20 May

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 environment (JRE) Installed.

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

Stages in CI-CD

Stage 1 : Continuous Download

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—DEV, QA, PROD

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 or open in putty.

(Note: it 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

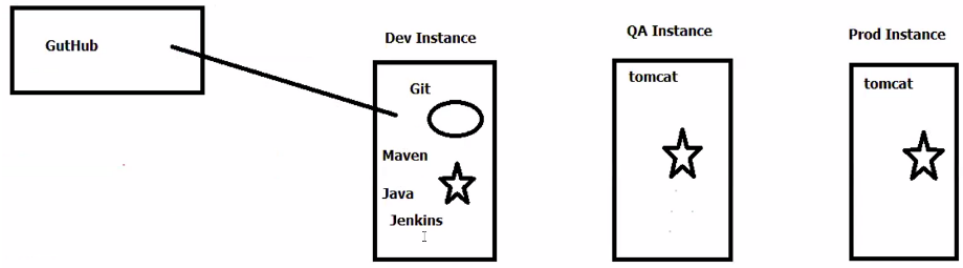
8) Just Type Yes

9) Now you are connect to the AWS Instance

Now we run any command that command run in the AWS DEV Instance

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

Install Jenkins in AWS DEV 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.

1) Update the apt repository

sudo apt update

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

3) Check the Java Version

java -version

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

```
ubuntu@ip-172-31-94-232:~$ git -version
unknown option: -version
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]
ubuntu@ip-172-31-94-232:~$ git --version
git version 2.34.1
ubuntu@ip-172-31-94-232:~$ mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 1.8.0_342, vendor: Private Build, runtime: /usr/lib/jvm/java-8-openjdk-amd64/jre
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.15.0-1017-aws", arch: "amd64", family: "unix"
ubuntu@ip-172-31-94-232:~$ wget https://get.jenkins.io/war-stable/2.277.2/jenkins.war
--2022-08-28 08:32:52--  https://get.jenkins.io/war-stable/2.277.2/jenkins.war
Resolving get.jenkins.io (get.jenkins.io)... 52.167.253.43
Connecting to get.jenkins.io (get.jenkins.io)|52.167.253.43|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://mirror.gruenehoelle.nl/jenkins/war-stable/2.277.2/jenkins.war [following]
--2022-08-28 08:32:52--  https://mirror.gruenehoelle.nl/jenkins/war-stable/2.277.2/jenkins.war
Resolving mirror.gruenehoelle.nl (mirror.gruenehoelle.nl)... 185.132.179.22, 2a00:7c80:0:de::2
Connecting to mirror.gruenehoelle.nl (mirror.gruenehoelle.nl)|185.132.179.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 70887351 (68M) [application/java-archive]
Saving to: 'jenkins.war'

jenkins.war                         100%[===================================================================================================>]  67.60M  12.2MB/s    in 6.6s

2022-08-28 08:32:59 (10.3 MB/s) - 'jenkins.war' saved [70887351/70887351]
```

11) *Start the Jenkins.war file*: java -jar jenkins.war

Every day if we want to run the Jenkins we need to run above command.

```
ubuntu@ip-172-31-94-232:~$ java -jar jenkins.war
Running from: /home/ubuntu/jenkins.war
webroot: $user.home/.jenkins
2022-08-28 08:40:27.866+0000 [id=1]     INFO    org.eclipse.jetty.util.log.Log#initialized: Logging initialized @931ms to org.eclipse.jetty.util.log.JavaUtilLog
2022-08-28 08:40:28.127+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction from war file
2022-08-28 08:40:29.984+0000 [id=1]     WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2022-08-28 08:40:30.100+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-9.4.38.v20210224; built: 2021-02-24T20:25:07.675Z; git: 288f3cc74549e8a
 jvm 1.8.0_342-8u342-b07-0ubuntu1~22.04-b07
2022-08-28 08:40:30.764+0000 [id=1]     INFO    o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclipse.jetty.jsp.JettyJspSer
2022-08-28 08:40:30.865+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager workerName=node0
2022-08-28 08:40:30.869+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: No SessionScavenger set, using defaults
2022-08-28 08:40:30.871+0000 [id=1]     INFO    o.e.j.server.session.HouseKeeper#startScavenging: node0 Scavenging every 660000ms
2022-08-28 08:40:31.749+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home directory: /home/ubuntu/.jenkins found at: $user.home/.jenkins
2022-08-28 08:40:32.205+0000 [id=1]     INFO    o.e.j.s.handler.ContextHandler#doStart: Started w.@ec2bf82{Jenkins v2.277.2,/,file:///home/ubuntu/.jenkins/war/,AVAILA
/war}
2022-08-28 08:40:32.270+0000 [id=1]     INFO    o.e.j.server.AbstractConnector#doStart: Started ServerConnector@14bdbc74{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
2022-08-28 08:40:32.270+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started @5337ms
2022-08-28 08:40:32.279+0000 [id=21]    INFO    winstone.Logger#logInternal: Winstone Servlet Engine running: controlPort=disabled
2022-08-28 08:40:34.119+0000 [id=26]    INFO    jenkins.InitReactorRunner$1#onAttained: Started initialization
2022-08-28 08:40:34.184+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2022-08-28 08:40:36.767+0000 [id=26]    INFO    jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2022-08-28 08:40:36.773+0000 [id=26]    INFO    jenkins.InitReactorRunner$1#onAttained: Started all plugins
2022-08-28 08:40:36.813+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2022-08-28 08:40:38.616+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: System config loaded
2022-08-28 08:40:38.617+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: System config adapted
2022-08-28 08:40:38.617+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2022-08-28 08:40:38.618+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2022-08-28 08:40:39.334+0000 [id=40]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Started Download metadata
2022-08-28 08:40:39.355+0000 [id=40]    INFO    hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2022-08-28 08:40:39.475+0000 [id=26]    INFO    jenkins.install.SetupWizard#init:

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

21669fdd14854a558414a169e10b7ad6

This may also be found at: /home/ubuntu/.jenkins/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************
```

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

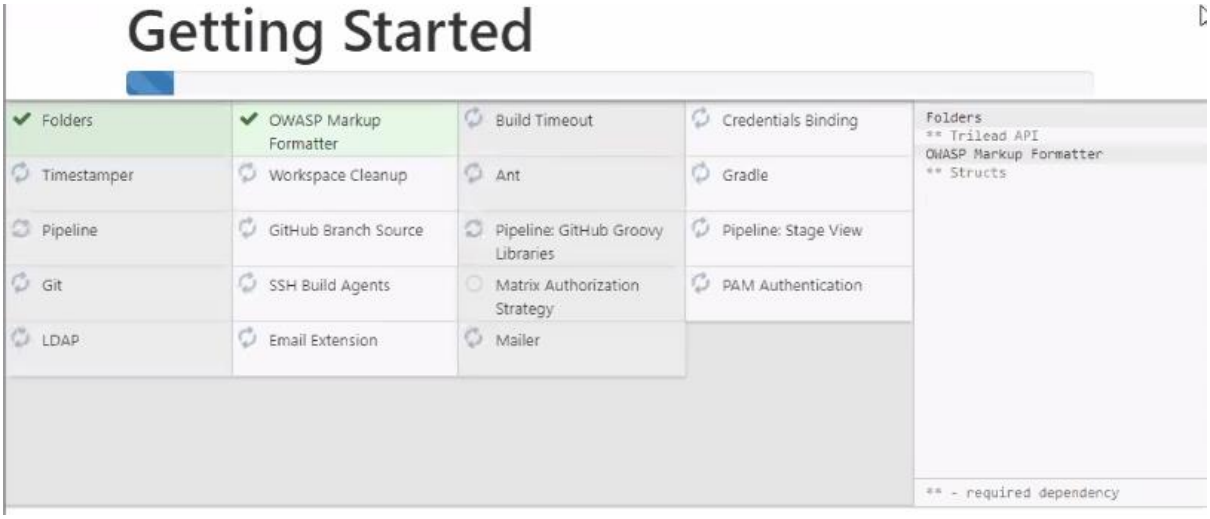
When we are installing jenkins it will automatically give you the password in the github terminal.

Copy the password and paste the browser.

```
Jenkins initial setup is required. An admin user has been created and a passwo
 generated.
Please use the following password to proceed to installation:

49e8e03581974cedaa46ec8545a481ed

This may also be found at: /home/ubuntu/.jenkins/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

2021-05-21 05:21:33.646+0000 [id=40]    INFO    h.m.DownloadService$Downloadab
#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2021-05-21 05:21:33.646+0000 [id=40]    INFO    hudson.util.Retrier#start: Per
rmed the action check updates server successfully at the attempt #1
2021-05-21 05:21:33.649+0000 [id=40]    INFO    hudson.model.AsyncPeriodicWork
ambda$doRun$0: Finished Download metadata. 23,475 ms
2021-05-21 05:21:35.574+0000 [id=26]    INFO    jenkins.InitReactorRunner$1#on
tained: Completed initialization
2021-05-21 05:21:35.595+0000 [id=20]    INFO    hudson.WebAppMain$3#run: Jenki
 is fully up and running
```

You will get the password on the step 11

14) Press Install Suggested Plugins

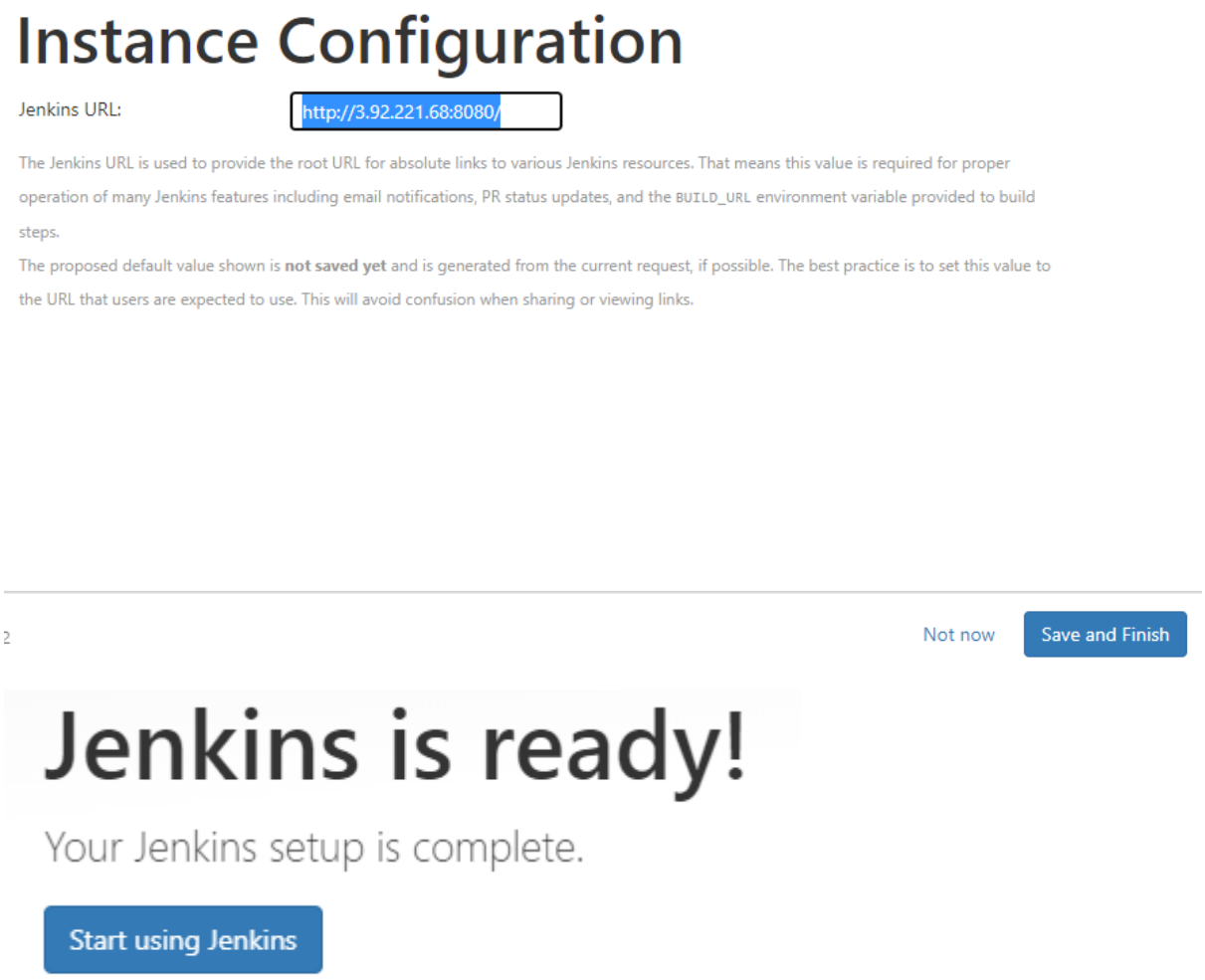


15) Create First admin user

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

Click on save and continue.

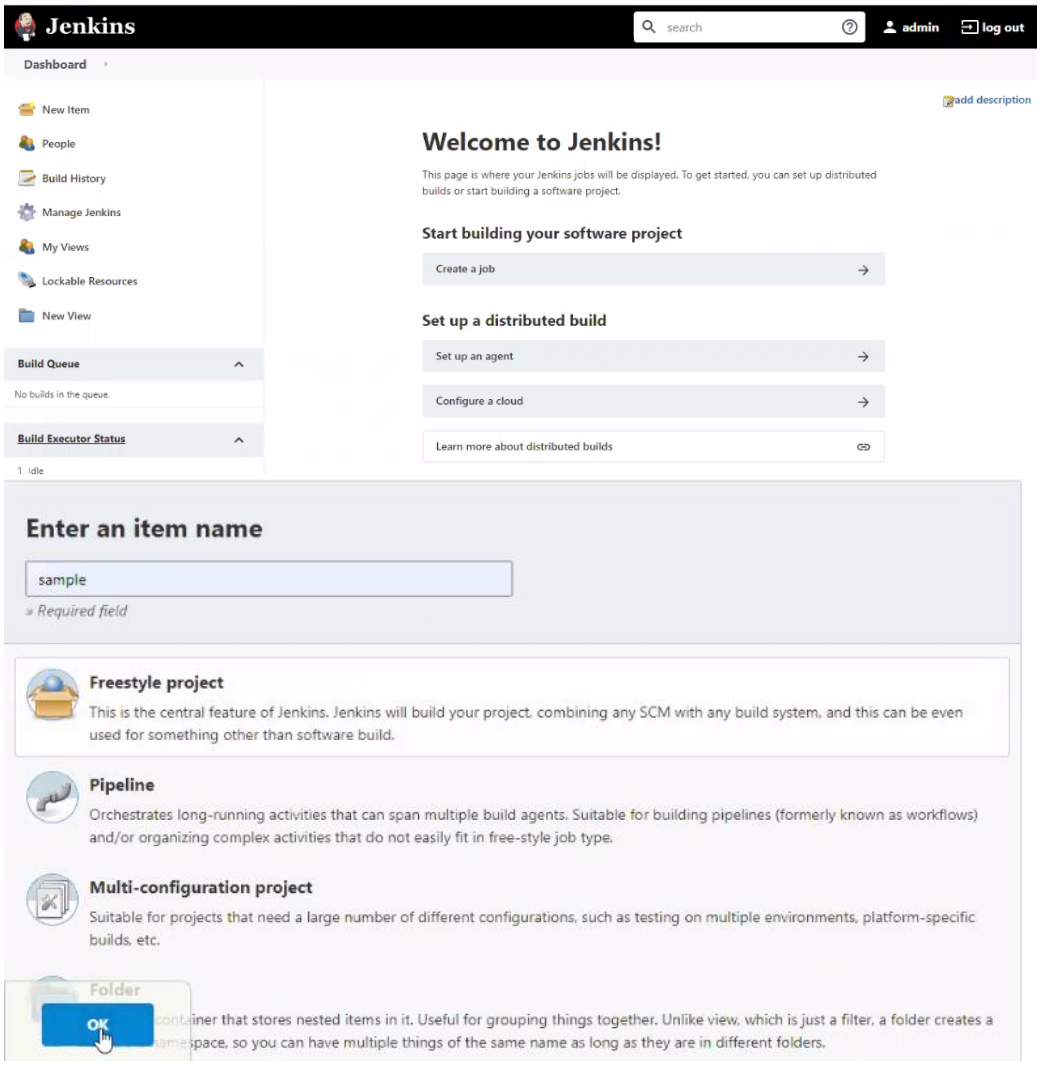
Click on save and finish



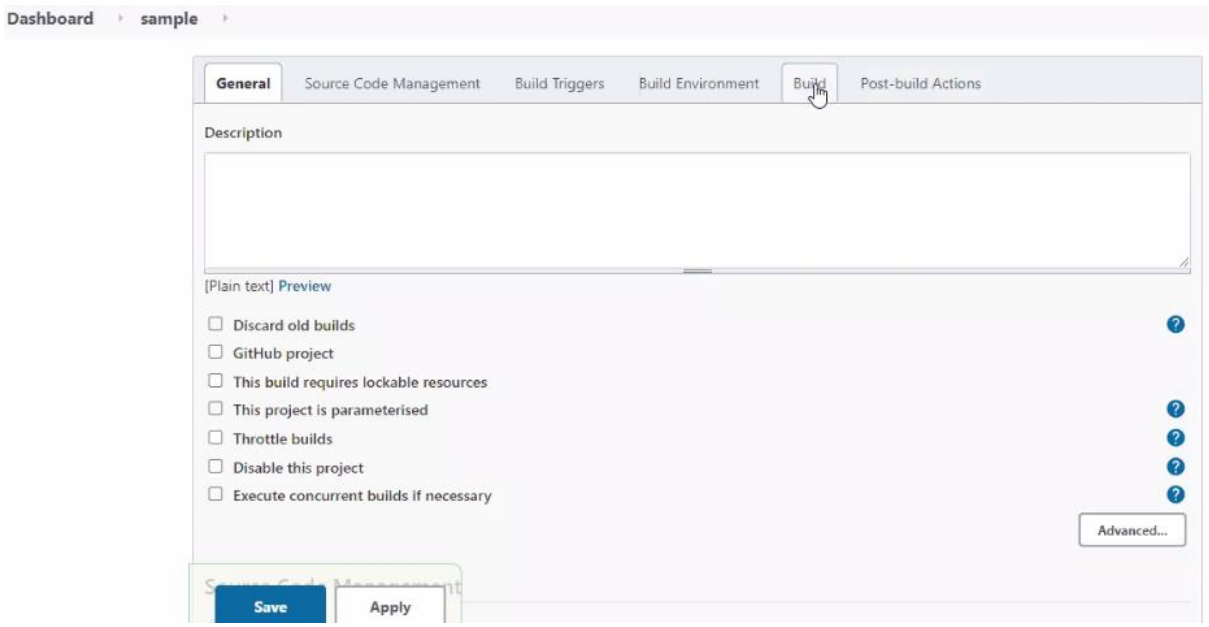
++

Create Sample Job in Jenkins:

