Jenkins is a self contained, open source automation server which can be used to automate all tasks related to building , testing and delivery activities.Jenkins can be installed even on standalone be any machine with a java runtime envirowment (JRE) Installed.

Jenkins is a tool for Implmenting CI-CD (Continuous Integration - Continuous Delivery)

Stages in CI-CD

Stage 1 : Continuous Download -->Download the source code from the Github/Repository

Stage 2: Continuous Build

Stage 3: Continuous Deployment

Stage 4: Continuous Testing

Stage 5: Continuous Delivery

1-4 ----- Continuous Integration

5 ---- Continuous Delivery

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

Create Instance in AWS

1) Create the AWS Account 2) Login with your aws account 3) Click on Services

4) Click on EC2 5) Click on Instance 6) Click on launch instance

7) Select Ubuntu Server 18 (Free For Eligble) 8) Select t2.micro (Free For Eligble)

9) Click on Next: Configure Instance Details 10) Enter 3 in Number of Instance

11) Click on Add storage 12) Click on Next : Add Tags 13) Click on Next : Configure Security Group

14) Click on Add Rule 15) Select Type as All Traffic 16) Select Source as Anywhere

17) Click on Review and launch 18) If you are doing first time then you need to select Create a new key pair

19) Enter any name in key pair name 20) Click on download key pair (This key pair helps us to connect us to our data center.)

21) Click on launch instance 22) Give all 3 instance proper name ( Dev Server, QA Server, Prod Server )

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

How to Connect with the AWS Instance

1) Select that Instance 2) Press Connect

3) You will get the SSH Command

4) Copy the SSH Command

5) Go to the folder where you have place your key then Open GITBASH in local machine

Git Bash you will get automatically when you have install GIT in your local machine.

6) Check your current location using pwd command

7) Now Paste the SSH Command in GITBASH

8) Just Type Yes

9) Now you are connect to the AWS Instance

Now we run any command that command run in the AWS Intance

Important Point : If you are not doing practice in AWS Stop all the instance.

Install Jenkins in AWS Instance

To install Jenkins the first thing we need java file so first we need to install java like we have done in the local instance.

We need to download Java 1.8 or more.

(For Ubuntu)

1) Update the apt repository

sudo apt update

2) sudo apt install openjdk-8-jdk -y

3) Check the Java Version

java -version

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

On DEV m/c we need maven git and should be downloade before jenkins.

Stages in CI-CD

Stage 1 : Continuous Download -->Download the source code from the Github/Repository

Stage 2: Continuous Build -->To build the code by using maven

Stage 3: Continuous Deployment

Stage 4: Continuous Testing

Stage 5: Continuous Delivery

1-4 ----- Continuous Integration

5 ---- Continuous Delivery

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

4) Install Maven & Git

sudo apt-get install -y git maven

5) Check the Verion of Git & Maven

For Git : git --version

For Maven : mvn --version

6) Download & install Jenkins

Open Jenkins website (<https://jenkins.io/download/>)

Go to Long Term Support

Select Generic Java Package (.war)

We are selecting generic java package file because jenkins will install on those machine where java is already install. If we have java install in windows machine jenkins will work. Only pre requirement is java needs to be install.

For Windows we just need to click on the file and it will download automatically.

For Linux machine enter command wget and paste the url to download the file.

To get the URL right click on generic java package and click on copy link address.

(wget <http://mirrors.jenkins.io/war-stable/latest/jenkins.war>)

wget <https://get.jenkins.io/war-stable/2.277.2/jenkins.war>

11) Start the Jenkins.war file

java -jar jenkins.war

Every day if we want to run the jenkins we need to run this command.

12) Access Jenkins Home Page

Select DEV Instance & Press Connect.

Copy the Domain Name On 4th point.

Paste the Domain name in the browser and in the end enter :8080 with the default port number.

We can access the jenkins with dev server Public IP.

Copy the public ip of the dev server and paste the ip address in the browser and in the end enter :8080 with the default port number.Public

13) Unclock Jenkins

When we are installing jenkins it will automatically give you the password in the github terminal.

Copy the password and paste the browser.

You will get the password on the step 11

14) Press Install Suggested Plugins

15) Create First admin user

The first user which we create here is the admin user of the jenkins.

Click on save and continue.

Click on save and finish

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

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

Create Sample Job

Newitem-->free style proj -->build -->shell excute -->save -->run job.

Build tab

Click on execute Shell

In Command Box Enter echo " Hello Jenkins"

Click on Console Output

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

29/06 -->to run the job from other m/c.

Install TOMCAT In QA & Production Server

1) Select QA Server and press connect

2) Copy the SSH Command

3) Open GIT Bash & paste the SSH Command

Press Yes

4) Update the apt repository

sudo apt-get update

5) Install tomcat8

sudo apt-get install -y tomcat8

After this we need to install one more package

sudo apt-get install -y tomcat8-admin

6) Check the tomcat is intall or not

Copy the public IP of the QA Server then paste in the browser and in the end enter :8080

qa\_server\_public\_ip:8080

Setting the path of tomcat in jenkins

7) enter linux command in QA Server - cd /etc/tomcat8/

8) enter linux command in QA Server - ls

9) You will find the file tomcat-users.xml

10) Open the file -- sudo vim tomcat-users.xml

11) In the end we need to add one statement

<user username="training" password="sunilsunil" roles="manager-script,manager-status,manager-gui"/>

save and quit

press esc

type :wq

press enter

12) When ever we do any changes done in any service we need to restart the service

sudo service tomcat8 restart

13) After this the same above 12 steps we need to do in the prod server also.

