CI/CD Pipeline Deployment with Jenkins and Kubernetes using Kubeadm

Assignment: Jenkins CICD Pipeline\ **Author**: Vinay Hinukale\ **Objective**: This guide documents the setup of a complete CI/CD pipeline using Jenkins Master-Slave architecture integrated with a Kubernetes cluster created via kubeadm.

Infrastructure Requirements

You will need **three AWS EC2 instances**:

EC2 Role	Instance Type	Description
Jenkins Master	t2.micro	Hosts Jenkins server and UI
Jenkins Slave + K8s Master	t2.medium	Dual-purpose node: Jenkins agent and Kubeadm master
Kubernetes Worker Node	t2.medium	Node that runs application workloads in the cluster

---

Step-by-Step Setup Instructions

1. Jenkins Master Setup (t2.micro EC2)

Refer to `Docker_Assignment_3`:

- Install Jenkins:

```
sudo yum install java-17-amazon-corretto.x86_64 git jenkins -y
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

- Access Jenkins UI:

```
http://<JENKINS_MASTER_PUBLIC_IP>:8080
cat /var/lib/jenkins/secrets/initialAdminPassword
```

- Install required plugins:
- Git
- Docker
- Pipeline

- SSH Slaves

---

2. Jenkins Slave + Kubernetes Master Node Setup (t2.medium EC2)

Refer to `Docker_Assignment_4`:

Install Jenkins Slave Prerequisites:

```
sudo yum install java-17-amazon-corretto docker git -y
sudo systemctl enable --now docker
```

Install Maven:

```
mkdir -p /home/ec2-user/build-tools
cd /home/ec2-user/build-tools
wget https://dlcdn.apache.org/maven/maven-3/3.9.10/binaries/apache-maven-3.9.10-bin.zip
unzip apache-maven-3.9.10-bin.zip
rm -f apache-maven-3.9.10-bin.zip
```

Configure Maven path:

```
echo 'export MAVEN_HOME=/home/ec2-user/build-tools/apache-maven-3.9.10' >> /home/ec2-user/.bash_profile
echo 'export PATH=$MAVEN_HOME/bin:$PATH' >> /home/ec2-user/.bash_profile
source /home/ec2-user/.bash_profile
```

Configure Kubeconfig Access:

```
echo 'export KUBECONFIG=/home/ec2-user/.kube/config' >> /home/ec2-user/.bash_profile
source /home/ec2-user/.bash_profile
```

Jenkins Slave Configuration (from Jenkins Master UI):

- Go to Manage Jenkins → Nodes → New Node
- Type: Permanent Agent
- Name: `slave`
- Remote Directory: `/home/ec2-user/Jenkins`
- Launch Method: SSH
- Username: `ec2-user`

- Paste SSH Private Key (from PEM file)
- Use label: `slave`

---

3. Kubernetes Cluster Setup with Kubeadm

Refer to `kubeadm_installation.md` and apply on:

- Jenkins Slave (also Kubernetes Master)
- Kubernetes Worker Node

Key Commands (run as root):

```
sudo kubeadm init
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Join Worker Node with the `kubeadm join` command output from above.

Apply Calico CNI plugin:

```
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/
manifests/calico.yaml
```

Jenkins Credentials Required

In Jenkins UI → Manage Jenkins → Credentials → Global domain:

1. GitHub Access (if private):
2. Type: Username/Password or SSH Key
3. SSH to Jenkins Slave:
4. Type: SSH Username with Private Key
5. Username: `ec2-user`
6. Docker Hub Credentials:
7. ID: `docker-hub-creds`

8. Type: Username/Password or Personal Access Token (PAT)

CI/CD Pipeline Execution

Source Project Repository

```
https://github.com/LegPro/SprintBootService-1.git
```

Contains:

- Jenkinsfile
- Dockerfile
- deploy.yaml
- Spring Boot code

Configure Pipeline Job

1. Go to Jenkins Dashboard
2. Click "New Item" → Pipeline
3. Name: `CICD_BUILD`
4. Choose: Pipeline script from SCM
5. SCM: Git
6. Repo URL: `https://github.com/LegPro/SprintBootService-1.git`
7. Script Path: `Jenkinsfile`
8. Click Save

Trigger Build

Click Build Now.

Result

- Jenkins checks out the code.
- Builds with Maven.
- Builds and pushes Docker image to Docker Hub.
- Deploys to the Kubernetes cluster.

You can verify deployment via:

```
kubectl get pods
kubectl get svc
```

And access the service using:

```
http://<K8S_WORKER_NODE_PUBLIC_IP>:<NODE_PORT>/getData
```

CI/CD Flow Diagram

```
[1] Jenkins Master (t2.micro)
     - Triggers build
     - Delegates job to Slave via SSH
        |
        V
[2] Jenkins Slave = Kubernetes Master (t2.medium)
     - Receives source code (GitHub)
     - mvn package -> creates JAR
     - Docker build and push to Docker Hub
     - kubectl apply to deploy.yaml
        |
        V
[3] Kubernetes Worker Node (t2.medium)
     - Pulls Docker image
     - Runs Spring Boot container
     - Exposes via NodePort service
```

Summary

Component	Configured On
Jenkins Master	t2.micro EC2
Jenkins Slave Agent	t2.medium + K8s Master Node
Kubernetes Cluster	Master + 1 Worker Node
Project Code & CI/CD	GitHub + Jenkins Pipeline

End-to-end DevOps pipeline powered by Jenkins, Docker, Maven, and Kubernetes using kubeadm.