JENKINS (CI/CD)

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 will trigger the maven and maven will build the code.

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)

CI-CD is a combination of 5 Stages :

Stage 1 : Continuous Download

Stage 2: Continuous Build

Stage 3: Continuous Deployment

Stage 4: Continuous Testing

Stage 5: Continuous Delivery

**Stage 1 : Continuous Download:**

Downloading the sourcecode from github to DevInstance continously.

**Stage 2: Continuous Build:**

To Build the source code into artifact (.war file)

**Stage 3: Continuous Deployment:**

Deploying the artifact which is available in the Dev Instance to the QA Instance.so that the application will be available at the qa team to perform testing activity.

**Stage 4: Continuous Testing:**

Running the testing scripts and testing application which is deployed is called "continoues Testing".

**Stage 5: Continuous Delivery:**

If the Artifact is passed . Then we deploy that artifact into "Prod Instance".So that the application goes live.

This process of making the application going live using jenkins is called as a CI/CD process.

Continuous Integration 🡺1-4

Continuous Delivery 🡺5

Without CI/CD we can’t image Devops and There is no Agile methodology.

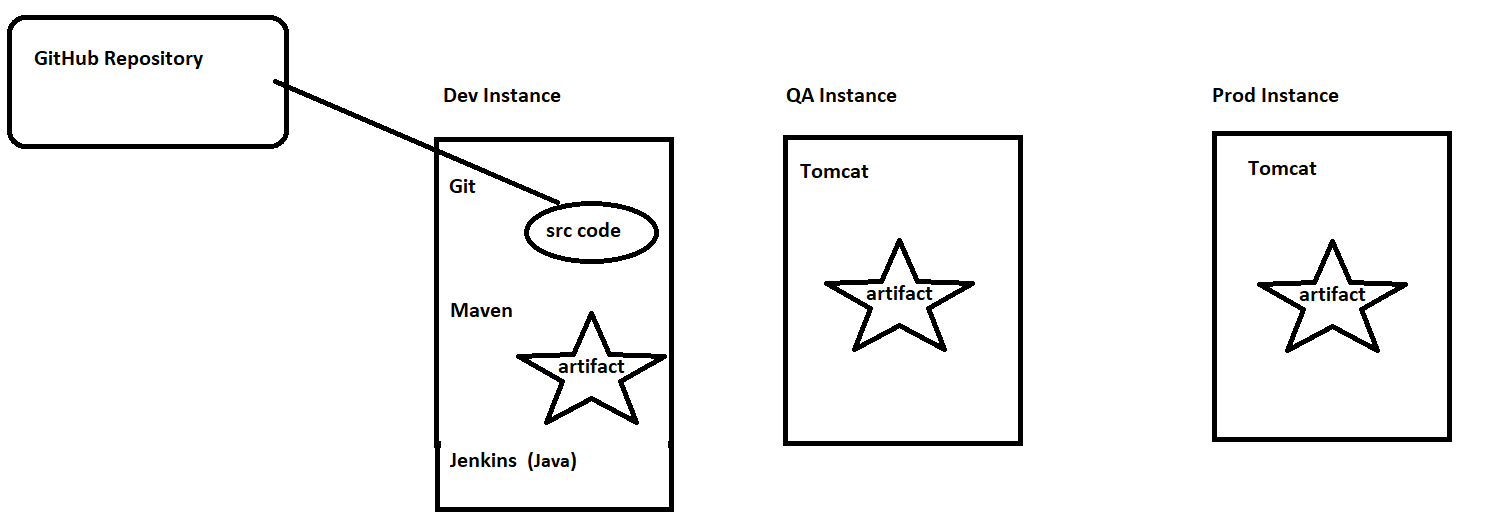
JENKINS is a browser based tool.we will access JENKINS from the browser.

|  |
| --- |
| Using jenkins we are making the following as Automation   1. Clone the code 2. Build the code 3. Deploy the code |

We need to create a (Job/pipeline/process/project) i.e integrating the commands using JENKINS.

That job includes

|  |
| --- |
| GIT  MAVEN  DEPLOYEMENT COMMANDS  SCHEDULE/POLLING  Trigger automation scripts ----------if 100% test cases are passed .we need to place these war file into production environment. |

****

**EXPLANATION HOW THE FLOW GOING IN JENKINS :**

If developer commits or not commits the particular code into git hub.These Jenkins job will interacts with the git hub for every 10min

.if a code is available immediatly it will clone, build and will deploy the code into a testing environment Automatically with the help of scheduler/polling.

In testing environment (QA environment) testing team will test the code using automation scripts or selenium scripts and the trigger command will be reside in the JENKINS tool .if the 100% test cases are passed .we need to place these war file into “production environment”

If the test cases failed then automatically email will shoot to both developers and QA team who ever involved in that email.

**How to install Jenkins /Run inside a AWS Instance:** **METHOD -1**

While installing instance give security group as (SSH,HTTP,ALLTRAFFIC ) source as (Anywhere)

sudo apt update

sudo apt install openjdk-8-jdk -y -To install Jenkins we need java .

java -version -To check java installed or not

sudo apt-get install -y git maven -To install git & maven (optional)

git --version & mvn –version -To check git & maven installed or not

Goto this Jenkins website 🡺 <https://jenkins.io/download/> 🡺Copy link from generic java pkg.

wget <https://get.jenkins.io/war-stable/2.303.3/jenkins.war> -To download the .war file.

java -jar jenkins.war -To start the Jenkins.(console will hung here because it will busy with log messages)

PUBLIC IP ADDRESS:8080 -To access the Jenkins homepage

It will open to create First admin user in Jenkins homage.

**Drawback:**

When we run java -jar Jenkins.war at that time only Jenkins server will opens. At that time instance is busy with logs.

If we give ctrl + c .Then jenkins server will won’t work.

**This drawback can be overcome by placing a war file into a Tomcat server.**

**Public Ip Address:8080/Jenkins**

**REMINDER :**

To unlock the Jenkins page we should copy the password which is generated at the time of running the war file. Simply we need to paste unlock Jenkins page.Otherwise goto that path and get password to unlock Jenkins plugins.