++++++++++++++++++=

Prod Instance

<user username="learning" password="sunilsunil" roles="manager-script,manager-status,manager-gui"/>

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

First Start All the AWS Machines.

Connect Dev Server

Start the Jenkins

java -jar jenkins.war

Stage 1 : Continuous Download START CI-CD

1) Create New item as free style project

2) Click on source code managment

3) Select GIT

4) Enter the URL of github reposiditory

<https://github.com/sunildevops77/maven.git>

5) Click on apply and save

6) Run the Job

7) Check the console output.

8) Connect to the dev server

9) Go to the location where code is downloaded

sudo su -

cd path of the folder

ls

Stage 2 : Continuous Build

Convert the java files in to artifact ( .war file)

10) Click on configure of the same job

11) Go to Build Section

12) Click on add build step

13) Click on Invoke top level maven targets

14) Enter the goal as package

15) click on apply and save

16) Run the Job

17) Click on number & click on console output

18) Copy the path of the war file and check the file in the linux machine

sudo su -

cd path

ls

Stage 3 :Continuous Deployment

Now we need to deploy the war file into the QA Server.

19) For this we need to install "deploy to container" plugin.

Go to Dasboard

Click on manage jenkins

Click on manage plugins

Click on avaiable section

Search for plugin ( deploy to container )

Select that plugin and click on install without restart.

20) Click on post build actions of the development job

21) Click on add post build actions

22) Click on deploy war/ear to container

23) Enter the path of the war file (or)

we can give \*\*/\*.war in war/ear files.

24) Context path: qaenv

25) Containers : select tomcat 8

Credentials : Click on add

select jenkins

enter tomcat user name and password

Click on add

Select credentials.

give the private ip of the QA server.

<http://private_ip:8080>

<http://172.31.32.183:8080>

26) Click on apply and save

27) Run the job

28) To access the home page

public\_ip\_Qa\_server:8080/qaenv

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

**+++++++++++++30/06 jenkins JOBS depend on other job+++++++++++++++++++++++**

First Start All the AWS Machines.

Connect Dev Server

Start the Jenkins

java -jar jenkins.war

Stage 1 : Continuous Download START CI-CD

1) Create New item as free style project

2) Click on source code managment

3) Select GIT

4) Enter the URL of github reposiditory

<https://github.com/sunildevops77/maven.git>

5) Click on apply and save

6) Run the Job

7) Check the console output.

8) Connect to the dev server

9) Go to the location where code is downloaded

sudo su - cd path of the folder ls

Stage 2 : Continuous Build

Convert the java files in to artifact ( .war file)

10) Click on configure of the same job

11) Go to Build Section

12) Click on add build step

13) Click on Invoke top level maven targets

14) Enter the goal as package

15) click on apply and save

16) Run the Job

17) Click on number & click on console output

18) Copy the path of the war file and check the file in the linux machine

sudo su - cd path ls

Stage 3 :Continuous Deployment

Now we need to deploy the war file into the QA Server.

19) For this we need to install "deploy to container" plugin.

* Go to Dasboard
* Click on manage jenkins
* Click on manage plugins
* Click on avaiable section
* Search for plugin ( deploy to container )
* Select that plugin and click on install without restart.

20) Click on post build actions of the development job

21) Click on add post build actions

22) Click on deploy war/ear to container

23) Enter the path of the war file (or) we can give \*\*/\*.war in war/ear files.

24) Context path: qaenv

25) Containers : select tomcat 8

* Credentials : Click on add
* select jenkins
* enter tomcat user name and password
* Click on add
* Select credentials.
* give the private ip of the QA server.
* <http://private_ip:8080>
* <http://172.31.47.36:8080>

26) Click on apply and save

27) Run the job

28) To access the home page

* public\_ip\_Qa\_server:8080/qaenv
* 13.127.177.32:8080/qaenv

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

<https://github.com/sunildevops77/TestingNew.git>

Step 1: Connect to Dev server from git bash

Step 2: Start Jenkins ( java -jar jenkins.war )

Step 3: Create new item ( Name - testing )

Source code management tab, Git

Repository URL - <https://github.com/sunildevops77/TestingNew.git>

Apply -- Save

Step 4: Run the job.

Step 5: Check the path of the files which are downloaded.

/home/ubuntu/.jenkins/workspace/testing

Step 6: Configure the same job ( testing )

Build -- Add build Step -- Execute shell

( Command: java -jar testing.jar )

Command: echo " Testing passed"

Now both are independent job.

To call testing job after development job is completed

Go to first job ( demo ) -- configure

Post build actions -- add post build action -- build other project -

Projects to build - testing ( name of the job)

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

Copying artifacts from development job to testing job

The artifacts (war) created by the development job should be passed to the testing job so that the testing job can deploy that into tomcat in the prod environment.