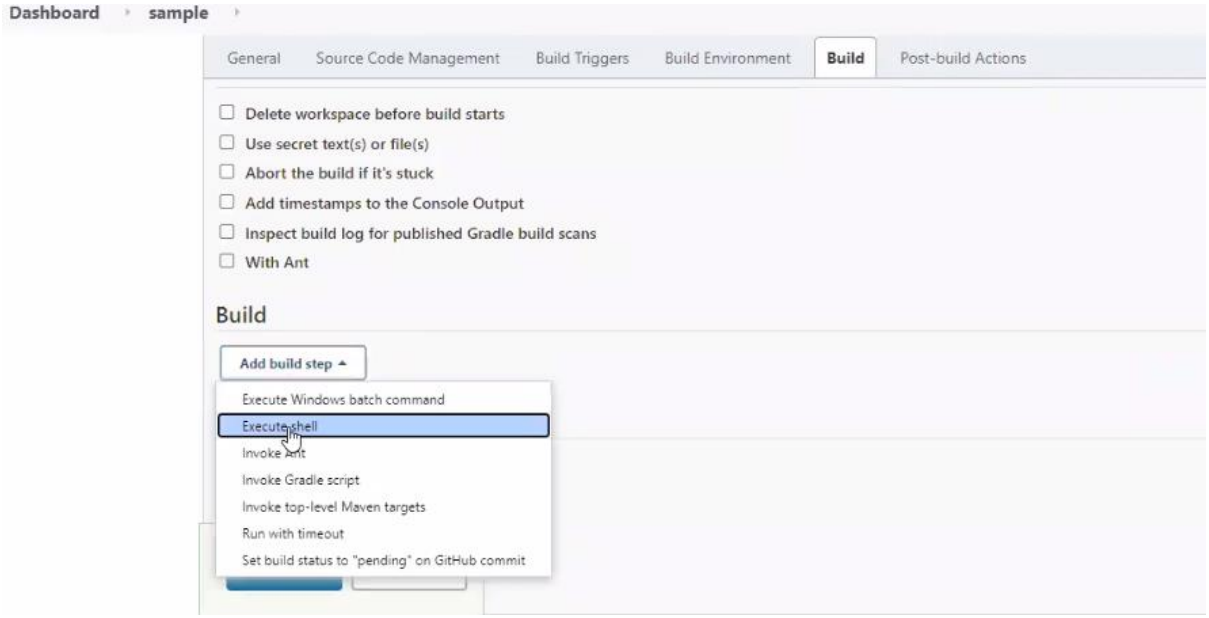
Click on new item or



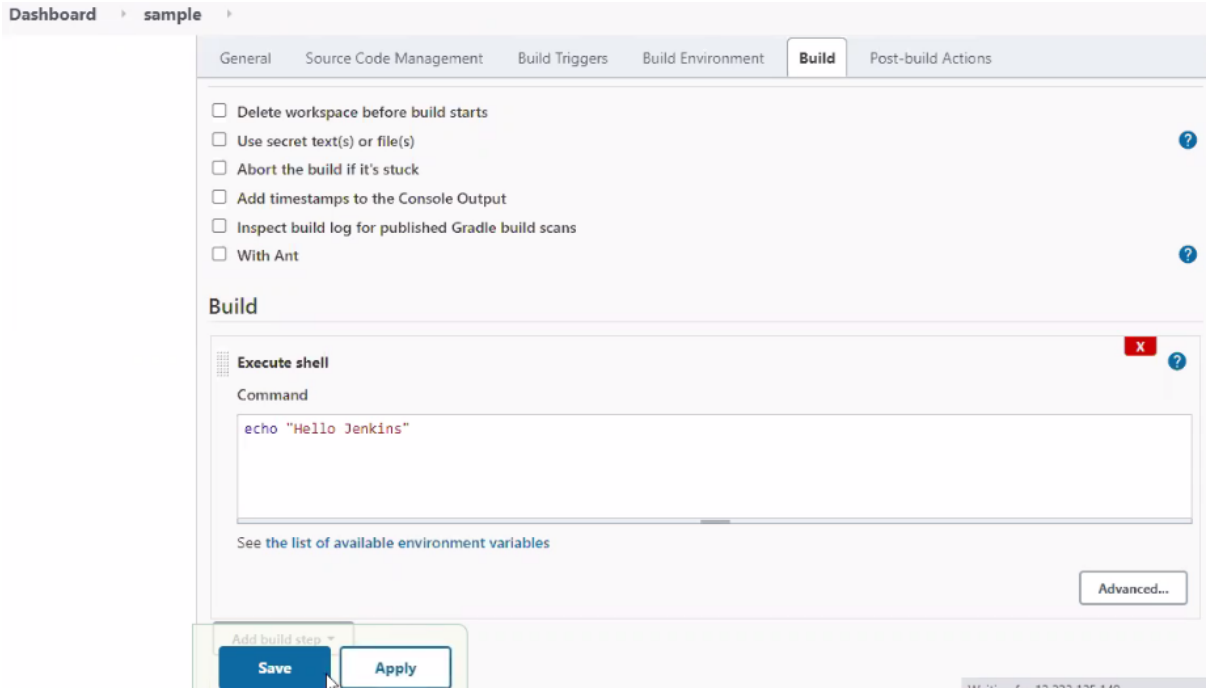
Build tab

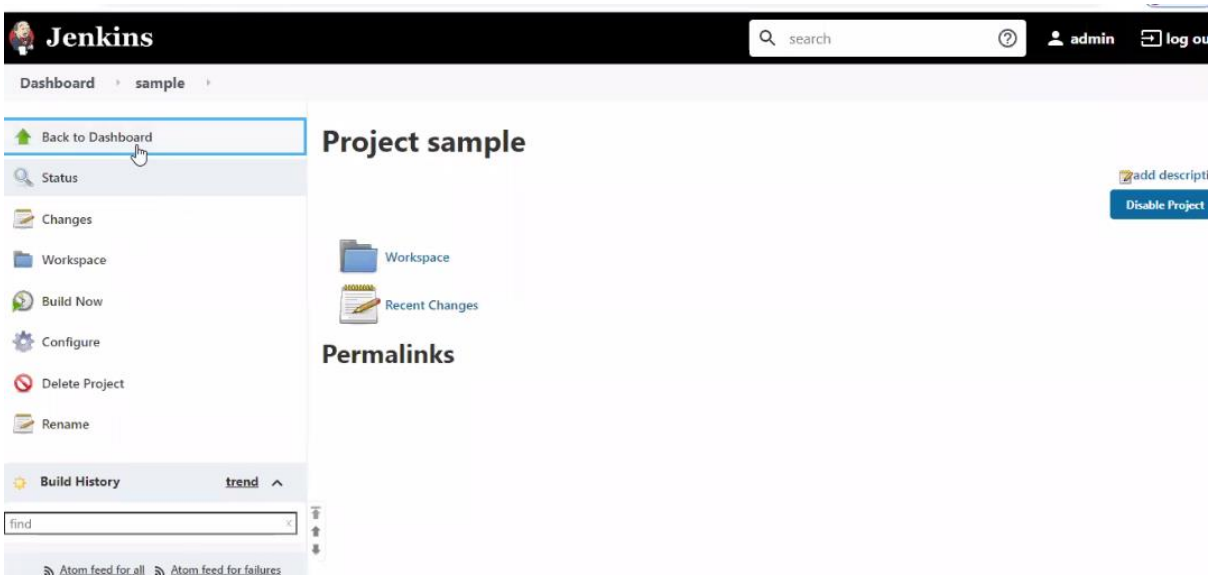


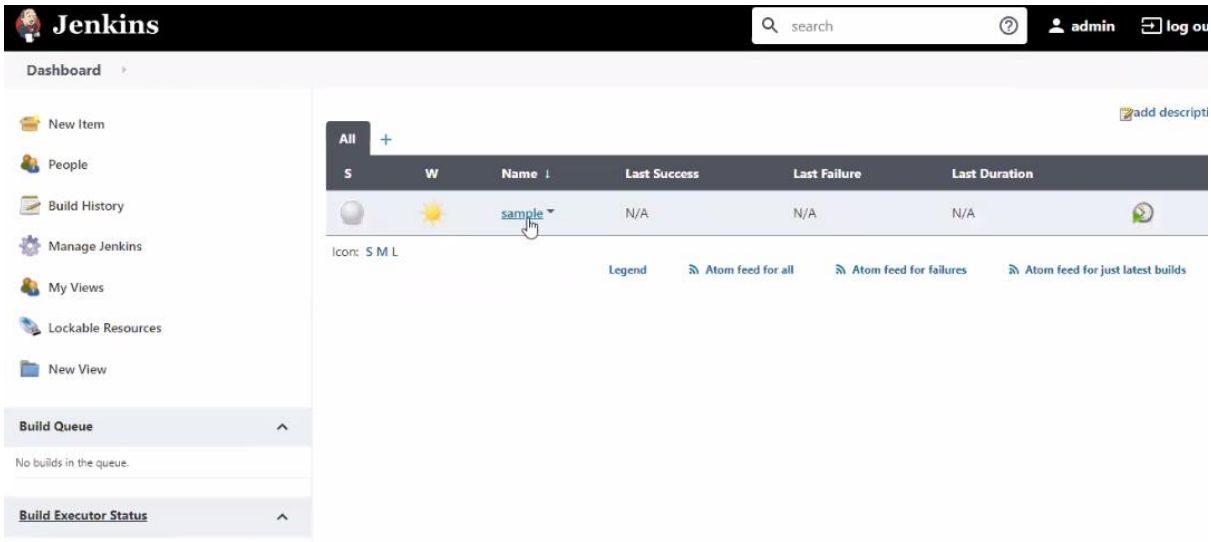
Click on execute Shell

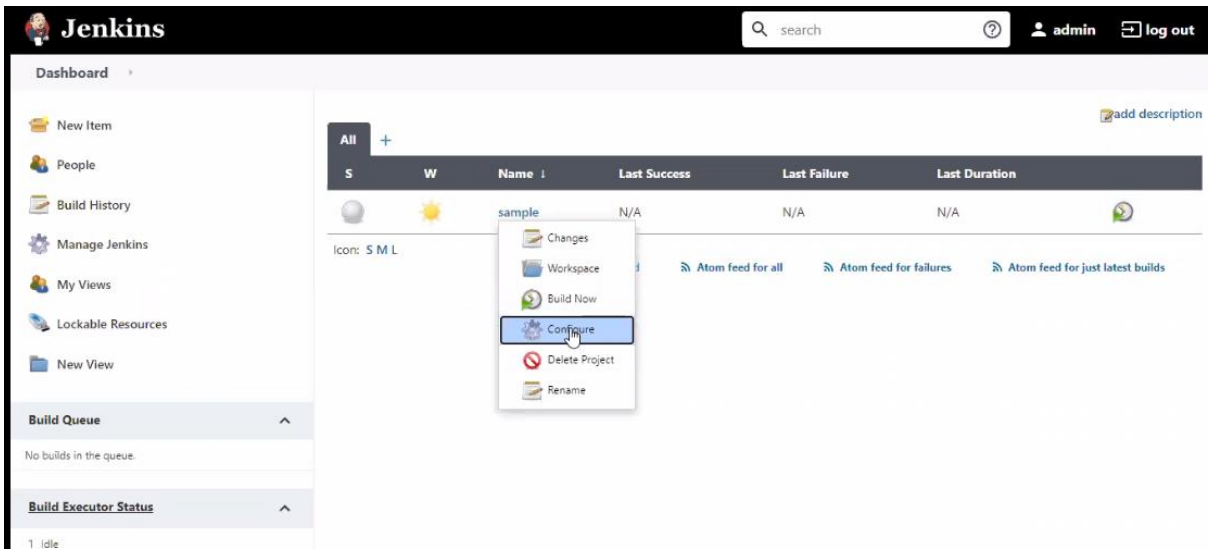


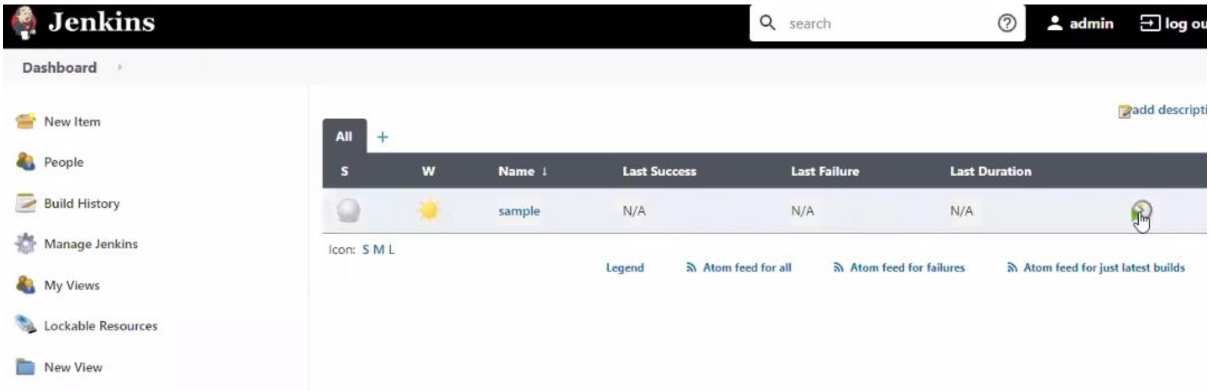
In Command Box Enter echo “Hello Jenkins"



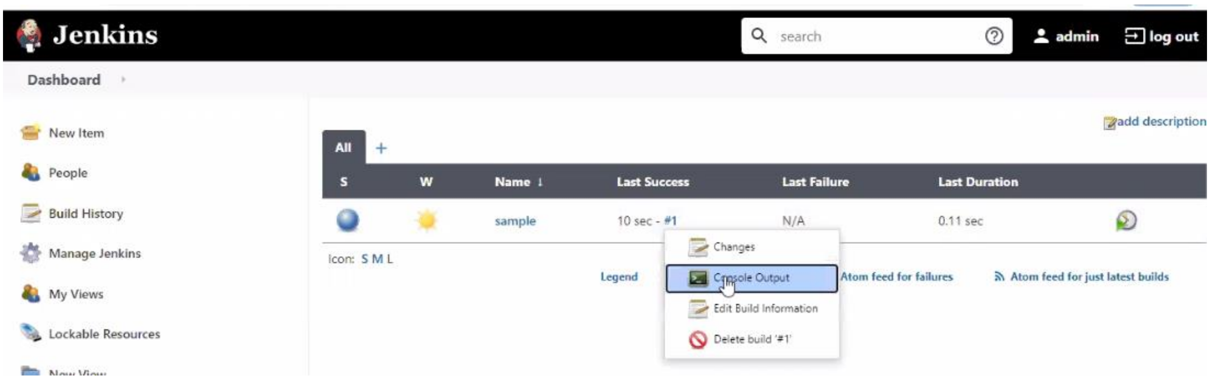

Jenkins
search
admin
log ou
Dashboard
sample
Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Project
Rename
Build History
trend
find
Atom feed for all
Atom feed for failures
Project sample
add descripti
Disable Project
Workspace
Recent Changes
Permalinks

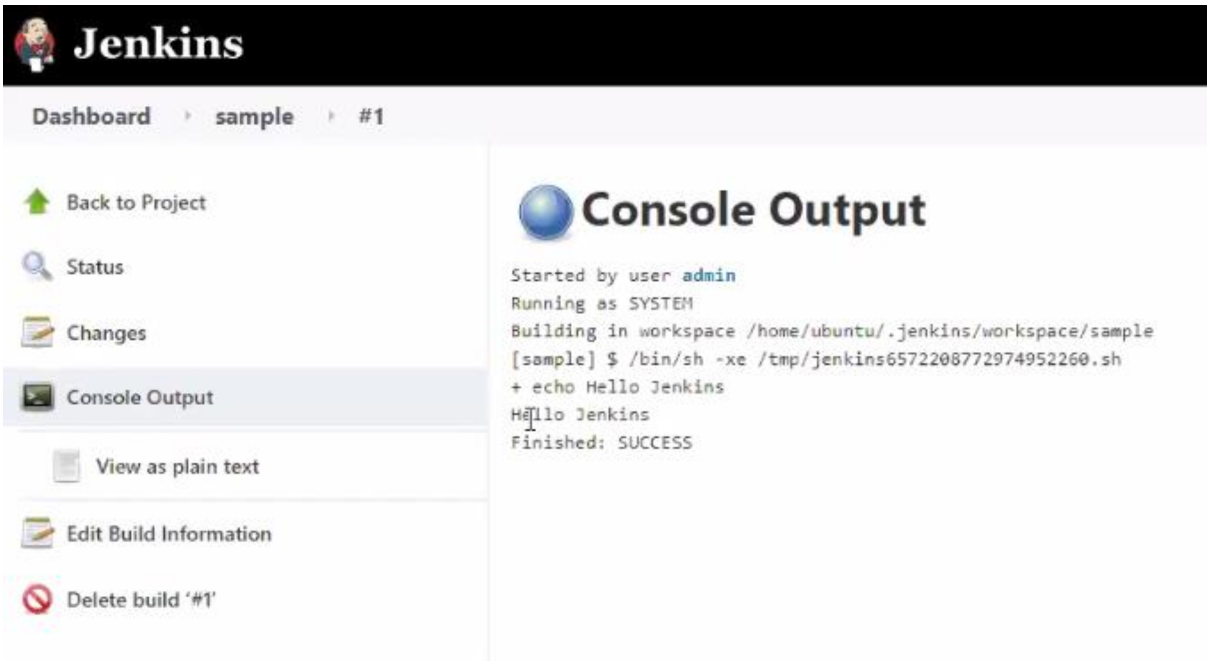

Jenkins
search
admin
log ou
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Lockable Resources
New View
Build Queue
No builds in the queue.
Build Executor Status
add descripti
All
S
W
Name
Last Success
Last Failure
Last Duration
sample
N/A
N/A
N/A
Icon: S M L
Legend
Atom feed for all
Atom feed for failures
Atom feed for just latest builds


Jenkins
search
admin
log out
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Lockable Resources
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
add description
All
S
W
Name
Last Success
Last Failure
Last Duration
sample
N/A
N/A
N/A
Icon: S M L
Changes
Workspace
Build Now
Configure
Delete Project
Rename
Atom feed for all
Atom feed for failures
Atom feed for just latest builds



Click on Console Output





++

Install TOMCAT in QA & Production Server

1) Select QA Server and press connect on AWS

2) Copy the SSH Command

3) Open GIT Bash & paste the SSH Command

Press Yes

4) Update the apt repository

sudo apt-get update **sudo apt install openjdk-8-jdk -y**

5) Install tomcat9

sudo apt-get install -y tomcat9

After this we need to install one more package

sudo apt-get install -y tomcat9-admin

6) Check the tomcat is install 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/*

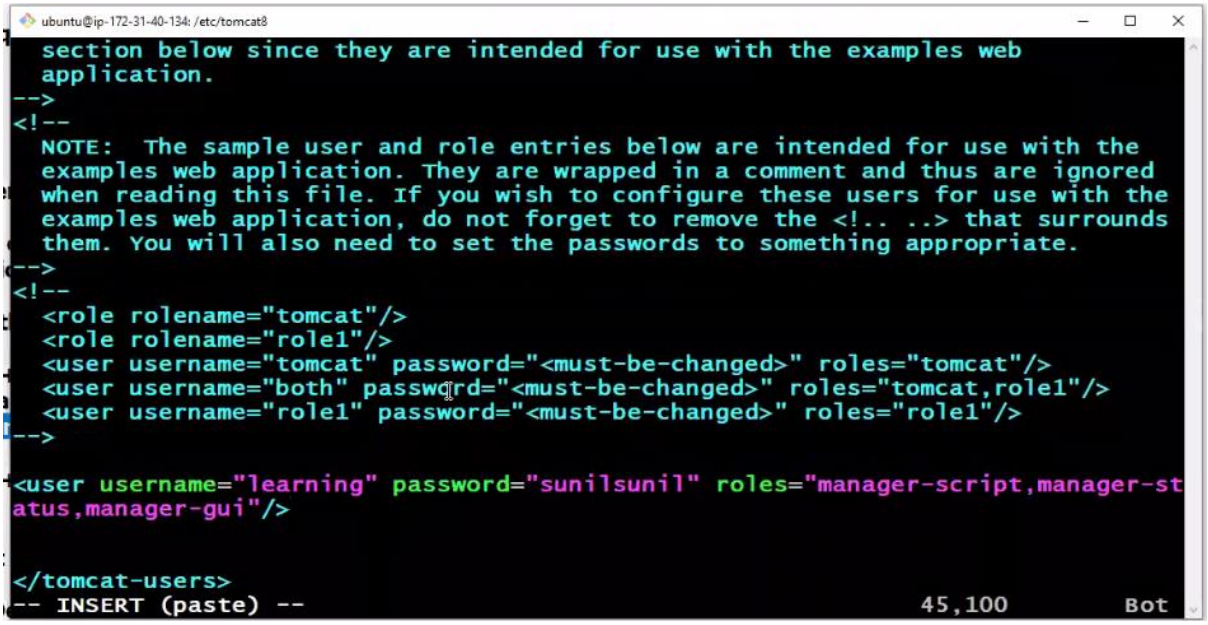
8) enter linux command in QA Server - ls

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



save and quit

press esc

type :wq

press enter

12) Whenever, 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 on jenkins

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 artefact (.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 Dashboard

Click on manage jenkins

Click on manage plugins

Click on available 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

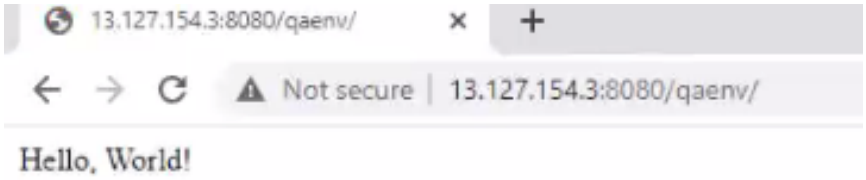
http://172.31.34.103:8080

26) Click on apply and save

27) Run the job

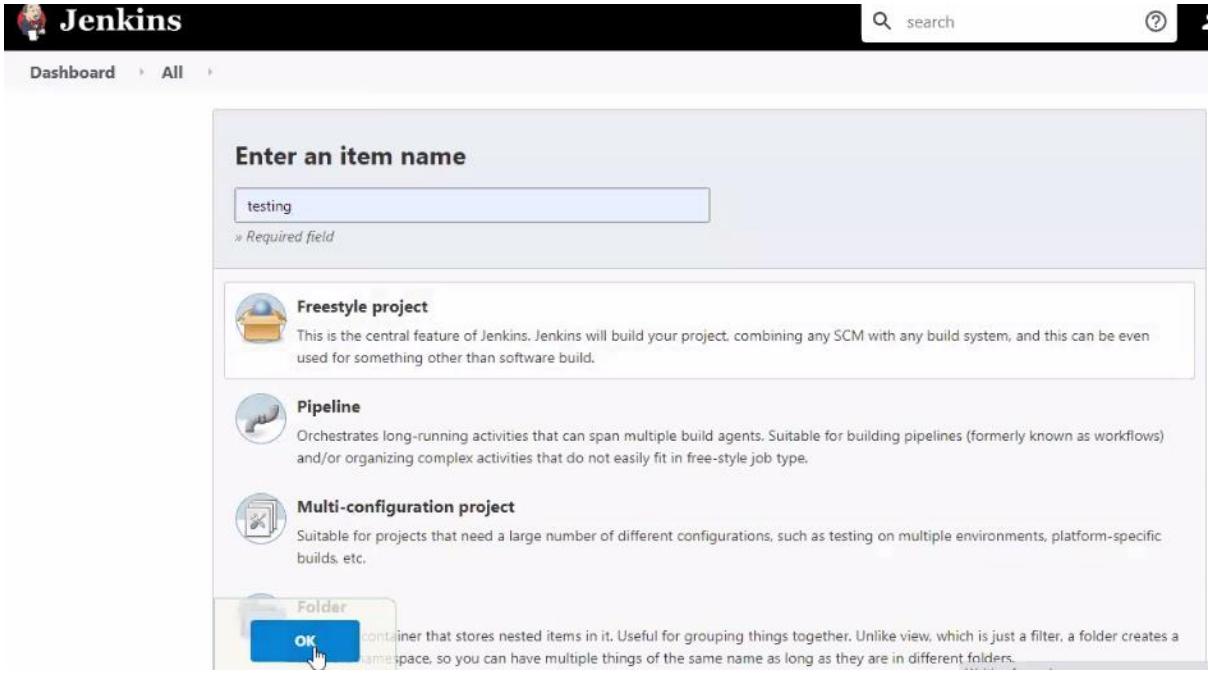
28) To access the home page

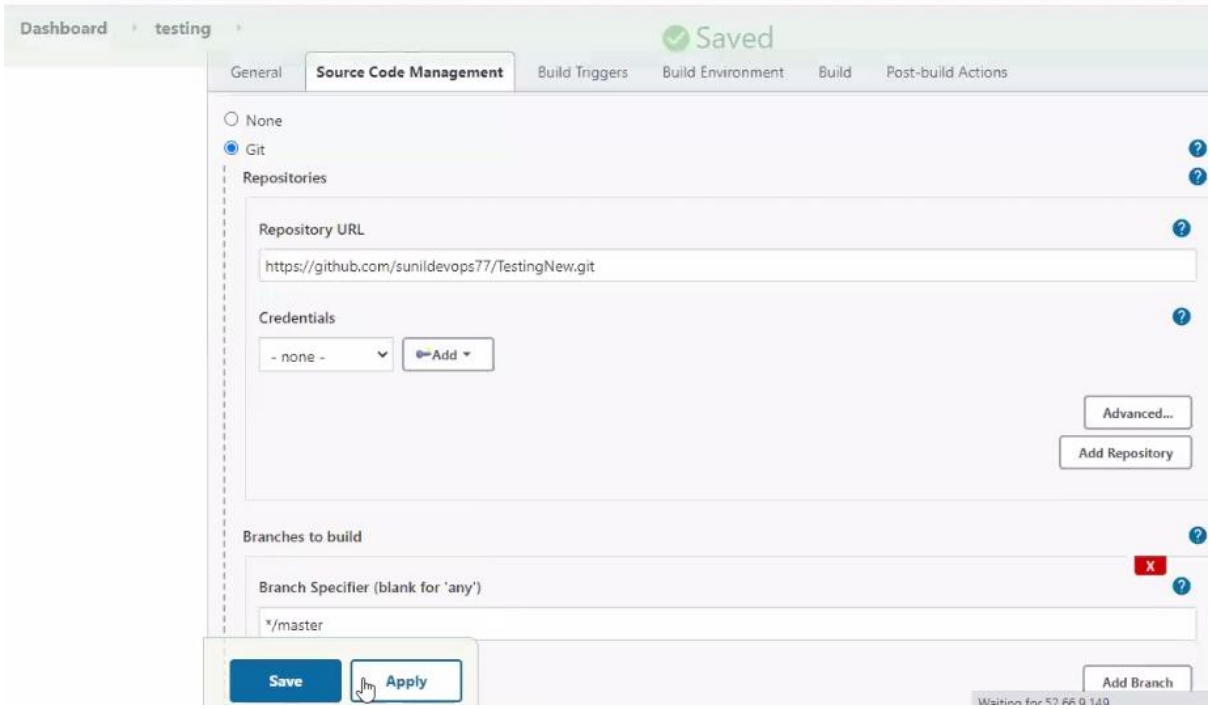
public_ip_Qa_server:8080/qaenv



++

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





Step 1: Connect to Dev server from git bash

Step 2: Start Jenkins (java -jar Jenkins.War)

