# **CI/CD & Jenkins Complete Notes**

## **1. Introduction to CI/CD**

### **What is CI/CD?**

CI/CD (Continuous Integration/Continuous Deployment) is a software development practice that automates the building, testing, and deployment of applications.

### **Key Concepts**

* **Continuous Integration (CI):** Automates the merging and testing of code changes.
* **Continuous Delivery (CD):** Ensures the application is always ready for deployment.
* **Continuous Deployment:** Automates deployment to production.

### **Benefits of CI/CD**

✅ Faster Development Cycles  
✅ Reduced Bugs and Errors  
✅ Automated Testing  
✅ Reliable and Reproducible Deployments

## **2. Introduction to Jenkins**

### **What is Jenkins?**

Jenkins is an open-source automation server that helps with CI/CD by automating the software development lifecycle.

### **Jenkins Features**

* Open-source and highly extensible
* Supports various plugins
* Works with Docker, Kubernetes, Ansible, Git, and more
* Provides a web-based UI for configuration

## **3. Installing Jenkins**

### **On Ubuntu/Debian**

sudo apt update

sudo apt install openjdk-11-jdk -y

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -

echo "deb http://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list

sudo apt update

sudo apt install jenkins -y

sudo systemctl start jenkins

sudo systemctl enable jenkins

### **On RHEL/CentOS**

sudo yum install java-11-openjdk -y

sudo yum install wget

wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo

rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key

sudo yum install jenkins -y

sudo systemctl start jenkins

sudo systemctl enable jenkins

### **Access Jenkins Web Interface**

After installation, Jenkins runs on port **8080**:

http://localhost:8080

To unlock Jenkins, retrieve the initial admin password:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

## **4. Configuring Jenkins**

### **Setting Up Plugins**

Navigate to **Manage Jenkins → Manage Plugins** and install: ✅ Git Plugin  
✅ Pipeline Plugin  
✅ Docker Plugin  
✅ Ansible Plugin  
✅ Kubernetes Plugin

### **Setting Up Users**

Navigate to **Manage Jenkins → Manage Users** to create users with appropriate roles.

## **5. Creating a Freestyle Job in Jenkins**

### **Steps to Create a Job**

1. Go to **Jenkins Dashboard** → Click **New Item**
2. Select **Freestyle Project**
3. Configure **Source Code Management (SCM)** (GitHub, GitLab, etc.)
4. Add **Build Steps** (Shell script, Maven, Gradle, etc.)
5. Add **Post-Build Actions** (Deploy, Notify, etc.)
6. Click **Save** and **Build Now**

### **Example Build Script**

echo "Building the application..."

mvn clean install

echo "Build complete!"

## **6. Creating a Jenkins Pipeline**

Jenkins Pipelines are used for complex workflows.

### **Example Pipeline (Declarative)**

pipeline {

agent any

stages {

stage('Checkout Code') {

steps {

git 'https://github.com/user/repository.git'

}

}

stage('Build') {

steps {

sh 'mvn clean install'

}

}

stage('Test') {

steps {

sh 'mvn test'

}

}

stage('Deploy') {

steps {

echo 'Deploying application...'

}

}

}

}

### **Run the Pipeline**

1. Go to **Jenkins Dashboard** → Click **New Item**
2. Select **Pipeline** → Configure Pipeline script
3. Click **Save & Build Now**

## **7. Jenkins Integration with GitHub**

### **Configuring Webhooks for Jenkins**

1. Go to **GitHub Repo → Settings → Webhooks**
2. Click **Add Webhook**
3. Set the **Payload URL** as:

http://<JENKINS\_IP>:8080/github-webhook/

1. Select **Just the push event**
2. Click **Add Webhook**

## **8. Jenkins with Docker**

### **Running Jenkins in Docker**

docker run -d -p 8080:8080 -p 50000:50000 --name jenkins -v jenkins\_home:/var/jenkins\_home jenkins/jenkins:lts

### **Pipeline to Build Docker Image**

pipeline {

agent any

stages {

stage('Build Docker Image') {

steps {

sh 'docker build -t myapp .'

}

}

stage('Push to Docker Hub') {

steps {

sh 'docker login -u username -p password'

sh 'docker tag myapp username/myapp:latest'

sh 'docker push username/myapp:latest'

}

}

}

}

## **9. Jenkins with Kubernetes**

### **Deploying to Kubernetes using Jenkins Pipeline**

pipeline {

agent any

stages {

stage('Deploy to Kubernetes') {

steps {

sh 'kubectl apply -f deployment.yaml'

}

}

}

}

### **Example Kubernetes Deployment YAML (**deployment.yaml**)**

apiVersion: apps/v1

kind: Deployment

metadata:

name: myapp

spec:

replicas: 2

selector:

matchLabels:

app: myapp

template:

metadata:

labels:

app: myapp

spec:

containers:

- name: myapp

image: username/myapp:latest

ports:

- containerPort: 80

## **10. Jenkins Best Practices**

✅ Use Pipelines instead of Freestyle Jobs  
✅ Keep Jenkins updated  
✅ Use **Jenkins Agents** for scalability  
✅ Secure Jenkins with authentication and backups  
✅ Monitor Jenkins logs for troubleshooting

## **Conclusion**

Jenkins is a powerful tool for CI/CD automation! 🚀 Let me know if you need additional topics or modifications! 😊