Install Plugins

1) Go to Jenkins dashboard

2) Go to manage jenkins

3) Click on Manage plugins

4) Search for "Copy Artifact" plugin

5) Install the plugin

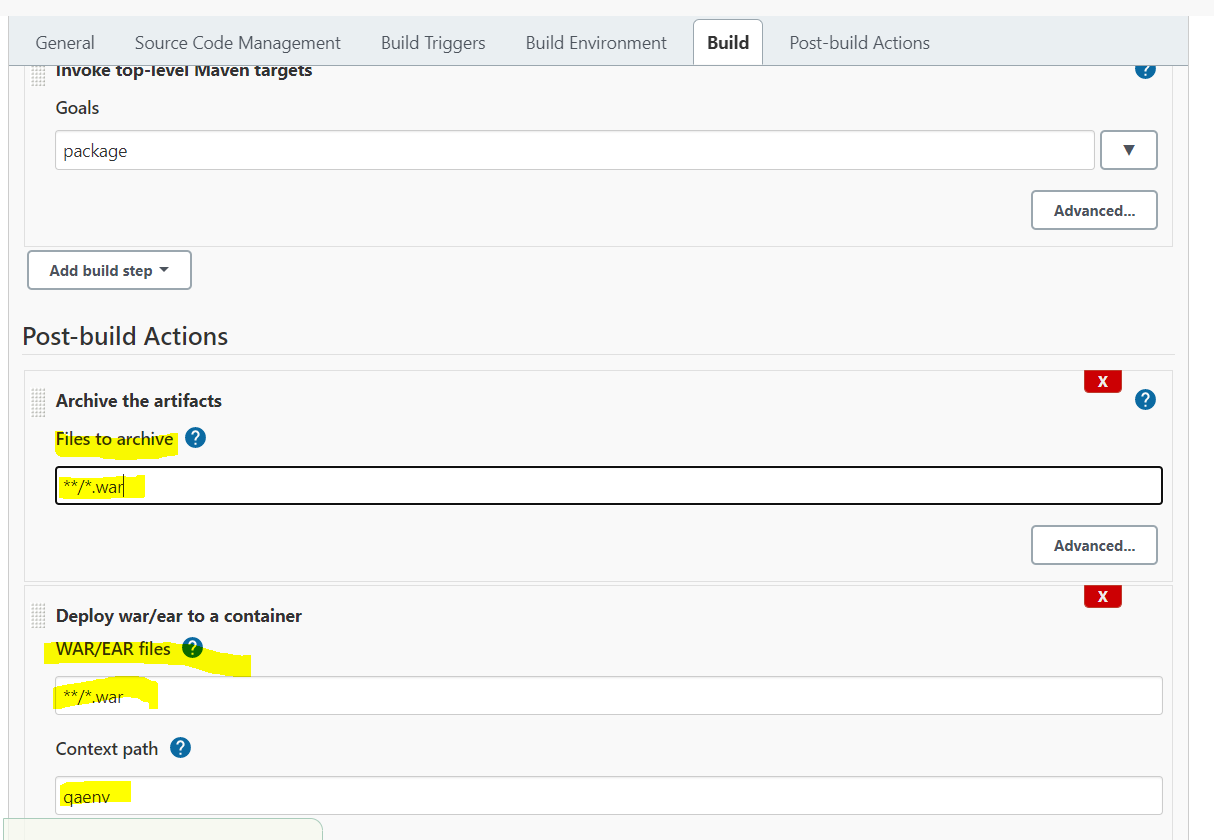
===========================================================

Stage 5 : Continous Delivery

1) Go to Development job 2) Go to Configure

3) Go to Post build actions tab 4) Click on add post build action

5) Click on Archive the artifacts 6) Enter \*\*/\*.war



7) Click on apply and save 8) Go to testing Job

9) Click on configure 10) Go to Build section

11) Click on add build steps 12) Click on copy artifacts from another project

13) Enter Development as project name



14) For Deployment Go to Post build actions section 15) Click on add post build action

16) Click on deploy war/ear to a container 17) Enter \*\*/\*.war in war/ear files

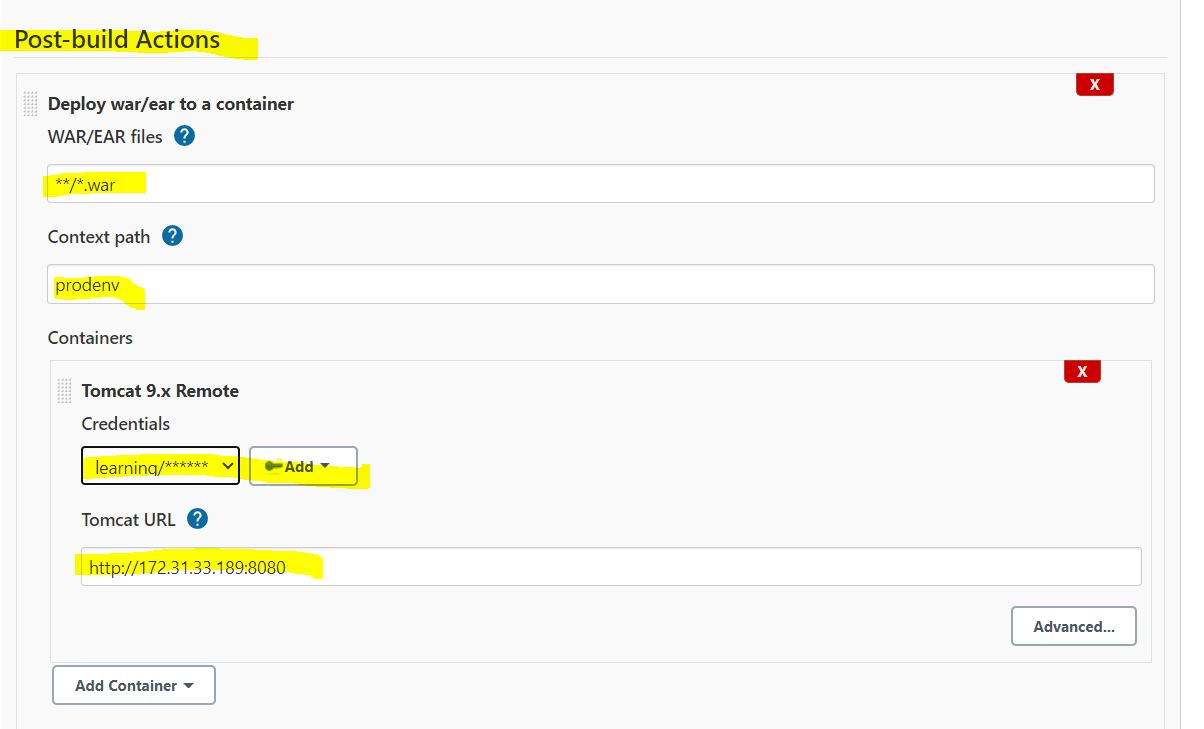
18) Context path : prodenv 19) Click on add container

