**jenkins:**

**========**

This is the tool which we can use to perform continus intigration and continus delevory

**Stages in CI-CD :**

**=================**

**Stage 1 (continus Download) :**

**=======================**

The developers creats some code an upload that code into the version controlling system. whenever developers upload any modified code

into git version controlling system jenkins will get a notificationn and it will immediately download the code. this is called CD

**Stage 2 (Continuos Build ):**

**====================**

The code downloaded in the privious stage has to be convertrd into an artifact. this artifact can come in to the format of war,ear,exe files etc.

This process is called build process and to perform this build process jenkins will talk the help of build plugins like MsBuild , Maven etc.

this stage is called continuos Build.

**Stage 3 (continuos Deployment ):**

**=========================**

The artifact created in the previous stage has to be deployed into the testing servers. The testing servers might be running on some applications servers like tomcat

weblogic etc. jenkins will deploy the artifact into these applications servers so that testers can access it. this stage os called continuos Deployment.

**Stage 4 (continuos testing ):**

**======================**

Once the applications is deployed into the testing environment jenkins will run the automation testing programs created by the testers and check if the application

is working correctly or not. these automation testing programs can be created by testers using tools like selenium, codedui ets. If testing failed jenkins will send

automated emails to the team members.

Developers will fix the defects and upload the modified code into git. jenkins will again trigger all the stages from stage 1.

**Stage 5 (continuos delivory):**

**=======================**

Once testing passes jenkins will take approvals from the delivery team and deploys that artifact into the production servers where the client or end users can start

accessing it. this is called continuos delivory

Note:

The first four stages are called continuos Integration The kast stage is called continuos Delivory.

**Installing jenkins:**

**==============**

1 Connect to the Devserver AWS instance using gitbash.

2 update the apt repository

apt-get update

3 Insatll Java

apt-get insatll -y openjdk-8-jdk

4 Insatllgit and maven

apt-get insatll -y git maven

5 Download jenkins.war

wget http://mirrors.jenkins.io/war-stable/latest/jenkins.war

6 To start jenkins

jav -jar jenkins.war

7 To unlock jenkins

Enter the admin password displayed in output of the step 6

8 click on insatll suggested pluggins

9 Create first admin user and the you click on continue --> click save and run.

**Setting up tomcat on QAServer and Prod Server:**

**======================================**

1 Connect to QaServer AWS instance using gitbash

2 Update the apt repository

apt-get update

3 Insatll tomcat8

apt-get insatll -y tomcat8

4 Insatll tomcat8-admin

apt-get insatll -y tomcat8-admin

5 Set usename and password in tomcat for jenkinsto do deployment

vi /etc/tomcat8/tomcat-users.xml

<user username="admin" password="admin" roles="manager-script"/>

6 Restart the tomcat

service tomcat8 restart

**Stage 1 ( Continuos Download ):**

**=========================**

1 open the dashboard of jenkins ( public\_ip\_Of DevServer:8080)

2 Click on New Item

3 Enter item name as Development

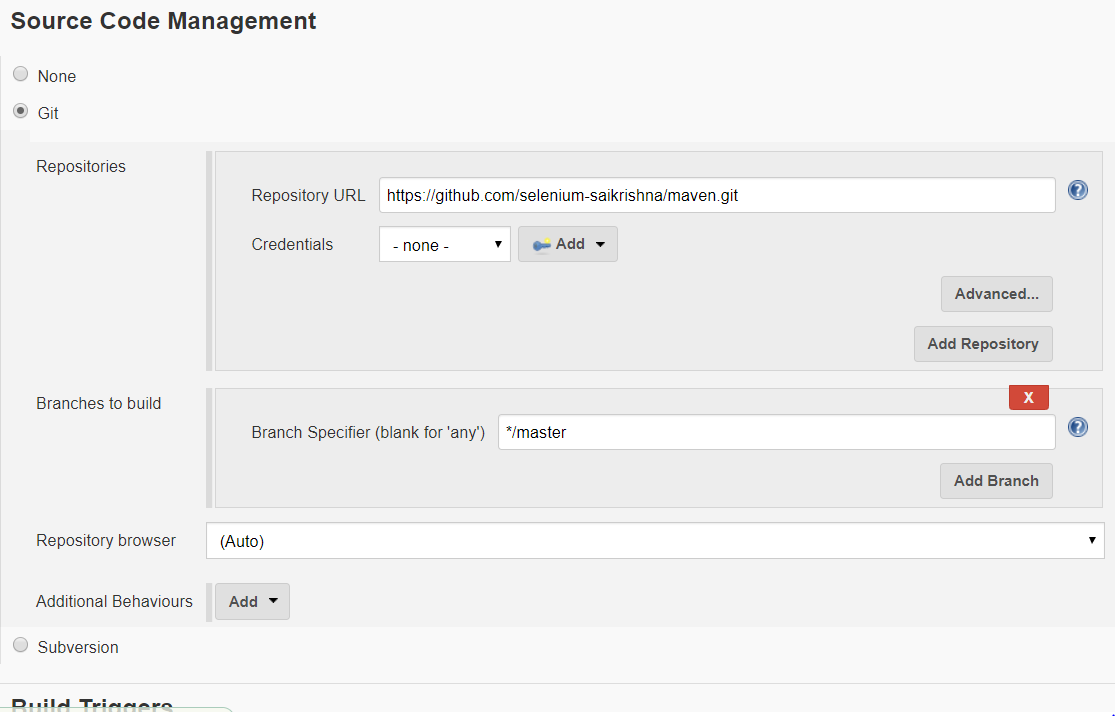
4 Select free style project-->Click on ok

