node('mvn') {

try {

// define commands

def mvnCmd = "/opt/apache-maven-3.5.4/bin/mvn"

stage ('Build') {

//git 'http://10.129.194.153/root/workflow-manager.git'

git '$git\_url'

sh "${mvnCmd} clean install -DskipTests=true -DproxyHost=10.158.100.1 -DproxyPort=8080"

}

stage('static code analysis') {

sh "${mvnCmd} sonar:sonar -Dsonar.host.url=http://10.129.210.114:8085 -DskipTests=true"

}

stage('artifactory') {

sh '''

curl -v http://10.129.211.174:8081/repository/operation-catalog-repo/sit/ --user admin:admin123 --upload-file $WORKSPACE/target/\*.jar

'''

}

stage('pulling artifact') {

sh '''

export nexusurl=http://10.129.211.174:8081

export repo=operation-catalog-repo

export group=hosted

export artifact\_id=digimopcatalog

export version=Release

export artifact\_name=digimopcatalog-1.0.jar

export nexus\_username=admin

export nexus\_password=admin123

url="$nexusurl/repository/$repo/$group/$artifact\_id/$version/$artifact\_name"

echo $url

curl -vvv -X GET -u $nexus\_username:$nexus\_password "$url" -H "accept:application/json" -O --progress-bar

'''

}

stage ('Deploy DEV') {

sh "/home/cloud-user/oc-tool/oc login $openshift\_url --token=$ocp\_token --insecure-skip-tls-verify"

sh '''

result=$(/home/cloud-user/oc-tool/oc get projects | grep $namespace\_name | awk '{print $1}')

if [ ! -z "$result" ];

then

echo "project already exists. Deleting old project"

/home/cloud-user/oc-tool/oc delete project $namespace\_name

sleep 30s

echo "creating new project with name $namespace\_name"

/home/cloud-user/oc-tool/oc new-project $namespace\_name

else

echo "creating new project with name $namespace\_name"

/home/cloud-user/oc-tool/oc new-project $namespace\_name

fi

'''

sh "rm -rf oc-build && mkdir oc-build"

sh "pwd"

sh "cp target/\*.jar oc-build/\*.jar"

sh "/home/cloud-user/oc-tool/oc new-build --name=$appname --image-stream=openshift/redhat-openjdk18-openshift:1.1 --strategy=source --binary=true --labels=app=$appname -n $namespace\_name || true"

sh "/home/cloud-user/oc-tool/oc start-build $appname --from-dir=oc-build --wait=true -n $namespace\_name"

sh "/home/cloud-user/oc-tool/oc new-app $appname:$version -n $namespace\_name"

sh "/home/cloud-user/oc-tool/oc delete svc $appname -n $namespace\_name"

sh "/home/cloud-user/oc-tool/oc expose dc $appname --port=$port -n $namespace\_name"

sh "/home/cloud-user/oc-tool/oc expose svc/$appname -n $namespace\_name"

}

notifySuccessful()

}

catch (e) {

currentBuild.result = "FAILED"

notifyJobFailed()

throw e

}

}

def notifySuccessful() {

emailext (

to: 'hruday.hanchate.ext@nokia.com',

subject: "SUCCESSFUL: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]'",

mimeType: 'text/html',

attachLog: true,

body: """<p>SUCCESSFUL: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]':</p>

<p>Check console output at "<a href="${env.BUILD\_URL}">${env.JOB\_NAME} [${env.BUILD\_NUMBER}]</a>"</p>"""

)

}

def notifyJobFailed() {

emailext (

to: 'hruday.hanchate.ext@nokia.com',

subject: "FAILED: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]'",

mimeType: 'text/html',

attachLog: true,

body: """<p>FAILED: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]':</p>

<p>Check console output at "<a href="${env.BUILD\_URL}">${env.JOB\_NAME} [${env.BUILD\_NUMBER}]</a>"</p>"""

)

}

checkout([$class: 'GitSCM', branches: [[name: '$branch']], doGenerateSubmoduleConfigurations: false, extensions: [[$class: 'LocalBranch', localBranch: '']], submoduleCfg: [], userRemoteConfigs: [[url: 'http://10.129.194.153/chandrag/digimopcatalog.git']]])

pipeline {

/\*

\* Run everything on an existing agent configured with a label 'docker'.