20) Select tomcat 8 21) Select your Credentials

22) Enter private ip:8080 of the prod server

<http://172.31.39.130:8080>

23) Click on Apply and save



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

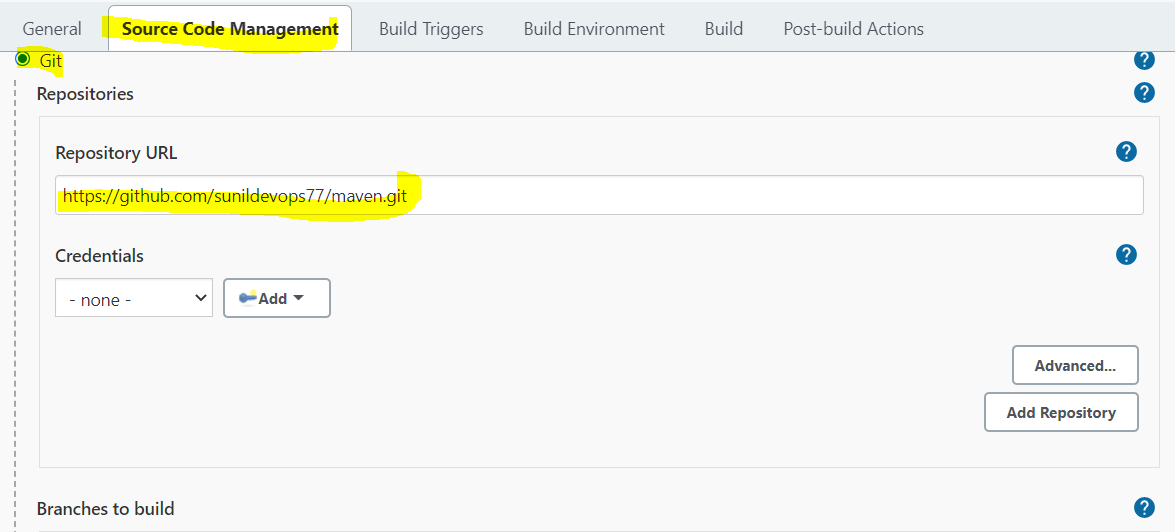
7) enter linux command in Prod Server - cd /etc/tomcat8/8) enter linux command in prod Server- ls

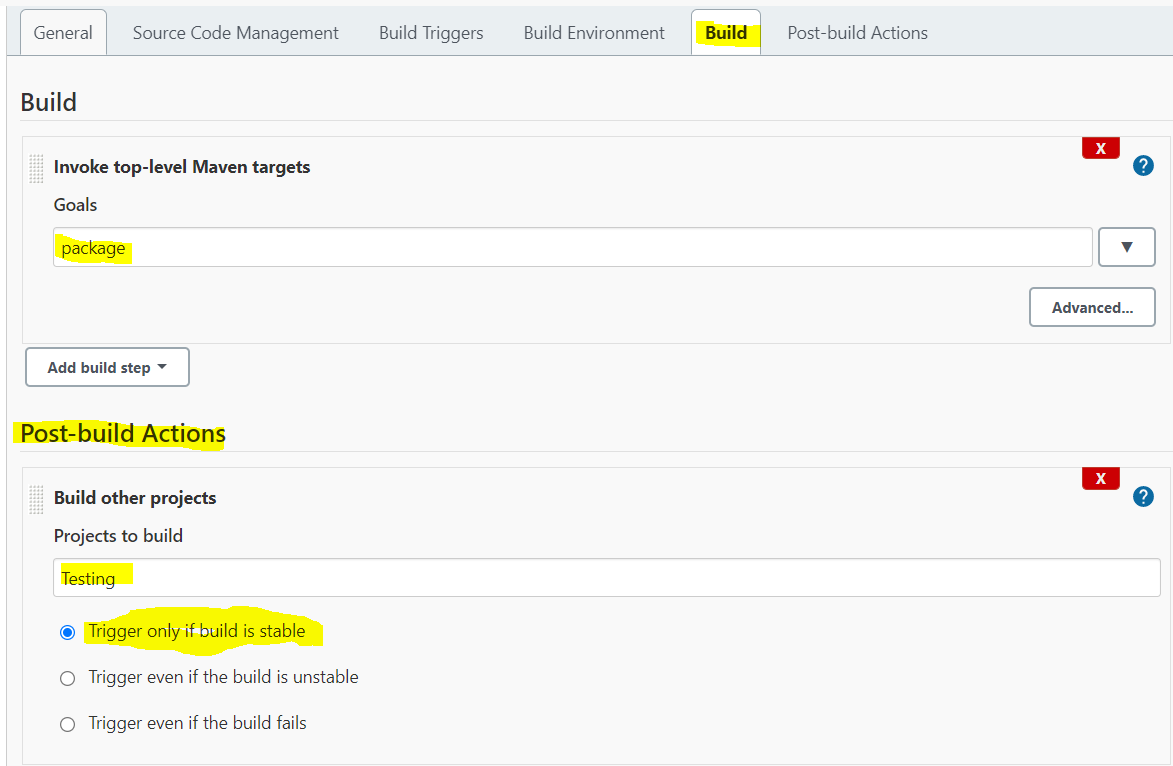
9) You will find the file tomcat-users.xml 10) Open the file -- sudo vim tomcat-users.xml 11) In the end we need to add one statement

