DevOps Workflow: Jenkins CI/CD Pipeline with DockerHub & K8s Deployment

1. Overview

This project demonstrates a complete DevOps pipeline using Jenkins, Docker, and Kubernetes. The CI/CD pipeline is automated with Jenkins, builds and pushes Docker images to DockerHub, and deploys the application to a Kubernetes cluster.

2. Tools & Technologies

- Jenkins
- Docker
- DockerHub
- Kubernetes (kubeadm cluster)
- Maven
- Tomcat

3. Jenkins Configuration

1. Installed Jenkins and Docker on the same Ubuntu server.
2. Accessed Jenkins through the browser.
3. Created a pipeline job.
4. Pulled source code from GitHub (contains Dockerfile and Jenkinsfile).
5. Configured Maven and DockerHub credentials in Jenkins.
6. Upon successful build, Docker image is pushed to DockerHub.

4. Dockerfile Used

```
FROM tomcat:9-jre9
MAINTAINER madhandeva249@gmail.com
RUN rm -rf /usr/local/tomcat/webapps/ROOT/*
COPY index.html /usr/local/tomcat/webapps/ROOT/index.html
EXPOSE 8081
```

5. Jenkins Pipeline Script

```
pipeline {
    agent any
    tools { maven 'maven' }
    environment { DOCKERHUB_USERNAME = 'madhand249' }
    stages {
        stage('Clean') { steps { sh 'mvn clean' } }
        stage('Validate') { steps { sh 'mvn validate' } }
        stage('Test') { steps { sh 'mvn test' } }
        stage('Package') { steps { sh 'mvn package' } }
        stage('Build Docker Image') { steps { sh 'docker build -t wesly .' } }
        stage('Push to Docker Hub') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'credendials', usernameVariable: 'DOCKER_USER',
passwordVariable: 'DOCKER_PASS')]) {
```

```
                sh '''
                    echo "$DOCKER_PASS" | docker login -u "$DOCKER_USER" --password-stdin
                    docker tag wesly $DOCKER_USER/wesly:latest
                    docker push $DOCKER_USER/wesly:latest
                '''
            }
        }
    }
    stage('Remove Docker Image Locally') {
        steps {
            sh 'docker rmi -f ${DOCKERHUB_USERNAME} wesly || true'
            sh 'docker rmi -f wesly || true'
        }
    }
    stage('Stop and Restart Container') {
        steps {
            sh 'docker rm -f app || true'
            sh 'docker run -d --name app -p 8081:8080 ${DOCKERHUB_USERNAME}/wesly:latest'
        }
    }
  }
}
```

6. Kubernetes Configuration

Two Ubuntu servers were launched and configured with Kubernetes using kubeadm.

Deployment YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flip
spec:
  replicas: 1
  selector:
    matchLabels:
      name: deployment
  template:
    metadata:
      name: dev
      labels:
        name: deployment
    spec:
      containers:
      - name: signup
        image: dockerhub image
        imagePullPolicy: Always
        ports:
          - containerPort: 8080
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: flip-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      nodePort: 30070
  selector:
    name: deployment
  type: NodePort
```

7. Summary

This project covers the full CI/CD lifecycle from code integration to deployment in Kubernetes using Jenkins. It demonstrates how to automate builds, testing, Docker image creation, pushing to DockerHub, and finally deploying to a Kubernetes cluster.