\* This agent will need docker, git and a jdk installed at a minimum.

\*/

agent {

node {

label 'docker'

}

}

// using the Timestamper plugin we can add timestamps to the console log

options {

timestamps()

}

environment {

//Use Pipeline Utility Steps plugin to read information from pom.xml into env variables

IMAGE = readMavenPom().getArtifactId()

VERSION = readMavenPom().getVersion()

}

stages {

stage('Build') {

agent {

docker {

/\*

\* Reuse the workspace on the agent defined at top-level of Pipeline but run inside a container.

\* In this case we are running a container with maven so we don't have to install specific versions

\* of maven directly on the agent

\*/

reuseNode true

image 'maven:3.5.0-jdk-8'

}

}

steps {

// using the Pipeline Maven plugin we can set maven configuration settings, publish test results, and annotate the Jenkins console

withMaven(options: [findbugsPublisher(), junitPublisher(ignoreAttachments: false)]) {

sh 'mvn clean findbugs:findbugs package'

}

}

post {

success {

// we only worry about archiving the jar file if the build steps are successful

archiveArtifacts(artifacts: '\*\*/target/\*.jar', allowEmptyArchive: true)

}

}

}

stage('Quality Analysis') {

parallel {

// run Sonar Scan and Integration tests in parallel. This syntax requires Declarative Pipeline 1.2 or higher

stage ('Integration Test') {

agent any //run this stage on any available agent

steps {

echo 'Run integration tests here...'

}

}

stage('Sonar Scan') {

agent {

docker {

// we can use the same image and workspace as we did previously

reuseNode true

image 'maven:3.5.0-jdk-8'

}

}

environment {

//use 'sonar' credentials scoped only to this stage

SONAR = credentials('sonar')

}

steps {

sh 'mvn sonar:sonar -Dsonar.login=$SONAR\_PSW'

}

}

}

}

stage('Build and Publish Image') {

when {

branch 'master' //only run these steps on the master branch

}

steps {

/\*

\* Multiline strings can be used for larger scripts. It is also possible to put scripts in your shared library

\* and load them with 'libaryResource'

\*/

sh """

docker build -t ${IMAGE} .

docker tag ${IMAGE} ${IMAGE}:${VERSION}

docker push ${IMAGE}:${VERSION}

"""

}

}

}

post {

failure {

// notify users when the Pipeline fails

mail to: 'team@example.com',

subject: "Failed Pipeline: ${currentBuild.fullDisplayName}",

body: "Something is wrong with ${env.BUILD\_URL}"

}

}

}

Contents

1. Introduction

2. Installation procedure

2.1 Prerequisites

2.2 Installation of Jenkins

2.3 Installation of Maven

2.4 Installation of SonarQube

2.5 Installation of Nexus Repository

3. Launching slave node in Jenkins

3.1 Creating slave node

3.2 Configure slave node to Maven

4. Setting-up CI/CD Pipeline

4.1 Creating Pipeline project

4.2 Stages in Continuous Integration

4.3 Stages in Continuous Deployment

4.4 Email Notifications for Build Status

1. Introduction

CI/CD Pipeline coordinate a wide variety of activities, such as checking out and building new versions of code, running tests, and deploying software. These integrations are handled by automated build and testing systems, which means that your developers are alerted to issues such as code clashes or broken unit tests as early as possible.

2. Installation procedure

2.1 Prerequisites:

1. Launch four VM's from Openstack cloud and Deploy CentOs 7 server instance in all.

2. Logged into your machine as a non-root user with sudo privileges.

3. Gather Project details from the Development teams.

2.2 Installation of Jenkins server:

Login to the Jenkins VM and update the packages.