**#Project 1:**

**Creating a simple job to print “Hellow World ” using Jenkins ?**

Click on New project > Give project name > Select Free style project

Build tab

Click on execute Shell

In Command Box Enter echo " Hello Jenkins"

Click on Console Output

**How to install Jenkins /Run inside a container:** **METHOD -2**

sudo docker run -it --name c1 -p 8080:8080 ubuntu /bin/bash -To create a container.

sudo apt update

sudo apt install openjdk-8-jdk -y -To install Jenkins we need java .

java -version -To check java installed or not

sudo apt-get install -y git maven -To install git & maven (optional)

git --version & mvn –version -To check git & maven installed or not

Goto this Jenkins website 🡺 <https://jenkins.io/download/> 🡺Copy link from generic java pkg.

wget <https://get.jenkins.io/war-stable/2.303.3/jenkins.war> -To download the .war file.

java -jar jenkins.war -To start the Jenkins. (console will hung here)

PUBLIC IP ADDRESS:8080 -To access the Jenkins homepage

It will open to create First admin user in Jenkins homage.

**Drawback:**

When we run java -jar Jenkins.war at that time only Jenkins server will opens. At that time instance is busy with logs.

If we give ctrl + c .Then jenkins server will won’t work.

**This drawback can be overcome by placing a war file into a Tomcat server.**

**Public Ip Address:8080/Jenkins**

**REMINDER :**

To unlock the Jenkins page we should copy the password which is generated at the time of running the war file. Simply we need to paste unlock Jenkins page.

**#Project 1:**

**Creating a simple job to print “Hellow World ” using Jenkins ?**

Click on New project > Give project name > Select Free style project

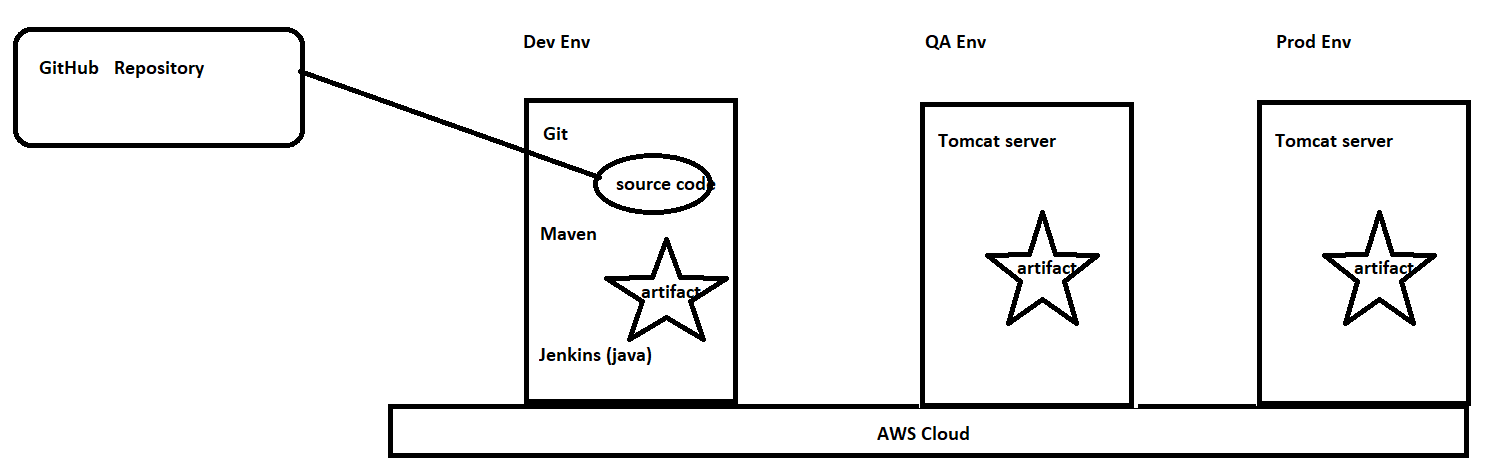
Build tab

Click on execute Shell

In Command Box Enter echo " Hello Jenkins"

Click on Console Output

**#Main Project : VVV IMP**

****

We are using private ip Address of the QA server. -***To connect with Dev Env and QA Env***

We are using private ip Address of the Prod server. - ***To connect with Dev Env and Prod Env***

**Stage 1 : Continuous Download: (DEV JOB )(SUCCESS)**

**First we need to install git, maven, Jenkins, java in Dev Env.**

**Downloading the sourcecode from github to DevInstance continously.**

1) New item >Dev Job > free style project > source code management > select Git

2) Enter the URL of github reposiditory

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

3) Click on apply , save and Run the Job

4) Check the console output and check whether files downloaded are not in Dev Server.

**Recommend:** Please install tomcat also otherwise if you try to run Jenkins with java -jar jenkins.war it will hung. So try to run Jenkins.war file from tomcat server. Public ip:8080/jenkins

**Stage 2: Continuous Build: (SUCCESS)**

**To Build the source code into artifact (. war file)**

1) Go to Build Section > add build step > Invoke top level maven targets

2) Enter the goal as package

3) Click on apply , save and Run the Job

4) Check the console output whether the war file/artifact is generated or not.

**Stage 3: Continuous Deployment: (FAILED TO DEPLOY) 24/05/2021 video.**

**Deploying the artifact which is available in the Dev Instance to the QA Instance.so that the application will be available at the qa team to perform testing activity.**

**For that we need to install tomcat server with username,password in QA server.**

**To deploy the artifact into QA instance**

Install "**Deploy to container**" plugin.

Go to Dasboard > manage Jenkins > manage plugins > avaiable section >

Search for plugin ( deploy to container )

Select that plugin and click on install without restart.

