**PIPELINE**

---------------------

Implementing CI-CD from the level of code.

This code is created using groovy script, and this file is also called as jenkins file.

**Advantages**

-----------------

As pipeline is implemented as code, it gives the developers the ability to upload into version controlling system from where they can edit and review the script.

Pipelines can accept interactive human input before continuing with specific stage in CI-CD

Ex: Before deployment into production environment, pipeline script can accept approval

from the delivery head and then continue.

Pipeline script support complex realtime scenario where we can implement conditional statements, loops etc.

Ex: If testing passes, we want to go to delivery.

If it fails, we want to send automated emails.

**Scripted pipeline syntax:**

-------------------------------------------------------------------------

node ('built-in/slave')

{

stage(' Stage in CI-CD')

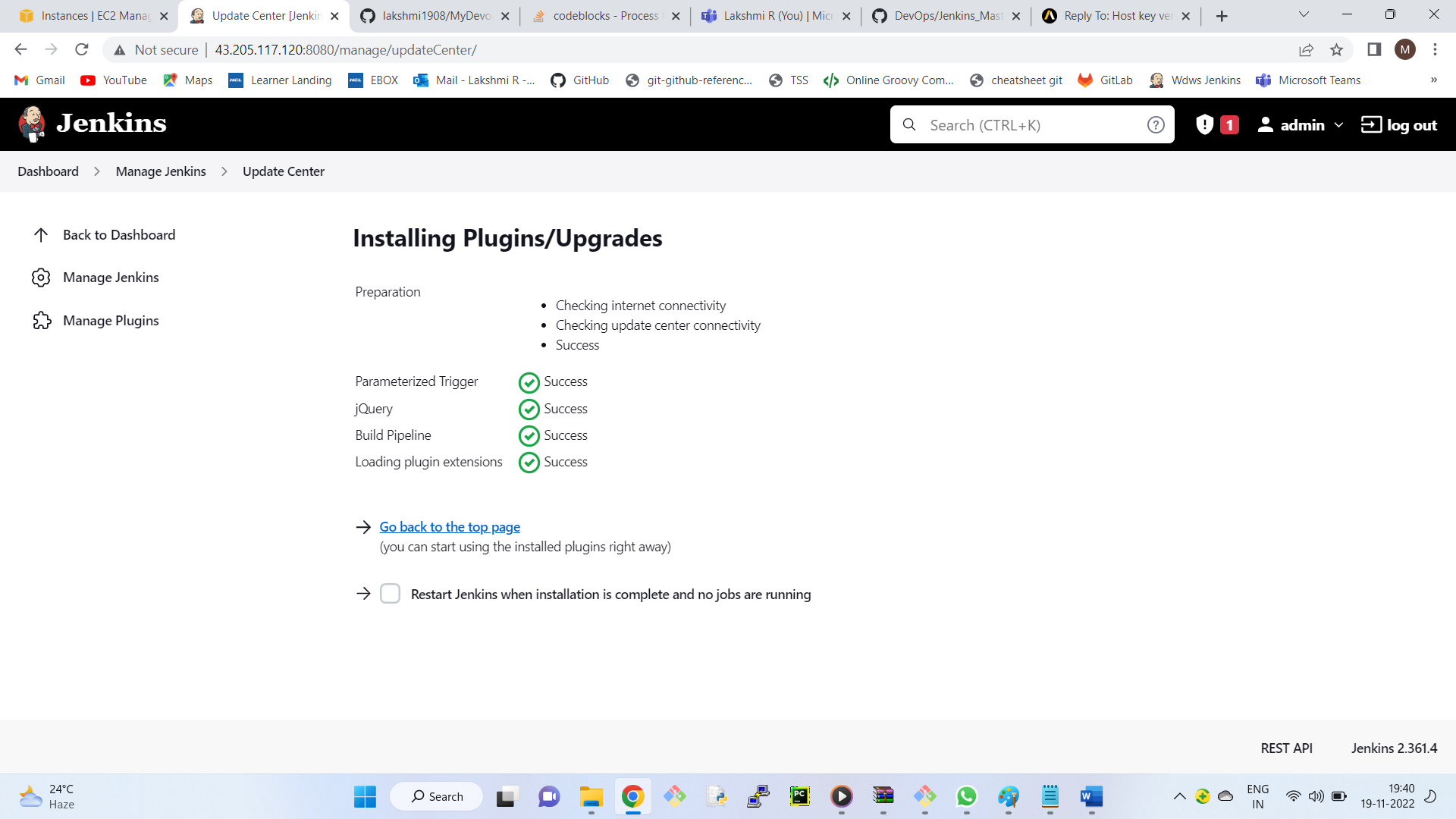
{

Groovy code for implementing the stage

}

}

Install Build pipeline plugin



Ex:

Create new item --- ScriptedPipeline

select pipeline --OK

Pipeline tab,

pipeline syntax

Sample step - node: Allocate node

label – built-in

Generate piplescript -- copy the groovy code and paste in pipeline tab.