Step 1: Update the system with stable packages.

sudo yum install epel-release

sudo yum update

sudo reboot

When the reboot finishes, login with the same sudo user.

Step 2: Install Java.

Before installing Jenkins, setup a Java virtual machine OpenJDK Runtime Environment 1.8.0

sudo yum install java-1.8.0-openjdk.x86\_64

After the installation, confirm it by running the following command:

java -version

Step 3: Install Jenkins

cd ~

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

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

yum install jenkins

Start the Jenkins service and set it to run at boot time:

sudo systemctl start jenkins.service

sudo systemctl enable jenkins.service

Now, test Jenkins by visiting the following address from your web browser:

http://10.129.211.118:8080

2.3 Installation of Maven:

Login to the Jenkins VM and update the packages.

Step 1: Install OpenJDK 8

sudo yum install -y java-1.8.0-openjdk-devel

Step 2: Install Apache Maven

First, download and extract the Apache Maven 3.5 archive.

cd

wget http://www-us.apache.org/dist/maven/maven-3/3.5.0/binaries/apache-maven-3.5.0-bin.tar.gz

tar -zxvf apache-maven-3.5.0-bin.tar.gz

Moving Apache Maven 3.5 files to a other location and change their ownership to root:root:

sudo mv ~/apache-maven-3.5.0 /opt

sudo chown -R root:root /opt/apache-maven-3.5.0

Add the path /opt/apache-maven to the PATH environment variable.

echo 'export PATH=$PATH:/opt/apache-maven/bin' | sudo tee -a /etc/profile

source /etc/profile

Below command to verify the installation.

mvn –version

2.4 Installation of SonarQube:

Login to the Jenkins VM and update the packages.

Step 1: Perform a system update

sudo yum -y install epel-release

sudo yum -y update

sudo shutdown -r now

Once the system has finished rebooting, log in again as the sudo user and proceed to the next step.

Step 2: Install Java

wget --no-cookies --no-check-certificate --header"Cookie:oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm"

Install the downloaded package by typing:

sudo yum -y localinstall jdk-8u131-linux-x64.rpm

java -version

Step 3: Install and configure PostgreSQL

Install PostgreSQL repository by typing:

sudo rpm -Uvh https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86\_64/pgdg-centos96-9.6- 3.noarch.rpm

Install PostgreSQL database server by running:

sudo yum -y install postgresql96-server postgresql96-contrib

Initialize the database:

sudo /usr/pgsql-9.6/bin/postgresql96-setup initdb

Edit the /var/lib/pgsql/9.6/data/pg\_hba.conf to enable MD5-based authentication.

sudo nano /var/lib/pgsql/9.6/data/pg\_hba.conf

Find the following lines and change peer to trust and idnet to md5.

# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all peer

# IPv4 local connections:

host all all 127.0.0.1/32 ident

# IPv6 local connections:

host all all ::1/128 ident

Once updated, the configuration should look like the one shown below.

# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all trust

# IPv4 local connections:

host all all 127.0.0.1/32 md5

# IPv6 local connections:

host all all ::1/128 md5

Start PostgreSQL server and enable it to start automatically at boot time by running:

sudo systemctl start postgresql-9.6

sudo systemctl enable postgresql-9.6

Change the password for the default PostgreSQL user.

sudo passwd postgres

Switch to the postgres user.

su – postgres

Create a new user by typing:

createuser sonar

Switch to the PostgreSQL shell.

Psql

Set a password for the newly created user for SonarQube database.

ALTER USER sonar WITH ENCRYPTED password 'StrongPassword';

Create a new database for PostgreSQL database by running:

CREATE DATABASE sonar OWNER sonar;

Exit from the psql shell:

\q

Switch back to the sudo user by running the exit command.