5 go to source code management

6 Select git

7 Enter github repository URL where developers have uploaded the cose

<https://github.com/selenium-saikrishna/maven.git>



8 click on apply--> save

9 Go to the dashboard of jenkins-->go to development job

click on Build icon

the above job will download all the code created by the developers from the remote github repository

**stage 2 (continuos Build):**

**====================**

1 Open the dashboard of jenkins

2 go to the development job --> click on configure

3 Go to build section

4 click on add build step

5 click on invoke top level maven targets

6 enter the gole as "package"



7 click on apply and svae

8 go to dashboard of jenkins

9 go to develpment job --> click on build icon

the above job will be converted the code downloaded in stage 1 into an artifact.

this artifact is generated in the format of a war files

**Installing deploy to container plugin:**

**==============================**

1 open the dashboard of jenkins

2 click on manage jenkins

3 click on manage pluggins --> go in available section

4 search for deploy to container pluggin

5 insatll it

stage 3 (continuous Deployment):

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

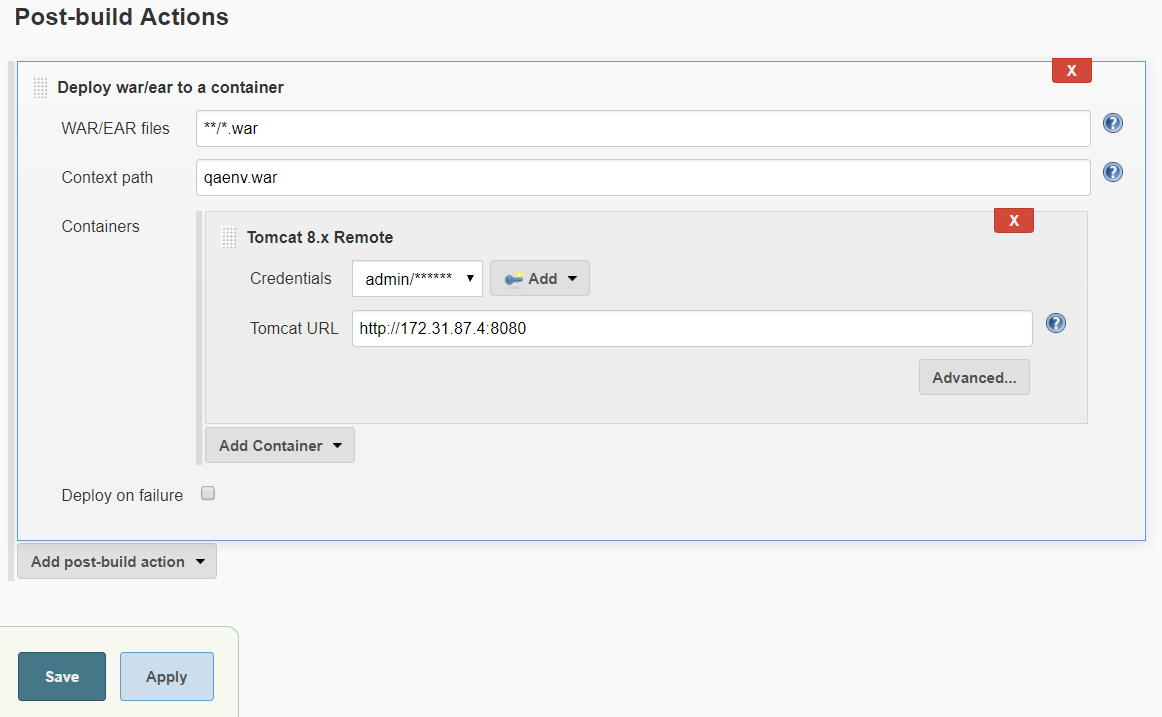
1 open the dashboard op jenkins

2 Go to development job -->click on configure

3 go to post build action

4 click on add post build action

5 click on deploy war/ear to container



6 Enter the private ip of the server

7 And enter the apply ans save

8 Go to dashboard of Jenkins

9 Go to development job and click on build icon

10 the above job deploy the artefact to the tomcat running on the qa server.

**Step 4 (Continuous Testing)**

**======================**

1 Open the dashboard of jenkins

2 Click on Newitem-->Give item name as "Testing"

3 Click on Free style project-->ok

4 The testing team will provide the url of github

where they have uploaded their automation testing programs

5 Go to source code management-->Click on git

6 Enter the github url of the testing project

https://github.com/selenium-saikrishna/testing.git

7 Click on apply-->save

8 once the testing code is downloaded into jenkins workspace

in the target we will find a jar file called testing.jar

this jar file can execute all the testing programs on the

application deployed into QA env

9 Go to dashboard of jenkins

10 go to the "Testing" job that we created-->click on downward

arrow-->configure

11 Go to build section-->click on "add build step"

12 click on execute windows batch command

13 java -jar target/testing.jar

14 apply-->Save

**Note:To link the development job with the testing job:**

**===========================================**

1 Go to the dashboard of jenkins

2 Go to development job

3 Click on downwar arrow-->configure

4 Go post build actions-->Click on build other project-->enter project name as testing