In pipeline syntax

Sample step - stage:Stage

Stage name - **Continuous Download**

Generate piplescript -- copy the groovy code and paste in pipeline tab.

-----

In pipeline syntax

Sample step - git:Git

Repository URL - <https://github.com/lakshmi1908/mavendeploy.git>

Generate piplescript -- copy the groovy code and paste in pipeline tab.

-------------

Apply --- Save --> Run the job

**2nd stage: Continous Build**

-----------------

We need to run 'mvn package' command.

This command can be executed as a shell script

In pipeline syntax:

Sample step - sh: Shell Script

Stage name - mvn package

Generate piplescript -- copy the groovy code and paste in pipeline tab.

Save and run.

++++++++++++++++++++++++++++

**Step 3: Deployment**

We need to establish password less SSH connection between Dev server and QA Server

Connect to QA server using gitbash

Set the password for ubuntu

$ sudo passwd ubuntu

Edit sshd\_config ( Password authentication -- yes)

$ cd /etc/ssh

$ sudo vim sshd\_config

Go to insert mode

change password authentication to yes

13) Save and quit :wq

14) Restart the service

$ sudo service ssh restart

15) Connect to dev server using gitbash and generate ssh keys

$ ssh-keygen

Overwrite ? n

18) copy the keys to QA server

ssh-copy-id ubuntu@private\_ip\_qa\_server

ssh-copy-id ubuntu@172.31.47.36

Test are you able to connect to qa?

$ ssh ubuntu@172.31.47.36

$ exit ( To come back to dev server)

Now, you can copy the files from dev server to QA server

Create a file in dev server

$ cat > file1

fdsfgfdsgfdsgd

Ctrl +d

$

To copy the file in QA server

Syntax:

$ scp source destination

$ scp file1 [ubuntu@172.31.47.36:/tmp/file2](mailto:ubuntu@172.31.47.36:/tmp/file2)