Step 4: Download and configure SonarQube

Download the SonarQube installer files archive.

wget https://sonarsource.bintray.com/Distribution/sonarqube/sonarqube-6.4.zip

Install unzip by running:

sudo yum -y install unzip

Unzip the archive using the following command.

sudo unzip sonarqube-6.4.zip -d /opt

Rename the directory:

sudo mv /opt/sonarqube-6.4 /opt/sonarqube

Open the SonarQube configuration file using your favorite text editor.

sudo nano /opt/sonarqube/conf/sonar.properties

Find the following lines.

#sonar.jdbc.username=

#sonar.jdbc.password=

Uncomment and provide the PostgreSQL username and password of the database that created.

sonar.jdbc.username=sonar

sonar.jdbc.password=StrongPassword

Next, find:

#sonar.jdbc.url=jdbc:postgresql://localhost/sonar

Uncomment the line, save the file and exit from the editor.

Step 5: Configure Systemd service

SonarQube can be started directly using the startup script provided in the installer package.

sudo nano /etc/systemd/system/sonar.service

Populate the file with:

[Unit]

Description=SonarQube service

After=syslog.target network.target

[Service]

Type=forking

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start

ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

User=root

Group=root

Restart=always

[Install]

WantedBy=multi-user.target

Start the application by running:

sudo systemctl start sonar

Enable the SonarQube service to automatically start at boot time.

sudo systemctl enable sonar

To check if the service is running, run:

sudo systemctl status sonar

http://10.129.210.114:8085

2.5 Installation of Nexus repository:

Step 1: Login to nexus vm and update.

sudo yum update -y

Step 2: Install OpenJDK 1.8

sudo yum install java-1.8.0-openjdk.x86\_64

Step 3: Create a directory named app and cd into the directory.

sudo mkdir /app && cd /app

Step 4: Download the latest nexus 3

sudo wget https://sonatype-download.global.ssl.fastly.net/nexus/3/nexus-3.0.2-02-unix.tar.gz

Untar the downloaded file.

sudo tar -xvf nexus-3.0.2-02-unix.tar.gz

Rename the untared file to nexus.

sudo mv nexus-3.0.2-02 nexus

Step 5: create a new user named nexus.

sudo adduser nexus

Change the ownership of nexus file to nexus user.

sudo chown -R nexus:nexus /app/nexus

Open /app/nexus/bin/nexus.rc file, uncomment run\_as\_user parameter and set it as following.

run\_as\_user="nexus"

Step 6: start the Nexus service, use the following command.

sudo service nexus start

The above command will start the nexus service on port 8081. To access the nexus dashboard

http://10.129.211.174:8081/

For stopping,

sudo service nexus stop

For restarting,

sudo service nexus restart

3. Launching slave node in Jenkins

3.1 Creating Slave node:

login to Jenkins

Navigate to Manage Jenkins and select Manage nodes

Click on new node and type node name

Select the permanent agent and click on 'ok'

3.2 Configure slave node to maven:

Navigate to configure in slave node

Type the remote directory path of maven vm & label names

Launch Method : Launch slave agents via SSH

Host : “maven vm ip”

Credentials: “Add username/password of maven vm”

Host key Verification Strategy : Non verifying Verification Strategy

Click on 'Save'

4. Setting-up CI/CD Pipeline

4.1 Creating Pipeline project:

On Jenkins home page, click on new item.

Type Job name and Select Pipeline project. Click on 'Ok'

In General section, select the project is paramatereized.

On Pipeline section, select the definition as 'Pipeline Script'

Start writing the pipeline script in the workspace.

4.2 Stages in Continuous Integration:

Build: Performing git checkout and building the source code with Maven project.

stage ('Build') {

git 'http://10.129.194.153/root/workflow-manager.git'

sh "${mvnCmd} clean install -DskipTests=true -DproxyHost=10.158.100.1 -DproxyPort=8080"

}

Static code analysis: checking code standards with Maven

stage('static code analysis') {

sh "${mvnCmd} sonar:sonar -Dsonar.host.url=http://10.129.210.114:8085 -DskipTests=true"

}

Artifactory: Uploading artifact to Nexus repository

stage('artifactory') {

sh “curl -v http://10.129.211.174:8081/repository/operation-catalog-repo/sit/ --user admin:admin123 --upload-file $WORKSPACE/target/\*.jar”