5 click on apply-->save

**Step 5 (Continuous Delivary)**

**=======================**

We have created2 jobs in jenkins(development,testing)

development has has done the below three activities

1 Downloading the development code from git

2 Creating a war file with the help of maven

3 deploying that war file into the tomcat server

on qaenvironment

The war file created in step no2 by development job

should be passed to testing job.

This can be done using a plugin called "copy artifact plugin"

1 Open the dashboard of jenkins

2 click on manage jenkins-->manage plugins

3 click on available section--search for copy artifact plugin and install it

4 Go to the dashboard of jenkins-->go to the development job-->Go to its configuration page

go to post build actions-->click on add post build action-->click on archive the artifacts

Files to archive----\*\*/\*.war

5 apply-->save

6 Go to the dashboard of jenkins-->Go to testing job

7 click on configure-->Go to build section

8 click on add build step-->copy artifacts from another project

9 enter the project name as Development

10 Apply

11 The war file which has been copied from the

development job into testing job should be deployed into tomcat server in prodenv

12 click on post build actions

13 click on add post build actions

14 deploy war/ear to container

15 apply-->save

16 http://<Private\_Ip>:8080/<env>

**User Administration :**

**=================**

1 Open the dashboard of jenkins

2 click on manage jenkins

3 click on manageusers

4 click on create users

5 enter user credentials

**Creating roles and assgning:**

**======================**

1 open the dashboard of jenkins

2 click on manage jenkins

3 click on manage pluggin

4 click on role based authorization strategy pluggin

5 install it

6 go to dashboard--> manage jenkins

7 click on configure global security

8 click enable security checkbox

9 go to authorization section on role based strategyradio button

10 apply--> save

11 go to dashboard of jenkins

12 click on manage jenkins

13 click on manage and assign roles

14 click on manage roles

15 go to global roles and create a role "employee"

16 for this employee in overall give read access

and in view section give all access

17 go to project roles--> give the role as developers

and patter as dev.\*(ie developer role can access only those jbos whose name start with Dev)

18 Similarly create anothe role as Tester and assign the pattern as "Test.\*"

19 give all permissions to developer and testers

20 apply-->save

21 click on assign role

22 go to global roles and ass user1 and user2

23 ckeck user1 and user2 as employee

24 go to item roles

25 add user1 and user2

26 check user1 as Developer and user2 as testers

27 apply-->save

If we login to jenkins as user1 we can access only the Development related job

and user2 can access only the Testing realted job.