1) post build actions > add post build actions > deploy war/ear to container

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

we can give \*\*/\*.war in war/ear files. -***It means all files which having an extension with .war***

3) Context path: qaenv -***we will access the application using public ip :8080/context path name.***

4) Containers : select tomcat 8

Credentials : Click on add -***To Deploy .war file Jenkins will contact with tomcat server.***

select jenkins

enter tomcat user name and password

Click on add

Select credentials.

give the private ip of the QA server. -***To connect with QA Env with Dev Env ,need some link.***

http://private\_ip:8080

5) Click on apply and save and Run the job

6) To access the home page/application in browser.

public\_ip\_Qa\_server:8080/qaenv

**Another way to install tomcat**

It's not working

sudo apt-get update

sudo apt-get install -y tomcat8

sudo apt-get install -y tomcat8-admin

**Stage 4: Continuous Testing: (TESTING JOB)**

Testing team will develop the testing scripts and they will also upload the scripts into GitHub repository.

Jenkins responsible to download the testing scripts and running the testing scripts and testing application which is deployed is called "continoues Testing".

1) New item >Testing Job > free style project > source code management > select Git

2) Enter the URL of github reposiditory

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

3) Click on apply , save and Run the Job

4) Check the console output and check whether the files are downloaded or not in Dev server.

Why we need to check in Dev server means , Jenkins running based on Dev ser.

5) 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 ( DevelopmentJob ) -- 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 while continoues delivery**

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

For this we need to install "**Copy Artifact**" plugin.

Dasboard > manage Jenkins > avaiable section > Search for plugin ( Copy Artifact)

Select that plugin and click on install without restart.

**Stage 5: Continuous Delivery:**

If the Artifact is passed . Then we deploy that artifact into "Prod Instance".So that the application goes live.This is called Continuous delivery.

**To Archieve the Artifact before deploying it into Prod Env :**

For this we need to install "**Copy Artifact**" plugin.

Dasboard > manage Jenkins > avaiable section > Search for plugin ( Copy Artifact)

Select that plugin and click on install without restart.

Click on post build actions of the **development job.**

Add post build actions > Archive the artifacts

Enter \*\*/\*.war

Click on apply and save

Click on Build Section on **Testing job**

Add build step > copy artifacts from another project

Enter Development as project name

**For Deployment application into live**

1) Click on post build actions of the Testing job

Add post build actions > deploy war/ear to container

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

we can give \*\*/\*.war in war/ear files. -***It means all files which having an extension with .war***

3) Context path: prodenv -***we will access the application using public ip :8080/context path name.***

4) Containers : select tomcat 8

Credentials : Click on add -***To Deploy .war file Jenkins will contact with tomcat server.***

select jenkins

enter tomcat user name and password

Click on add

Select credentials.

give the private ip of the Prod server. -***To connect with Prod Env with Dev Env ,need some link.***

http://private\_ip:8080

http://172.31.34.103:8080

5) Click on apply , save and ) Run the job

6) To access the home page/application in browser.

public\_ip\_Qa\_server:8080/prodenv

**How to install Jenkins /Run inside a container using Tomcat server: METHOD -3**

|  |
| --- |
|  |

We are installing Jenkins inside the container. Jenkins war file should kept in tomcat server, So tomcat required.

We can use our image also we followed Dockerfile II approach

**Docker run -it --name c1 -p 8080:8080 pavan0423/Jenkins-server /bin/bash**

**[OR]**

|  |
| --- |
| docker run-it --name c5000 -p 8080:8080 ubuntu /bin/bash  apt-get update  apt-get install openjdk-8-jdk |

|  |
| --- |
| apt-get update  apt-get install openssh-server -----sometimes we need to interact Jenkins in some outside machies so we need its just a key.  service ssh start ----to start the ssh service  service ssh status -----to check whether ssh is instlled and running or not. |

|  |
| --- |
| **whenever I login to Jenkins with other machine I need to login with ssh and we need to provide username and password.**  useradd -m -d /home/pavan -s /bin/bash pavan  passwd -----🡪pavan |

|  |
| --- |
| mkdir distross  cd distross  wget https://downloads.apache.org/tomcat/tomcat-10/v10.0.7/bin/apache-tomcat-10.0.7.tar.gz  tar -zxvf apache-tomcat-10.0.7.tar.gz  cd apache-tomcat-10.0.7  cd webapps  wget <https://get.jenkins.io/war-stable/2.289.2/jenkins.war>  ./startup.sh (bin folder lo unnaka run cheayli) |

Ipaddress:8080/Jenkins

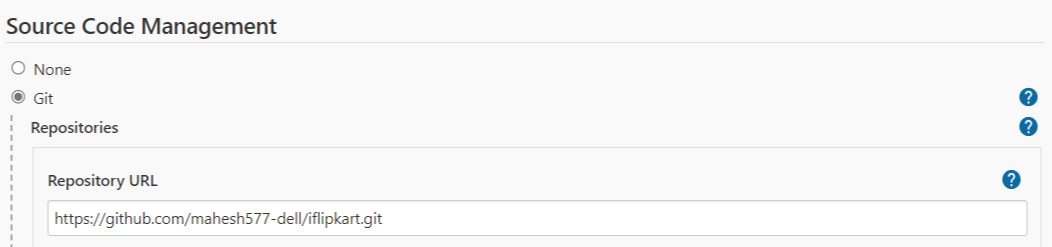
Copy the passwordlink and use cat link then copy the password you will get

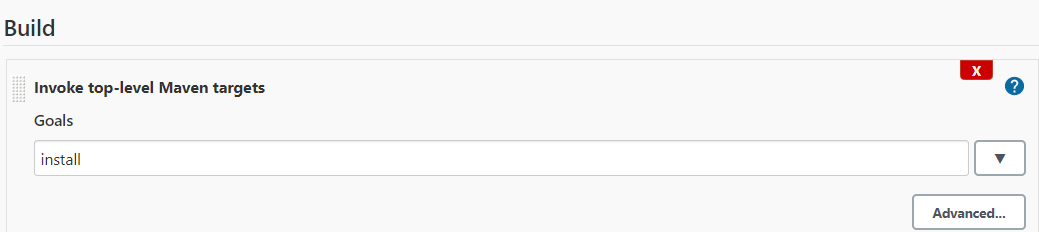
Click on Install suggested plugins

Now your Jenkins will ready to use.

**# Project 1 (FlipkartProject) : (integrating Git & Maven)**

General :This is a Jenkins pipeline project





Then click on Apply > Save > Build > console output >

Click on Full Log





Goto container and enter you will see project details

cd /root/.jenkins/workspace/Pavanproject

NOTE:

* Everything we are doing inside a container.
* As a devops engineer we are the admins for the Jenkins we can cutdown the access for some particular options from developers and testers .