}

4.3 Stages in Continuous Deployment:

Pulling Artifact: Downloading artifact from Nexus repository to Maven target folder.

stage('pulling artifact') {

sh '''

export nexusurl=http://10.129.211.174:8081

export repo=operation-catalog-repo

export group=hosted

export artifact\_id=digimopcatalog

export version=Release

export artifact\_name=digimopcatalog-1.0.jar

export nexus\_username=admin

export nexus\_password=admin123

url="$nexusurl/repository/$repo/$group/$artifact\_id/$version/$artifact\_name"

echo $url

curl -vvv -X GET -u $nexus\_username:$nexus\_password "$url" -H "accept:application/json"

'''

}

Deploy to OCP: Deploying artifact to Openshift platform

String parameters passed in the pipeline job for the below details

- Namespace name

- app name

- Port no

- Version

- Openshift URL

- OCP token

After creating namespace, create folder “oc-build” in openshift

Copying artifact from maven target/\*.jar to oc-build/\*.jar

Building & deploying the artifact in the openshift namespace

Exposing the application to the external acccess via routes

stage ('Deploy DEV') {

sh "/home/cloud-user/oc-tool/oc login $openshift\_url --token=$ocp\_token --insecure-skip-tls-verify"

sh '''

result=$(/home/cloud-user/oc-tool/oc get projects | grep $namespace\_name | awk '{print $1}')

if [ ! -z "$result" ];

then

echo "project already exists. Deleting old project"

/home/cloud-user/oc-tool/oc delete project $namespace\_name

sleep 30s

echo "creating new project with name $namespace\_name"

/home/cloud-user/oc-tool/oc new-project $namespace\_name

else

echo "creating new project with name $namespace\_name"

/home/cloud-user/oc-tool/oc new-project $namespace\_name

fi

'''

sh "rm -rf oc-build && mkdir oc-build"

sh "pwd"

sh "cp target/\*.jar oc-build/\*.jar"

sh "/home/cloud-user/oc-tool/oc new-build --name=$appname --image-stream=openshift/redhat-openjdk18- openshift:1.1 --strategy=source --binary=true --labels=app=$appname -n $namespace\_name || true"

sh "/home/cloud-user/oc-tool/oc start-build $appname --from-dir=oc-build --wait=true -n $namespace\_name"

sh "/home/cloud-user/oc-tool/oc new-app $appname:$version -n $namespace\_name"

sh "/home/cloud-user/oc-tool/oc delete svc $appname -n $namespace\_name"

sh "/home/cloud-user/oc-tool/oc expose dc $appname --port=$port -n $namespace\_name"

sh "/home/cloud-user/oc-tool/oc expose svc/$appname -n $namespace\_name"

}

4.4 Email notifications for Build Status:

Notifies recipients with build status and logs through emails

Email notification configuration:

Configure Email:

SMTP Server : mailrelay.int.nokia.com

SMTP Port : 25

Configuring email in Pipeline code:

node('mvn') {

try {

-------

-------

notifySuccessful()

}

catch (e) {

currentBuild.result = "FAILED"

notifyJobFailed()

throw e

}

}

def notifySuccessful() {

emailext (

to: 'recipient@nokia.com',

subject: "SUCCESSFUL: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]'",

mimeType: 'text/html',

attachLog: true,

body: """<p>SUCCESSFUL: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]':</p>

<p>Check console output at "<a href="${env.BUILD\_URL}">${env.JOB\_NAME} [${env.BUILD\_NUMBER}]</a>"</p>"""

)

}

def notifyJobFailed() {

emailext (

to: 'recepient@nokia.com',

subject: "FAILED: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]'",

mimeType: 'text/html',

attachLog: true,

body: """<p>FAILED: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]':</p>

<p>Check console output at "<a href="${env.BUILD\_URL}">${env.JOB\_NAME} [${env.BUILD\_NUMBER}]</a>"</p>"""

)

}

Jenkins Pipeline is a suite of plugins that supports implementing and integrating continuous delivery pipelines into Jenkins.

Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the Pipeline DSL.

Since Pipeline code (i.e. Scripted Pipeline in particular) is written in Groovy-like syntax,

if your IDE is not correctly syntax highlighting your Jenkinsfile,

try inserting the line #!/usr/bin/env groovy at the top of the Jenkinsfile, [4] [5] which may rectify the issue.

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

Freestyle job - Ansible Jenkins

cd Ansible

ansible-playbook ansible\_sample.yaml -e uid=$uid -e resturl=http://yangrestconf-yangrestconf.haproxy.project.matrix.com/restconf/data/NokiaModule:Ansible-Template

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

Python script template: freesyle

String Parameter:

Name uid

Default Value 000|TEST|002

Description 000|TEST|002 userid workflowid taskid

Restrict where this project can be run : vnf slave

Sourcecode Mgmt: Git

Execute shell:

cd Python27

python Python27.py $uid http://yangrestconf-yangrestconf.haproxy.project.matrix.com/restconf/data/NokiaModule:PythonScript-Template

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

Ansible\_Template - freesyle

String Parameter:

Name uid

Default Value 000|TEST|002

Description 000|TEST|002 userid workflowid taskid

Restrict where this project can be run : vnf slave

Sourcecode Mgmt: Git

Execute shell:

cd Ansible

ansible-playbook ansible\_sample.yaml -e uid=$uid -e resturl=http://yangrestconf-yangrestconf.haproxy.project.matrix.com/restconf/data/NokiaModule:Ansible-Template

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

Shell template - freesyle

String Parameter:

Name uid

Default Value 000|TEST|002

Description 000|TEST|002 userid workflowid taskid

Restrict where this project can be run : vnf slave

Sourcecode Mgmt: Git

Branches to build :\*\*

Execute shell:

cd ShellScript

chmod +x \*.sh

./shell\_sample.sh $uid http://yangrestconf-yangrestconf.haproxy.project.matrix.com/restconf/data/NokiaModule:Shell-Template

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

VNF-Ansible-Configure

String Parameter:

Name uid

Default Value 000|TEST|002

Description 000|TEST|002 userid workflowid taskid

Restrict where this project can be run : vnf slave

Sourcecode Mgmt: Git

Branches to build :\*\*

Execute shell:

cd vnf-configure-dir

chmod +x \*.sh

./sample.sh $uid http://yangrestconf-yangrestconf.haproxy.project.matrix.com/restconf/data/NokiaModule:VNF-Ansible-Configure

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

workflow;

Pipeline script:

node('maven') {

// define commands

def mvnCmd = "mvn"

stage ('Build') {

git 'http://10.129.194.153/root/workflow-manager.git'

// git 'http://10.129.194.153/root/Jenkins-Adapter.git'

//git '$gitpath'

sh "${mvnCmd} clean install -DskipTests=true -DproxyHost=10.158.100.1 -DproxyPort=8080"

}

stage ('Deploy DEV') {

// sh "oc delete bc,dc,is,svc,route -l app=cbam-adapter -n matrix-demo"

// sh "rm -rf oc-build && mkdir oc-build"

// sh "cp target/\*.jar oc-build/\*.jar"

// sh "oc new-build --name=workflow --image-stream=openshift/redhat-openjdk18-openshift:1.1 --strategy=source --binary=true --labels=app=workflow -n matrix-demo || true"

// sh "oc start-build workflow --from-dir=oc-build --wait=true -n matrix-demo"

// sh "oc new-app cbam-adapter:latest -n matrix-demo"

// sh "oc delete svc cbam-adapter -n matrix-demo"

// sh "oc expose dc cbam-adapter --port=8080 -n matrix-demo"

// sh "oc expose svc/cbam-adapter -n matrix-demo"