File1 is copied as file2

Connect qa server and check

file1 will be copied into qa server with the name file2

Lets check for the file, by connecting to qa server

$ ssh ubuntu@172.31.47.36

$ cd /tmp

$ ls

$ cat file2

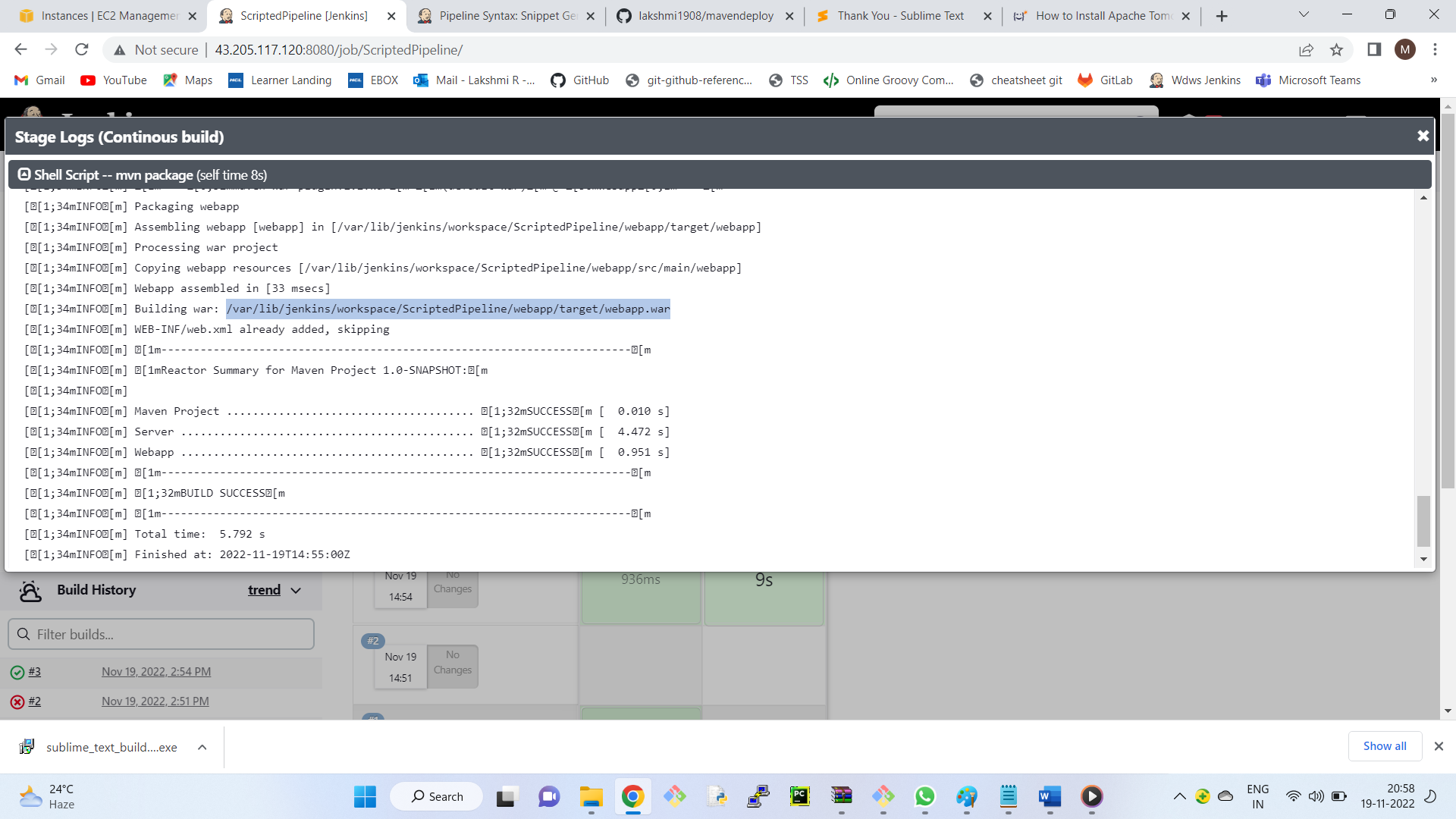
$ exit

**Stage-3: Deployment**

Deployment is nothing but, copying the war file from dev server to qa server

Get the location of war file from log

We will get the location from log message of stage 2.



$ scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war

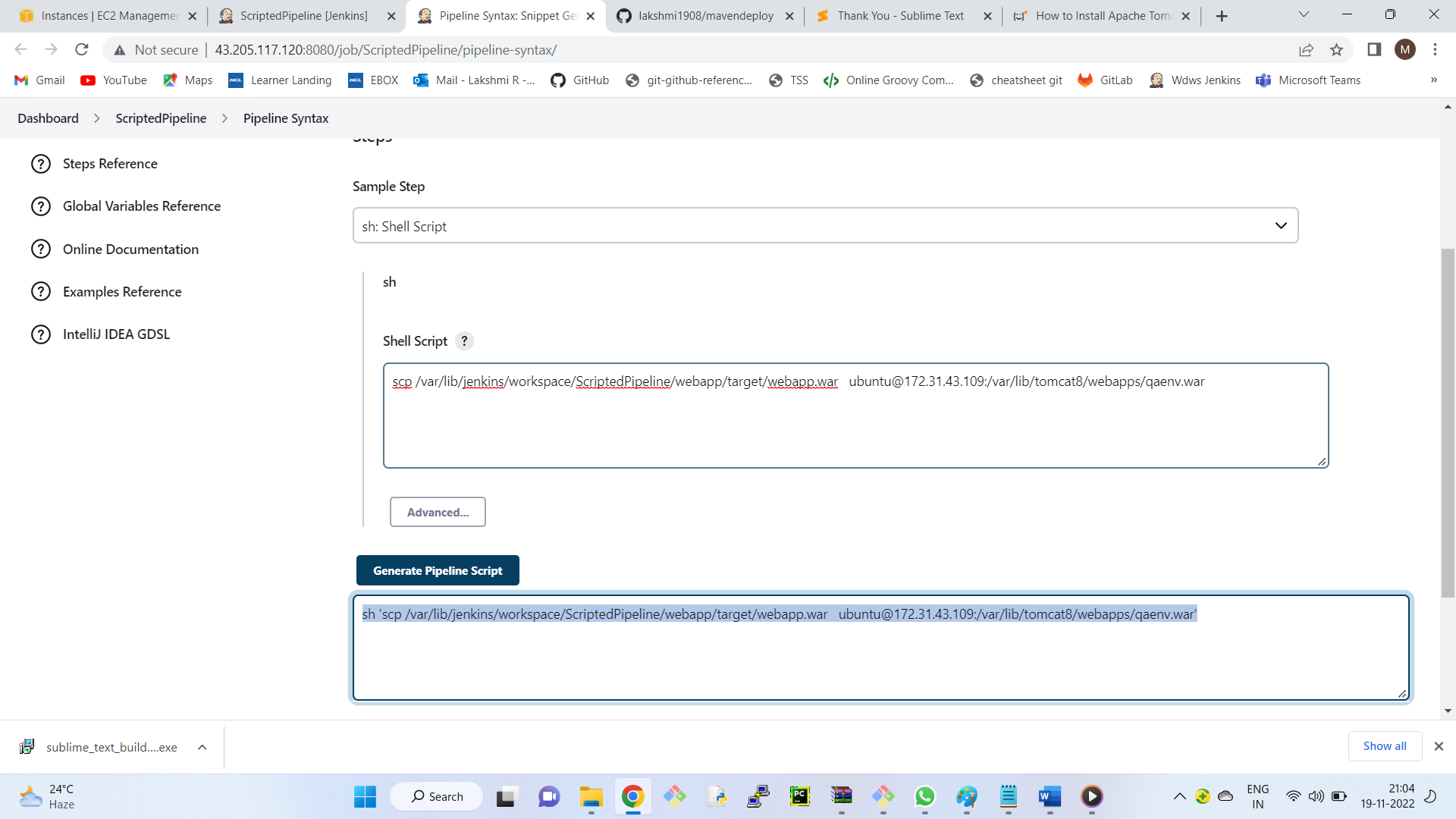
ubuntu@pvt\_ip\_of\_qa\_server:/var/lib/tomcat9/webapps/qaenv.war