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

**System Configurations:**

**1.Configure System:**

|  |
| --- |
| **For plain text message as a Notification:**  Manage jenkins > Configure system > System message  Enter some Plain Text message  Apply > save |

|  |
| --- |
| **For HTML text message as a Notification:**  Manage Jenkins > Configure Global Security > Markup formatter  (select Safe HTML)  > Manage jenkins > Configure system > System message (Enter HTML Code). |

**E-mail Configuration:**

In case, if a job fails , we need to send notificiation.

For that we need to integrate jenkins to smtp server. (Simple mail transfer protocol)

**Due to some security issue we are giving from and two address as same**

**From:**

|  |
| --- |
| Manage jenkins > Configure system > Extended E-mail Notifications >  SMTP server : smtp.gmail.com  Port :465  Click on Advance  SMTP uername :[Ppasupuleti50@gmail.com](mailto:Ppasupuleti50@gmail.com)  SMTP uername: Give password  Check SSL  Apply & save |

**To:**

|  |
| --- |
| Post build Actions > Editable Email Notifications >  Project receipt list: [ppasupuleti50@gmail.com](mailto:ppasupuleti50@gmail.com)  Advanced settings > Click on Add trigger >Always |

**2. Manage Plugins :**

What is a Plugin?

Plugin is an extra feature which will be added and installed in your Jenkins system.

Jenkins has 1000’s of plugins.

Green balls plugin, shelve ….etc

|  |
| --- |
| How to install Plugins:  Manage Jenkins > Manage plugins > available > install without restart > click check box. |

|  |
| --- |
| How to uninstall Plugins:  Manage Jenkins > Manage plugins > installed > search for required plugins that you need to uninstall > uninstall |

Ex 2:

Jenkins may become slow if we keep on creating jobs.If we remove unused jobs and that you need to take a backup of that particular job. To take the back up of that project we need to install shelve plugin.

Once installed then we can see the option shelved projects in dashboard.

Just click on drop down arrow on any project then select shelve and we can also unshelve the poject.

**3. Global Tool Configuration : (Declaring user defined versions)**

Ex:

We have 100 jobs in Jenkins dashboard

* Few jobs/projects ---50jobs ----jdk8 version
* Maven 3.0
* Few jobs/projects ---50jobs ----jdk9 version
* Maven 4.0

Manage Jenkins > global tool configuration settings. Give your required version for java/maven.

Refresh the page

Click on any project > configure > you can choose your required versions.

**4.Manage Clouds and nodes:**

Master slave Configuration concept/Distributed Builds:

if server have limited hardware resources like memory,hardware,processer.When there are more number of jobs are running because of the limited resources there is a chance to go jenkins server go down.

In that case we can avoid down time by creating one more machine called “slave machine”

The machine which have Jenkins installed is also called as “master machine”.

**Conclusion:**

When we want to run multiple jobs parallel we will use this concept.

Ex:

If we try to run 1000 jobs in master Jenkins server then master Jenkins server will become slows. To avoid this we are going to the concept called “Master slave concept”.

**Dev Instance**

Master Jenkins server

1000 jobs server will become slow

Slave node 3/Agent 3

250 jobs

Slave node 2/Agent 2

250 jobs

Slave node 1/Agent 1

250 jobs

**Ec2 Instance 1 Ec2 Instance 2 Ec2 Instance 3**

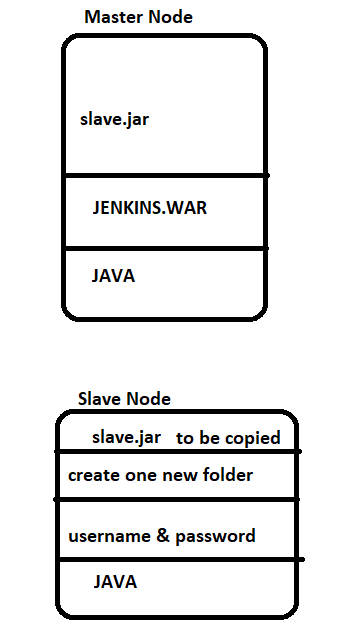
How to implement master slave concept:

1. We should know how to attach an agent/slave node to master Jenkins server.
2. We should know how to route the jobs to a particular agent/slave node.

EXAMPLE 1:

**(Q) We want to run the jobs on slave .This will reduce the load in master server.**

Let One Master & One Slave : Launch agent via execution of command on the controller



**Step 1: Master Instance**

Update the libraries and install the java package.

Download the Jenkins.war file from browser , run it and access the home page.

**Step 2: Slave Instance**

***In Master Instance ,***

Exchange keys with **Slave Instance** .

***In Slave Instance ,***

Define password for the existing user or create new user & password .

In sshd\_config file make sure that PasswordAuthentication is YES & Restart the ssh services.

Then connect to **Slave Instance** Through **Master Instance** without password.(To test connection)

Download slave.jar in slave machine (Whenever you install jenkins by default slave.jar will be downloaded in master node .That should be copied into slave node)

sudo wget <http://pvt_ip> address\_master:8080/jnlpJars/slave.jar ===> should run in slave machine.

**Note:**

At this time jenkins should run then only it will download.otherwise it will thow error as connection refused.