**Scheduling the jenkins job for a particuler time:**

**=====================================**

1 open the dashboard of the jenkins

2 go to the job that we want to schedule click on configure

3 click on build triggers

3 click on build periodically

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW

MINUTE Minutes within the hour (0–59)

HOUR The hour of the day (0–23)

DOM The day of the month (1–31)

MONTH The month (1–12)

DOW The day of the week (0–7) where 0 and 7 are Sunday.

To specify multiple values for one field, the following operators are available. In the order of precedence,

\* specifies all valid values

M-N specifies a range of values

M-N/X or \*/X steps by intervals of X through the specified range or whole valid range

A,B,...,Z enumerates multiple values

4 enter the particuler build schedule

5 click on apply and save

**Exicuting the jenkins job whenever developer makes any modification to the code:**

**================================================================**

1 Open the dashboard of the jenkins

2 Go to respective job click on configure

3 click on build triggers

4 click on poll SCM

5 enter the vallues

\* \* \* \* \*

6 click on apply and save.

**Notifications:**

**===========**

1 Whenever the jenkins job executed we can send notification to the team members these notification can send through

catilight

email notification

**Catilight:**

**=======**

Catilight is a third parth s/w wigh can be intigated with jenkins and it will send notification

in the format of popup masseges

catilight is a client side s/w and it has to be insatlled and all the teamm members machines

1 Open https://catlight.io/downloads

2 Download catlight for windows and install it

3 Open catlight-->It will dispaly the list of CI tools that it can connect with

4 select jenkins

5 enter the jenkind public\_ip\_Of The\_AWS insatlce\_jenkins:8080

enter the user name and password

6 select the jobs for which we want notification

7 open jenkins and run the job

A small popup massege come on the screen of the team members where the catlight is insatlled which show the status of the jenkins job.

**Email Notifications :**

**================**

1 Open the dashboard of the jenkins

2 Go to job for which we want to send email notification

3 Click on configure

4 Go to post build actions-->Click on add post build actions

5 Click on email notification

6 Enter the email id's of the team members seperated by a space

7 Click on apply and save

8 Go to the dashboard of the jenkins

9 click on manage jenkins-->click on configure system

10 search of email notifications

11 SMTP server --> give the smtp.gmail.com

12 click on advanced

13 check SMTP authentication

14 enter the email id and password

15 check use SSL

16 Enter the SMTP port number as 465

17 check test configuration by sending test email enter gmail id

18 click on test configuration cheeck for massege email was successfully sent

**master slave setup of jenkins:**

**=======================**

When we want run multiple jenkins job on the same server it might downgrade the performance of the jenkins server in such a senario we can use master slave setup

Where we can distribute jenkins work lode on to multiple servers this is also calld as distributed jenkins build

Slave machine can be any linux server with the same version of java installed

the only pre conditions is passwordless ssh should be establised between master and slave.

1 Create a new AWS ubuntu18 insatnce and name ans it slave1

2 To establish passwordless ssh between master and slave

a) Connect to slave with gitbash

b) set password for ubuntu user

sudo passwd ubuntu

enter some new password for ubuntu user

c) sudo vim /etc/ssh/sshd\_config

Go into insert mode pressing i

Search for password Authentication and change it from no to yes

PermitRootLogin yes

save and quit

d) restart ssh

service ssh restart

e) connect to master using git bash

f) generate the ssh key

ssh-keygen

the above command will create 2keys called public key and private key in the .ssh folder

g) ssh-copy-id ubuntu@private\_ip\_of\_slave

This command will copy the content of the public key into the slave machine into a file called

"authorised\_key". once these keys are copied the master machine can do ssh to the slave without password

3 Install the same version of java that is present on master

COnnect to slave using gitbash

sudo apt-get update

sudo apt-get insatll -y openjedk-8-jdk

4 Download the slave.jar from master into slave

wget http://private\_ip\_Of\_master:8080/jnlpjars/slave.jar

5 Give execute permissions on the slave.jar

chmod u+x slave.jar

6 create an empty directory that will act as a workspace for jenkins

to the job on the slave

mkdir workspace

7 Open the dashboard of jenkins click on manage jenkins

8 click on manage nodes

9 click on new node enetr some nade name slect permanant agent

10 remote root directory enter as /home/ubuntu/worksapce

11 lable name as "myslave" (this lable should assosiated with the job in jenkins and then the job will run on the corresponding slave)

12 go to launch method select as "launch agent via execution of command on the master "

private\_ip\_of\_te\_server java -jar slave.jar

13 click on the save

14 go to dash boardboard of jenkins that should on slave

15 click on configure

16 go into general section

17 click on Restrict where this project can be run

and enter the slave lable (myslave)

18 click on apply ans click on save.

