Your Jenkinsfile is mostly correct, but there are a few improvements and best practices that can be applied to make it more secure, efficient, and maintainable. Below is an updated version of your Jenkinsfile with explanations for the changes.

Shape

**Updated Jenkinsfile**

groovy

Copy

pipeline {

agent any

environment {

DOCKER\_HUB\_CREDENTIALS = credentials('docker-hub-token') // Reference the credential ID

KUBECONFIG = credentials('kubeconfig') // Reference the kubeconfig credential ID

}

stages {

// Stage 1: Checkout Code

stage('Checkout') {

steps {

git(

url: 'https://github.com/Mahananda-Kawale/node-js-sample.git',

branch: 'Mahananda-Kawale-patch-1'

)

}

}

// Stage 2: Build and Push Docker Image

stage('Build and Push Docker Image') {

steps {

script {

// Use environment variables for credentials (SECURE)

def dockerRegistry = "https://index.docker.io/v1/"

def dockerUsername = "srikantb1"

def dockerImageName = "node-js-sample"

def dockerImageTag = "1.0"

// Log in to Docker registry using credentials

withCredentials([usernamePassword(credentialsId: 'docker-hub-token', usernameVariable: 'DOCKER\_USERNAME', passwordVariable: 'DOCKER\_PASSWORD')]) {

sh "echo ${DOCKER\_PASSWORD} | docker login -u ${DOCKER\_USERNAME} --password-stdin ${dockerRegistry}"

}

// Build Docker image

sh "docker build -t ${dockerUsername}/${dockerImageName}:${dockerImageTag} ."

// Push Docker image to the registry

sh "docker push ${dockerUsername}/${dockerImageName}:${dockerImageTag}"

}

}

}

// Stage 3: Deploy to Kubernetes

stage('Deploy to Kubernetes') {

steps {

// Apply Kubernetes manifests

sh 'kubectl apply -f k8s/deployment.yaml'

// sh 'kubectl apply -f k8s/network-policy.yaml'

// sh 'kubectl apply -f k8s/persistent-volume.yaml'

}

}

// Stage 4: Clean Workspace

stage('Clean Workspace') {

steps {

cleanWs() // Safely clean the workspace

}

}

}

// Post-build actions

post {

success {

echo 'Pipeline completed successfully!'

}

failure {

echo 'Pipeline failed!'

}

}

}

Shape

**Key Changes and Improvements**

1. **Secure Handling of Credentials**:

* Removed hardcoded credentials (e.g., dockerPassword).
* Used Jenkins' withCredentials block to securely inject Docker Hub credentials (docker-hub-token).

1. **Platform-Agnostic Commands**:

* Replaced bat (Windows-specific) with sh (Unix/Linux-specific) for better compatibility. If you're running Jenkins on Windows, you can switch back to bat.

1. **Environment Variables**:

* Used environment variables (DOCKER\_HUB\_CREDENTIALS and KUBECONFIG) to manage sensitive data securely.

1. **Comments and Readability**:

* Added comments to explain each stage and step for better maintainability.

1. **Kubernetes Deployment**:

* Added comments for applying additional Kubernetes manifests (network-policy.yaml and persistent-volume.yaml). Uncomment these lines if needed.

1. **Post-Build Actions**:

* Added post block to log success or failure messages.

**How It Works**

1. **Checkout Stage**:

* Clones the Git repository (node-js-sample) from the specified branch (Mahananda-Kawale-patch-1).

1. **Build and Push Docker Image Stage**:

* Logs in to Docker Hub using credentials stored in Jenkins.
* Builds the Docker image (node-js-sample:1.0).
* Pushes the Docker image to Docker Hub.

1. **Deploy to Kubernetes Stage**:

* Applies the Kubernetes deployment manifest (deployment.yaml).
* (Optional) Applies network policy and persistent volume manifests if uncommented.

1. **Clean Workspace Stage**:

* Cleans the Jenkins workspace to free up disk space.

1. **Post-Build Actions**:

* Logs a success or failure message based on the pipeline's outcome.

**Prerequisites**

1. **Jenkins Plugins**:

* Ensure the following plugins are installed:
* Docker Pipeline
* Kubernetes CLI
* Credentials Binding

1. **Credentials in Jenkins**:

* Add the following credentials in Jenkins:
* docker-hub-token: A username/password credential for Docker Hub.
* kubeconfig: A file credential containing the Kubernetes kubeconfig file.

1. **Kubernetes Manifests**:

* Ensure the following files exist in the k8s directory:
* deployment.yaml
* network-policy.yaml (optional)
* persistent-volume.yaml (optional)

**Example Kubernetes deployment.yaml**

Here’s an example deployment.yaml for reference:

yaml

Copy

apiVersion: apps/v1

kind: Deployment

metadata:

name: node-js-sample

spec:

replicas: 2

selector:

matchLabels:

app: node-js

template:

metadata:

labels:

app: node-js

spec:

containers:

- name: node-js-sample

image: srikantb1/node-js-sample:1.0

ports:

- containerPort: 5000

---

apiVersion: v1

kind: Service

metadata:

name: node-js-service

spec:

type: NodePort

ports:

- port: 5000

targetPort: 5000

nodePort: 30007

selector:

app: node-js

Shape

**Running the Pipeline**

1. Save the Jenkinsfile in the root of your repository.
2. Create a new pipeline job in Jenkins and point it to this repository.
3. Run the pipeline and monitor the logs for success or failure.

This updated Jenkinsfile is secure, efficient, and adheres to best practices for CI/CD pipelines. Let me know if you need further assistance!