate to the Jenkinsfile in the project folder and update the environment variables:

   environment {

   DOCKER_IMAGE = "{docker-hub-username}/my-web-app"

   DOCKER_TAG = "latest"

   }

3. Setting Up GitHub Repository and Docker Hub

1. Create a public repository in your personal GitHub account with the name “my-web-app” using the provided link.
2. After the repository creation, navigate to the terminal and to the project path. Fill in the GitHub username and the GitHub email id.
3. Initialize the repository, add all files, set the remote GitHub repository, and push the changes to GitHub.
4. Ensure the repository is publicly accessible. If it is private, generate a Personal Access Token (PAT) to access it. Update the changes in the Jenkinsfile with the GitHub credentials. The project should be in the master branch.
5. Use the Repository link and add it to the checkout stage of the Jenkinsfile in the project folder.
6. Once all updates with your Docker Hub and GitHub details are complete, clone the application folder or project folder to the GitHub repository created earlier using the Linux terminal.

4. Configuring Jenkins Pipeline

1. Go to Manage Jenkins > Credentials > Global > Add Credentials.
2. Add your Docker Hub credentials and save them with the ID “docker-hub-credentials.”
3. If the GitHub repository created is private, you will need to add the credentials for GitHub.
4. Create a new Jenkins pipeline named "webapplication-pipeline."
5. In the Pipeline section, change the pipeline definition to "Pipeline script from SCM."
6. Select and add the link to the GitHub repository, then save the pipeline.
7. Go to the webapplication-pipeline project in Jenkins and click on Build Now.
8. After a successful build, the Maven app will be visible on port 8081 in Chrome within the VM lab.

Testcases

1. Checking if the Docker image ‘my-web-app’ is present. [10 marks]
2. Checking if the Docker webapp_container exists. [15 marks]
3. Checking if the Docker container is running. [10 marks]
4. Checking if the application is active on the Docker container. [15 marks]
5. Checking the Jenkins pipeline name. [15 marks]

6. Checking the stage in the Jenkins pipeline. [10 marks]
7. Checking if the latest build of the Jenkins pipeline is successful. [10 marks]
8. Checking if the application is accessible on the desired port. [15 marks]