**Step 3 :**

Creating node

Manage Jenkins > manage nodes and clouds > new node

Node name = slavenode1 - select permanent agent

Click on ok

Description = This is a slavenode going to be attached to master Jenkins server

# of executors = 1

Remote root directory = /home/ubuntu/newfolder

Labels = nandhyal

Launch method = **Launch agent via execution of command on the controller**

In Launch command

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

Save

Lauch agent1

**Step 4 :**

New item > flipkartproject > Freestyle project > ok

This is a flipkart real time project

Restrict where this project can be run = nandhyal

Build > Add build step > Execute shell > echo “Hellow world”

Apply & save 🡺 Run the job

[or]

**Step 4 :**

New item > flipkartproject > Freestyle project > ok

This is a flipkart real time project

Restrict where this project can be run = nandhyal

<https://github.com/mahesh577-dell/iflipkart>

invoke top-level maven projects

install

Apply & save 🡺 Run the job

EXAMPLE 2:

**(Q) We want to run the jobs on worker .This will reduce the load in master server.**

Let one master & one slave (***launch agents via SSH***)

We need two machines,

1. One machine will acts as a master Jenkins server.

jdk, tomcat server, Jenkins. War, started the server, ssh (maheshdevops123/Jenkins-server)

1. Second machine will acts as a Agent/slave node

jdk, git, maven, user with password

**BUILD PART  *BUILT SUCCESSFULLY***

**STEP1: MASTER**

|  |
| --- |
| docker run -it --name master -h master maheshdevops123/jenkins-server /bin/bash  start tomcat server  login into Jenkins server |

**STEP 2: WORKER**

|  |
| --- |
| docker run -it --name worker -h worker ubuntu /bin/bash  useradd -m -d /home/pavan -s /bin/bash pavan  passwd pavan  apt-get install openssh-server  service ssh start  connect to master and provide  ssh username@pvt\_ipaddress of slave 🡺 To test the connection    Create an empty folder which will work like workspace for jenkins to use on the slave machine and given full permissions (Copy path) |

**STEP 3:** You must need to create slavenode first then only this option ***Restrict where this project can be run*** will be enable while creating project .Otherwise you can’t see that option.

Creating slavenode

|  |
| --- |
| Manage Jenkins > manage nodes and clouds > new node  Node name = slavenode1  Click on ok  Description = This is a slavenode going to be attached to master Jenkins server  # of executors = 1  Remote root directory = /home/ubuntu/newfolder  Labels = nandhyal  Launch method =**launch agents via SSH**  Hostname = ipaddress of slave  Credentials > add > Jenkins = pavan & pavan  Save  Lauch agent1 |

**STEP 4:**

|  |
| --- |
| New item > flipkartproject > Freestyle project > ok  This is a flipkart real time project  Restrict where this project can be run = nandhyal  <https://github.com/mahesh577-dell/iflipkart>  invoke top-level maven projects  install  Apply & save |

**STEP 5:**

Build project

|  |
| --- |
| Note:  Without jdk agent won’t launch (Jenkins failed to launch the agent process) 🡪must need jdk in slavenode.  Building remotely on Slavenode1 (nandhyal) in workspace /newfolder/workspace/flipkartproject  ERROR: Error cloning remote repo 'origin' 🡪must install git  Build step 'Invoke top-level Maven targets' marked build as failure 🡪must install mvn  For suppose if you would like to give remote root directory as /apache-tomcat-8.5.70/webapps.you will get remoting, remoting.jar, workspace . workspace/flipkartproject2/target/mahesh.war file will be get. |

**(BUILD & DEPLOYMENT) *BUILT SUCCESSFULLY***

STEP1: Master

|  |
| --- |
| docker run -it --name master -h master maheshdevops123/jenkins-server /bin/bash  apt-get install sshpass 🡪to pass the password  start the tomcat server and launch jenkins machine. |

STEP 2: Worker

|  |
| --- |
| docker run -it --name worker -h worker ubuntu /bin/bash  apt-get install openjdk-8-jdk  apt-get install git  apt-get install maven  useradd -m -d /home/pavan -s /bin/bash pavan  passwd pavan  apt-get install openssh-server  service ssh start  ssh [pavan@172.17.0.3](mailto:pavan@172.17.0.3)  ssh [pavan@172.17.0.3](mailto:pavan@172.17.0.3) 🡪 Atleast we need to connect 2nd machine through SSH from 1st machine i.e master and exit from that.  If you face conncection refused then use this command vim /etc/ssh/sshd\_config give Authentication=permitRootLogin yes  service ssh restart  we need to give full permission ----🡪chmod 777 -R apache-tomcat-8.5.70  ls -ld apache-tomcat-8.5.70 ----------🡪 To check permissions |

STEP 3:

|  |
| --- |
| New item > Eveningproject > Freestyle project > ok  This is a build & deployment project project  --Restrict where this project can be run = ora-jdk-2021  <https://github.com/mahesh577-dell/iflipkart>  invoke top-level maven projects  install  Apply & save  Build project --🡪war file will be generated. |

STEP 4:

|  |
| --- |
| Eveningproject > configure > build > Add build step > Execute shell >  sshpass -p "pavan" scp target/mahesh.war pavan@172.17.0.3:/Distros/apache-tomcat-8.5.66/webapps  sshpass -p "pavan" ssh pavan@172.17.0.3 "/Distros/apache-tomcat-8.5.66/bin/startup.sh"  Apply & save  Build project  Now you can see in 2nd machine (worker). There will be a war file. |

**Security:**

As a devops engineer we need to restrict the developers,testors from some access.

Authentication ---Who can login into Jenkins

Authorization ---once enter into Jenkins account , what can he do

**1.Manage users**: (Authentication)

Pavan on-boarded ---Jenkins----create username and password ?