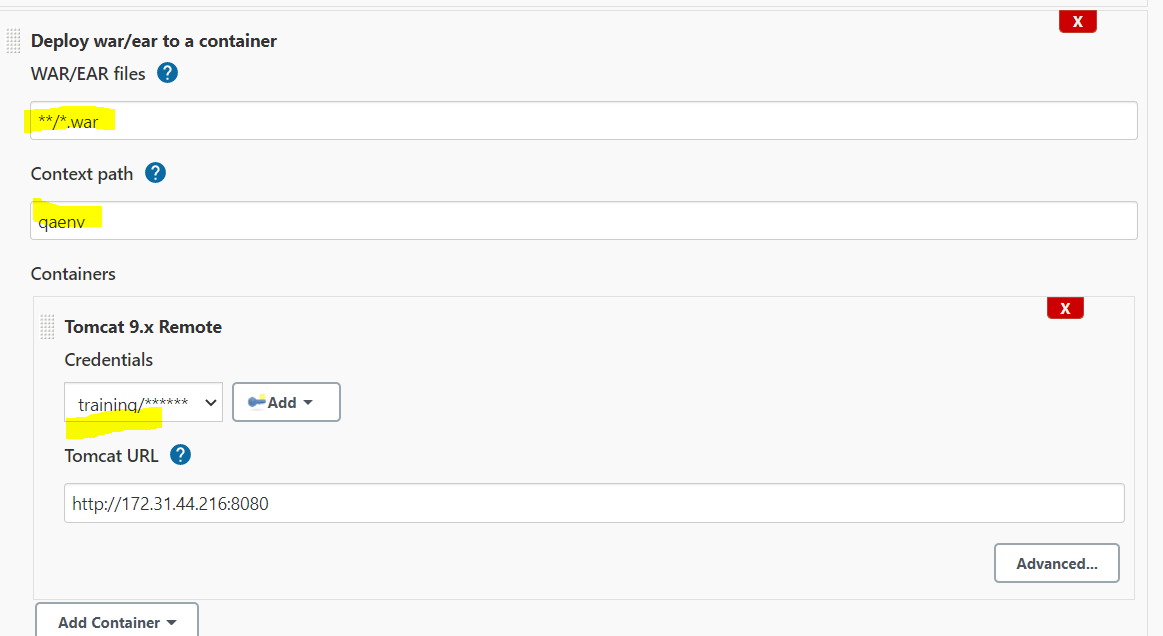
<user username="learning" password="sunilsunil" roles="manager-script,manager-status,manager-gui"/>

12) we need to restart the service sudo service tomcat8 restart

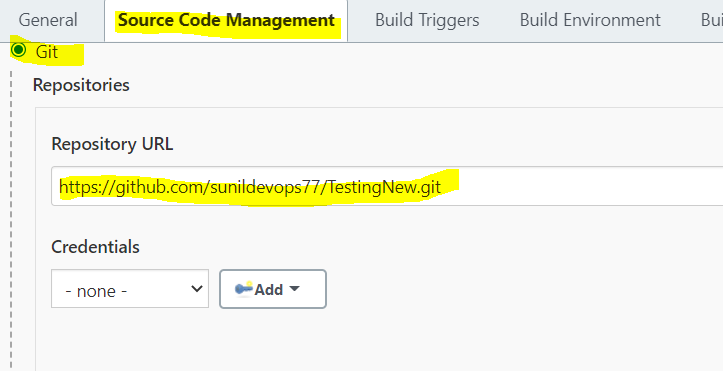
DEV JOB ::