**Build Pipeline:**

**===========**

This is a pluggin which is user for getting better GUI for handling jenking job. this shows the link of upstream and downstream jobs in one single window

form where we can control all the activities.

1 open the dashboard of jenkins

2 click on manage jenkins

3 click on manage pluggins

4 go into available section search for buildpipeline pluggin (pipeline) install it

5 go back to the dashboard of the jenkins click on new view icon enter some view name slect build pipeline view click on ok

6 go to upstream downstream config section and seclet the inticial job

7 click on apply and save.

**Pipeline as code:**

**=============**

This is the feature of jenkins where we perform all the activities of jenkins CICD form the lelval of the script.

This script is created using groove code and it is calls as jenkins file.

this jenkins file jenerally created by developers and uploaded into the version controling system (Git).

Form the version controling system itself and it will perform CICD.

**Advantages :**

**==========**

1 SInce the pipeline code is uploaded into version controling system. it gives the team members ability to review the script

and also maintine multiple versions paralally.

2 pipeline scripts can servive both planed and unplaned restarts of jenkins master

3 pipeline scripts can stopped and it can be accept untractive human input before proceding to the next step that is we can make the pipeline script to take approvals

From the delevary team before deploying into prod enviroment.

**Pipeline script support :**

**==================**

Complex real world continuos intigration senarious where we can exicute steps based on a conditions that is if a spcificat step passes

we want to perform a set of activities and if it failes we can perform another set of activies.

Pipeline scripts can perform CICD using minimum number of pluggins duto thi reason the performance of jenkins will be much faster.

pipeline line can be implemented in two ways

1 scripted pipeline

2 declarative pipeline .

**scripted pipeline:**

**----------------------**

Sample syntax:

node('maste/slave')

{

stage('stage name in CI-CD')

{

Groovy code for implementing this stage

}

}

**Declarative pipeline:**

**--------------------------**

Sample syntax:

pipeline

{

agent any

stages

{

stage ('stage name in CI-CD')

{

steps

{

Groovy code for implementing this stage

}

}

}

}

**Scripted pipeline :**

**=============**

A )

Open the dashboard of jenkins

1 click on new item

2 enter some item name click on pipeline click on okay.

3 go to pipeline section and create the scripted pipeline code

node('master')

{

stage('ContinuousDownload')

{

git 'https://github.com/selenium-saikrishna/maven.git'

}

stage('ContinuousBuild')

{

sh label: '', script: 'mvn package'

}

stage('ContinuousDeployment')

{

sh label: '', script: 'scp /root/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war root@172.31.87.4:/var/lib/tomcat8/webapps/qaenv.war'

}

stage ('ContinuousTesting')

{

git 'https://github.com/selenium-saikrishna/FunctionalTesting.git'

sh label: '', script: 'java -jar /root/.jenkins/workspace/ScriptedPipeline/testing.jar'

}

stage('ContinuousDelivory')

{

input message: 'waiting for approvals from DM', submitter: 'Ramu'

sh label: '', script: 'scp /root/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war root@172.31.83.120:/var/lib/tomcat8/webapps/prodenv.war'

}

}

B)

the jenkins file is some time created by developers as part of application code and it is uploaded into git hub repository.

This jenkins file from git hub repository will execute will effect build execute all the stages of CI-CD

1 open the dashboard of the jenkins

2 click on new item enter some item name select pipeline job click on ok

3 Go pipeline section

4 go to deffination black select pipeline scripts from SCM

5 in the SCM select git

6 enter theremote git hub URL where developer has uploaded code

7 go into build trigger section click on poll SCM give as \* \* \* \* \*

8 click on apply and save.

**Developer Activity:**

**===============**

1 Clone the maven repository

git clone https://github.com/selenium-saikrishna/maven.git

2 Move into this maven project

cd maven

3 Delete the .git folder

rm -rf .git

4 Initilise a new git repository

git init

5 send all files into stagging and local repository

fit add .

git commit -m "First commit"

6 open github.com--> Signin into your account--> create a new repository

7 push the local repository into the remote server

8 create a file called jenkinsfile on the local machine

vim jenkinsfile

Go into insert mode by pressing i

node('master')

{

stage('ContinuousDownload')

{

git 'https://github.com/selenium-saikrishna/maven.git'

}

stage('ContinuousBuild')

{

sh label: '', script: 'mvn package'

}

stage('ContinuousDeployment')

