Building and Deploying a Scalable Java Web Application

Assessments

Contents

Title: Building and Deploying a Scalable Java Web Application

Difficulty Level

Practitioner

Duration

90 minutes

What you will learn

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

- Managing Git repositories and initializing a project structure.
- Building and packaging a Java application with Maven.
- Writing a Dockerfile to containerize applications.
- Deploying a containerized application to a server using Docker.

What you will be provided

- A Linux Virtual Machine with necessary software such as Visual Studio Code, docker, maven, jenkins and java libraries are available in the lab.
- The Project folder is available with the required files on the location **Desktop>Project**.

Prerequisites

- Github Account
- Docker Hub Account

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.

Activities

1. Setting Up Docker

- In the VM lab, in the applications find terminal and open it.
- Run the following commands to assign Jenkins to the Docker group and restart Jenkins:

  sudo usermod -aG docker jenkins

  sudo systemctl restart jenkins

2. Building the Docker Image

- In the application drop-down, find the development option and click on it. You can find Visual Studio code and open it. Open the project folder in VS code. Build the Docker image using the terminal.
- Navigate to the project directory and run the following command to build the Docker image with the name “my-web-app" with a tag 1.0:

  Cd Desktop/Project

  docker build -t my-web-app:1.0 .

- Once the image is built, run the image on a container on port 8081:

  docker run -p 8081:8081 my-web-app:1.0

- Verify if the output is visible on http://localhost:8081. Once confirmed, exit the process.
- Login to docker hub using the link . If you don't have a docker hub account, create a new account.
- Create a repository by using this link with the name “my-web-app" and make the repo public.
- Once the repository is created. Navigate to the Jenkinsfile in the project folder in VS code and update the environment variables: (change the {docker-hub-username} with your docker hub username and save the file)

```
environment {
DOCKER_IMAGE = "{docker-hub-username}/my-web-app"
DOCKER_TAG = "latest"
}
```

3. Configuring Jenkins for Docker Integration

- Open the Jenkins server at http://localhost:8080 and log in using the credentials provided in the readme file in the desktop. (username: jenkinsuser password : Jenkinsuser@123)

4. Setting Up GitHub Repository and Docker Hub

- Open your github using this link and create a repository with name “my-web-app”. Make it a private repository.
- Once the repository is created, navigate to the terminal and to the project path. Fill the GitHub username and the GitHub email id.

```
cd Desktop/Project
git config user.name "New GitHub Username"
git config user.email "your-email@example.com"
```

- Initialize the repository

```
git init
```

- Add all files to the repository

```
git add .
```

- Commit the changes

```
git commit -m "Initial commit"
```

- Add the GitHub remote repository

git remote add origin https://github.com/username/my-maven-app.git

- Push to GitHub

git push -u origin master

- Log in with your GitHub email ID and password as the PAT token if the authenticator is enabled for your GitHub account. By this the repository is uploaded into git
- Open the Jenkinsfile in VS code and update the Jenkinsfile with the latest repository git url in the Checkout Stage.

5. Configuring Jenkins Pipeline

- Create a new Jenkins pipeline named maven-docker-pipeline.
- In the Pipeline section, change the pipeline definition to "Pipeline script from SCM".
- Go to Manage Jenkins > Credentials > Global > Add Credentials.
- Add your Docker Hub credentials and save it with ID docker-hub-credentials.
- Go to the maven-docker-pipeline project in Jenkins and click on Build Now.
- After a successful build, the Maven app will be visible on port 8081 in chrome of VM lab.

Testcases

1. Docker Image Name.
2. Checking the Jenkins pipeline name.
3. Checking the credentials of the Jenkins user.
4. Checking the status of the jenkins pipeline.
5. Checking the output of the pipeline.