// sh "oc delete dc,bc,is,svc,route -l app=workflow -n logs-backup"

sh "rm -rf oc-build && mkdir oc-build"

sh "cp target/\*.jar oc-build/\*.jar"

sh "oc new-build --name=workflow --image-stream=openshift/redhat-openjdk18-openshift:1.1 --strategy=source --binary=true --labels=app=workflow -n logs-backup || true"

//sh "oc policy add-role-to-user edit system:serviceaccount:executionjenkins:jenkins -n matrix-demo-test-logs"

sh "oc start-build workflow --from-dir=oc-build --wait=true -n logs-backup"

// sh "oc new-app workflow:latest -n logs-backup"

// sh "oc delete svc workflow -n logs-backup"

// sh "oc expose dc/workflow --port=8000 -n logs-backup"

// sh "oc expose svc/workflow -n logs-backup"

}

}

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

file input:

File parameter

file location: file.txt

Restrict where this project can be run: jjcs1

Execute shell:

ls

pwd

hostname

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

oc whoami -t

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

Declarative Pipeline supports an environment directive, whereas users of Scripted Pipeline must use the withEnv step.

Declarative Pipeline supports parameters out-of-the-box, allowing the Pipeline to accept user-specified parameters at runtime via the parameters

directive. Configuring parameters with Scripted Pipeline is done with the properties step, which can be found in the Snippet Generator.

If you configured your pipeline to accept parameters using the Build with Parameters option,

those parameters are accessible as members of the params variable.

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

Scripted Pipeline however relies on Groovy’s built-in try/catch/finally semantics for handling failures during execution of the Pipeline.

creating a map in groovy:

git url: 'git://example.com/amazing-project.git', branch: 'master'

git([url: 'git://example.com/amazing-project.git', branch: 'master'])

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

Sctrited pipeline: executing in parallel:

stage('Build') {

/\* .. snip .. \*/

}

stage('Test') {

parallel linux: {

node('linux') {

checkout scm

try {

unstash 'app'

sh 'make check'

}

finally {

junit '\*\*/target/\*.xml'

}

}

},

windows: {

node('windows') {

/\* .. snip .. \*/

}

}

}

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

exit

2 ls

3 exit

4 ls

5 yum update

6 sudo yum -y install epel-release

7 sudo yum -y update

8 yum install wget

9 wget --no-cookies --no-check-certificate --header "Cookie:oraclelicense=a ccept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u131-b1 1/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm"

10 sudo yum -y localinstall jdk-8u131-linux-x64.rpm

11 java -version

12 sudo rpm -Uvh https://download.postgresql.org/pub/repos/yum/9.6/redhat/rh el-7-x86\_64/pgdg-centos96-9.6-3.noarch.rpm

13 sudo yum -y install postgresql96-server postgresql96-contrib

14 sudo /usr/pgsql-9.6/bin/postgresql96-setup initdb

15 vi /var/lib/pgsql/9.6/data/pg\_hba.conf

16 sudo systemctl start postgresql-9.6

17 sudo systemctl enable postgresql-9.6

18 sudo passwd postgres

19 su - postgres

20 wget https://sonarsource.bintray.com/Distribution/sonarqube/sonarqube-6.4 .zip

21 sudo yum -y install unzip

22 sudo unzip sonarqube-6.4.zip -d /opt

23 sudo mv /opt/sonarqube-6.4 /opt/sonarqube

24 sudo nano /opt/sonarqube/conf/sonar.properties

25 sudo vi /opt/sonarqube/conf/sonar.properties

26 sudo vi /etc/systemd/system/sonar.service

27 sudo systemctl start sonar

28 sudo systemctl enable sonar

29 sudo systemctl status sonar

30 sudo systemctl stop sonar

31 sudo systemctl status sonar

32 sudo systemctl start sonar

33 systemctl status firewalld

34 sudo vi /opt/sonarqube/conf/sonar.properties

35 sudo vi /opt/sonarqube/conf/sonar.properties

36 q!