Get the groovy code of scp command

Sample Step - sh: Shell Script

Shell script -- copy the scp command which we have created

Generate the code and paste in pipeline script



Apply --- save -- run

Deployment fails

Observe the log file (permissions denied)

To give the permissions

Connect to qa server using git bash

$ cd /var/lib

$ ls -ld tomcat9

(Observation: tomcat9 directory -- others is not having write permissions)

$ sudo chmod -R o+w tomcat9/

Now run the job

+++++++++++++++++++

Connect qa server and check

+++++++++++++++++++++++++++++++++

**4th Stage: Continuous testing**

In pipeline -- add a new stage

Shell script -- echo "Tesing Passed"

Generate the groovy code and copy paste

Apply -- save-- run

+++++++++++++++++++++++++++++++++++

**5th Stage: continuous delivery**

In pipeline -- add a new stage

Copy the code in the – continuous deployment and change the qa\_ipaddress to prod\_Ip\_address

Also change the context path - prodenv

(We need to establish password less ssh between devserver and prodserver)

(we should change tomcat9 permissions)

Connect to prod server using gitbash

Set the password for ubuntu

$ sudo passwd ubuntu

Edit sshd\_config (Password authentication -- yes)

$ cd /etc/ssh

$ sudo vim sshd\_config

Go to insert mode

)change password authentication to yes

13) Save and quit:wq

14) Restart the service

$ sudo service ssh restart

15) Connect to dev server using gitbash and generate ssh keys

$ ssh-keygen Overwrite ? n

18) copy the keys to Prod server

ssh-copy-id ubuntu@private\_ip\_prod\_server

ssh-copy-id [ubuntu@172.31.40.134](mailto:ubuntu@172.31.40.134)

Test are you able to connect to prod?

$ ssh [ubuntu@172.31.40.134](mailto:ubuntu@172.31.40.134)

$ exit (To come back to dev server)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

To give the permissions

Connect to prod server using git bash

$ cd /var/lib

$ ls -ld tomcat9

(Observation: tomcat9 directory -- others is not having write permissions)

$ sudo chmod -R o+w tomcat9/

Now run the job Connect prod server and check http://13.126.45.247:8080/prodenv/ +++++++++++++++++++++++++++++++++++++++++++++

**Pipeline code:**

node('built-in')

{

stage('continous download') {

git 'https://github.com/lakshmi1908/mavendeploy.git'

}

stage('continous build') {

sh 'mvn package'

}

stage('continous deployment') {

sh 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.33.129:/var/lib/tomcat9/webapps/qaenv.war'

}

stage('continous testing') {

sh 'echo "Testing Passed"'

}

stage('continous delivery') {

sh 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.38.195:/var/lib/tomcat9/webapps/prodenv.war'

}

}

Another sample script:

pipeline{

agent any

stages {

stage('build'){

steps{

sh 'echo "Running the build phase"'

}

}

stage('test'){

steps{

echo "Running test phase"

}

}

stage('QA') {

steps{

echo "Running the QA phase"

}

}

stage('Deploy'){

steps{

echo "Running the Deploy phase"

}

}

stage('Monitor'){

steps{

echo "Running the Monitoring phase"

}

}

}

}

<https://github.com/jabedhasan21/java-hello-world-with-maven.git>

<https://github.com/kitconcept/jenkins-pipeline-examples>