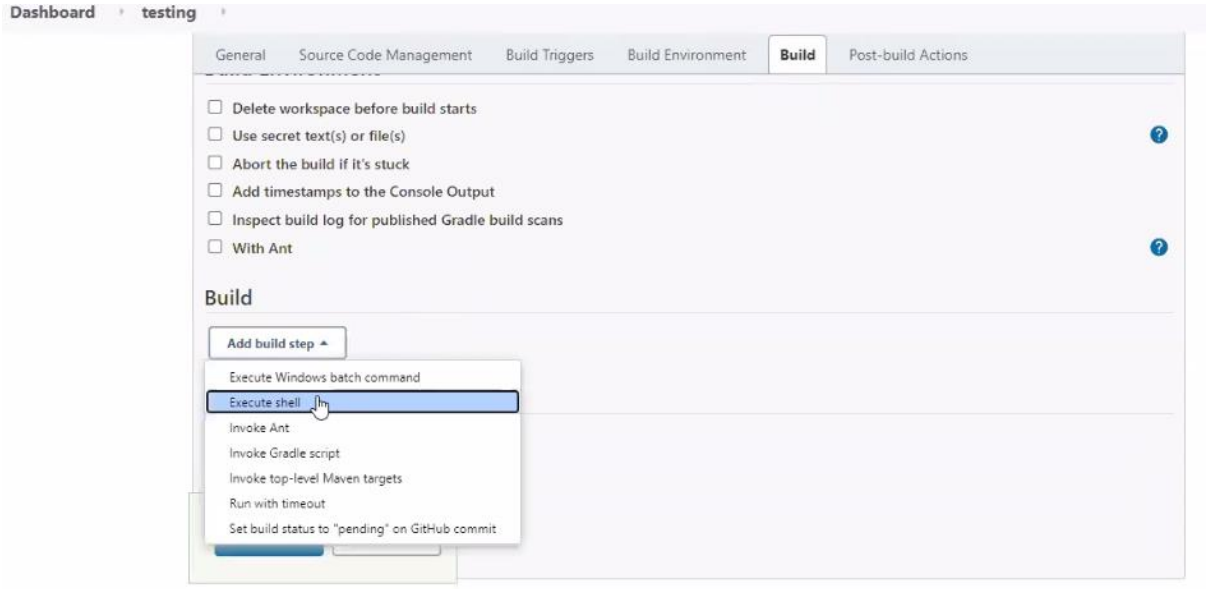
Step 3: Create new item (Name - testing)

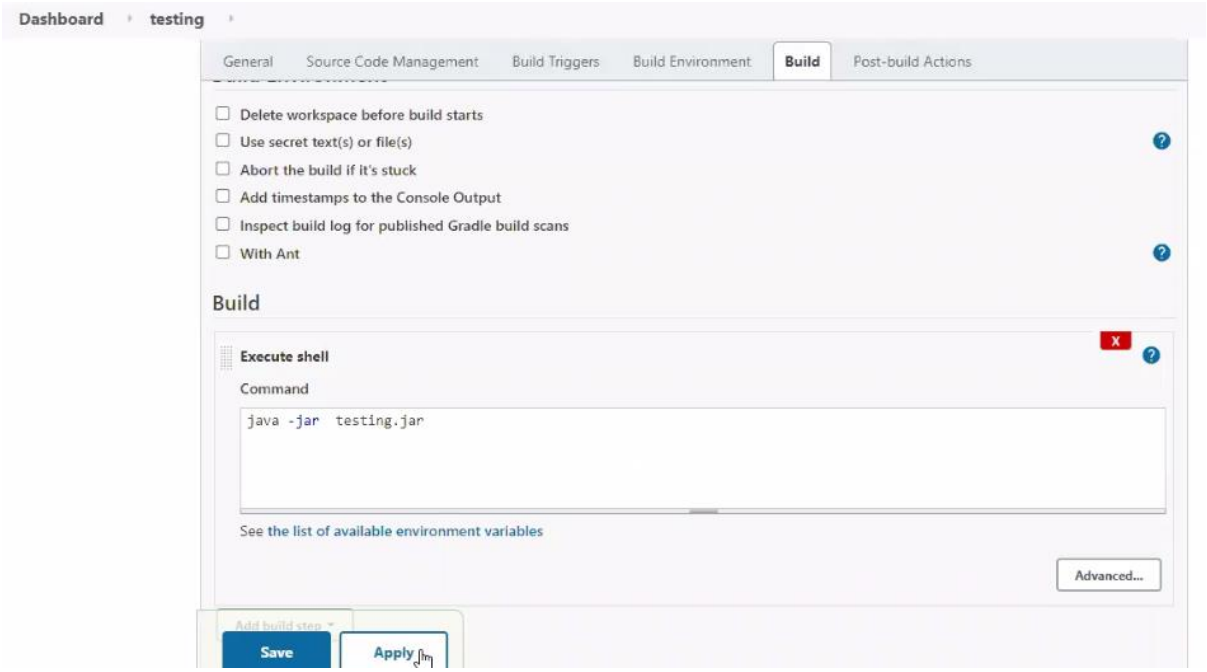
Source code management tab, Git

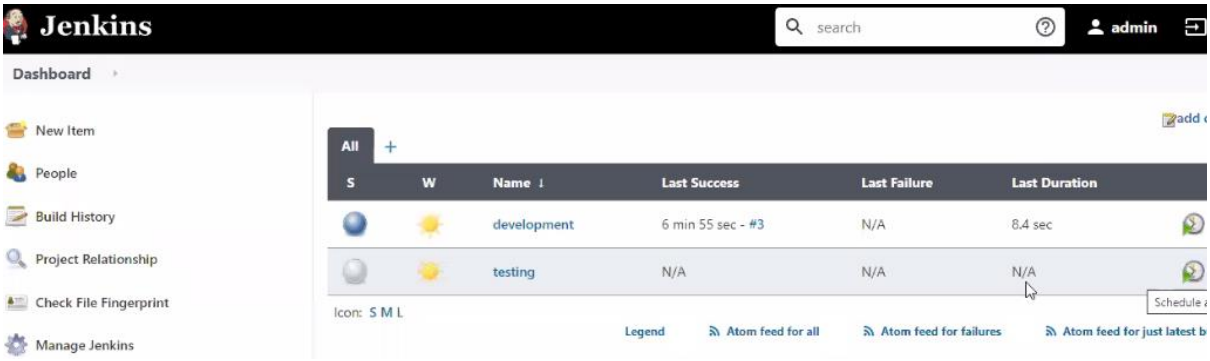
Repository URL - https://github.com/sunildevops77/TestingNew.git

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"

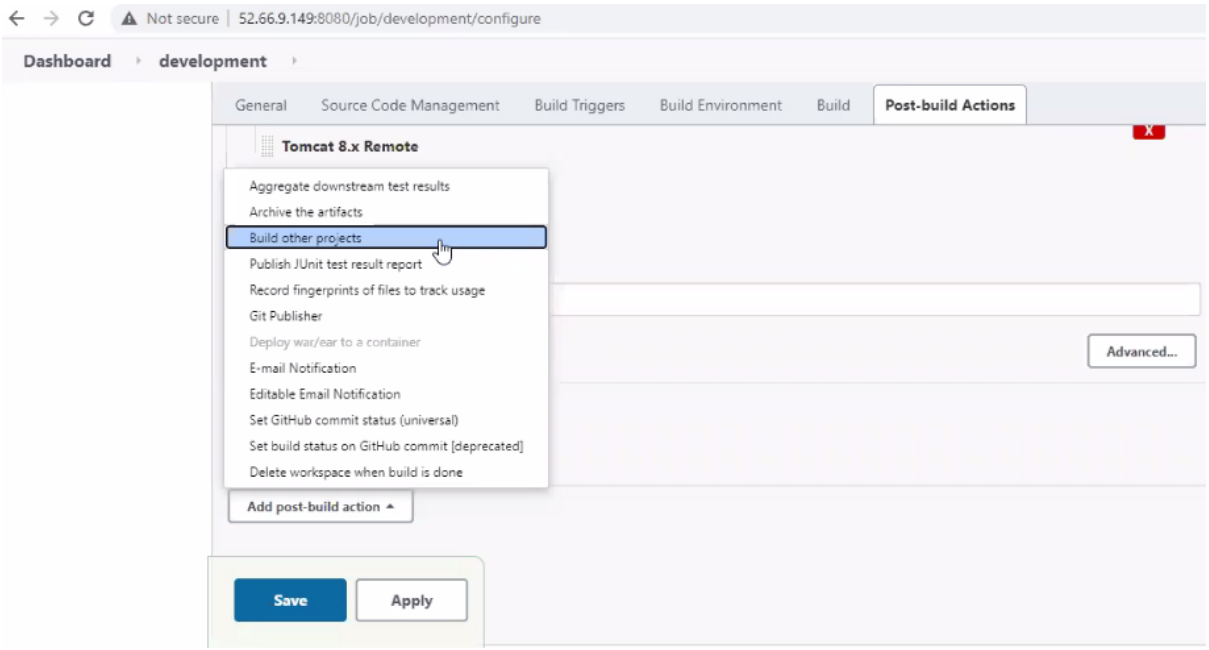
Now both are independent job.

Step:4

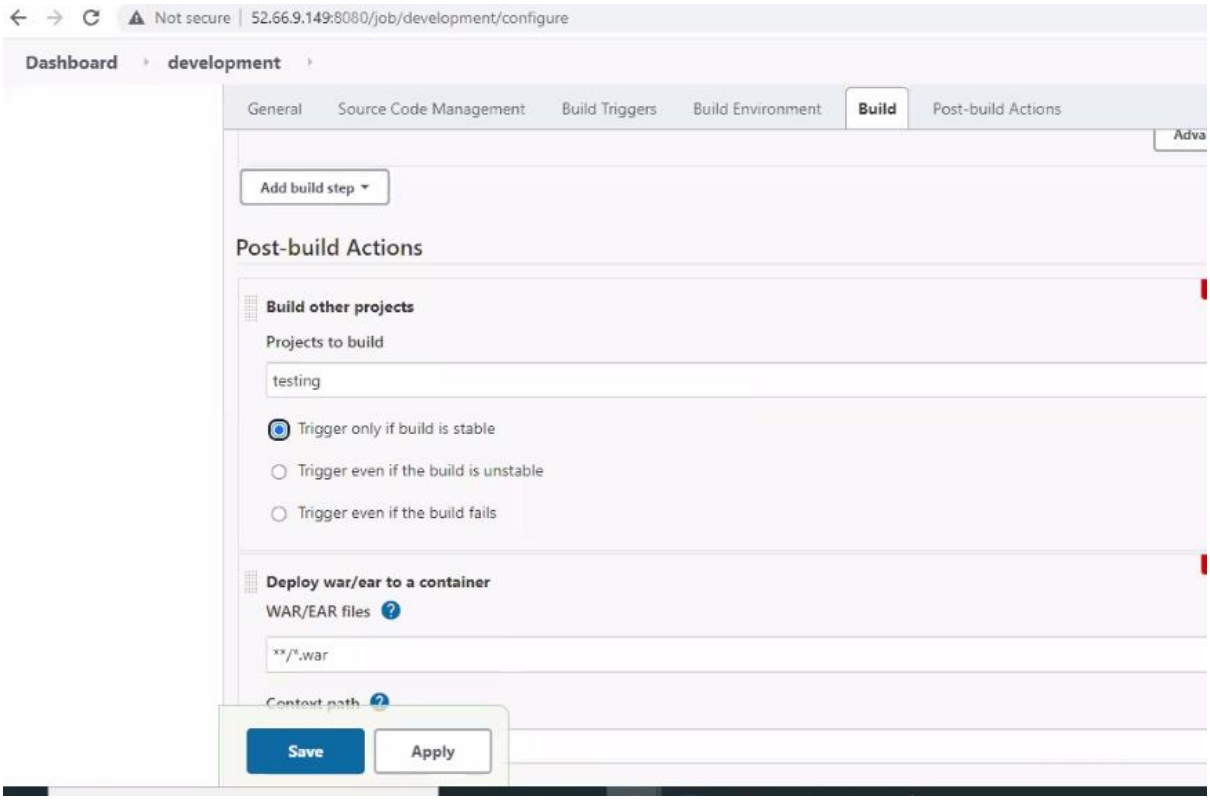
To call testing job to link for auto calling after development job is completed

Go to first job (Development) - configure

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



Projects to build - testing (name of the job)



Once you run the Development job the testing job triggered automatically because we have linked the both as shown above.

++

Copying artefacts from development job to testing job

The artefacts (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: Continuous 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 artefacts