{

sh label: '', script: 'scp /root/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war root@172.31.87.4:/var/lib/tomcat8/webapps/qaenv.war'

}

stage ('ContinuousTesting')

{

git 'https://github.com/selenium-saikrishna/FunctionalTesting.git'

sh label: '', script: 'java -jar /root/.jenkins/workspace/ScriptedPipeline/testing.jar'

}

stage('ContinuousDelivory')

{

input message: 'waiting for approvals from DM', submitter: 'Ramu'

sh label: '', script: 'scp /root/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war root@172.31.83.120:/var/lib/tomcat8/webapps/prodenv.war'

}

}

9 push this into github

git add .

git commit -m "second commit"

git push

**Declarative Pipeline :**

**================**

pipeline

{

agent any

stages

{

stage('ContinuousDownload')

{

steps

{

git 'https://github.com/selenium-saikrishna/maven.git'

}

}

stage('ContinuousBuild')

{

steps

{

sh label: '', script: 'mvn package'

}

}

stage('ContinuousDeployment')

{

steps

{

sh label: '', script: 'scp /root/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war root@172.31.87.4:/var/lib/tomcat8/webapps/qaenv.war'

}

}

stage('ContinuousTesting')

{

steps

{

git 'https://github.com/selenium-saikrishna/FunctionalTesting.git'

sh label: '', script: 'java -jar /root/.jenkins/workspace/DeclarativePipeline/testing.jar'

}

}

}

post

{

success

{

sh label: '', script: 'scp /root/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war root@172.31.83.120:/var/lib/tomcat8/webapps/prodenv.war'

}

failure

{

mail bcc: '', body: 'jenkins job has failed check logs', cc: '', from: '', replyTo: '', subject: 'jenkins job failed', to: 'ravi.ilavarapu92@gmail.com'

}

}

}

**Execption handling :**

**===============**

this is the process of overcomeing the potentional error and reuming the execution of the program

genarally when ever the stage jenkins failed the program execution stopped over there and it comes into the post sections

If you dont want the program executions to stop on encontering an error

If you dont want the program exections to stop on we can use execptions handling

**execptions handling is implemented using try catch:**

**=========================================**

The pice of code which might genarate an error is given try section if it fails the control comes to the catch section and the program execution continue.

pipeline

{

agent any

stages

{

stage('contiDownload')

{

steps

{

script

{

try

{

git 'https://github.com/selenium-saikrishna/maven.git'

}

catch(Exception e)

{

mail bcc: '', body: 'notification sent to user ', cc: '', from: '', replyTo: '', subject: 'Download failed', to: 'ravi.ilavarapu92@gmail.com'

exit(1)

}

}

}

}

stage('contiBuild')

{

steps

{

script

{

try

{

sh label: '', script: 'mvn package'

}

catch(Exception e)

{

mail bcc: '', body: 'notification sent to user ', cc: '', from: '', replyTo: '', subject: 'Download failed', to: 'ravi.ilavarapu92@gmail.com'

exit(1)

}

}

}

}

stage('contiDeployment')

{

steps

{

script

{

try

{

sh label: '', script: 'scp /root/.jenkins/workspace/DPipeline/webapp/target/webapp.war root@172.31.87.4:/var/lib/tomcat8/webapps/qaenv.war'

}

catch(Exception e)

{

mail bcc: '', body: 'notification sent to user ', cc: '', from: '', replyTo: '', subject: 'Download failed', to: 'ravi.ilavarapu92@gmail.com'

exit(1)

}

}

}

}

stage('contiTesting')

{

steps

{

script

{

try

{

git 'https://github.com/selenium-saikrishna/FunctionalTesting.git'

sh label: '', script: 'java -jar /root/.jenkins/workspace/DPipeline/testing.jar'

}

catch(Exception e)

{

mail bcc: '', body: 'notification sent to user ', cc: '', from: '', replyTo: '', subject: 'Download failed', to: 'ravi.ilavarapu92@gmail.com'

exit(1)

}

}

}

}

stage('contiTesting')

{

steps

{

script

{

try

{

sh label: '', script: 'scp /root/.jenkins/workspace/DPipeline/webapp/target/webapp.war root@172.31.83.120:/var/lib/tomcat8/webapps/prodenv.war'

}

catch(Exception e)

{

mail bcc: '', body: 'notification sent to user ', cc: '', from: '', replyTo: '', subject: 'Download failed', to: 'ravi.ilavarapu92@gmail.com'

exit(1)

}

}

}

}

}

}