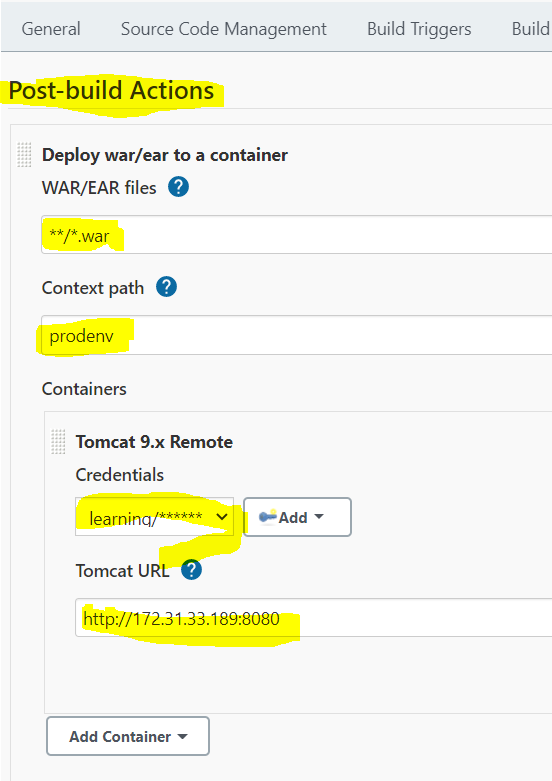




TESTING JOB to prod :







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

++++++++++++02/07---->HOW to create NEW USERS/Assign roles ++++++++++++++++++++

Creating users in Jenkins

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

1 Open the dashboard of jenkins

2 click on manage jenkins

3 click on manage users

4 clcik on create users

5 enter user credentials

Creating roles and assgning

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

1 Install "role based authorization strategy" plugin

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

6 go to dashboard-->manage jenkins

7 click on configure global security

8 check enable security checkbox

9 go to authorization section-->click on role based strategy radio 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 mange roles

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

16 for this employee in overall give read access in view section give all access

17 go to project roles-->Give the role as developer and patter as Dev.\* (ie developer role can access only those jobs whose name start with Dev)

18 similarly create another role as tester and assign the pattern as "Test.\*"

19 give all permisiinons to developrs and tester

20 apply--save

21 click on assign roles

22 go to global roles and add user1 and user2

23 check user1 nad user2 as employees

24 go to item roles

25 add user1 and user2

26 check user1 as developer and user2 as tester

27 apply-->save

Restart Jenkins

<http://13.233.127.59:8080/restart>

If we login into jenkins as user1 we can access on the development related jobs and user2 can access only the testing related jobs

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

++++++++++++++++++++++++06/07--->MASTER SLAVE++++++++++++++++++++++++

Master - Slave configuration

In Dev server we install Jenkins and as there are too many jobs load will increase bec which master will down and Jenkins will down to overcome this load we used the slave m/c so that load is distributed.

Same version of java should exist.

Master and slave should have password less SSH

Step 1: Create slave machine , connect to slave

1) Update the apt repository

sudo apt-get update

2) sudo apt install openjdk-8-jdk -y

3) Check the Java Version

java -version

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

We need to establish password less connection between Dev server and Slave machine

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

Connect to slave

7) Check you user

$ whoami ( ubuntu )

8) set password for ubuntu user

syntax: sudo passwd <user\_name>

Ex: sudo passwd ubuntu

enter password

$ cd /etc/ssh

$ ls ( we get list of files ) Look for sshd\_config

To edit sshd\_config

$ 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

Lets test the connection

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

15) Connect to the development server ( Master )

16) connect to slave server through dev server

ssh ubuntu@private\_ip\_slave\_machine

$ ssh ubuntu@172.31.1.107

exit ( to come back to master )

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

17) To connect to slave without password

$ ssh-keygen ( In master)

18) copy the keys to slave server

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

ssh-copy-id ubuntu@172.31.1.107

19) now we are able to connect to the slave user without password

$ ssh ubuntu@172.31.1.107

Download slave.jar in slave machine

sudo wget http://172.31.41.7:8080/jnlpJars/slave.jar

Check the file is download or not

$ ls

check the file permissions

$ ls -l

we want rwxrw-r--

3) Give execute permissions of this file

sudo chmod u+x slave.jar

4) Create an empty folder which will work like workspace for jenkins to use on the slave machine