6) Enter **/*.war

7) Click on apply and save

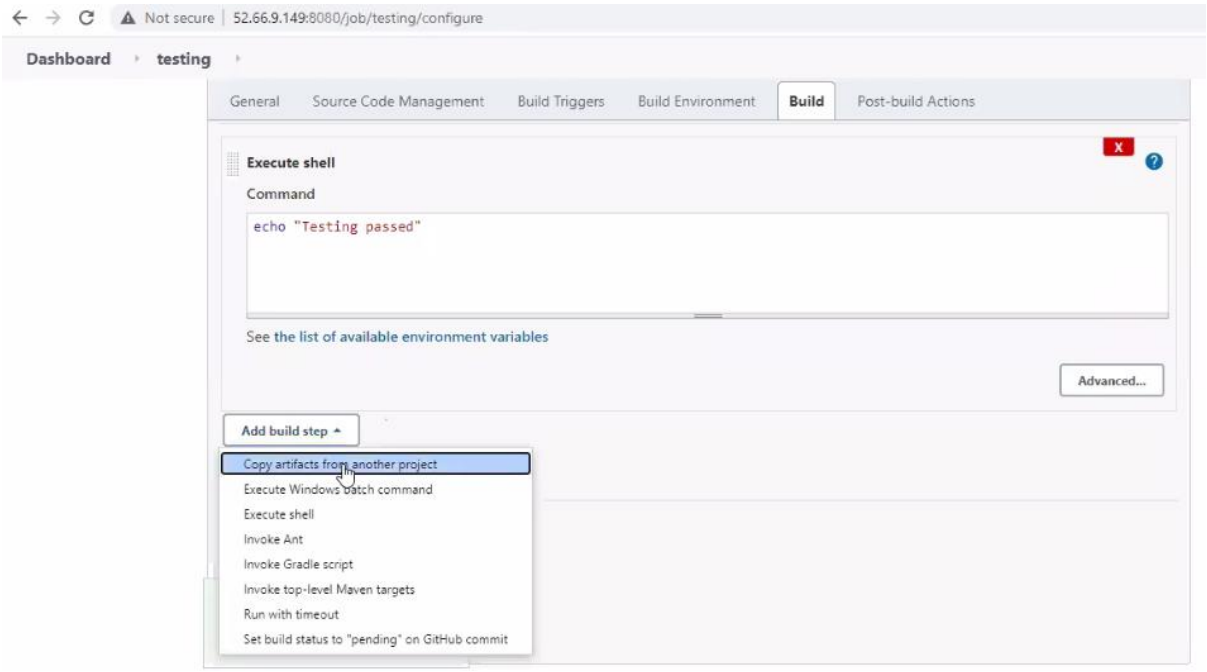
8) Go to testing Job

9) Click on configure

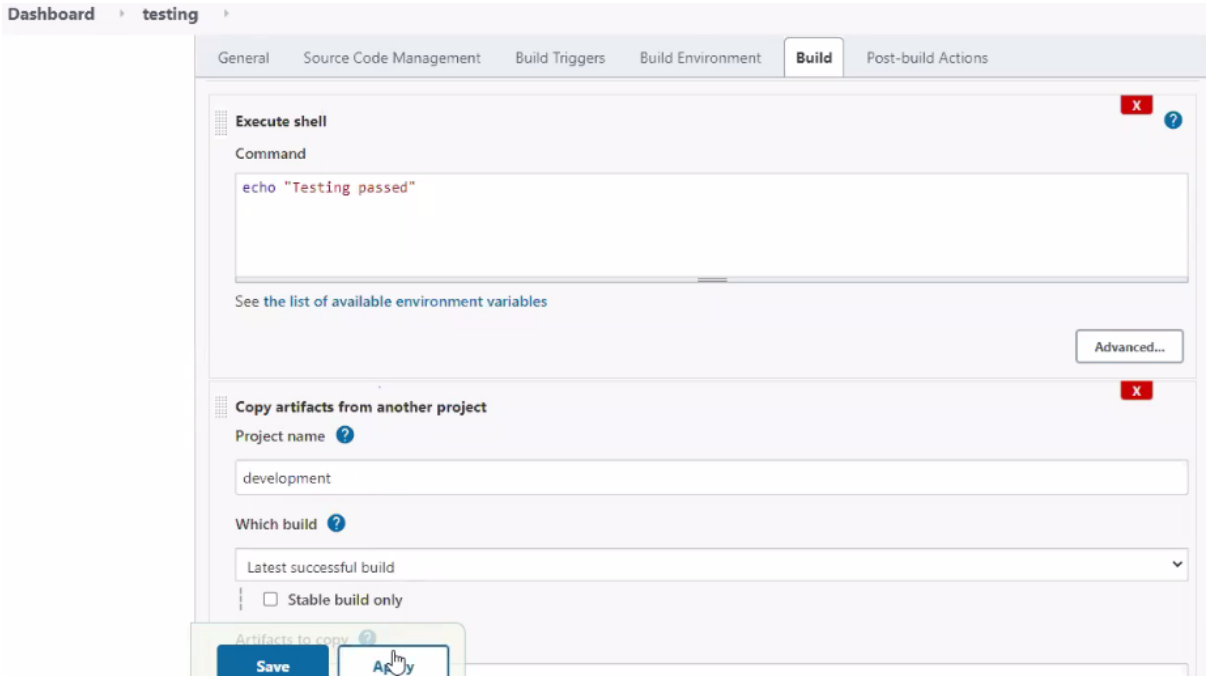
10) Go to Build section

11) Click on add build steps

12) Click on copy artifacts from another project

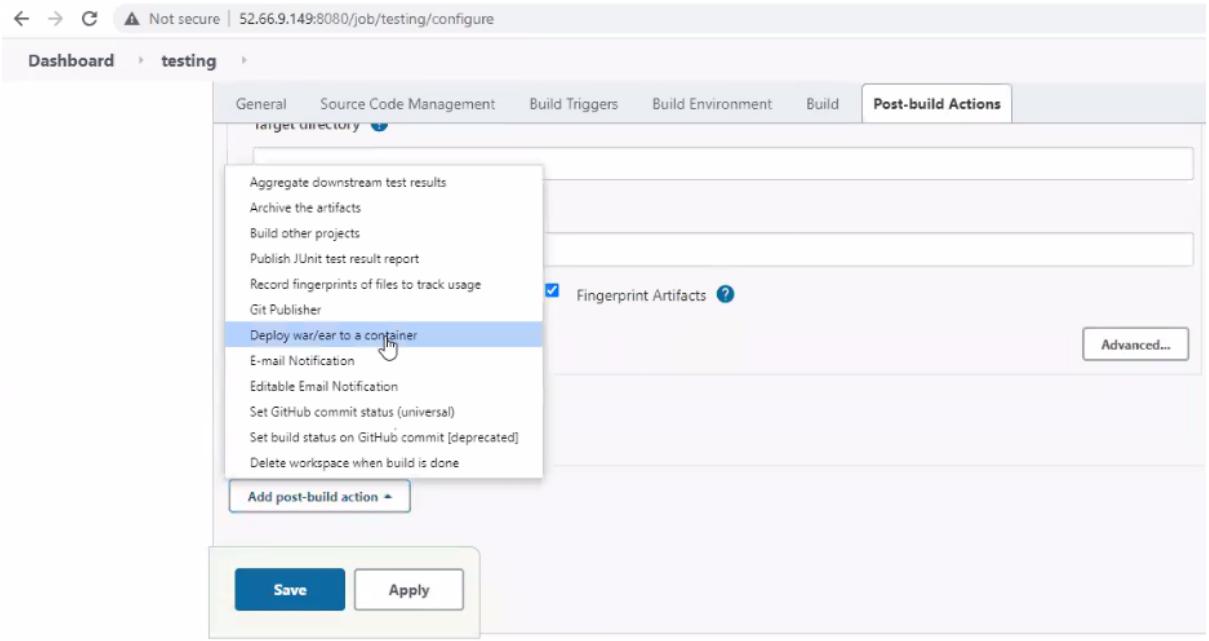


13) Enter Development as project name



14) For Deployment Go to Post build actions section

15) Click on add post build action

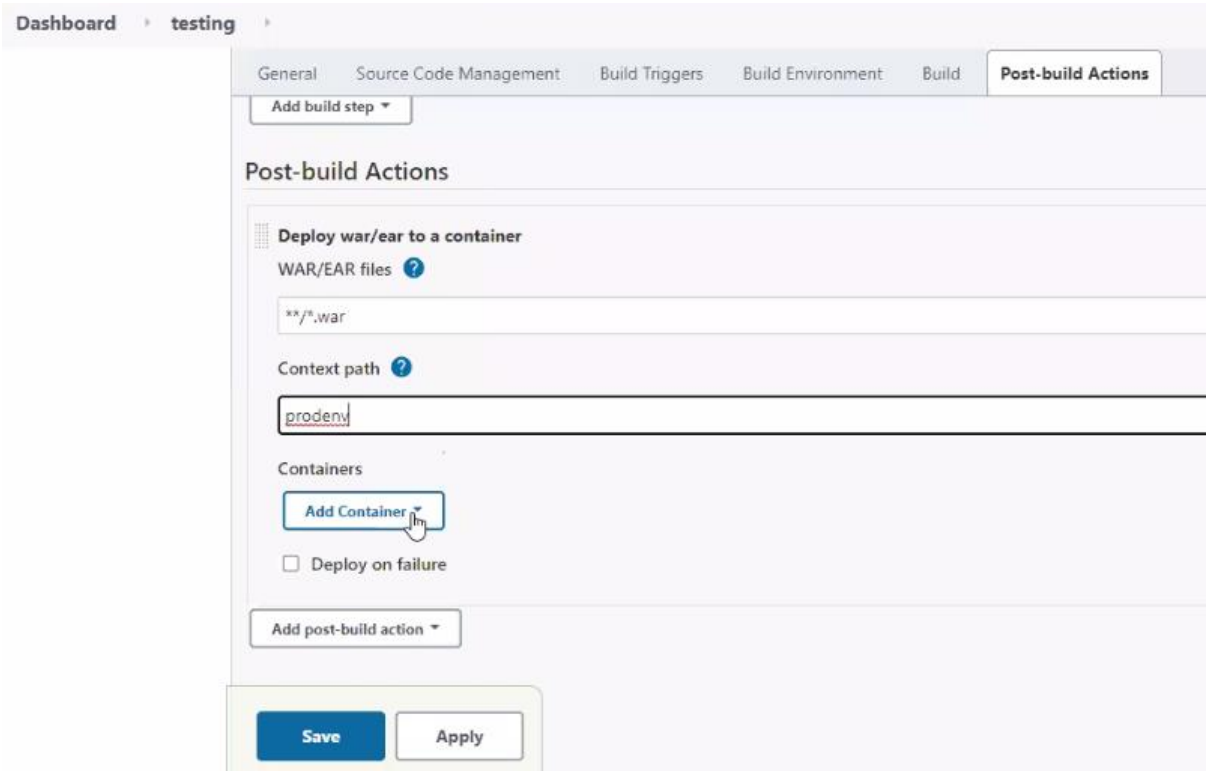


16) Click on deploy war/ear to a container

17) Enter **/*.war in war/ear files

18) Context path: prodenv

19) Click on add container

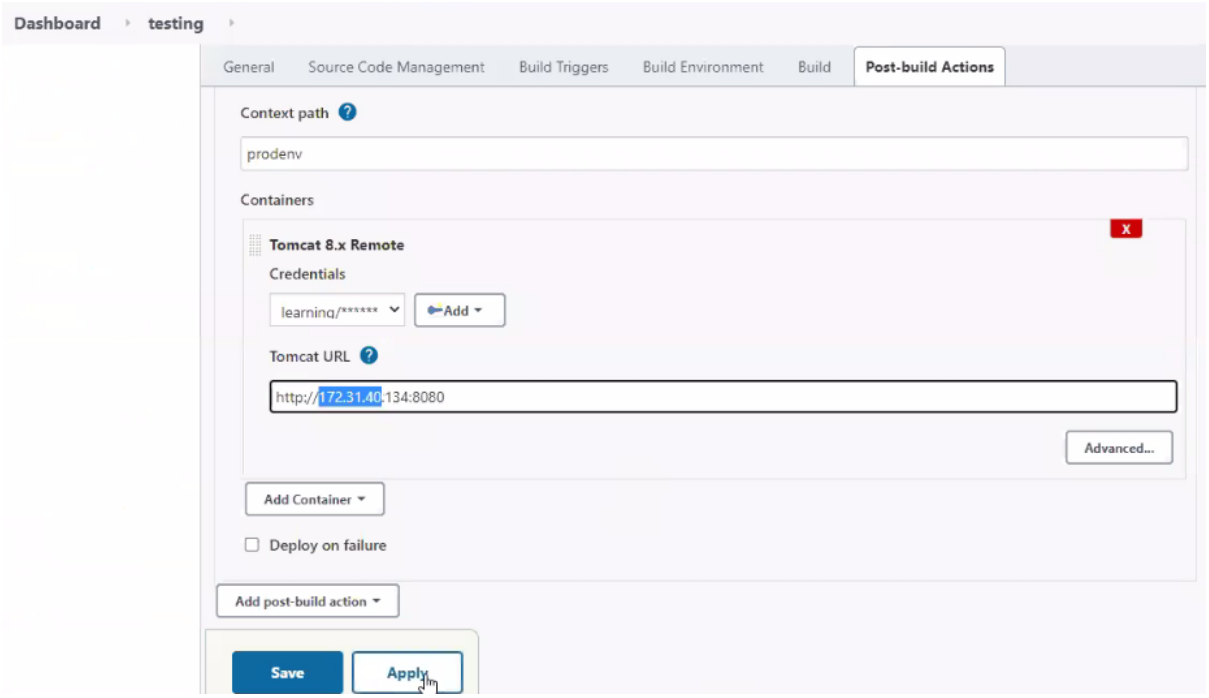


20) Select tomcat 8

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

++

26th May

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 assigning

==========

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 pattern as Dev.* (i.e. 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 and 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

http://13.233.127.59:8080/saferestart-- to restart in Jenkins idle state when the jobss are running

++

27th May

Master - Slave configuration

Same version of java should exist in master and slave (Dev1 and Dev2 on).

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 who is the 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

Let's 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)-→enter until get the key

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 (whenever install Jenkins slave.jar will be downloaded)

Take private ip of the dev server

Run below command in slave machine (start java –jar Jenkins.war in dev)

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

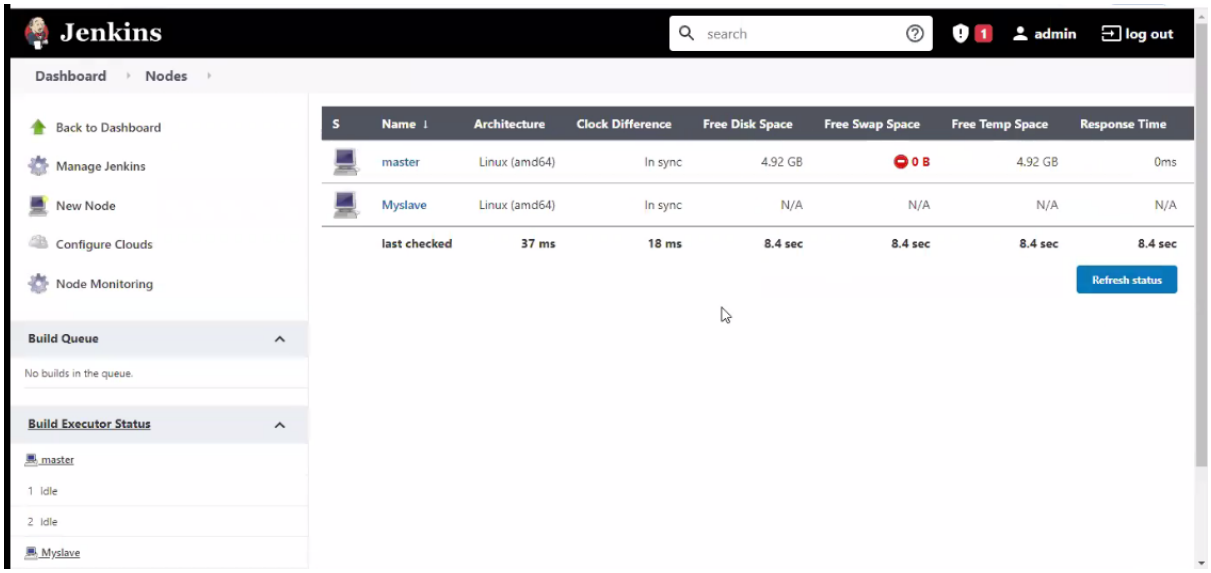
Select Launch agent via execution of command on the controller

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

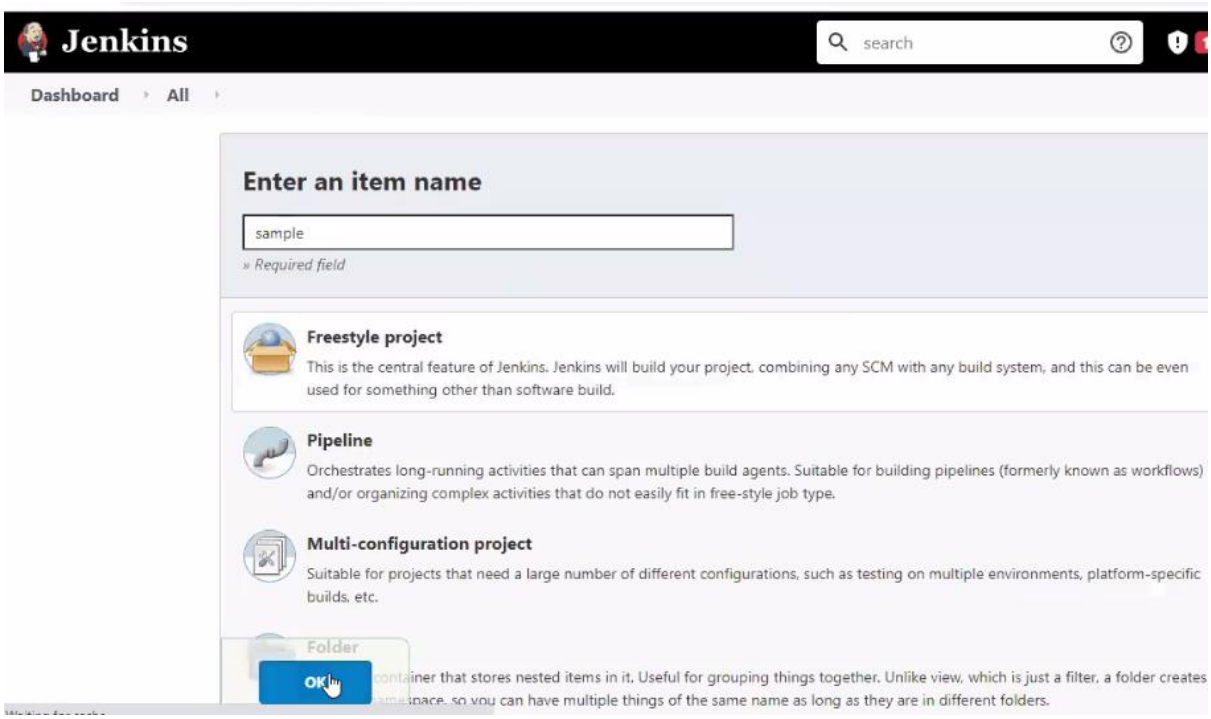


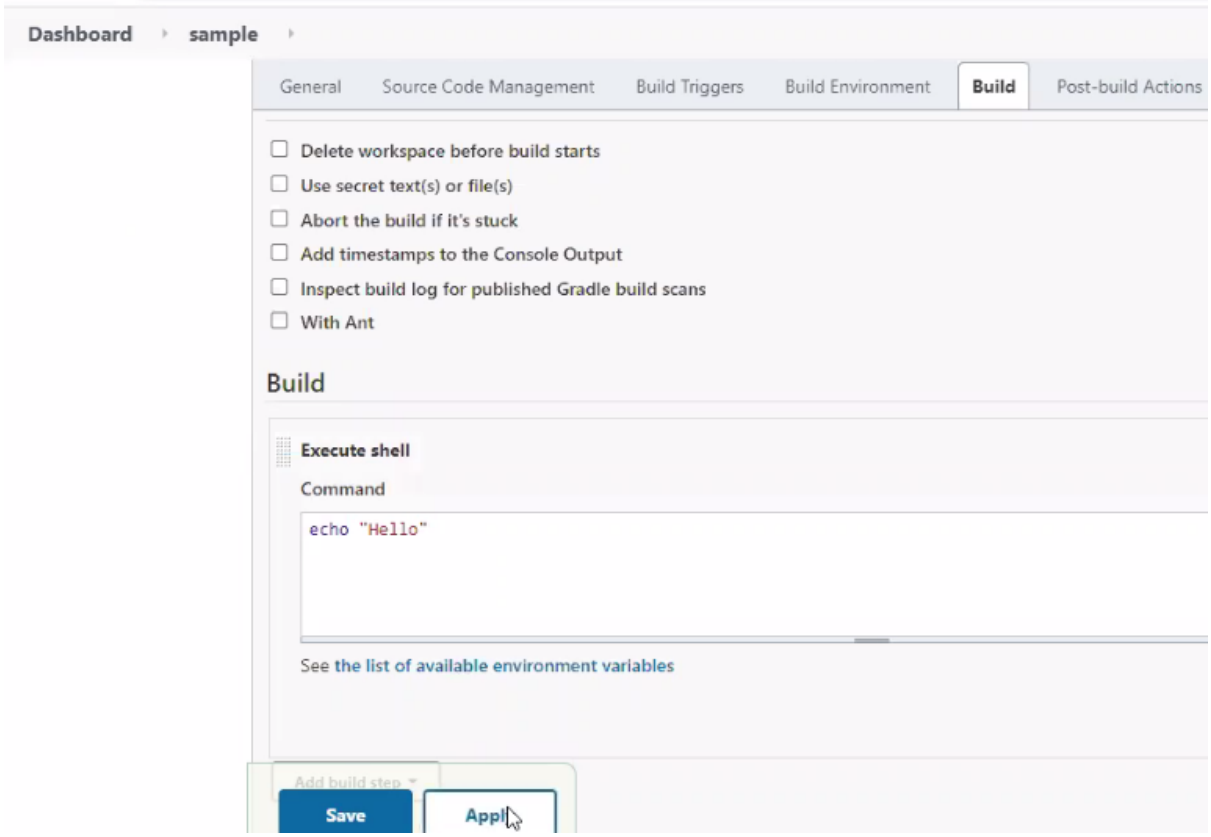
++

Configure job to run on slave

---

14) Select Testing Job

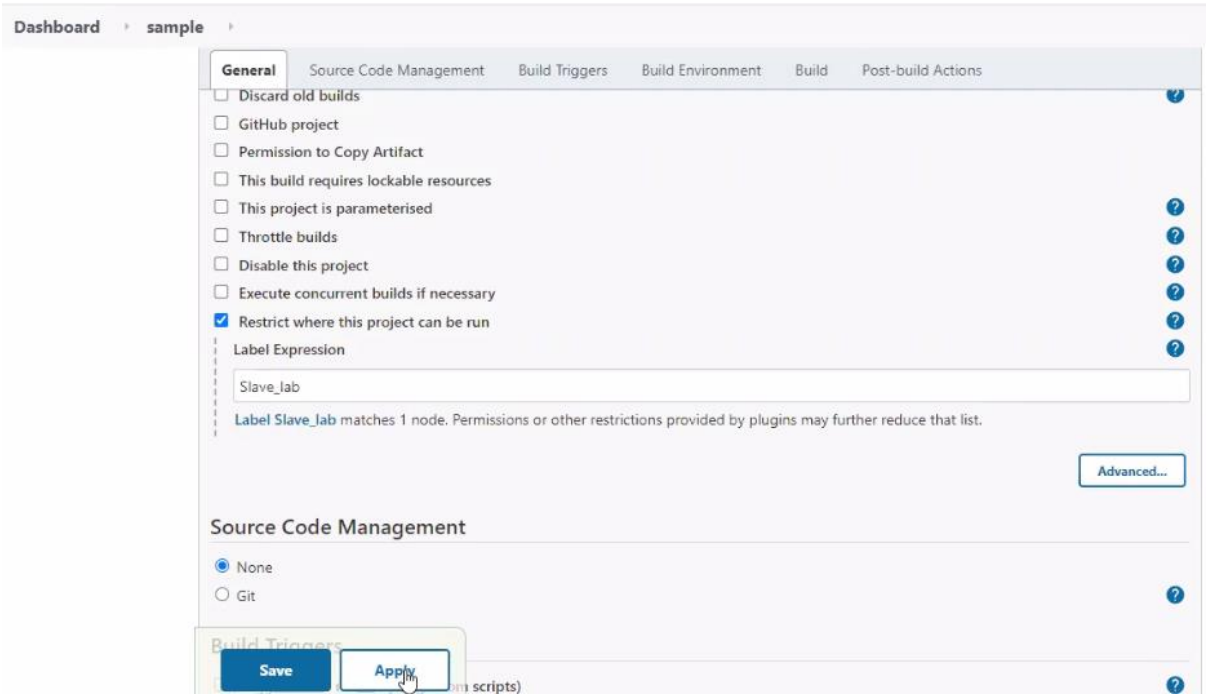




15) Go to Configure --> General Tab

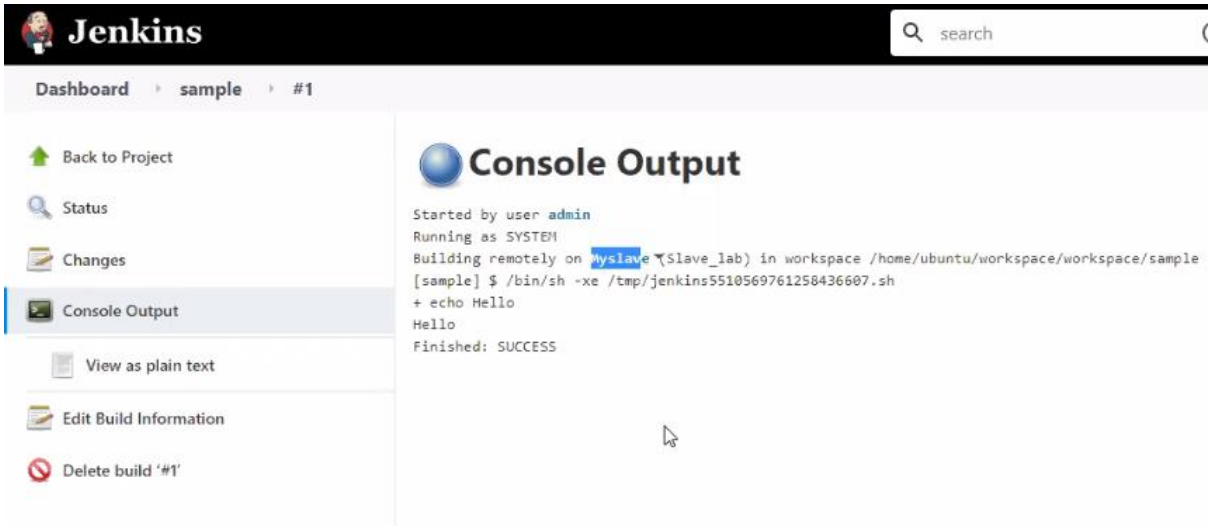
17) Check Restrict where this project can be run (then the load will be on slave and the job will run on salve with salve hardware)

18) Enter Label Expression (Slave_lab) ()



Apply ---> Save

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



++

++

31st May

PIPELINE

Implementing CI-CD from the level of code.

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

Advantages:

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

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

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

From the delivery head and then continue.

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

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

If it fails, we want to send automated emails.

