Deploying a Robust and Scalable Task Management Application

Beginner Assessment

Contents

Title: Deploying a Robust and Scalable Task Management Application 3
Difficulty Level 3
Duration 3
What you will learn 3
What you will be provided 3
What you need to know 3
Skill Tags 3
What you will do 3
Activities 4
1. Setting Up Docker 4
2. Building the Docker Image and docker-compose.yaml 4
3. Setting Up GitHub Repository and Docker Hub 5
4. Configuring Jenkins Pipeline 5
5. Test the Application 5
Testcases 6

Title: Deploying a Robust and Scalable Task Management Application

Difficulty Level

Beginner

Duration

90 minutes

What you will learn

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

- How to containerize a Java based web application using Docker and Docker-Compose.
- Managing dependencies and building the application using Maven.
- Setting up Jenkins pipelines for automation.
- Using Git for version control and seamless code management.

What you will be provided

- A Linux Virtual Machine with the necessary software, including Visual Studio Code, Docker, Maven, Jenkins, web deployment, 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, Docker and Git.
- Basics of deploying applications in production environments.

Skill Tags

- Docker
- Jenkins
- Git
- CI/CD
- Maven
- Docker Hub
- Web Deployment

What you will do

You are tasked with deploying a Task Management Application to improve team collaboration. Your goal is to create a scalable CI/CD pipeline by automating the build process with Jenkins and Maven,

containerizing the application with Docker, and deploying it to a production server. You will then test the application to ensure its scalability and reliability.

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. You can find the details of the jenkins credentials and MySQL database local passwords in Readme.txt file on Desktop.
3. Run the following commands to add Jenkins to the Docker group and restart Jenkins:
   sudo usermod -aG docker jenkins
   sudo systemctl restart jenkins

Note: Use the `docker compose` command instead of the legacy `docker-compose`, as it is integrated into Docker CLI (v20.10+), eliminating the need for a separate binary. It provides better performance, consistency, and is actively maintained, unlike the legacy command.

2. Building the Docker Image and docker-compose.yaml

1. Navigate to the Project folder on the Desktop. Open the Dockerfile using Visual Studio Code in the VM lab.
2. Build the project using **maven:3.9.9-eclipse-temurin-17**.
3. Run the project using **openjdk:17-slim**
4. Expose the application port on ***port 8081***.
5. Build the image with name “taskmanagement-app-image" with tag latest.
6. Navigate to the Project folder on the Desktop. Open the docker-compose.yaml using Visual Studio Code in the VM lab.
7. In the docker-compose.yaml , the service names and container names of the components should be taskmanagement-app and mysql-db.
8. The local host port and container port should be 8082:8081 for the taskmanagement-app service.
9. Use “root” as the username and “Root@123” as password for mysql server.
10. Create a custom bridge network named taskmanagement-network.
11. Use database name as “taskmanagementdb”.
12. The local host port and container port should be 3307:3306 for the mysql-db service.
13. Use volumes to persist data. The volume name should be “mysql-data”.
14. Build and run the docker-compose.yaml file.

15. Verify if the output is visible on http://localhost:8082. Once confirmed, down the docker compose file.
16. Create a repository with the name “taskmanagement-app” in Docker Hub using this link.
17. Navigate to the Jenkinsfile in the project folder and update the environment variables:

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

3. Setting Up GitHub Repository and Docker Hub

1. Create a public repository with the name “taskmanagement-app” in your personal GitHub account 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. Once all updates with your Docker Hub and GitHub details are complete, commit 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 "taskmanagement-app-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 taskmanagement-app-pipeline project in Jenkins and click on Build Now.

5. Test the Application

1. After a successful build, the Maven app will be visible on port 8082 in Chrome within the VM lab.
2. Create a new task and add any details of your choice to check the application working.
3. Open the terminal. To access the MySQL container, run the following command to start a bash session inside the MySQL container:

docker exec -it taskmanagement-app-pipeline-mysql-db-1 /bin/bash

4. Once inside the container, log in to the MySQL CLI using the root user:

   mysql -u root -p

5. When prompted, enter the password as **Root@123**.
6. To view the list of all databases in MySQL, run:

   show databases;

7. Switch to the taskmanagementdb database by running:

   use taskmanagementdb;

8. To see the tables within the taskmanagementdb database, run:

   show tables;

9. To view all the data stored in the tasks table, execute:

   select * from tasks;

10. Make any change in the application and check the output by running the previous step again.

Testcases

1. Checking if the user 'jenkins' is part of the 'docker' group. [5 marks]
2. Checking if the Docker image ‘taskmanagement-app-image’ is created. [5 marks]
3. Checking if the Docker image ' taskmanagement-app-image ' is tagged correctly with latest. [5 marks]
4. Checking if the Docker volume 'mysql-data ' is created after executing the Jenkins pipeline. [5 marks]
5. Checking if the Docker network ' taskmanagement-network' is created after executing the Jenkins pipeline. [5 marks]
6. Checking if the application created as part of the Jenkins pipeline execution is active in the Docker container. [10 marks]
7. Checking if the MySQL database container ' mysql-db-1' was created and exists as part of the Jenkins pipeline execution. [5 marks]
8. Checking if the Task management application container 'taskmanagement-app-1' was created and exists as part of the Jenkins pipeline execution. [5 marks]
9. Checking if the latest build of 'taskmanagement-app-pipeline' is successful. [10 marks]

10. Checking if the database 'taskmanagementdb' exists in the MySQL container. [5 marks]