CI/CD Pipeline with Jenkins, Docker, and Kubernetes

SUJAN K S

DEVOPS 20012025 E BATCH

1. Overview

This document explains how to set up a CI/CD pipeline using Jenkins, Docker, and Kubernetes (MicroK8s). The process involves building a Docker image, pushing it to Docker Hub, and deploying it to a Kubernetes cluster.

2. Jenkins Pipeline Configuration

The Jenkinsfile automates the CI/CD process. Below is the Jenkinsfile used for this pipeline:

Jenkinsfile

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'suanx/gfx:latest'
    }
    stages {
        stage('Git Clone - Pull files from repo') {
            steps {
                git 'https://github.com/suanx/CICD-Project.git'
            }
        }
        stage('Build Docker Image') {
            steps {
                sh 'docker build -t $DOCKER_IMAGE .'  // Corrected build command
            }
        }
        stage('Login to Docker Hub') {
            steps {
                script {
                                    withCredentials([usernamePassword(credentialsId:
'DOCKER_HUB_CREDENTIALS',    passwordVariable:    'passdochub',    usernameVariable:
'usernamedochub')]) {
                          sh 'echo "$passdochub" | docker login -u "$usernamedochub"
--password-stdin'
                    }
                }
            }
        }
        stage('Push Docker Image to Hub') {
            steps {
                sh 'docker push $DOCKER_IMAGE'
            }
        }
        stage('Deploy to Kubernetes') {
            steps {
                sh 'microk8s.kubectl apply -f Deploy.yaml'
            }
        }
    }
}
```

3. Dockerfile

```dockerfile
FROM nginx:latest

# Copy the custom index.html file
COPY index.html /usr/share/nginx/html/index.html

# Expose port 80 for web access
EXPOSE 80

# Start Nginx in the foreground
CMD ["nginx", "-g", "daemon off;"]
```

4. Kubernetes Deployment Configuration

```
Deploy.yaml defines the Kubernetes deployment and service.
```

Deploy.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-dep
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
        - image: suanx/sugfxindia
          name: c1
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: srvc
spec:
  type: NodePort
  selector:
    app: myapp
  ports:
    - port: 80            # Exposes service on port 80
      targetPort: 80       # Connects to containerPort 80
      protocol: TCP
      nodePort: 30350      # External access via NodePort
```

5. User Permissions for Docker and Kubernetes

Jenkins needs permission to execute Docker and Kubernetes commands. Run the following commands:

```
# Add Jenkins to the Docker group
sudo usermod -aG docker jenkins

# Change ownership of the Docker socket (to avoid permission errors)
sudo chown jenkins:docker /var/run/docker.sock

# Add Jenkins to the MicroK8s group for Kubernetes access
sudo usermod -a -G microk8s jenkins

# Change ownership of the kubeconfig directory
sudo chown -R jenkins ~/.kube
```

6. Summary

- **Jenkinsfile** automates Git cloning, Docker image building, pushing to Docker Hub, and deploying to Kubernetes.
- **Dockerfile** creates a containerized Nginx web application.
- **Deploy.yaml** defines a Kubernetes deployment and service for the application.
- **User permissions** are adjusted for Jenkins to access Docker and Kubernetes smoothly.

This setup enables a seamless CI/CD pipeline for deploying a web application with Jenkins, Docker, and Kubernetes.