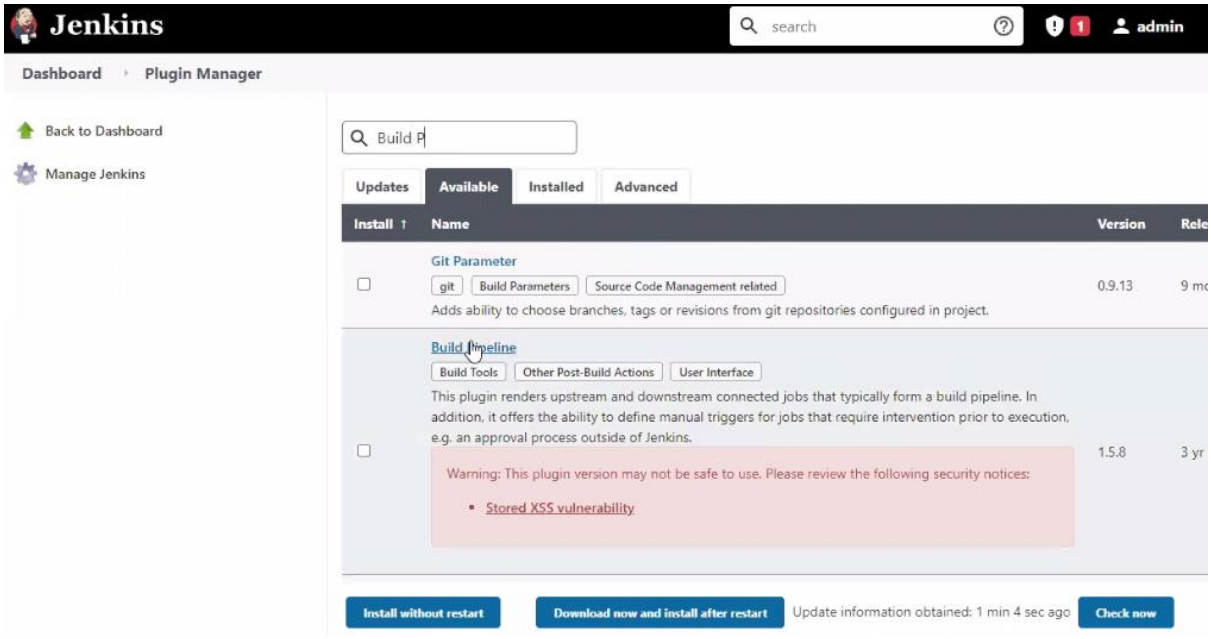
Scripted pipeline syntax:

```
node ( 'master/slave')
{
    stage(' Stage in CI-CD')
     {
          Groovy code for implementing the stage
     }
}
```

++

Install Build pipeline plugin

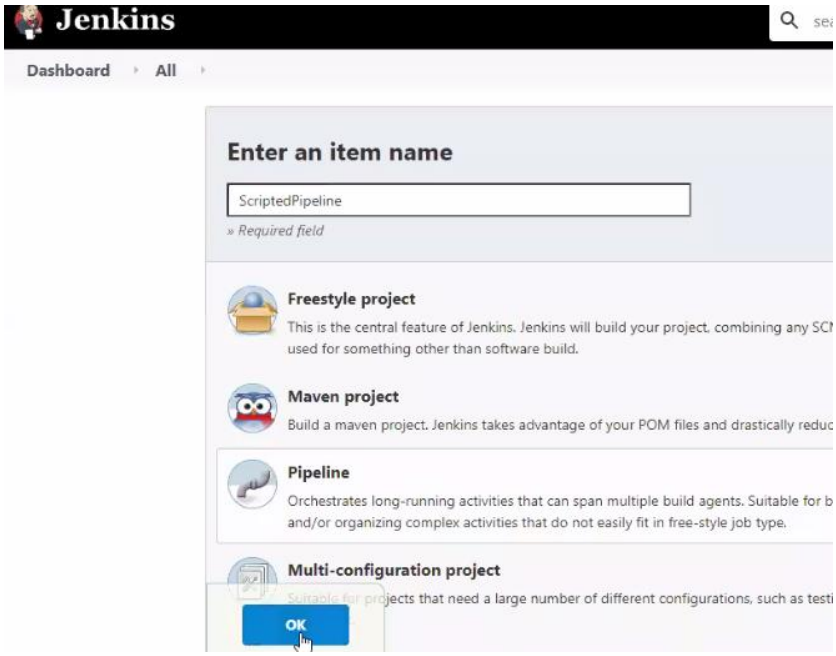
++++++++++++++++++++++++



Ex:

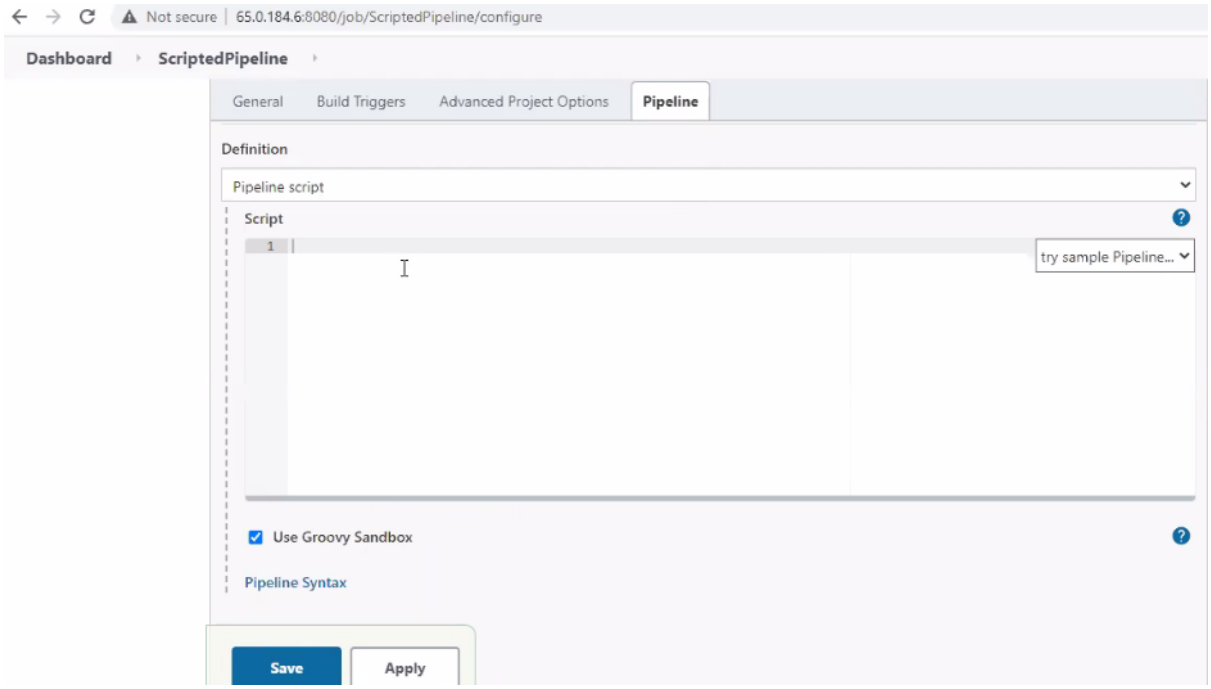
Create new item --- Scripted Pipeline

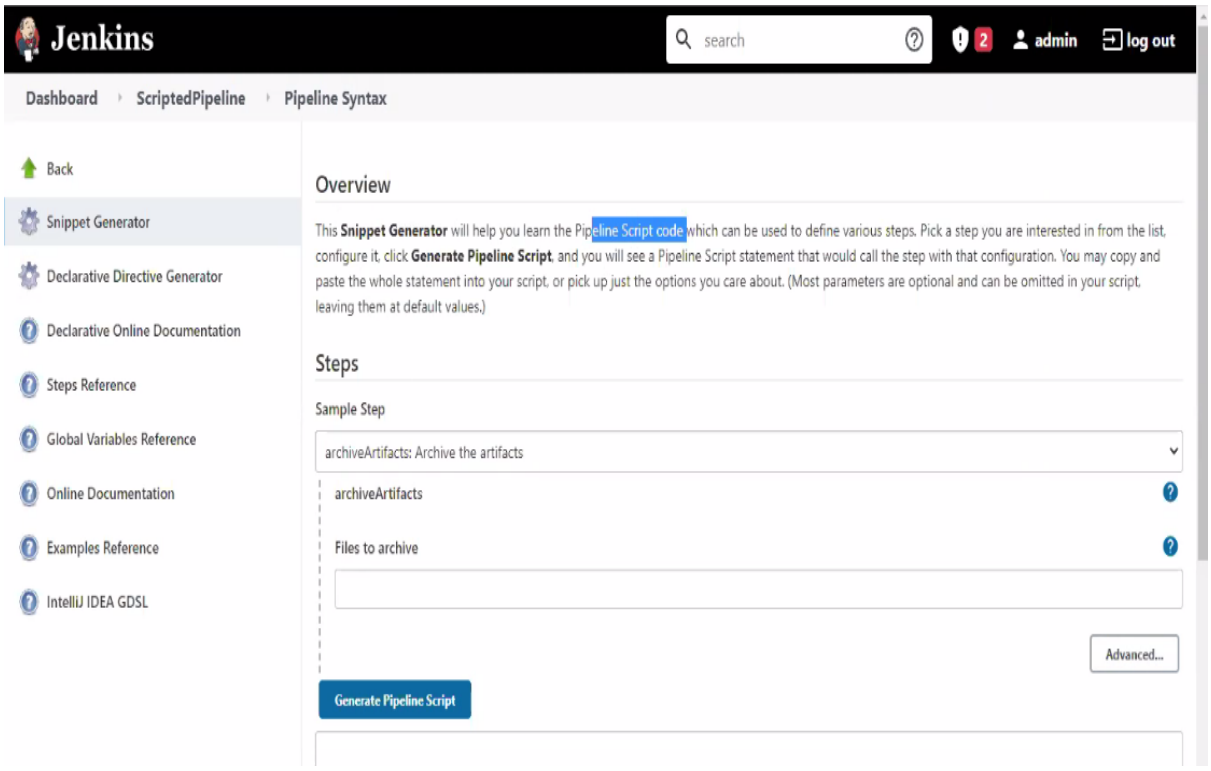
Select pipeline –OK

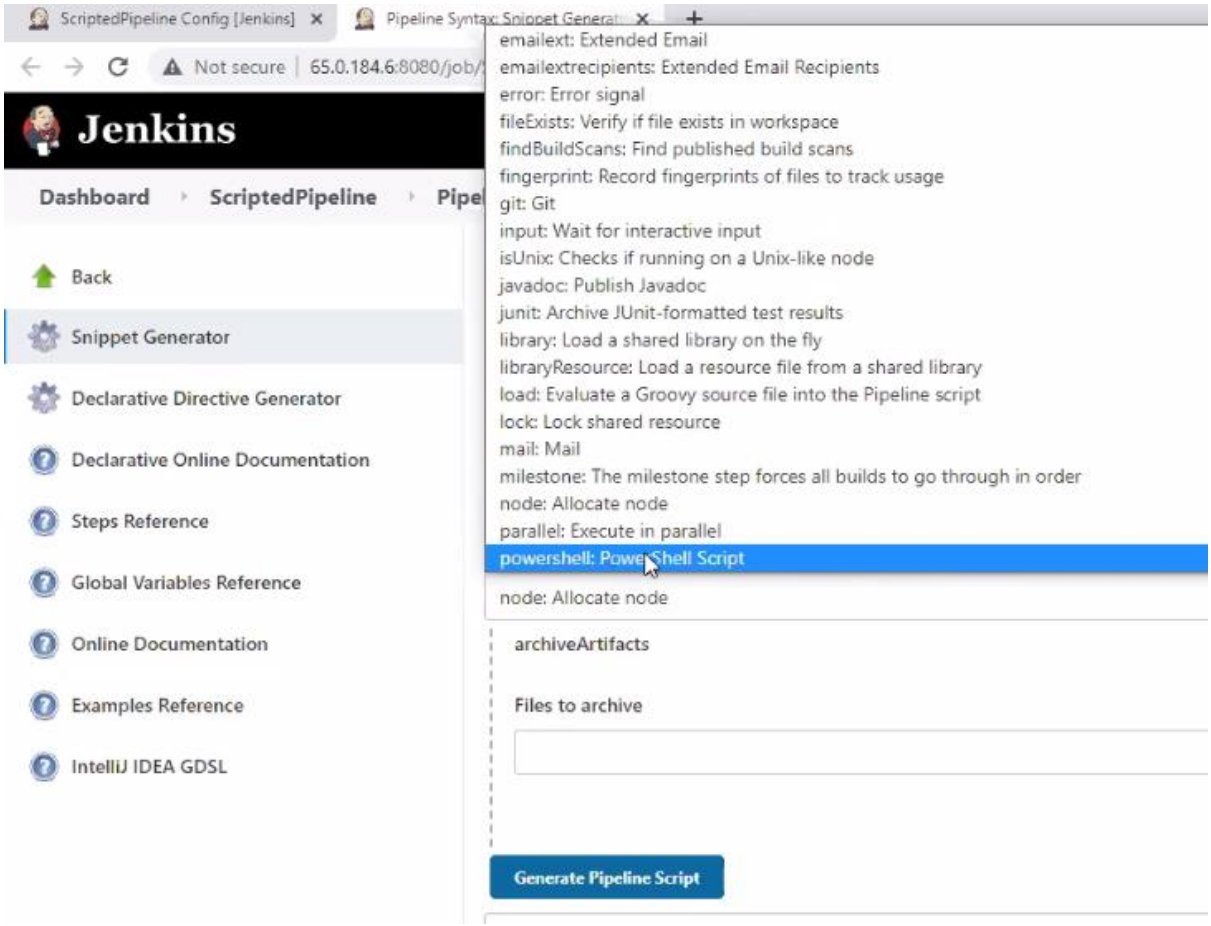


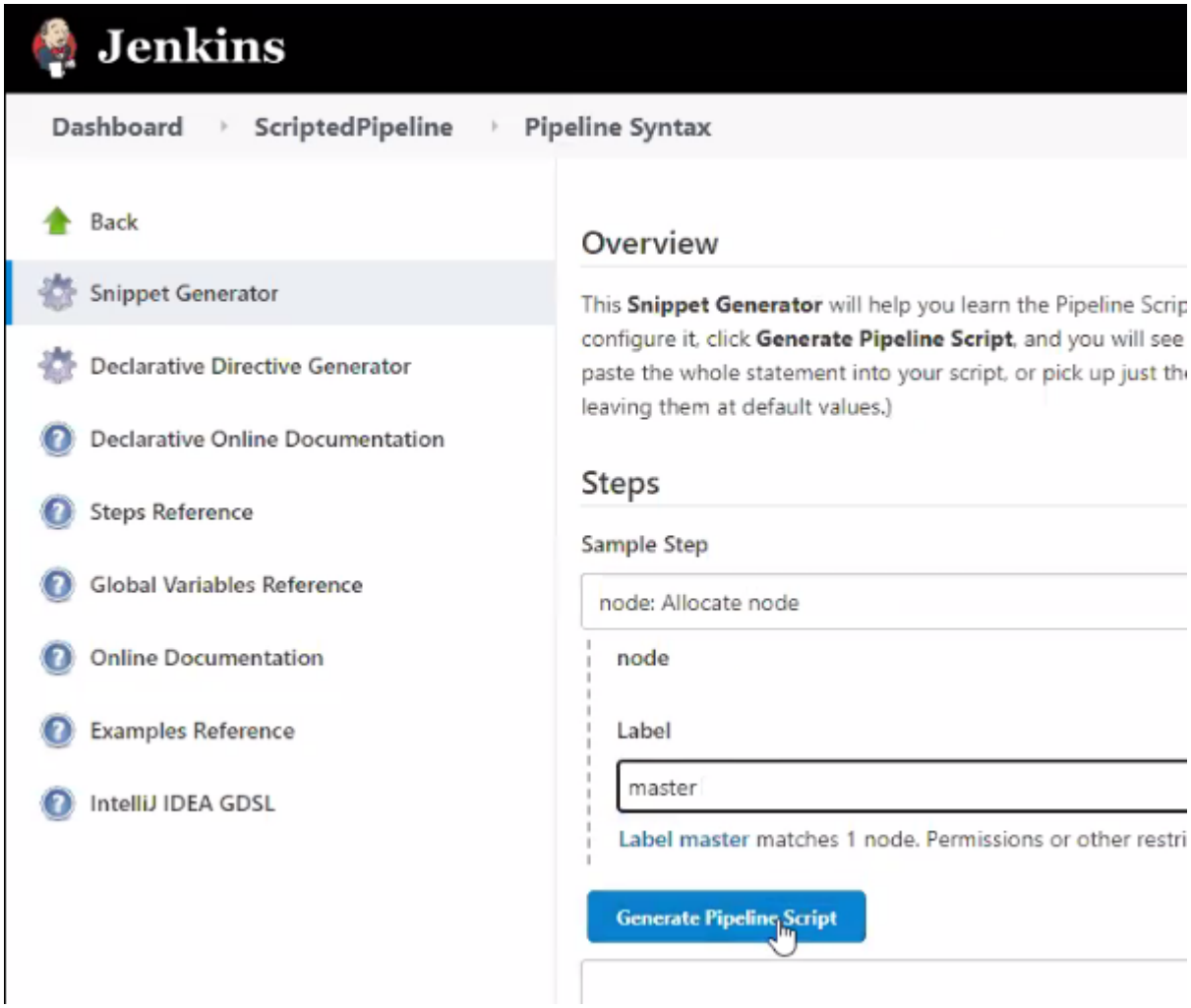
Go to →Pipeline tab,

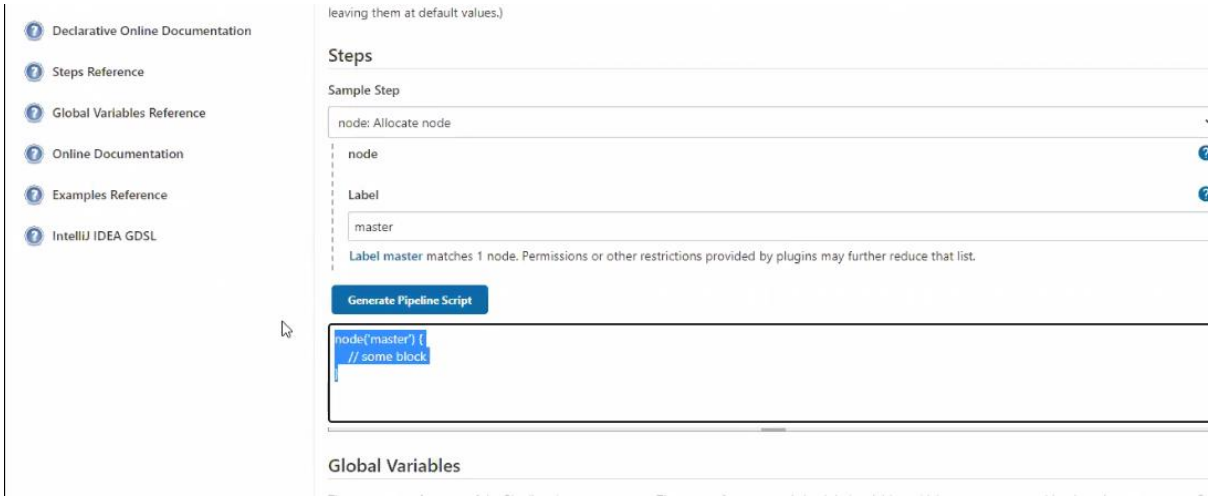
Pipeline syntax


Not secure | 65.0.184.6:8080/job/ScriptedPipeline/configure
Dashboard
ScriptedPipeline
General
Build Triggers
Advanced Project Options
Pipeline
Definition
Pipeline script
Script
1
try sample Pipeline...
Use Groovy Sandbox
Pipeline Syntax
Save
Apply


Jenkins
search
2
admin
log out
Dashboard
ScriptedPipeline
Pipeline Syntax
Back
Snippet Generator
Declarative Directive Generator
Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL
Overview
This Snippet Generator will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click Generate Pipeline Script, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)
Steps
Sample Step
archiveArtifacts: Archive the artifacts
archiveArtifacts
Files to archive
Advanced...
Generate Pipeline Script


ScriptedPipeline Config [Jenkins]
Pipeline Syntax: Snippet Generat
Not secure | 65.0.184.6:8080/job/
Jenkins
Dashboard
ScriptedPipeline
Pipel
Back
Snippet Generator
Declarative Directive Generator
Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL
emailext: Extended Email
emailextrecipients: Extended Email Recipients
error: Error signal
fileExists: Verify if file exists in workspace
findBuildScans: Find published build scans
fingerprint: Record fingerprints of files to track usage
git: Git
input: Wait for interactive input
isUnix: Checks if running on a Unix-like node
javadoc: Publish Javadoc
junit: Archive JUnit-formatted test results
library: Load a shared library on the fly
libraryResource: Load a resource file from a shared library
load: Evaluate a Groovy source file into the Pipeline script
lock: Lock shared resource
mail: Mail
milestone: The milestone step forces all builds to go through in order
node: Allocate node
parallel: Execute in parallel
powershell: PowerShell Script
node: Allocate node
archiveArtifacts
Files to archive
Generate Pipeline Script

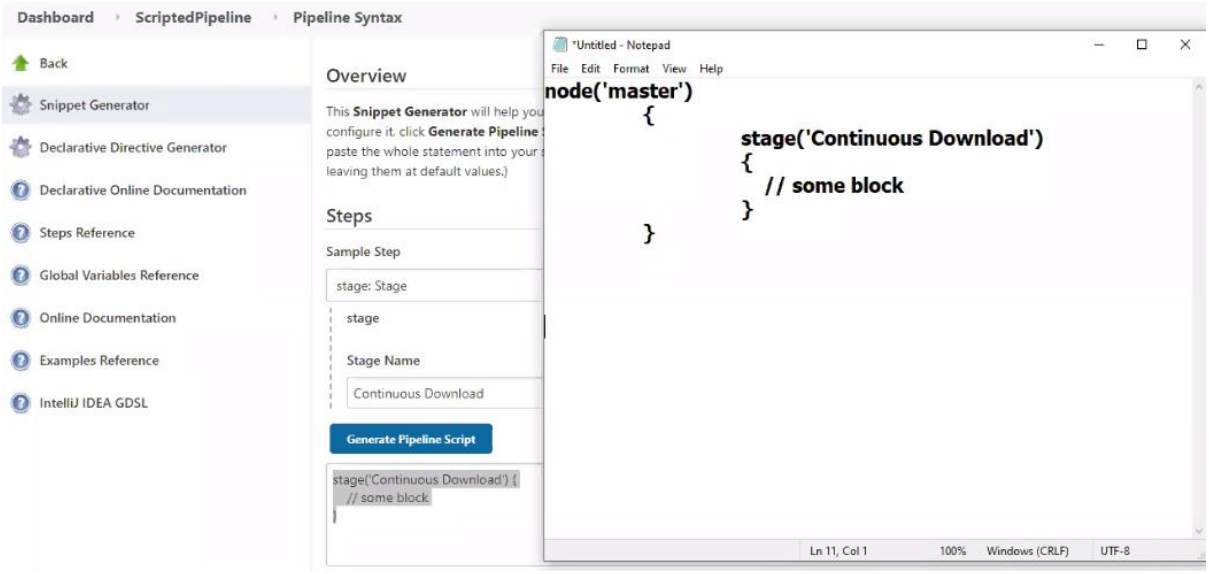




Sample step - node: Allocate node

label - master

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



In pipeline syntax

Sample step - stage: Stage

Stage name - Continuous Download

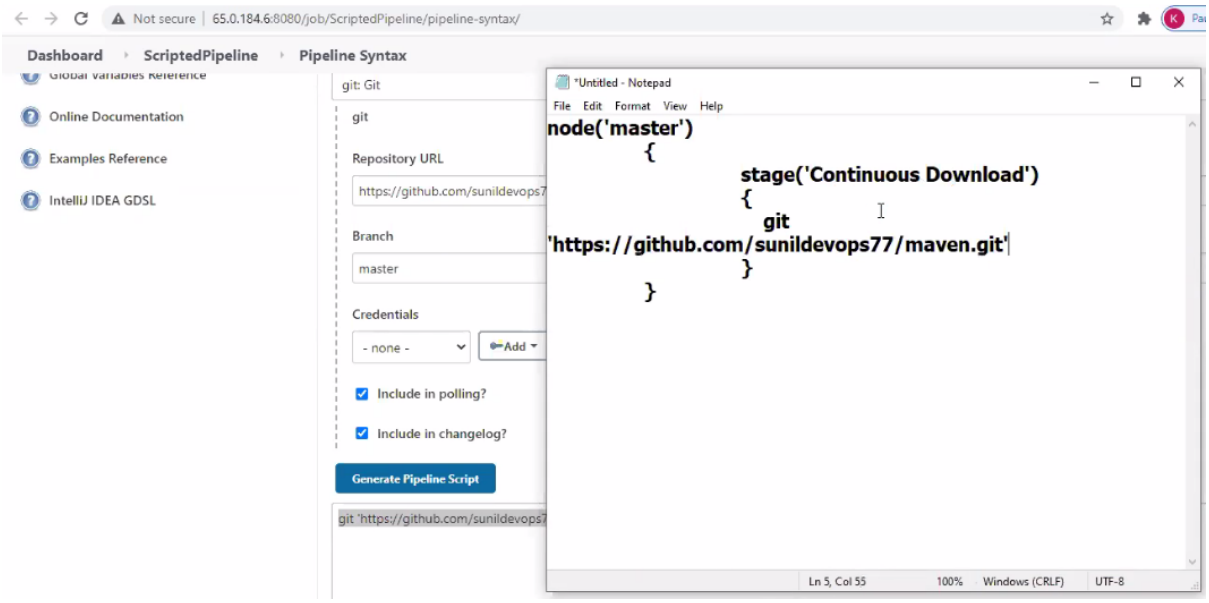
Generate piplescript -- copy the groovy code and paste in pipeline tab.

