Lab 4

Objective

Learn how to use Jenkins for Continuous Integration and Continuous Deployment (CI/CD).

Create and configure a Jenkins pipeline.

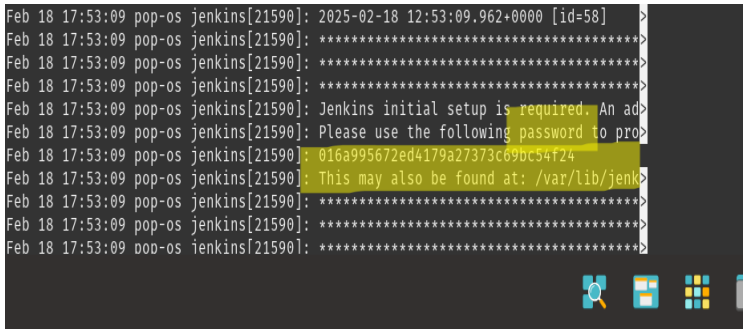
Deploy a Python application using Jenkins, Docker, and Kubernetes.

**********Install Jenkins**********

1. sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
2. echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
3. sudo apt-get update
4. sudo apt-get install fontconfig openjdk-17-jre
5. sudo apt-get install jenkins
6. sudo systemctl start jenkins
7. sudo systemctl enable jenkins

**********Access Jenkins

8. httP://localhost:8080
   ********** find the administrative password of jenkins
9. journalctl -u jenkins.service OR
10. sudo cat /var/lib/jenkins/secrets/initialAdminPassword



1. **Install Suggested Plugins:**
    a. On the Jenkins setup page, select Install suggested plugins.
2. Create an Admin User:
    a. Fill in the details to create an admin user.
3. Install Additional Plugins:
    a. Go to Manage Jenkins > Manage Plugins > Available.
    b. Install the following plugins:
        i. Git
        ii. Docker
        iii. Kubernetes
        iv. Pipeline

Step 2: Prepare the Python Application

4. Create a Project Directory:
    a. Open the terminal and create a new directory for the project:

```
mkdir python-app
cd python-app
```

    b. Create a Python Application: Create a file named app.py:

```
gedit app.py
# add the following python code in app.py

from flask import Flask app = Flask(name)

@app.route('/') def hello_world(): return 'Hello, DevOps!'

if name == 'main': app.run(host='0.0.0.0', port=5000)
```

    c. Create a requirements file: Create a file named requirements.txt:

```
gedit requirements.txt
Flask==2.0.1
```

    d. Create a Dockerfilefile: Create a file named Dockerfile:

```
#Use an official Python runtime as a parent image
```

```
FROM python:3.9-slim

#Set the working directory in the container

WORKDIR /app

#Copy the requirements file into the container

COPY requirements.txt .

#Install any needed packages specified in requirements.txt

RUN pip install --no-cache-dir -r requirements.txt

#Copy the rest of the application code

COPY . .

#Make port 5000 available to the world outside this container

EXPOSE 5000

#Run the application

CMD ["python", "app.py"]
```

e. Create Kubernetes Deployment file: Create a file named deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: python-app
  template:
    metadata:
      labels:
        app: python-app
    spec:
      containers:
```

```
      - name: python-app
        image: python-app
        ports:
        - containerPort: 5000
```

f. Create Jenkins file : create file named Jenkinsfile

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        git branch: 'main', url: '<your-repository-url>'
      }
    }
    stage('Build Docker Image') {
      steps {
        sh 'docker build -t python-app .'
      }
    }
    stage('Deploy to Kubernetes') {
      steps {
        sh 'kubectl apply -f deployment.yaml'
      }
    }
  }
  post {
    success {
      echo 'Pipeline succeeded!'
    }
    failure {
      echo 'Pipeline failed!'
    }
  }
}
```

g. Commit the files

```
#Create a repository in github,

#Clone that repository
 git init
Git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
#Now add these files separately
Git add app.py
```

```
Git add requirements.txt
Git etc.
Git status
Git log
Git remote add origin https://github.com/repository
Git commit –m “ deployment “
git push -u origin main
```

h. Create Jenkins Pipeline
    i. Create a New Pipeline Job:
        1. In Jenkins, click New Item.
        2. Enter a name (e.g., python-app-pipeline), select Pipeline, and click OK.
    ii. Configure the Pipeline:
        1. In the pipeline configuration, go to the Pipeline section.
        2. Select Pipeline script from SCM.
        3. Choose Git as the SCM.
        4. Enter the repository URL (e.g., https://github.com/your-username/python-app.git).
        5. Specify the branch (e.g., main).
        6. Set the script path to Jenkinsfile.

i. Run Jenkins Pipeline
    i. In Jenkins, go to the python-app-pipeline job
    ii. Click Build Now to trigger the pipline
j. Monitor the pipline
    i. View the pipeline progress in the jenkins console output
    ii. Ensure all stages (checkout, build Docker Image, Deploy to Kubernetes) Complete successfuly