Manage jenkins > Manage users > Create user/delete user/Modify user

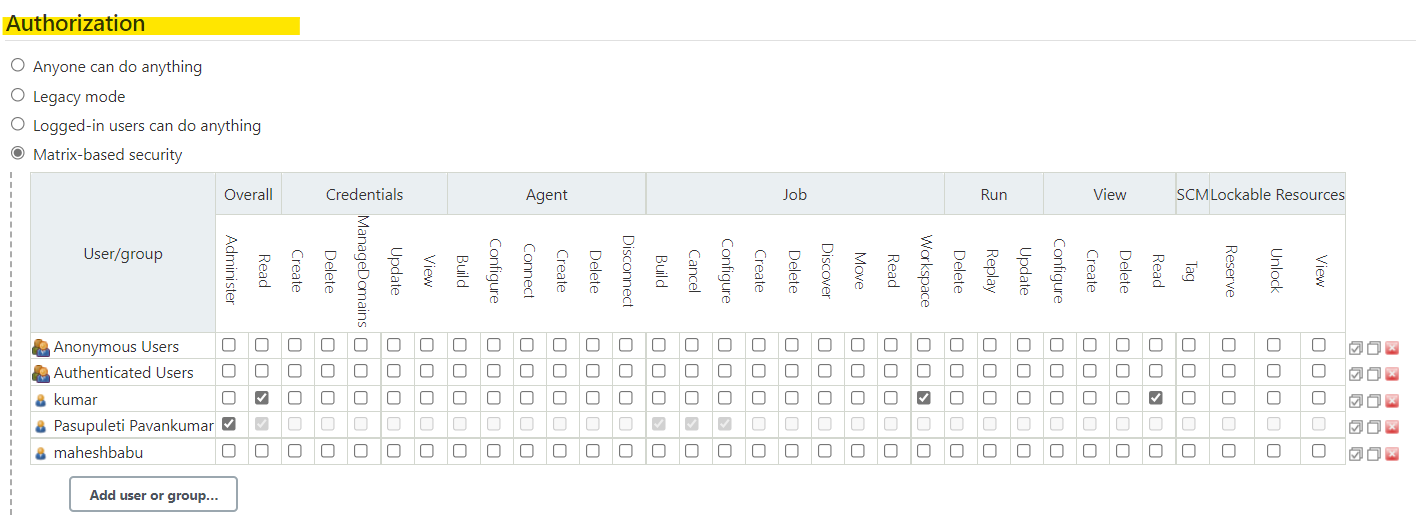
**2.Configure Global Security**: (Authorization. Giving permissions/Retrieving permissions.)

1. Once we provide any password and username no one has access to change it only an admin has access to change username and password.Because we need to restrict newuser .if we allow newuser to do anything .he/she may do anything.

Almost he/she can remove access like administrator to the user.

**Method 1:**

By default newUsers will have all permissions.so newusers equal to admin.So we need to restrict new users.



**Method 2 :** (Creating Roles & Assigning Roles To give permissions / To Retrieve the Permissions).

By default newUsers will have all permissions.so newusers equal to admin.So we need to restrict new users. For that we need to install pulgin called "role based authorization strategy".

**Step 1:**

manage jenkins > configure global security > click on role based strategy.

**Step 2:**

manage jenkins > manage and assign roles > mange roles > global roles and create a role

for this employee in overall give read access.

in view section give all access

**Step 3**:

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)

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

give all permisiinons to developrs and tester

apply--save

**Step 4:**

click on assign roles

go to global roles and add user1 and user2 . check user1 and user2 as employees.

go to item roles add user1 and user2 check user1 as developer and user2 as tester.

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.***

**Configures can restricted using global roles**

**Jobs can be restricted using patterns by using item roles**

2. LDAP server:

It’s a commercial official server LDAP.We will integrate usernames/paswwords for all applications of a user.

If the user going to another company then we need to retirve/delete the usrnames/passwords from each application is difficult.

To avoid that difficult we can integrate all usernames/paswwords for all applications of a user. Once we delete here .Then users have no longer to access applications.

DAY 26 11/07/2021

***Docker file to create image***

FROM ubuntu:latest

RUN apt-get update && apt-get install openjdk-8-jdk -y

##javapath

RUN echo "JAVA\_HOME=/user/" >>.bashrc

ADD https://dlcdn.apache.org/tomcat/tomcat-8/v8.5.70/bin/apache-tomcat-8.5.70.tar.gz /tmp/

RUN cd /tmp && tar -zxvf apache-tomcat-8.5.70.tar.gz

RUN cd /tmp && mv apache-tomcat-8.5.70 /opt/

EXPOSE 8080

ADD https://get.jenkins.io/war-stable/2.303.1/jenkins.war /opt/apache-tomcat-8.5.70/webapps/

ENTRYPOINT /opt/apache-tomcat-8.5.70/bin/startup.sh && bash

**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.

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.This cannot be achieved from freestyle job.

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.

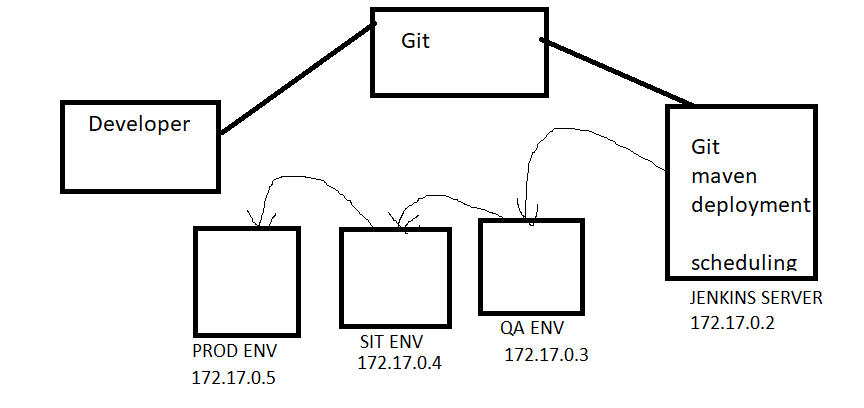
Ex: If testing passes, we want to go to delivery.

If its fails, we want to send automated emails.

**We have 2 kinds of pipeline**

1. **Manual pipeline**
2. **Scripted pipeline/Declarative pipeline**

Pipeline is nothing but interconnection of multiple jobs.



***1. MANUAL PIPELINE***

maheshdevops123/jenkins-server

Maheshdevops123/tomcat-server-image

STEP 1:

|  |
| --- |
| docker run -it --name master -h master-jenkins maheshdevops123/jenkins-server /bin/bash  apt-get install sshpass  start the tomcat server and launch jenkins machine. |