In pipeline syntax

Sample step - git: Git

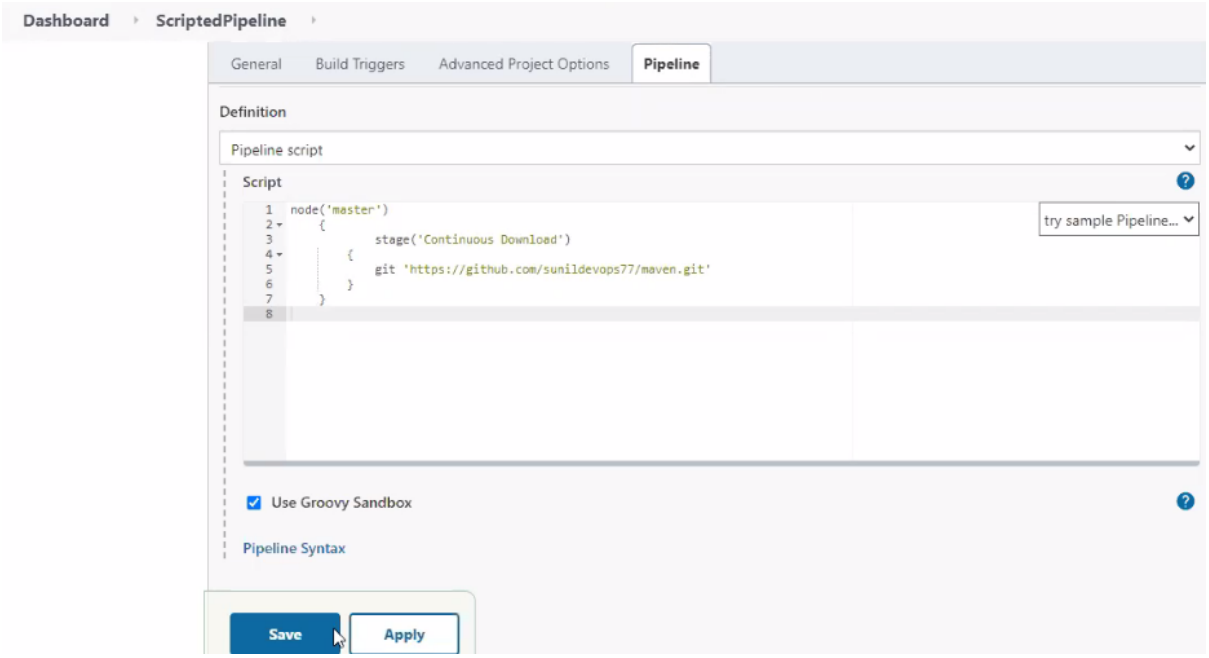
Repository URL - https://github.com/sunildevops77/maven.git

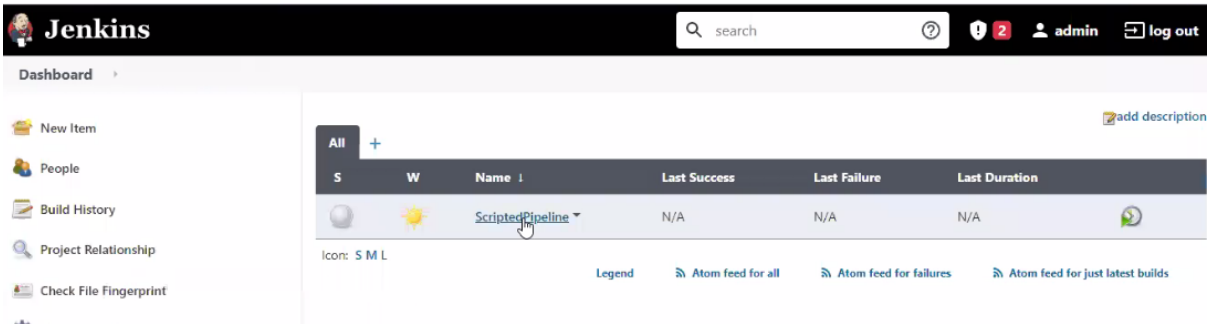
Generate piplescript -- copy the groovy code and paste in pipeline tab.



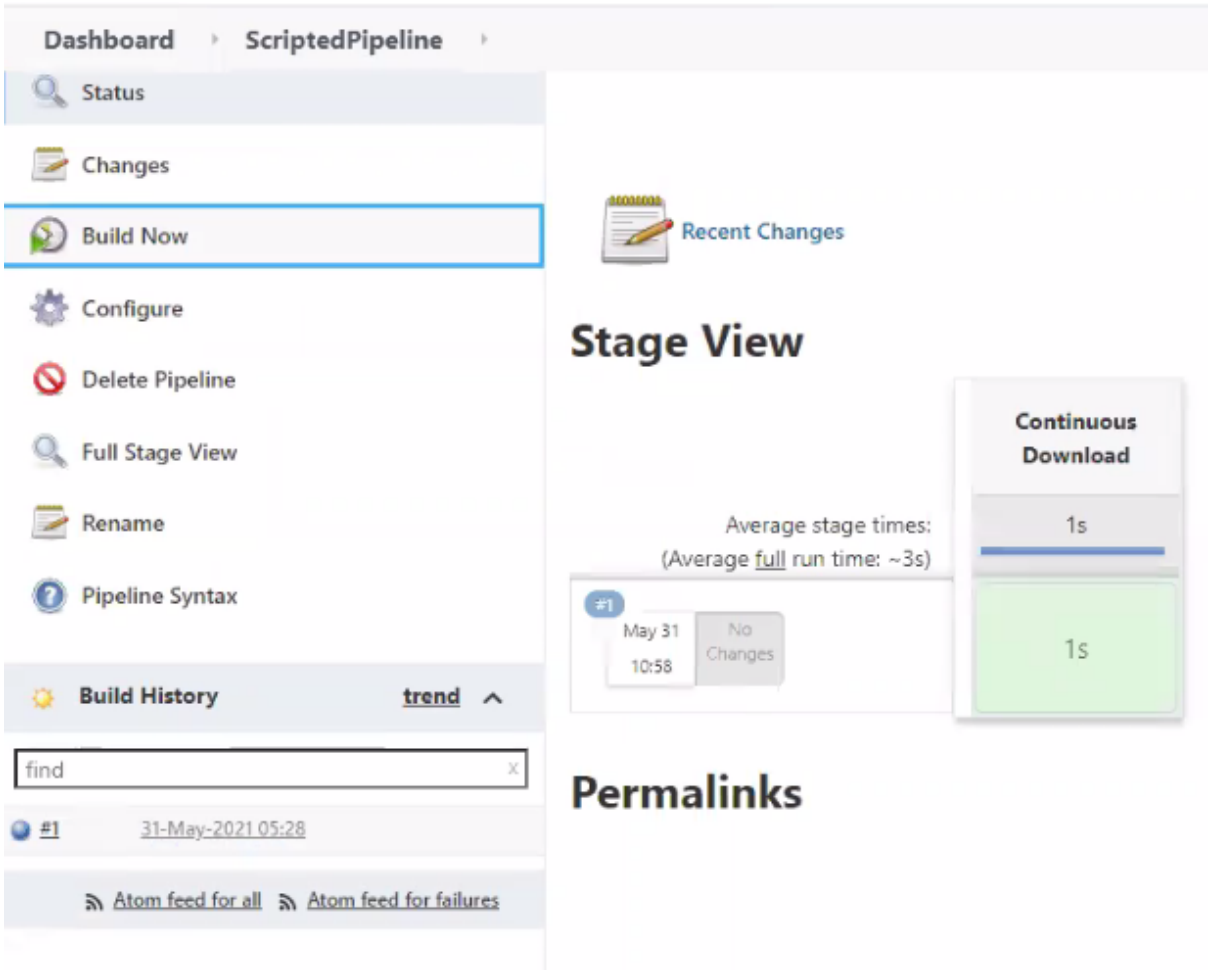
In Jenkins, scripted pipeline paste the code

Apply --- Save --> Run the job





Build now and check the status below in pipeline



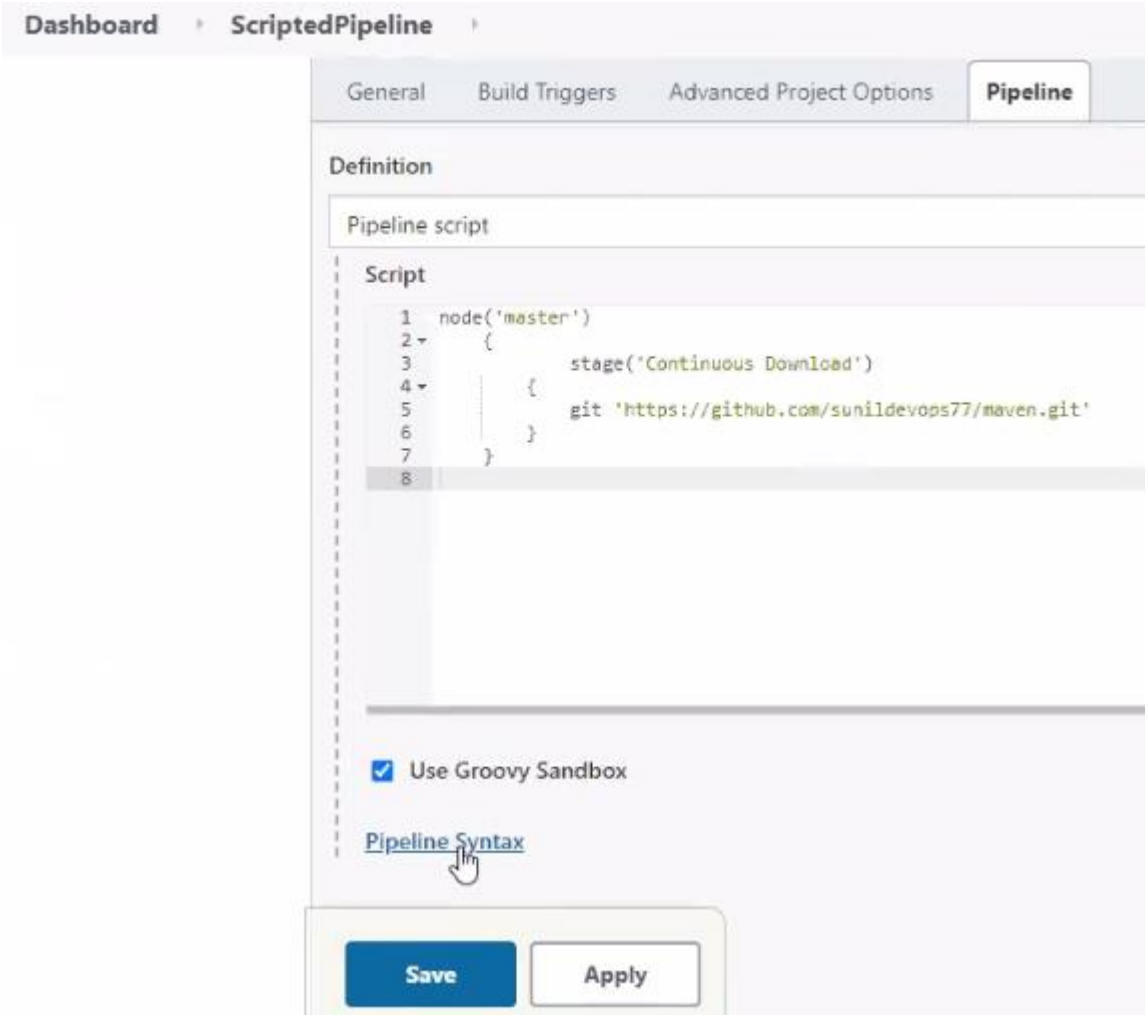
++

2nd stage

We need to run 'mvn package' command.

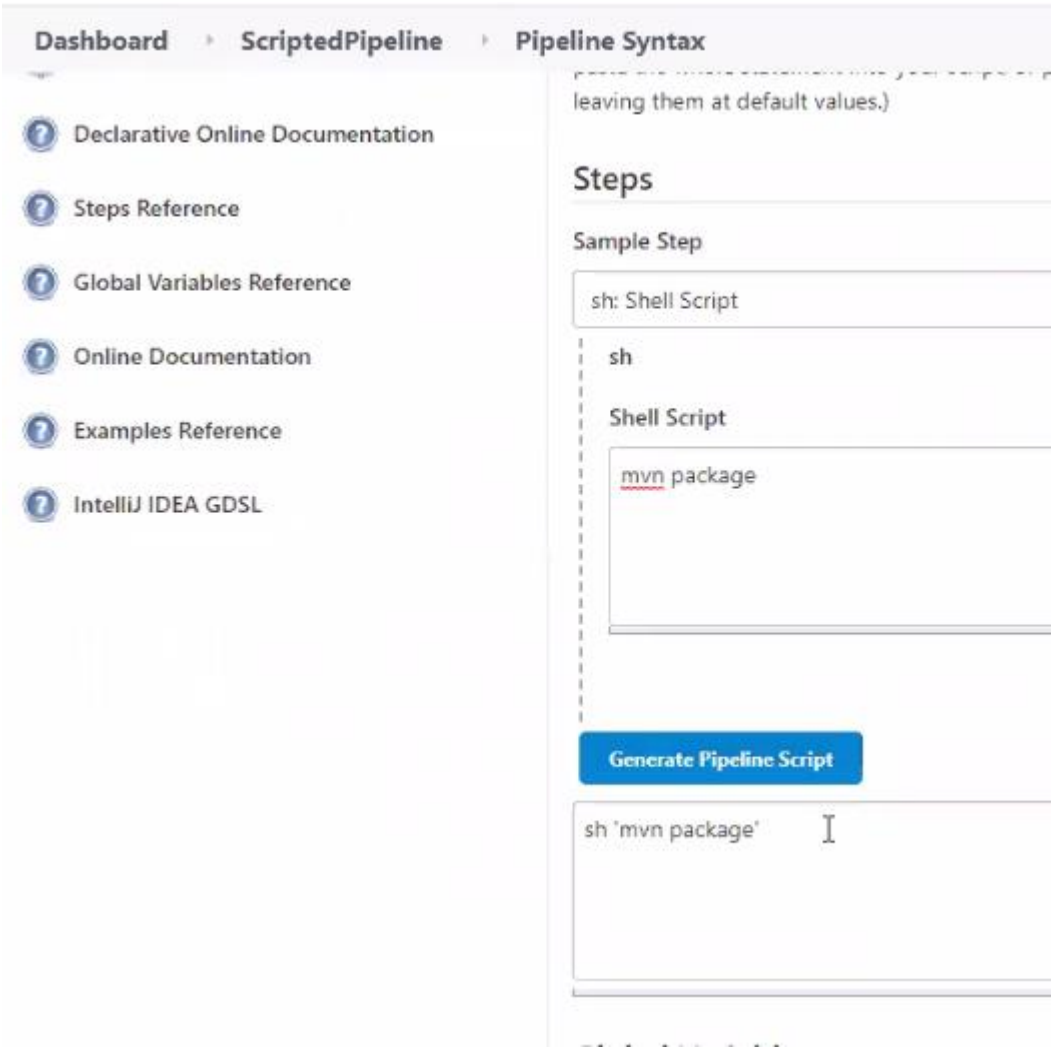
This command can be executed as a shell script

In pipeline syntax:



