**On Your Local Computer: Prepare the Project**

You need to create a specific folder structure with all the necessary files. Jenkins will later read this structure from your Git repository.

**### Step 1: Create the Main Project Folder**

Create a new folder for your project and open a terminal or command prompt inside it.

Bash

mkdir my-jenkins-project

cd my-jenkins-project

**### Step 2: Create Folders for Each Application**

Inside my-jenkins-project, create three folders, one for each application.

Bash

mkdir app-one

mkdir app-two

mkdir app-three

**### Step 3: Create a Simple Dockerfile**

For this exercise, you can use the same simple Dockerfile for all three apps. Create a file named **Dockerfile** inside the app-one folder with this content:

Dockerfile

FROM nginxdemos/hello:plain-text

Now, copy that exact same Dockerfile into the app-two and app-three folders.

**### Step 4: Create the Kubernetes Configuration**

1. Inside my-jenkins-project, create a new folder named k8s.
2. Inside the k8s folder, create a file named **deployment.yaml** and paste the entire configuration below into it. This single file defines all three applications and the shared Ingress.

YAML

# k8s/deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: app-one-deployment

spec:

replicas: 1

selector:

matchLabels:

app: app-one

template:

metadata:

labels:

app: app-one

spec:

containers:

- name: app-one-container

image: your-docker-hub-username/app-one:latest # Placeholder image

---

apiVersion: v1

kind: Service

metadata:

name: app-one-service

spec:

selector:

app: app-one

ports:

- port: 80

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: app-two-deployment

spec:

replicas: 1

selector:

matchLabels:

app: app-two

template:

metadata:

labels:

app: app-two

spec:

containers:

- name: app-two-container

image: your-docker-hub-username/app-two:latest # Placeholder image

---

apiVersion: v1

kind: Service

metadata:

name: app-two-service

spec:

selector:

app: app-two

ports:

- port: 80

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: app-three-deployment

spec:

replicas: 1

selector:

matchLabels:

app: app-three

template:

metadata:

labels:

app: app-three

spec:

containers:

- name: app-three-container

image: your-docker-hub-username/app-three:latest # Placeholder image

---

apiVersion: v1

kind: Service

metadata:

name: app-three-service

spec:

selector:

app: app-three

ports:

- port: 80

---

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: example-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

ingressClassName: nginx

rules:

- http:

paths:

- path: /app1

pathType: Prefix

backend:

service:

name: app-one-service

port:

number: 80

- path: /app2

pathType: Prefix

backend:

service:

name: app-two-service

port:

number: 80

- path: /app3

pathType: Prefix

backend:

service:

name: app-three-service

port:

number: 80

**Important:** The image names are placeholders. Our Jenkins pipeline will build images with these names and push them to your Docker Hub.

**### Step 5: Create the Jenkinsfile**

In the root of your my-jenkins-project folder, create a file named **Jenkinsfile**. Paste the pipeline script below into it.

**You must change your-docker-hub-username to your actual Docker Hub username.**

Groovy

// Jenkinsfile

pipeline {

agent any

environment {

// !! CHANGE THIS VALUE !!

DOCKER\_HUB\_USERNAME = "your-docker-hub-username"

KUBE\_NAMESPACE = "default"

}

stages {

stage('Checkout Code') {

steps {

checkout scm

}

}

stage('Build & Push App One') {

steps {

script {

def fullImageName = "${env.DOCKER\_HUB\_USERNAME}/app-one"

def dockerImage = docker.build(fullImageName, './app-one')

docker.withRegistry('https://registry.hub.docker.com', 'docker-hub-creds') {

dockerImage.push("latest")

}

}

}

}

stage('Build & Push App Two') {

steps {

script {

def fullImageName = "${env.DOCKER\_HUB\_USERNAME}/app-two"

def dockerImage = docker.build(fullImageName, './app-two')

docker.withRegistry('https://registry.hub.docker.com', 'docker-hub-creds') {

dockerImage.push("latest")

}

}

}

}

stage('Build & Push App Three') {

steps {

script {

def fullImageName = "${env.DOCKER\_HUB\_USERNAME}/app-three"

def dockerImage = docker.build(fullImageName, './app-three')

docker.withRegistry('https://registry.hub.docker.com', 'docker-hub-creds') {

dockerImage.push("latest")

}

}

}

}

stage('Deploy All Apps & Ingress') {

steps {

withCredentials([file(credentialsId: 'kubeconfig-file', variable: 'KUBECONFIG')]) {

sh "kubectl apply -f k8s/deployment.yaml --namespace ${KUBE\_NAMESPACE}"

}

}

}

stage('Verify All Deployments') {

steps {

withCredentials([file(credentialsId: 'kubeconfig-file', variable: 'KUBECONFIG')]) {

sh "kubectl rollout status deployment/app-one-deployment --namespace ${KUBE\_NAMESPACE}"

sh "kubectl rollout status deployment/app-two-deployment --namespace ${KUBE\_NAMESPACE}"

sh "kubectl rollout status deployment/app-three-deployment --namespace ${KUBE\_NAMESPACE}"

}

}

}

}

}

**### Step 6: Push to Git**

Jenkins needs to get these files from a Git repository.

1. Create a new, empty repository on GitHub (or another Git provider).
2. In your terminal, inside the my-jenkins-project folder, run these commands:

Bash

git init

git add .

git commit -m "Initial project setup"

git branch -M main

git remote add origin https://github.com/your-username/your-repo.git # Use your new repo URL

git push -u origin main

**You are now finished with your local computer.**

**## On Jenkins: Create and Run the Pipeline 🚀**

Now, go back to your Jenkins dashboard in your web browser.

**### Step 1: Create a New Pipeline Job**

1. On the Jenkins dashboard, click **+ New Item** on the left menu.
2. Enter a name for your job (e.g., deploy-my-apps).
3. Select the **Pipeline** option.
4. Click **OK**.

**### Step 2: Connect Jenkins to Your Git Repository**

1. You will be on the configuration page for your new job. Scroll down to the **Pipeline** section.
2. In the **Definition** dropdown menu, select **Pipeline script from SCM**.
3. The **SCM** option will appear. Select **Git**.
4. In the **Repository URL** box, paste the URL of the Git repository you just pushed to.
5. Check that the **Branch Specifier** is correct (usually \*/main).

**### Step 3: Save and Run**

1. Click the blue **Save** button at the bottom of the page.
2. This will take you to the job's main page. Click **Build Now** in the left menu.

Jenkins will now connect to your Git repository, read your Jenkinsfile, and start running the stages you defined: building each app, pushing them to Docker Hub, and deploying everything to Kubernetes. You can watch the progress in the "Stage View".