Jenkins Deployment Shared Libraries

1. **Shared Library Repository Structure (jenkins-sharedlibraries)**

**Create a different repo just for the groovy files to use shared libraries**

Your GitHub repository should look like this:

jenkins-sharedlibraries/

└── vars/

├── hello.groovy

├── clone.groovy

├── dockerBuild.groovy

└── dockerPush.groovy

**hello.groovy**

def call() {

echo "Test Jenkins"

}

**clone.groovy**

def call(String url, String branch) {

git url: "${url}", branch: "${branch}"

}

**dockerBuild.groovy**

def call(Map config = [:]) {

def imageName = config.imageName ?: error("Image name is required")

def imageTag = config.imageTag ?: 'latest'

def dockerfile = config.dockerfile ?: "${config.context}/Dockerfile"

def context = config.context ?: '.'

echo "Building Docker image: ${imageName}:${imageTag} using ${dockerfile}"

sh """

docker build -t ${imageName}:${imageTag} -f ${dockerfile} ${context}

"""

}

**dockerPush.groovy**

def call(Map config = [:]) {

def imageName = config.imageName ?: error("Image name is required")

def imageTag = config.imageTag ?: 'latest'

def credentials = config.credentials ?: 'dockerHubCred'

echo "Pushing Docker image: ${imageName}:${imageTag}"

withCredentials([usernamePassword(

credentialsId: credentials,

usernameVariable: 'DOCKER\_USERNAME',

passwordVariable: 'DOCKER\_PASSWORD'

)]) {

sh """

echo "\$DOCKER\_PASSWORD" | docker login -u "\$DOCKER\_USERNAME" --password-stdin

docker push ${imageName}:${imageTag}

"""

}

}

**2. Jenkinsfile (should be placed in the root directory of your project repo)**

Save this file as Jenkinsfile in your main application repository, e.g., Kubernetes-three-tier-deployment/.

@Library("shared") \_

pipeline {

agent { label "k8s" }

environment {

FRONTEND\_IMAGE = "spquantum/frontend"

BACKEND\_IMAGE = "spquantum/backend"

IMAGE\_TAG = "q1"

}

stages {

stage("Hello") {

steps {

script {

hello()

}

}

}

stage("Clone Code") {

steps {

script {

clone("https://github.com/projectrepo.git", "main")

}

}

}

stage("Setup Docker & Compose") {

steps {

sh "chmod +x scripts/docker-setup.sh"

sh "scripts/docker-setup.sh"

}

}

stage("Build Images") {

steps {

script {

dockerBuild(imageName: env.FRONTEND\_IMAGE, imageTag: env.IMAGE\_TAG, context: 'frontend')

dockerBuild(imageName: env.BACKEND\_IMAGE, imageTag: env.IMAGE\_TAG, context: 'backend')

}

}

}

stage("Push To DockerHub") {

steps {

script {

dockerPush(imageName: env.FRONTEND\_IMAGE, imageTag: env.IMAGE\_TAG)

dockerPush(imageName: env.BACKEND\_IMAGE, imageTag: env.IMAGE\_TAG)

}

}

}

stage("Deploy Application") {

steps {

echo "This is deploying application"

sh "docker-compose down && docker-compose up --build -d"

}

}

}

}

**3. Configure Jenkins Shared Library**

* Go to **Manage Jenkins → Configure System**
* Scroll to **Global Pipeline Libraries**
  + Name: shared
  + Default version: main
  + Retrieval method: Modern SCM: Github
  + Repository URL: https://github.com/sharedlibraries-repo-url.git

**4. Setup DockerHub Credentials**

* Navigate to **Manage Jenkins → Credentials → (global)**
* Click **Add Credentials**
  + Kind: **Username and Password**
  + ID: dockerHubCred
  + Enter DockerHub username and password or token
  + Scope: Global

**Step-by-Step: Add Jenkins Webhook in GitHub**

1. **Go to Your Repository Settings**
   * Navigate to your repo: Kubernetes-three-tier-deployment
   * Click on **Settings** (top-right corner)
2. **Access Webhooks**
   * On the left sidebar, click **Webhooks**
   * Then click the **Add webhook** button
3. **Configure the Webhook**
   * **Payload URL**:  
     http://<your-jenkins-ip>:8080/github-webhook/  
     Replace <your-jenkins-ip> with the public IP or domain of your Jenkins server.
   * **Content type**:  
     application/json
   * **Secret**:  
     Leave empty unless Jenkins is configured to verify webhook secrets.
   * **Which events would you like to trigger this webhook?**  
     Select **Just the push event**
   * **Active**:  
     Keep this box checked.
4. **Save**
   * Click the **Add webhook** button to finish