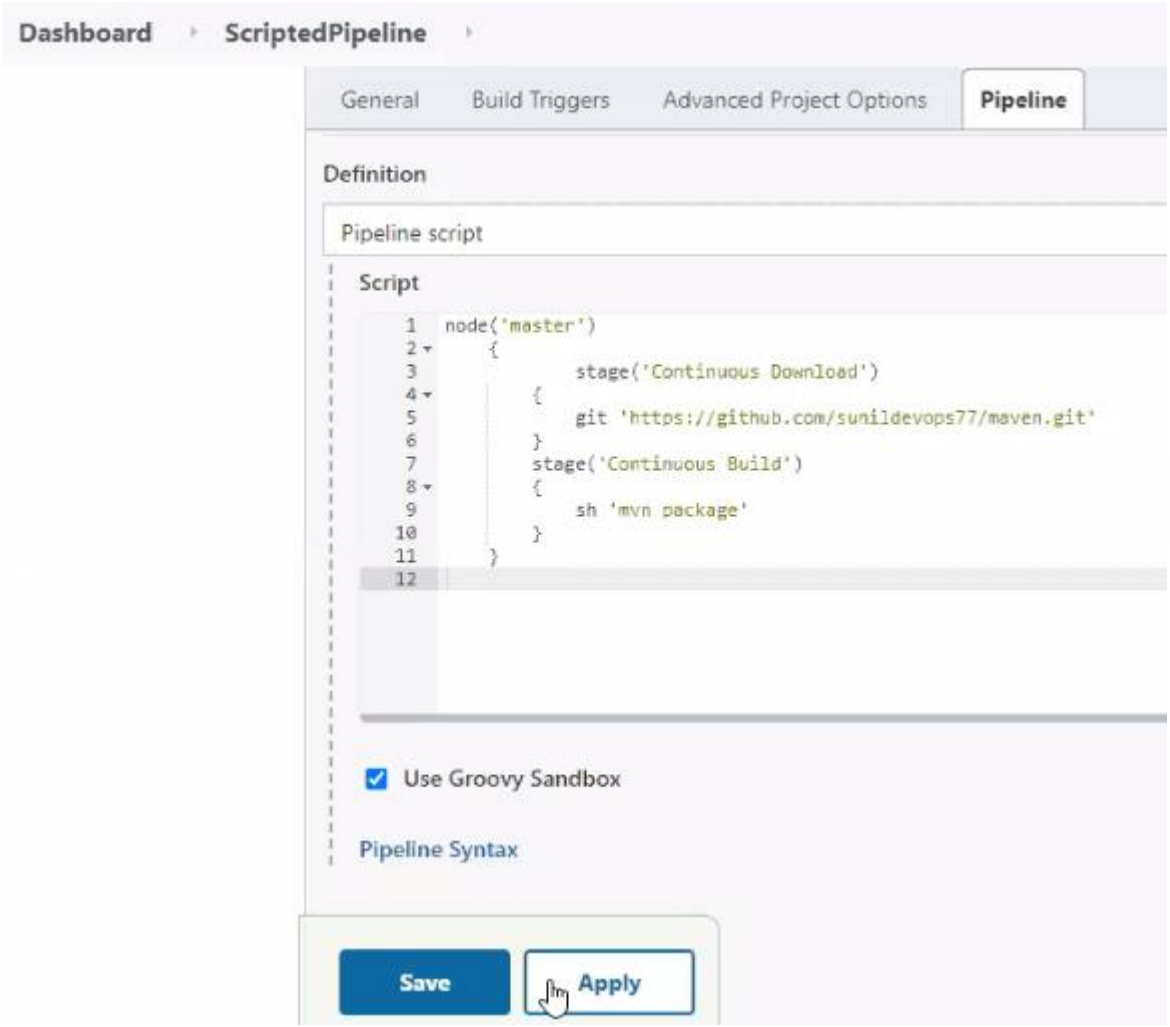
Sample step - sh: Shell Script

Stage name - mvn package

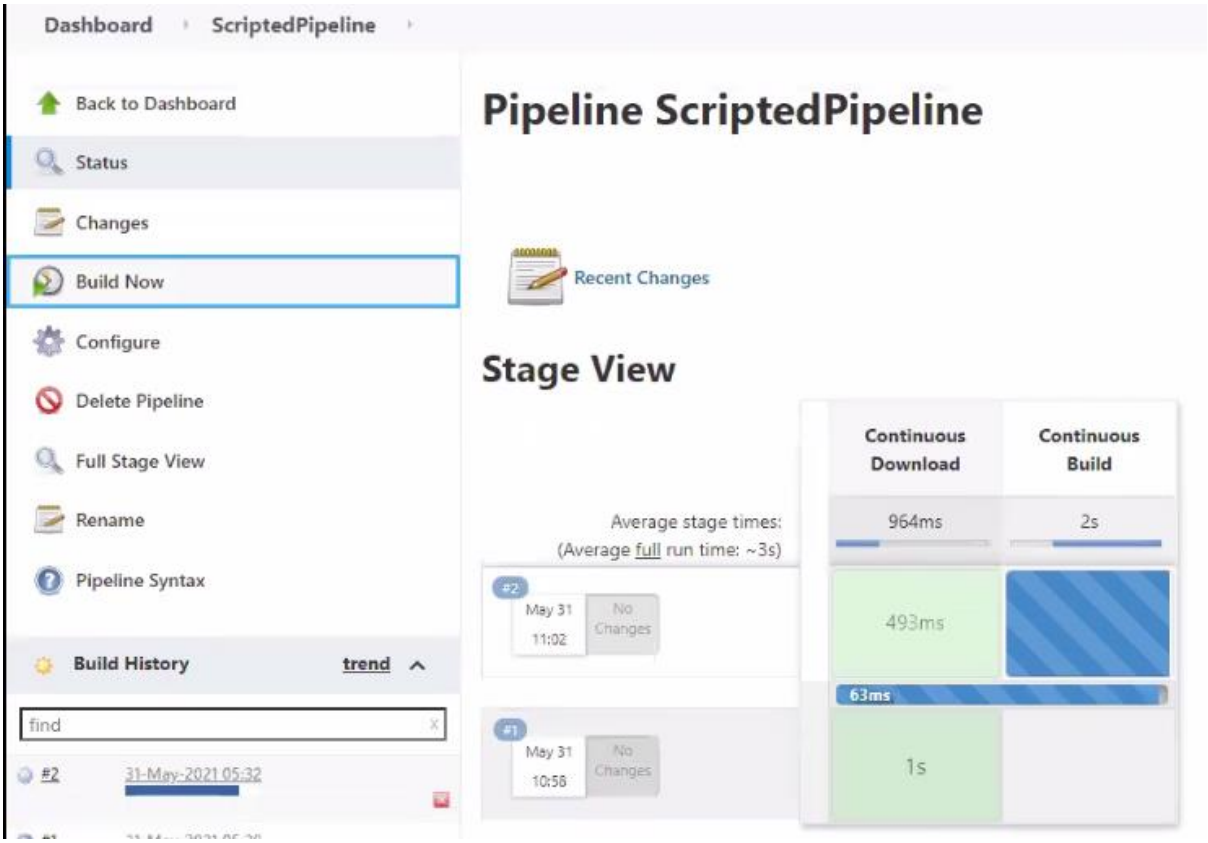



```
node('master')
        {
                stage('Continuous Download')
                {
                  git 'https://github.com/sunildevops77/maven.git'
                }
                stage('Continuous Build')
                {
                  sh 'mvn package'
                }
        }
```

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


Dashboard
ScriptedPipeline
General
Build Triggers
Advanced Project Options
Pipeline
Definition
Pipeline script
Script
node('master')
{
stage('Continuous Download')
{
git 'https://github.com/sunildevops77/maven.git'
}
stage('Continuous Build')
{
sh 'mvn package'
}
}
Use Groovy Sandbox
Pipeline Syntax
Save
Apply

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

$

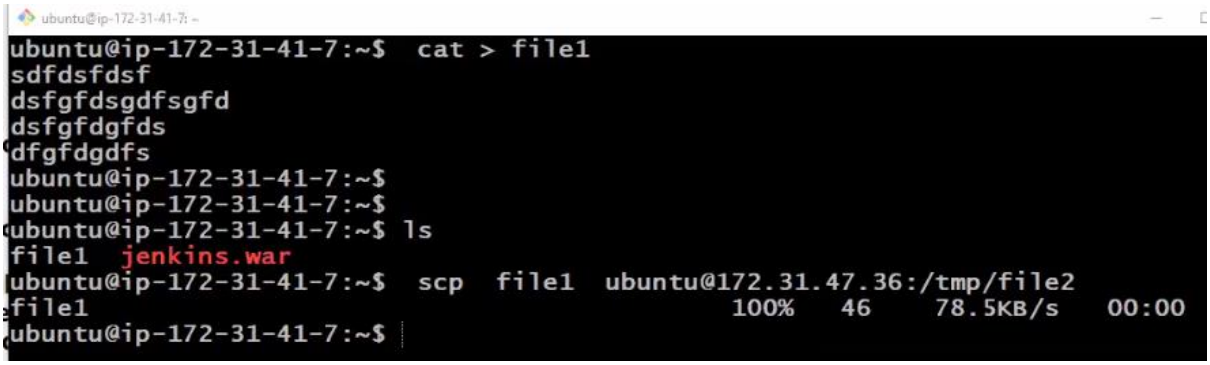
To copy the file in QA server

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

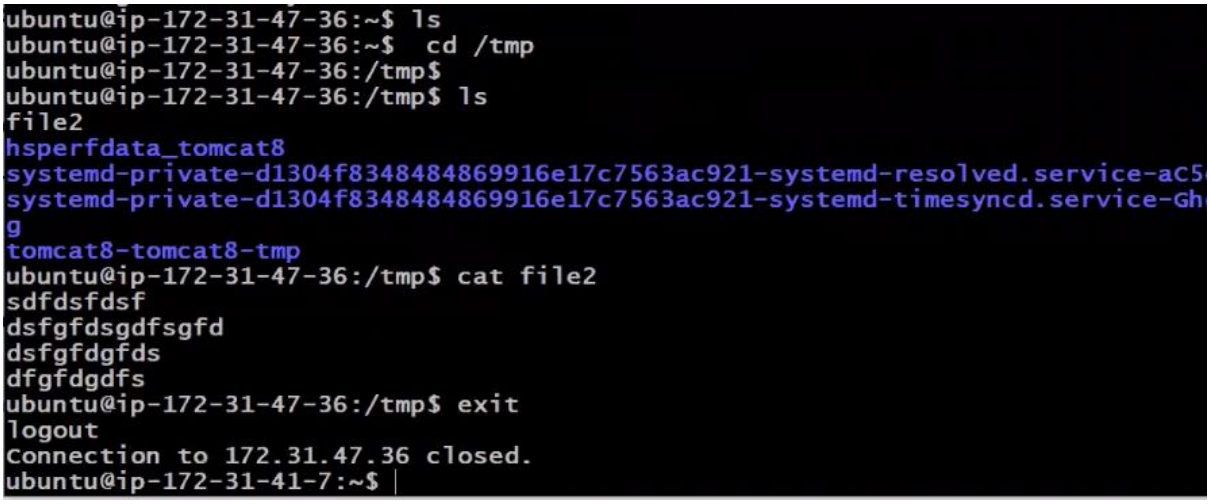
$ ssh ubuntu@172.31.47.36

$ cd /tmp

$ ls

$ cat file2

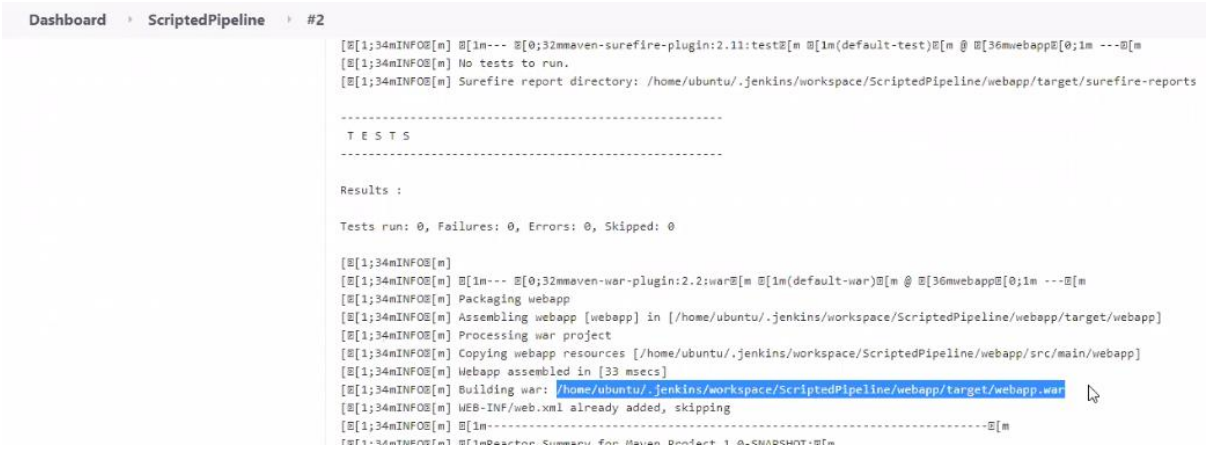
$ exit



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

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

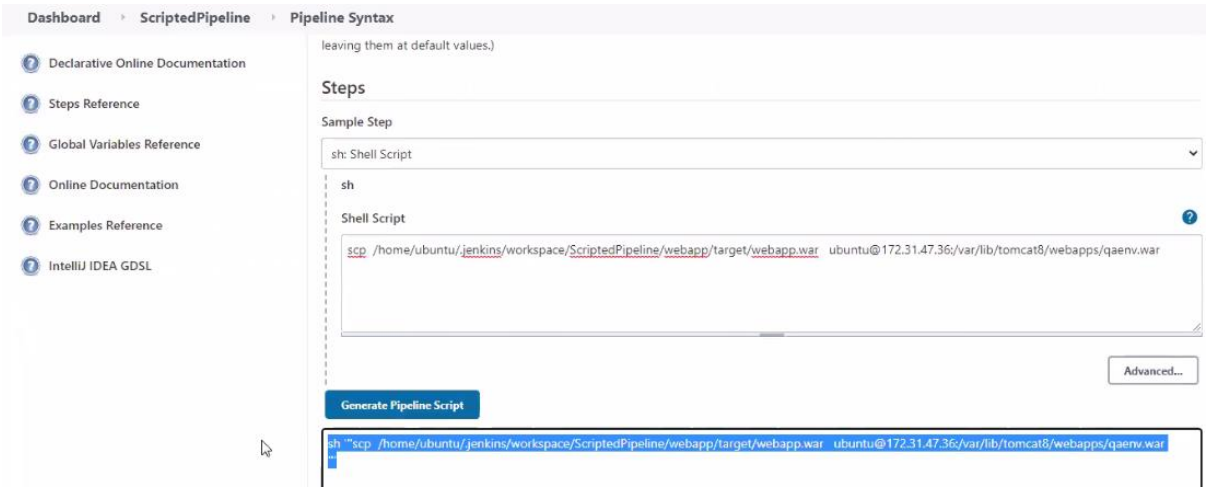
Get the location of war file from log



$ 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

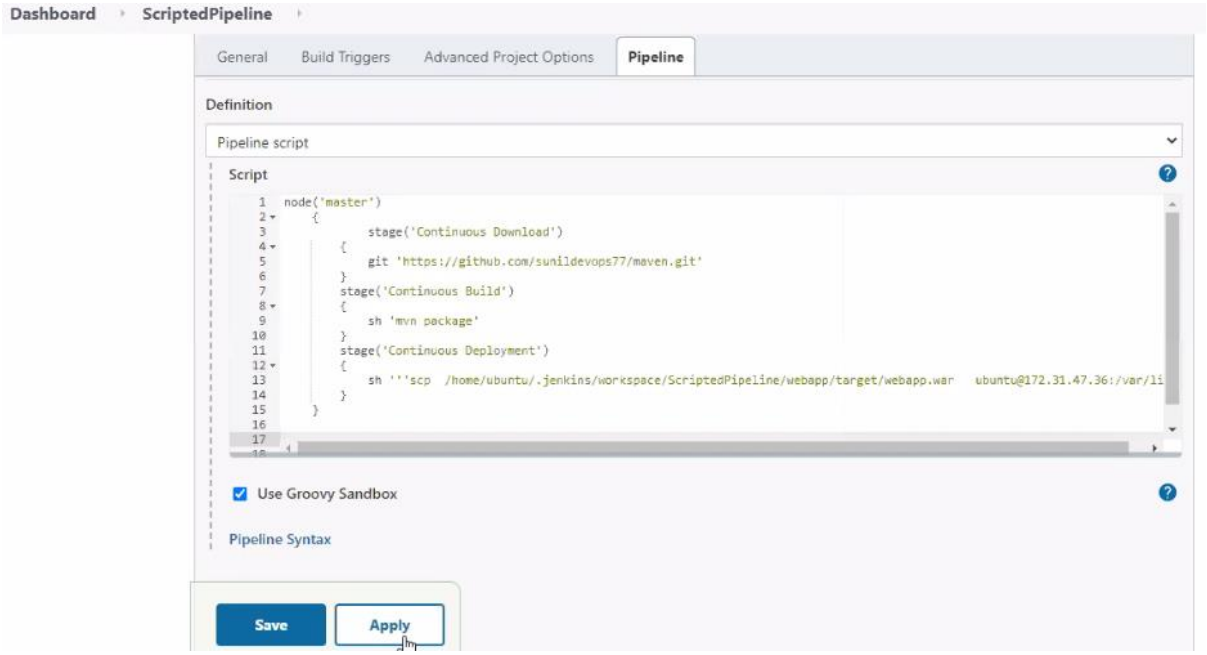


Sample Step - sh: Shell Script

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

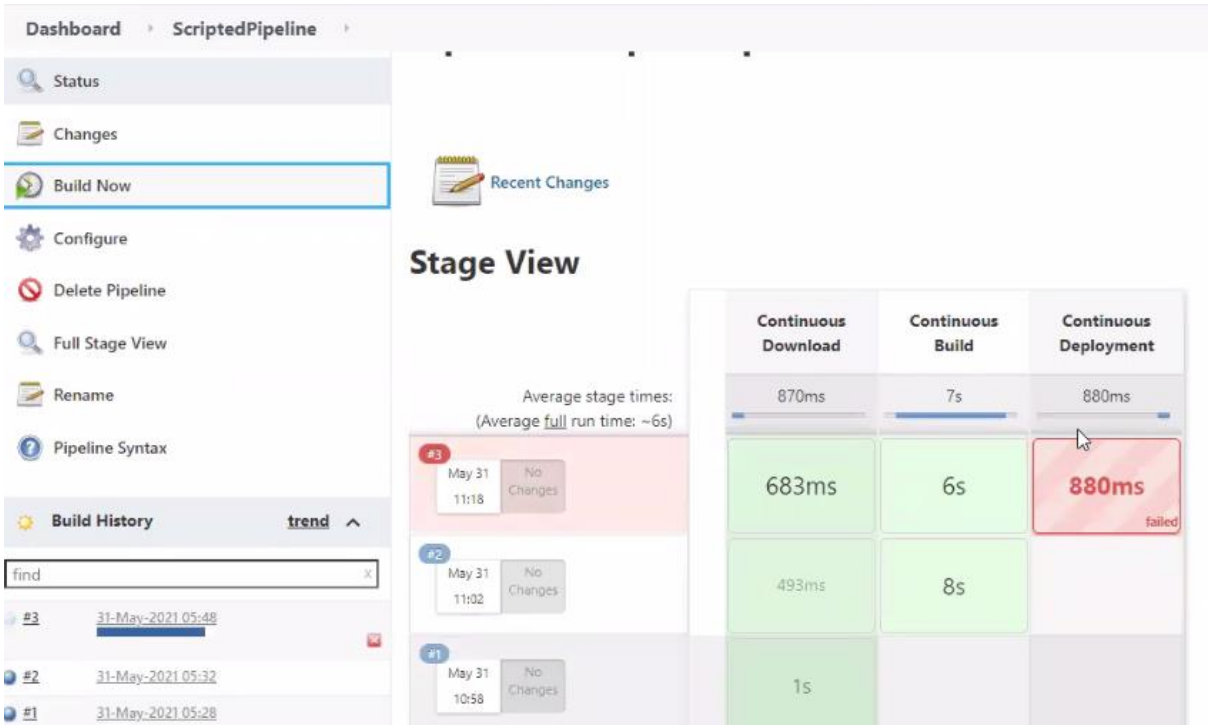
```
node('master')
        {
                stage('Continuous Download')
                {
                  git 'https://github.com/sunildevops77/maven.git'
                }
                stage('Continuous Build')
                {
                  sh 'mvn package'
                }
                stage('Continuous Deployment')
                {
                  sh '''scp
/home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war
ubuntu@172.31.47.36:/var/lib/tomcat8/webapps/qaenv.war
'''
                }
        }
```

Generate the code and paste in pipeline script

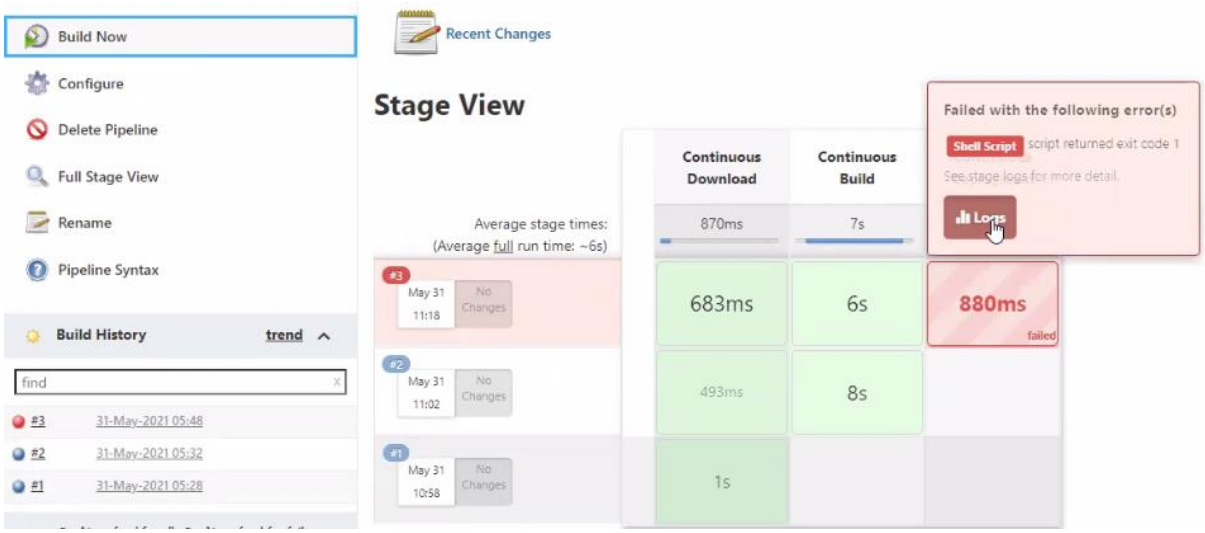


Apply --- save -- run

Deployment fails



Observe the log file (permissions denied)



To give the permissions

Connect to qa server using git bash

$ cd /var/lib

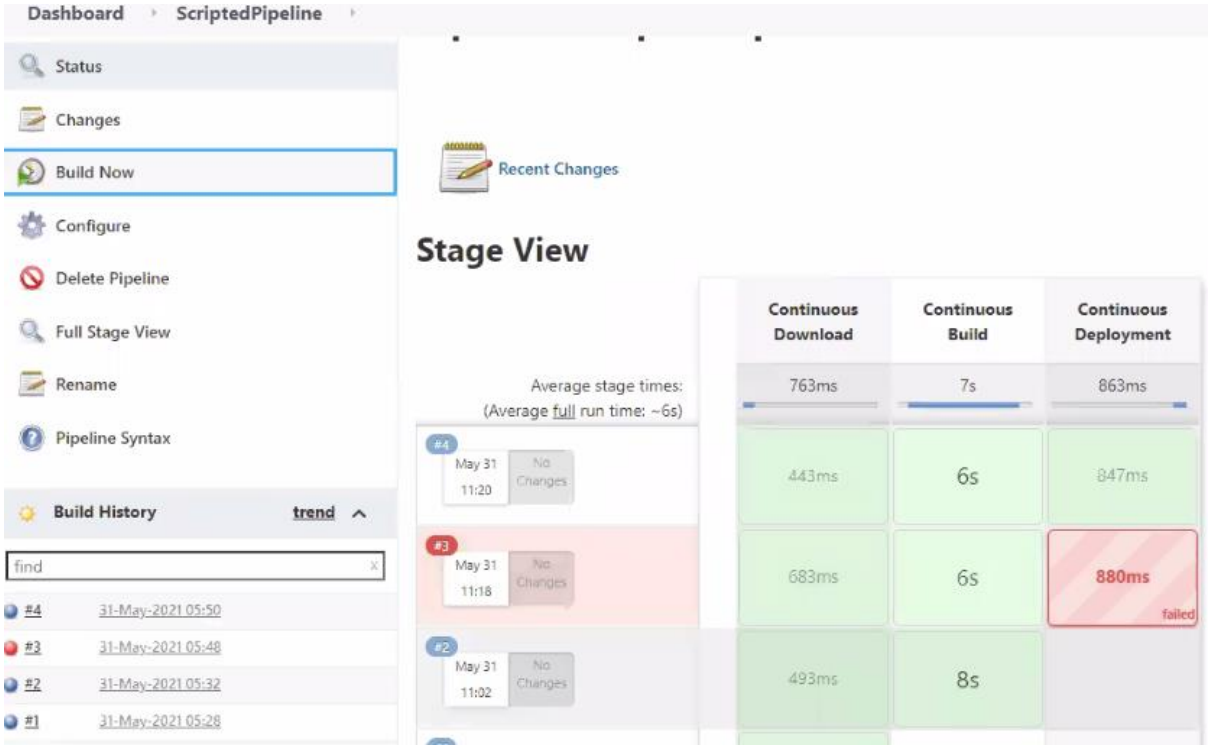
$ ls -ld tomcat8

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

$ sudo chmod -R o+w tomcat8/

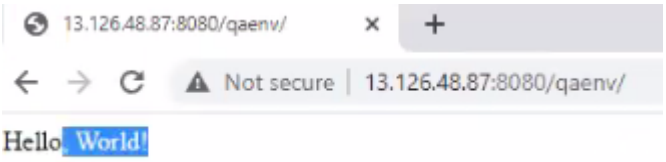
```
ubuntu@ip-172-31-47-36:/etc/ssh$
ubuntu@ip-172-31-47-36:/etc/ssh$
ubuntu@ip-172-31-47-36:/etc/ssh$  cd  /var/lib
ubuntu@ip-172-31-47-36:/var/lib$
ubuntu@ip-172-31-47-36:/var/lib$
ubuntu@ip-172-31-47-36:/var/lib$ ls
AccountsService     grub             os-prober  tomcat8
amazon              initramfs-tools  pam        ubuntu-release-upgrader
apt                 landscape        plymouth   ucf
cloud               logrotate        polkit-1   unattended-upgrades
command-not-found   lxcfs            private    update-manager
dbus                lxd              python     update-notifier
dhcp                man-db           snapd      ureadahead
dpkg                misc             sudo       usbutils
git                 mlocate          systemd    vim
ubuntu@ip-172-31-47-36:/var/lib$  sudo  chmod  -R  o+w  tomcat8/
ubuntu@ip-172-31-47-36:/var/lib$
ubuntu@ip-172-31-47-36:/var/lib$
```