37 ls

38 ps -eaf | grep sonar

39 systemctl status sonar

40 systemctl stop sonar

41 ps -eaf | grep sonar

42 sestatus

43 systemctl status firewalld

44 systemctl start sonar

45 systemctl status sonar

46 exit

47 ls

48 systemctl start sonar

49 systemctl status sonar

50 sudo vi /opt/sonarqube/conf/sonar.properties

51 systemctl stop sonar

52 systemctl start sonar

53 systemctl status sonar

54 sudo vi /opt/sonarqube/conf/sonar.properties

55 systemctl start sonar

56 systemctl stop sonar

57 sudo vi /opt/sonarqube/conf/sonar.properties

58 systemctl stop sonar

59 systemctl status sonar

60 systemctl start sonar

61 vi /opt/sonarqube/conf/sonar.properties

62 setenforce 0

63 systemctl status sonar

64 vi /opt/sonarqube/conf/sonar.properties

65 systemctl stop sonar

66 vi /opt/sonarqube/conf/sonar.properties

67 systemctl start sonar

68 systemctl stop sonar

69 vi /opt/sonarqube/conf/sonar.properties

70 systemctl start sonar

71 ip add

72 vi /opt/sonarqube/conf/sonar.properties

73 systemctl restart sonar

74 systemctl status sonar

75 vi /opt/sonarqube/conf/sonar.properties

76 cat ~/.bashrc

77 env | grep proxy7

78 env | grep proxy

79 vi /opt/sonarqube/conf/sonar.properties

80 systemctl stop sonar

81 vi /opt/sonarqube/conf/sonar.properties

82 systemctl start sonar

83 vi /opt/sonarqube/conf/sonar.properties

84 systemctl restart sonar

85 vi /opt/sonarqube/conf/sonar.properties

86 systemctl restart sonar

87 systemctl status sonar

88 vi /opt/sonarqube/conf/sonar.properties

89 systemctl stop sonar

90 systemctl status sonar

91 systemctl start sonar

92 systemctl status sonar

93 vi /opt/sonarqube/conf/sonar.properties

94 systemctl stop sonar

95 systemctl status sonar

96 systemctl start sonar

97 vi /opt/sonarqube/conf/sonar.properties

98 systemctl stop sonar

99 systemctl status sonar

100 systemctl start sonar

101 systemctl status sonar

102 vi /opt/sonarqube/conf/sonar.properties

103 nestat

104 netstat | grep 9000

105 netstat | grep 194464

106 vi /opt/sonarqube/conf/sonar.properties

107 systemctl stop sonar

108 systemctl start sonar

109 vi /opt/sonarqube/conf/sonar.properties

110 systemctl start sonar

111 systemctl stop sonar

112 netstat | grep 9090

113 vi /opt/sonarqube/conf/sonar.properties

114 systemctl start sonar

115 netstat | grep 9090

116 netstat -a

117 netstat -a | grep LISTENING

118 sudo netstat -plnt

119 sudo netstat -plnt | grep ':9090'

120 netcat

121 vi /etc/ssh/sshd\_config

122 vi /opt/sonarqube/conf/sonar.properties

123 sudo netstat -plnt | grep ':9090'

124 netstat -a | grep ':9090'

125 vi /opt/sonarqube/conf/sonar.properties

126 netstat -a | grep ':9090'

127 netstat -a | grep 'sonar'

128 systemctl restart sonar

129 netstat -a | grep 'sonar'

130 vi /opt/sonarqube/conf/sonar.properties

131 netstat -a | grep '10.129.210.114'

132 vi /opt/sonarqube/conf/sonar.properties

133 netstat -a | grep 'sonar'

134 ps -aef

135 ps -aef | grep 'sonar'

136 systemctl status sonar

137 vi /opt/sonarqube/conf/sonar.properties

138 curl localhost:9090

139 curl localhost:9000

140 vi /opt/sonarqube/conf/sonar.properties

141 systemctl restart sonar

142 systemctl status sonar

143 curl localhost:8085

144 vi /opt/sonarqube/conf/sonar.properties

145 systemctl restart sonar

146 systemctl status sonar

147 curl localhost:8085