$ mkdir workspace

$ cd workspace

$ pwd ( note the path of the workspace )-- /home/ubuntu/workspace

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

Creating node in Jenkins

Open the dashboard of jenkins

manage jenkins --- manage nodes

7) Click on new node ---- node name - Myslave

- select permanent agent

Remote root directory -/home/ubuntu/workspace

Label - Slave\_lab

10) Go to Launch method

Select Launch agent via execution of command on the controller

11) In Launch command

ssh ubuntu@private\_ip\_of\_slave java -jar slave.jar

ssh ubuntu@172.31.1.107 java -jar slave.jar

13) Click on save

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

Configure job to run on slave

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

14) Select Testing Job

15) Go to Configure --> General Tab

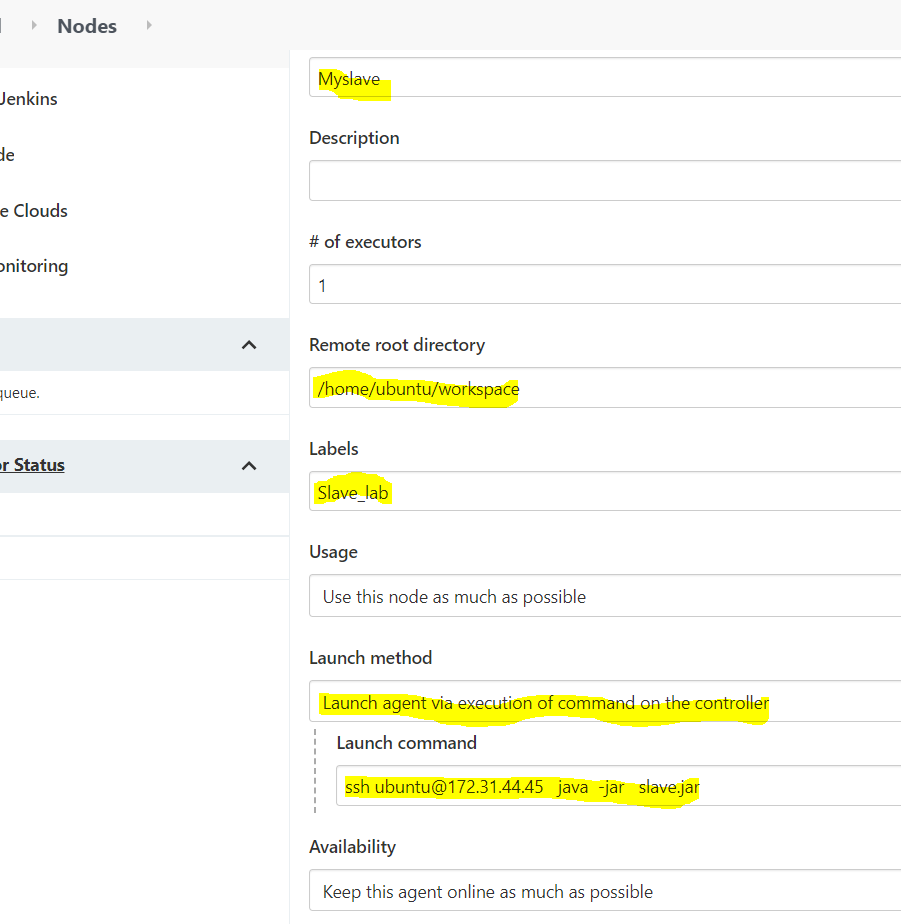
17) Check Restrict where this project can be run

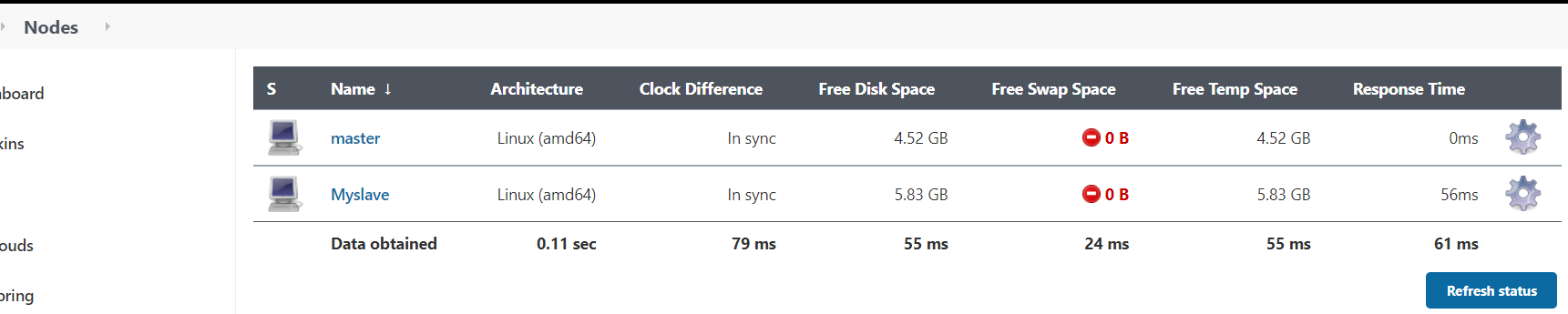
18) Enter Label Expression ( Slave\_lab)