STEP 2:

|  |
| --- |
| docker run -it --name qa-env -h qa-env maheshdevops123/tomcat-server-image /bin/bash  service ssh start  ssh [mahesh@172.17.0.3](mailto:mahesh@172.17.0.3) 🡪 Atleast we need to connect 2nd machine through SSH from 1st machine i.e master and exit from that.  If you face conncection refused then use this command vim /etc/ssh/sshd\_config give Authentication=permitRootLogin yes  service ssh restart  we need to give full permission ----🡪chmod 777 -R apche-tomcat  ls -ld apche-tomcat / ----------🡪 To check permissions |
| |  | | --- | | Sudo docker run -it --name sit-env -h sit-env maheshdevops123/tomcat-server-image /bin/bash  service ssh start  ssh mahesh@172.17.0.4 🡪 Atleast we need to connect 3nd machine through SSH from 1st machine i.e master and exit from that.  If you face conncection refused then use this command vim /etc/ssh/sshd\_config give Authentication=permitRootLogin yes  service ssh restart  we need to give full permission ----🡪chmod 777 -R apche-tomcat  ls -ld apche-tomcat / ----------🡪 To check permissions | | |  | | --- | | Sudo docker run -it --name prod-env -h prod-env maheshdevops123/tomcat-server-image /bin/bash  service ssh start  ssh mahesh@172.17.0.5 🡪 Atleast we need to connect 4th machine through SSH from 1st machine i.e master and exit from that.  If you face conncection refused then use this command vim /etc/ssh/sshd\_config give Authentication=permitRootLogin yes  service ssh restart  we need to give full permission ----🡪chmod 777 -R apche-tomcat  ls -ld apche-tomcat / ----------🡪 To check permissions | | |

STEP 3: masternode

|  |
| --- |
| New item > flipkart-qa > Freestyle project > ok  This is a build & deployment project project  <https://github.com/mahesh577-dell/iflipkart>  Build Tirggers > Poll SCM = \* \* \* \* \*  invoke top-level maven projects  install  Apply & save  Build the project --🡪war file is generated just checking it will build successfully or not |

STEP 4:

|  |
| --- |
| Flipkart-qa > configure > build > Execute shell >  sshpass -p “mahesh” scp target/mahesh.war mahesh@172.17.0.3: /Distross/apache-tomcat-8.5.69/webapps  sshpass -p “mahesh” ssh [mahesh@172.17.0.3](mailto:mahesh@172.17.0.3) “/Distross/apache-tomcat-8.5.69/bin/startup.sh”  Apply & save  Build project  72.17.0.3:8080/Mahesh ----🡪 check in browser |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

STEP 5:

|  |
| --- |
| Flipkart-sit > configure > build > Execute shell >  cd /root/.jenkins/workspace/Flipkart-qa  sshpass -p “mahesh” scp target/mahesh.war mahesh@172.17.0.4: /Distross/apache-tomcat-8.5.69/webapps  sshpass -p “mahesh” ssh mahesh@172.17.0.4 “/Distross/apache-tomcat-8.5.69/bin/startup.sh”  Apply & save  Now we need to link qa-env with sit-env so .  Flipkart-qa > configure > build >Post Build Actions > Build other projects  Projects to Build =Flipkart-sit  Now Build project from Flipkart-qa .it will build both projects i.e  Flipkart-qa & Flipkart-sit  172.17.0.3:8080/Mahesh ----🡪 check in browser  172.17.0.4:8080/Mahesh ----🡪 check in browser |

STEP 6:

|  |
| --- |
| Flipkart-prod > configure > build > Execute shell >  cd /root/.jenkins/workspace/Flipkart-qa  sshpass -p “mahesh” scp target/mahesh.war mahesh@172.17.0.5: /Distross/apache-tomcat-8.5.69/webapps  sshpass -p “mahesh” ssh mahesh@172.17.0.5 “/Distross/apache-tomcat-8.5.69/bin/startup.sh”  Apply & save  Now we need to link sit-env with prod-env so .  Flipkart-sit > configure > build >Post Build Actions > Build other projects  Projects to Build =Flipkart-prod  Now Build project from Flipkart-qa .it will build three projects i.e  Flipkart-qa , Flipkart-sit & Flipkart-prod  172.17.0.3:8080/Mahesh ----🡪 check in browser  172.17.0.4:8080/Mahesh ----🡪 check in browser  172.17.0.5:8080/Mahesh ----🡪 check in browser |

**2. Scripted pipeline/Declarative pipeline**

We can write groovy script or we can create Jenkinsfile

Scripted pipeline syntax:

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

node ( 'master/slave')

{

stage(' Stage in CI-CD')

Groovy code for implementing the stage

}

}

We need to Install **Build pipeline** plugin.

***Example1 :***

Create new item --- ScriptedPipeline

select pipeline --OK

Pipeline tab,

Now prepare groovy script with the help of **snippet generator**.

**1.Continous Download (Dev server)**

First we need to install java. And we need to Download Jenkins.war file

node: Allocate node

label - master

stage:Stage

Stage name : Continuous Download

git:Git

Repository URL - https://github.com/sunildevops77/maven.git

Apply --- Save --> Run the job

node('master') {

stage('ContinousDownload') {

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

}

}

**2.Continous Build (Dev server)**

sudo apt-get install maven

sh: Shell Script

Shell script: mvn package

node('master') {

stage('ContinousDownload') {

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

}

stage('ContinousBuild') {

sh 'mvn package'

}

}

**3. Continous Testing (QA server)**

**Step 1:**

Deployment is nothing but , copying the war file from Dev server to QA server.

*For that we need to establish password less SSH connection between Dev server and QA Server.*

**In Dev server,**

Exchange keys with QA server .

**In QA server,**

Define password for the existing user or create new user & password .

In sshd\_config file make sure that PasswordAuthentication is YES.

Restart the services.

Then connect to QA server Through Dev server without password.(To test connection)

Install Tomcat & java in Prod server

Provide the write permission to tomcat server folder 🡺 so that war file will copied from Dev to QA

**Step 2:**

sh: Shell Script

shell script : scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war [ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/qaenv.war](mailto:ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/qaenv.war)

node('master') {

stage('ContinousDownload') {

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

}

stage('ContinousBuild') {

sh 'mvn package'

}

stage('ContinousDeployment') {

sh 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/qaenv.war'

}

}

Start the tomcat services and run the job in browser. **Public\_ip\_address:8080/qaenv/**

**Note** :

1. To start the tomcat services java must be installed .
2. While copying file we can copy the file with same name / we can copy the file and we can give the file name whatever we want.

