Scripted Vs Declarative Pipelines Key Differences:

<https://e.printstacktrace.blog/jenkins-scripted-pipeline-vs-declarative-pipeline-the-4-practical-differences/>

======================DECLARATIVE-PART-2===============

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building..'

}

}

stage('Test') {

steps {

echo 'Testing..'

}

}

stage('Deploy') {

steps {

echo 'Deploying....'

}

}

}

}

#!/usr/bin/groovy

// Scripted Pipeline

node {

stage('Build') {

echo 'Building....'

}

stage('Test') {

echo 'Building....'

}

stage('Deploy') {

echo 'Deploying....'

}

}

===========================================================================

MyPipeline

node {

stage('Clone Repo') {

sh 'rm -rf dockertest1'

sh 'git clone<https://github.com/mavrick202/dockertest1.git'>

}

stage('Build Docker Image') {

sh 'cd /var/lib/jenkins/workspace/pipeline1/dockertest1'

sh ' cp /var/lib/jenkins/workspace/pipeline1/dockertest1/\* /var/lib/jenkins/workspace/pipeline1'

sh 'docker build -t sreeharshav/pipelinetest:v1 .'

}

stage('Push Image to Docker Hub') {

sh 'docker push sreeharshav/pipelinetest:v1'

}

stage('Deploy to Docker Host') {

sh 'docker -H tcp://10.1.1.250:2375 run --rm -dit --name webapp1 --hostname webapp1 -p 9000:80 --network ansible\_nw sreeharshav/pipelinetest:v1'

}

stage('Check WebApp Rechability') {

sh 'curl http://ec2-3-80-165-76.compute-1.amazonaws.com:9000'

}

}

=======================================================================

#!/usr/bin/groovy

pipeline {

agent any

options {

disableConcurrentBuilds()

}

stages {

stage("Build") {

steps { buildApp() }

}

stage("Deploy - Dev") {

steps { deploy('dev') }

}

}

}

// steps

def buildApp() {

dir ('section\_4/code/cd\_pipeline' ) {

def appImage = docker.build("hands-on-jenkins/myapp:${BUILD\_NUMBER}")

}

}

def deploy(environment) {

def containerName = ''

def port = ''

if ("${environment}" == 'dev') {

containerName = "app\_dev"

port = "8888"

}

else {

println "Environment not valid"

System.exit(0)

}

sh "docker ps -f name=${containerName} -q | xargs --no-run-if-empty docker stop"

sh "docker ps -a -f name=${containerName} -q | xargs -r docker rm"

sh "docker run -d -p ${port}:5000 --name ${containerName} hands-on-jenkins/myapp:${BUILD\_NUMBER}"

}

==========================================

pipeline {

environment {

registry = "gustavoapolinario/docker-test"

registryCredential = 'dockerhub'

dockerImage = ''

}

agent any

stages {

stage('Cloning Git') {

steps {

git 'https://github.com/gustavoapolinario/microservices-node-example-todo-frontend.git'

}

}

stage('Building image') {

steps{

script {

dockerImage = docker.build registry + ":$BUILD\_NUMBER"

}

}

}

stage('Deploy Image') {

steps{

script {

docker.withRegistry( '', registryCredential ) {

dockerImage.push()

}

}

}

}

stage('Remove Unused docker image') {

steps{

sh "docker rmi $registry:$BUILD\_NUMBER"

}

}

}

}

======================================================

Multi-Job-Pipeline:

node {

stage('Build Job1') {

build 'Job1'

}

stage('Build Job2') {

build 'Job1'

}

stage('Build Job3') {

build 'Job1'

}

}