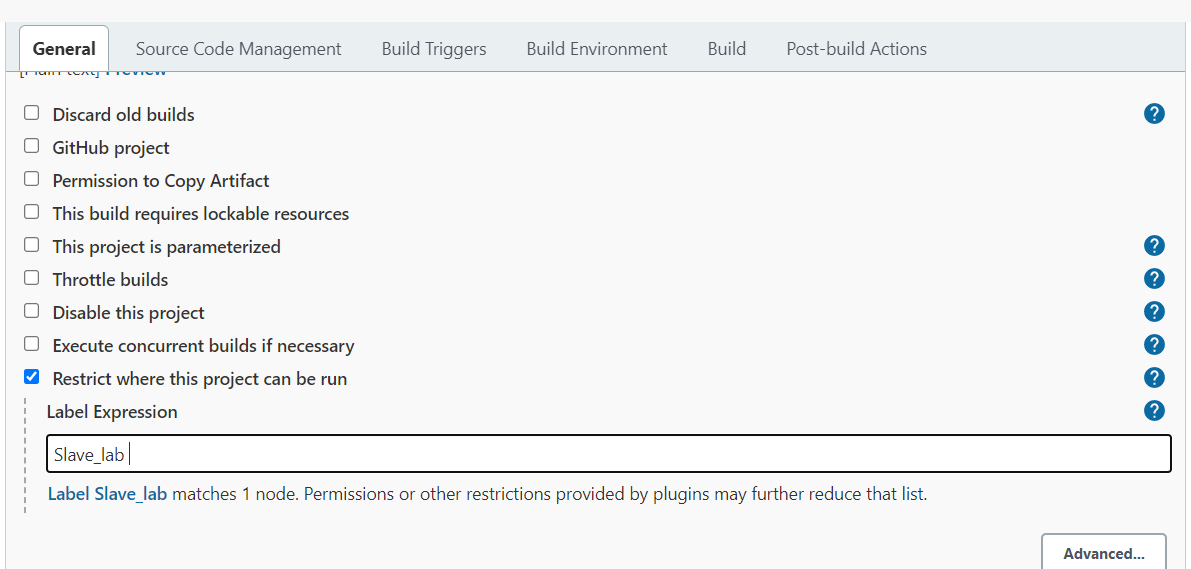
Apply ---> Save

Run the job, In console output, we can see the job is executed in slave machine

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







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

PIPELINE 07/07

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

Implementing CI-CD from the level of code.

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

Advantges

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

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

Pipelines can accept interactive human input before continuning 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 its fails, we want to send automated emails.

Scripted pipeline syntax:

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

node ( 'master/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 - master

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/sunildevops77/maven.git

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

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

Apply --- Save --> Run the job

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

2nd stage

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

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

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

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

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

Get the location of war file from log

$ scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.47.36:/var/lib/tomcat8/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 tomcat8

( Observation: tomcat8 directory -- others is not having write permissions )

$ sudo chmod -R o+w tomcat8/

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 - continuousdeployment 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 tomcat8 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

Test are you able to connect to prod?

$ ssh 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 tomcat8

( Observation: tomcat8 directory -- others is not having write permissions )

$ sudo chmod -R o+w tomcat8/

Now run the job

Connect prod server and check

http://13.126.45.247:8080/prodenv/

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

Script

---------

node('master')

{

stage('Continuous Download')

{

git 'https://github.com/sunildevops77/maven.git'

}

stage('Continuous build')

{

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

}

stage('Continuous Deployment')

{

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.21.16:/var/lib/tomcat8/webapps/qaenv.war'

}

stage('Continuous Testing')

{

sh label: '', script: 'echo "Testing Passed"'}

stage('Continuous Delivery')

{

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.28.16:/var/lib/tomcat8/webapps/prodenv.war'

}

}

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

13.126.48.87:8080/qaenv

13.127.24.219:8080/prodenv

08/07 MULTIBRANCHING

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

Multibranch pipeline

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

When developer creates code for multiple functionalities, he will generally do that on separate branches.

Every branch will contains specific code related to one functionality.

Along with the code, the developer will also create separate jenkins file for every branch.

This jenkins file will contain the stages of CI-CD, that should be performed on that branch.

All these branches along with jenkins file will be uploaded by into the github repository.

We should create a jenkins job, which will work on these branches parallely and execute the steps present in different jenkins files.

Steps performed by the developer

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

$ mkdir multibranch

$ cd multibranch

Download the files of maven repository

$ git clone https://github.com/sunildevops77/maven.git

Remove the hidden folder

$ cd maven

$ rm -rf .git ( Will break the link to maven repository )

$ git init ( create a new working directory )

$ git status

$ git add .

$ git commit -m "a"

$ git log

Developer creates branch

$ git checkout -b loans

$ git log

$ git checkout master

$ ls

Make changes to the jenkins file

$ vim Jenkinsfile

( Lets make it only two stages )

node('master')

{

stage('ContinuousDownload\_master')

{

git 'https://github.com/sunildevops77/maven.git'

}

stage('Continuousbuild\_master')

{

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

}

}

:wq

(Onservation, we have done the changes in master branch )

$ git add .

$ git commit -m "b"

$ git checkout loans

$ ls

$ vim Jenkinsfile

( Lets make it only two stages )

node('master')

{

stage('ContinuousDownload\_loans')

{

git 'https://github.com/sunildevops77/maven.git'

}

stage('Continuousbuild\_loans')

{

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

}

}

:wq

$ git add .

$ git commit -m "c"

Observe ( master branch is having jenkins file.

Loans branch is having jenkins file )

$ git checkout master

Create new repository in github

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

Repository name - Jenkins\_multiBranch24

$ git remote add origin https://github.com/sunildevops77/Jenkins\_multiBranch.git

$ git push -u origin --all ( as we want to push all branches )

( Check the remote repository )

+++This is developers activity+++++++++

Login to jenkins

New item -- MultiBranchPipeline

Select multibranch Pipeline

Branch Sources

Add source

Git

Project Repository -- https://github.com/sunildevops77/Jenkins\_multiBranch24.git

Scan multiline pipeline triggers

Check periodically if not otherwise

Interval - 1 minute

Apply --- Save

By this time it will be started.

This job will check github every minute.

Select multibranch pipeline

You will find two branches

Select loans , we can see two stages

Select master , we can see two stages

Lets say, developer will make changes and push to the repostitory

$ vim README.md ( Make some changes )

$ git add .

$ git commit -m "d"

Similarly, lets repeat in loans branch

$ git checkout loans

$ vim README.md ( Make some changes )

$ git add .

$ git commit -m "e"

$ git checkout master

To push all the branches

$ git push -u origin --all

Observation: Job will start automatically.

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