**4. Continous Testing**

sh: Shell Script

Shell script: echo “Testing Passed”

node('master') {

stage('ContinousDownload') {

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

}

stage('ContinousBuild') {

sh 'mvn package'

}

stage('ContinousDeployment') {

sh 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/qaenv.war'

}

stage('ContinousTesting') {

sh 'echo "Testing Passed"'

}

}

**5.Continous Delivery (Prod server)**

**Step 1:**

Deployment is nothing but , copying the war file from Dev server to Prod server.

*For that we need to establish password less SSH connection between Dev and Prod .*

**In Dev server,**

Exchange keys with Prod server .

**In Prod server,**

Define password for the existing user or create new user & password .

In sshd\_config file make sure that PasswordAuthentication is YES.

Restart the services.

Then connect to Prod server Through Dev server without password.(To test connection)

Install Tomcat & java in Prod server

Provide the write permission to tomcat server folder 🡺 so that war file will copied from Dev to Prod

**Step 2:**

sh: Shell Script

shell script : scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war [ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/prodenv.war](mailto:ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/prodenv.war)

node('master') {

stage('ContinousDownload') {

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

}

stage('ContinousBuild') {

sh 'mvn package'

}

stage('ContinousDeployment') {

sh 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/qaenv.war'

}

stage('ContinousTesting') {

sh 'echo "Testing Passed"'

}

stage('ContinousDelivery') {

sh 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/prodenv.war'

}

}

Start the tomcat services and run the job in browser. **Public\_ip\_address:8080/prodenv/**

**Note** :

1. To start the tomcat services java must be installed .
2. While copying file we can copy the file with same name / we can copy the file and we can give the file name whatever we want.

**Trigger the job periodically**

To build the job periodically, we use cron job format

Build Triggers > Build periodically > Schedule

We need to give 5 values

Min - 0-59

hour - 0-23

dom - 1-31

month - 1-12

dow - 0-6

**Lets say you want to run the job every day at 9:30 PM**

30 21 \* \* \*

**Lets say you want to run the job every day at 9:30 PM from Mon to Friday**

30 21 \* \* 1-5

**To run the job every minute**

\* \* \* \* \*

**Multibranch pipeline** : (branching increases the modularity)

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.**

Example:

**++++++++++++++++++++++++++Developers Activity+++++++++++++++++++++++++++++++++++++++++**

mkdir multibranch

cd multibranch

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

git add .

git commit -m "a"

git log --oneline

***Developer creates a new branch***

git checkout -b loans ===>checkout -b is used to create branch and will move into that new branch.

git log --oneline

**git checkout master**

Make changes to the jenkins file 🡺( Lets make it only two stages.Then it will be a untracked file )

git add .

git commit -m "b"

**git checkout loans**

Make changes to the jenkins file 🡺( Lets make it only two stages.Then it will be a untracked file )

git add .

git commit -m "c"

Observation :

1.master branch is having jenkins file.

2.Loans branch is having jenkins file .

**Create new repository in github ==> Jenkins\_multiBranch24**

**git checkout master**

git remote add origin https://github.com/sunildevops77/Jenkins\_multiBranch.git ===>This will link with LR to RR.

git push -u origin --all ( refers to push all branches .Now we can check in github repository)

**++++++++++++++++++++++++++Ends developers Activity++++++++++++++++++++++++++**

Login to jenkins

New item > MultiBranchPipeline > Select multibranch Pipeline > Branch Sources > Add source > Git

Enter the repository URL

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.

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

vim README.md ( Make some changes )

git add .

git commit -m "d"

git push -u origin --all

Observation: Job will start automatically.

Note : Only the branch in which the code is updated will only execute.it won't execute other branches.

Observation:

Select job , You will find two branches.

Select loans , we can see two stages (because jenkinsfile have only two stages)

Select master , we can see two stages (because jenkinsfile have only two stages)

**Waiting for the approval from delivery head**: ( it is only possible with pipeline job)

**Jobs wait for specific stage .Let for continuous Delivery.**

input:wait for interactive input

Message: Waiting for Approval

input 'Waiting for Approval' ===>Script generated using snippet generator.

node('master') {

stage('ContinousDownload') {

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

}

stage('ContinousBuild') {

sh 'mvn package'

}

stage('ContinousDeployment') {

sh 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/qaenv.war'

}

stage('ContinousTesting') {

sh 'echo "Testing Passed"'

}

stage('ContinousDelivery') {

**input 'Waiting for Approval'**

sh 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.25.11:/home/ubuntu/Tomcat/apache-tomcat-8.5.73/webapps/prodenv.war'

}

}

**How to create users & passwords:**

sudo passwd username ===>To define password for the existing user.

useradd -m -d /home/pavan -s /bin/bash pavan ==>To create new user and password.

passwd pavan

**How to connect to remote server :**

**With password:**

**In remote server,**

Define password for the existing user or create new user & password .

In sshd\_config file make sure that PasswordAuthentication is YES .

Restart the services .

Then connect remote server through local server

**Without password:**

**In local server,**

Exchange keys with remote server .

**In remote server,**

Define password for the existing user or create new user & password .

In sshd\_config file make sure that PasswordAuthentication is YES.

Restart the services.

Then connect remote server through local server without password.

**While installing instances in AWS**

In security configurations give,

SSH => Anywhere

AllTraffic => Anywhere

HTTP => Anywhere

**How to run Jenkins using command**

**Java -jar jenkins.war**

**Email Integration in Jenkins:**

In case, if a job fails , we need to send notificiation.

For that we need to integrate jenkins to smtp server.

We are now integrating jenkins with gmail smtp server.

Manage Jenkins > Configure System > Email Notification

SMTP Server : smtp.gmail.com

**Click on Advance button**

USE SMTP Authentication

Username - sunildevops77@gmail.com

Password - password for the above email

use SSL

SMTP Port - 465

**Gmail Settings to get email from Jenkins:**

1) Goto google account -- Less secure app access --- Allow less secure apps: ON

2) "Disable captcha gmail"

Search in google "Disable captcha gmail" -- Continue

Test Configuration by sending e-mail.

Test email Receipent - [sunildevops77@gmail.com](mailto:sunildevops77@gmail.com)

Post Build Actions Tab > email notification > Enter email address which you want to send.

Note: only Failed jobs we will get in mail.