Now run the job



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

Connect qa server and check



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

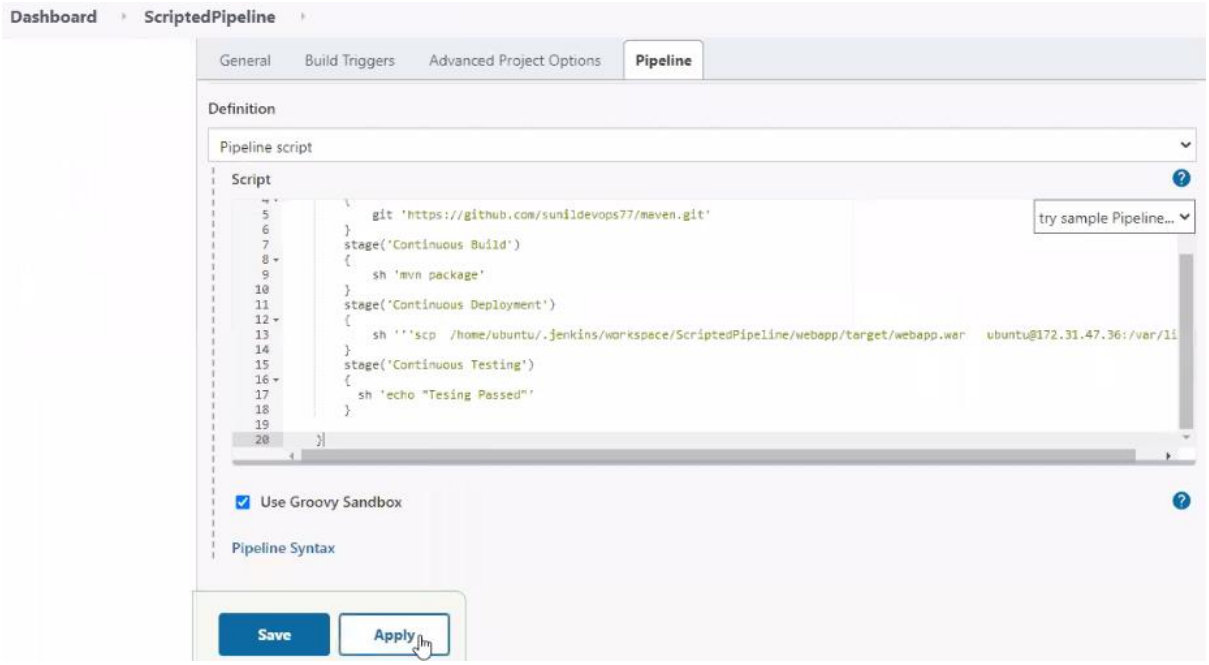
4th Stage: Continuous testing

In pipeline -- add a new stage

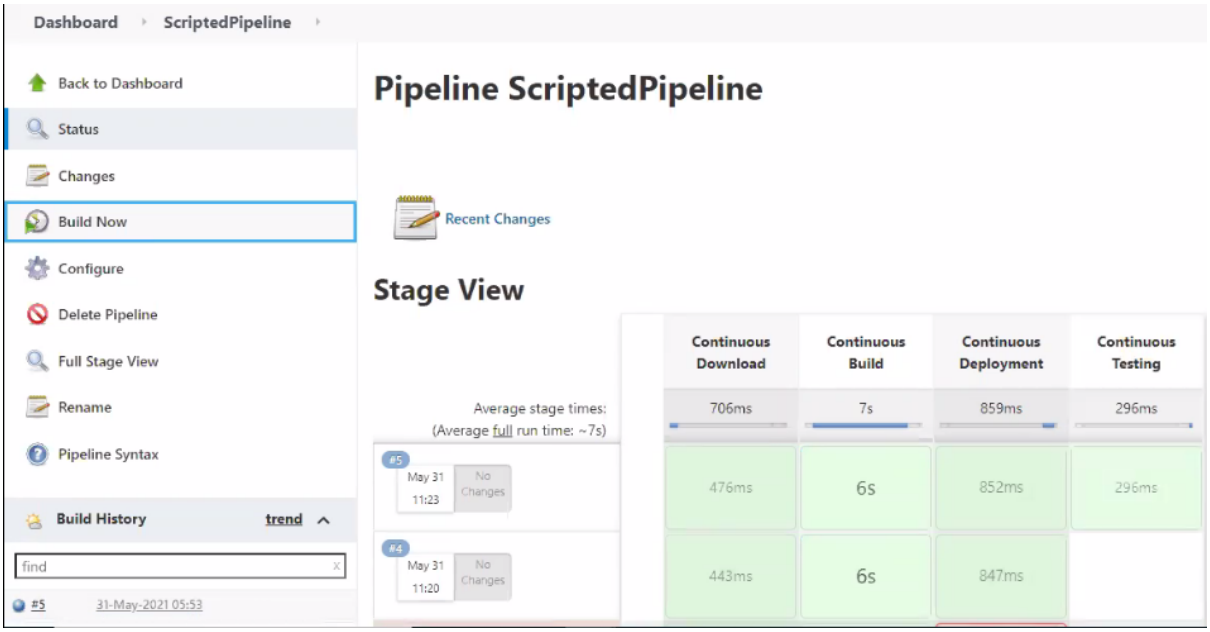
Shell script -- echo "Tesing Passed"


```
node('master')
        {
                stage('Continuous Download')
                {
                  git 'https://github.com/sunildevops77/maven.git'
                }
                stage('Continuous Build')
                {
                  sh 'mvn package'
                }
                stage('Continuous Deployment')
                {
                  sh '''scp  /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war
ubuntu@172.31.47.36:/var/lib/tomcat8/webapps/qaenv.war'''
                }
                stage('Continuous Testing')
                {
                 sh 'echo "Tesing Passed"'
                }

        }
```

Generate the groovy code and copy paste



Apply -- save-- run



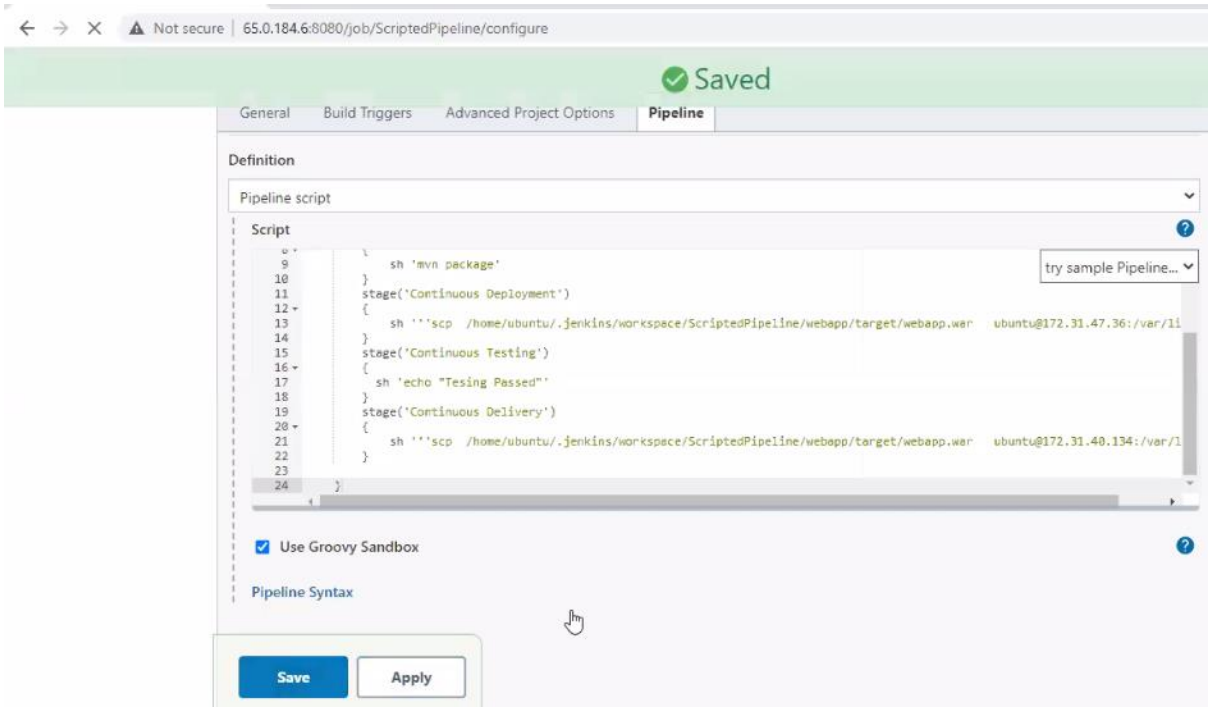
++++++++++++++++++++++++++++++++++

5th Stage : continuous delivery

continuous deployment is the same as the continuous delivery, the only change is qa and prod in path.

```
node('master')
        {
                stage('Continuous Download')
                {
                  git 'https://github.com/sunildevops77/maven.git'
                }
                stage('Continuous Build')
                {
                  sh 'mvn package'
                }
                stage('Continuous Deployment')
                {
                  sh '''scp  /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war
ubuntu@172.31.47.36:/var/lib/tomcat8/webapps/qaenv.war'''
                }
                stage('Continuous Testing')
                {
                 sh 'echo "Tesing Passed"'
                }
                stage('Continuous Delivery')
                {
                  sh '''scp  /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war
ubuntu@172.31.40.134:/var/lib/tomcat8/webapps/prodenv.war'''
                }

        }
```

In pipeline -- add a new stage



Copy the code in the - continuous deployment and change the qa_ipaddress to prod_Ip_address

Also change the context path - prodenv

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

(we should change 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

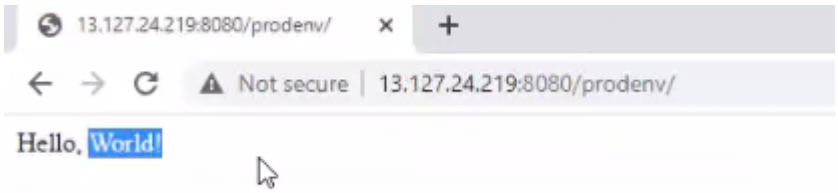
(Observation: tomcat8 directory -- others is not having write permissions)

$ 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

++
++
++

1st June

++

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.

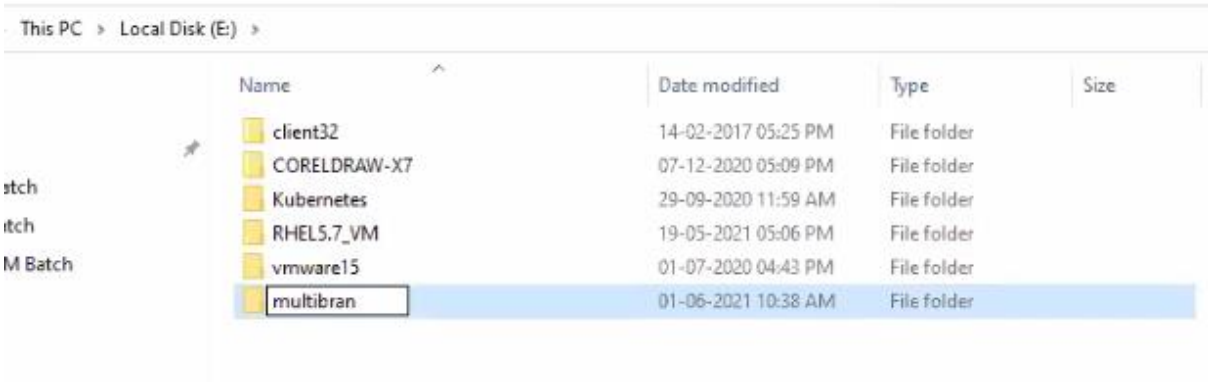
Along with the code, the developer will also create separate Jenkins file for every branch.

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 parallelly and execute the steps present in different Jenkins files.

Steps performed by the developer



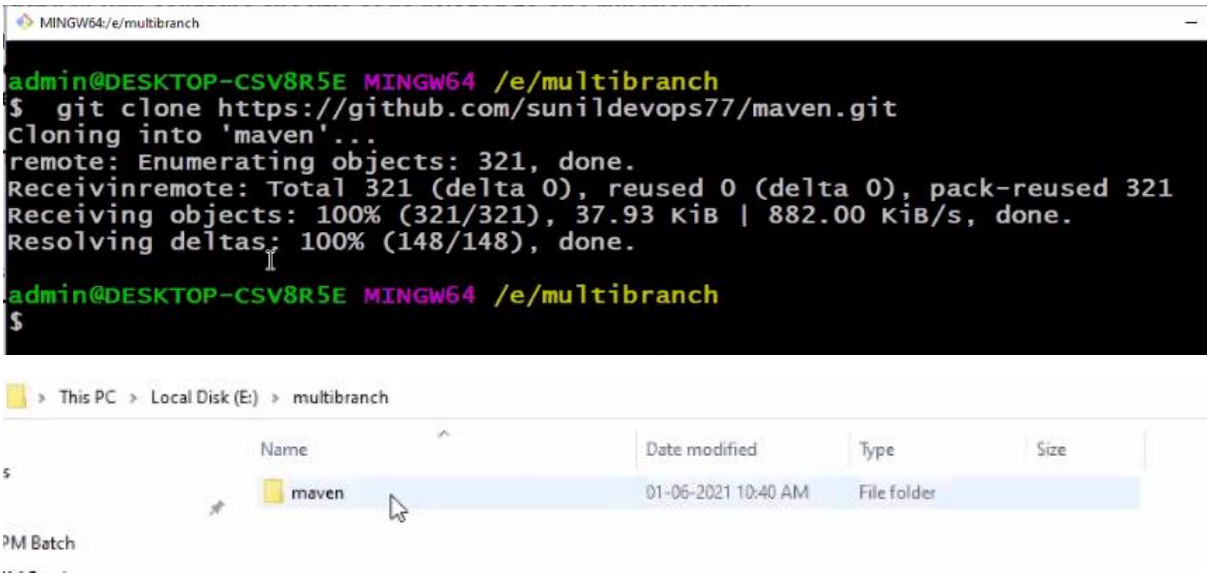
Create a new folder with Multi branch. Then opene Gitbash here.

$ mkdir multibranch

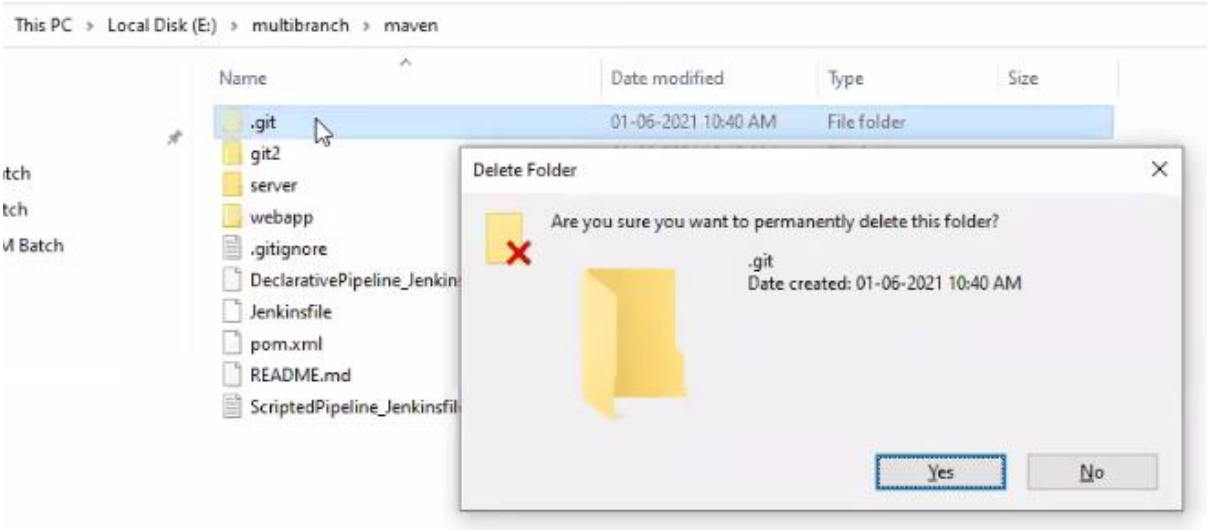
$ cd multibranch

Download the files of maven repository—code is downloaded and Maven folder is created.

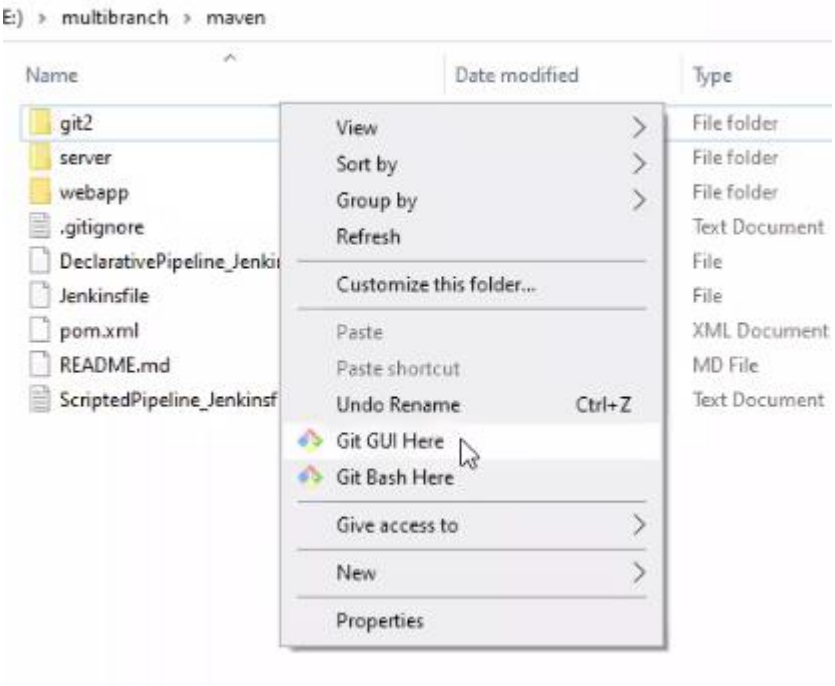
$ git clone https://github.com/sunildevops77/maven.git



Remove the hidden folder .git in maven folder

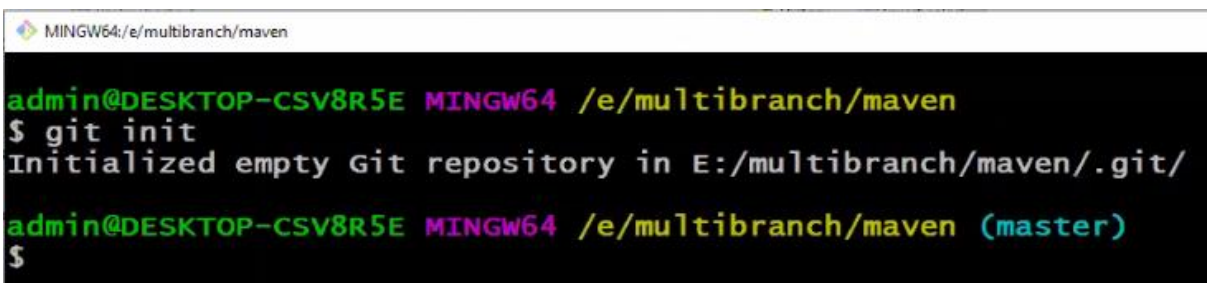


$ cd maven



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

$ git init (create a new working directory)



By default Master branch has created.

$ git status--- every file is untracked file as shown below.

```
$ git init
Initialized empty Git repository in E:/multibranch/maven/.git/

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        DeclarativePipeline_Jenkinsfile
        Jenkinsfile
        README.md
        ScriptedPipeline_Jenkinsfile.txt
        pom.xml
        server/
        webapp/

nothing added to commit but untracked files present (use "git add" to track)

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$
```

$ git add .---moving to staging area.

```
MINGW64:/e/multibranch/maven
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        DeclarativePipeline_Jenkinsfile
        Jenkinsfile
        README.md
        ScriptedPipeline_Jenkinsfile.txt
        pom.xml
        server/
        webapp/

nothing added to commit but untracked files present (use "git add" to track)

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$  git add .
```

$ git commit -m "a"--- staging area to local repository.

```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$  git add .

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$  git commit  -m "a"
[master (root-commit) e9b922b] a
 14 files changed, 508 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 DeclarativePipeline_Jenkinsfile
 create mode 100644 Jenkinsfile
 create mode 100644 README.md
 create mode 100644 ScriptedPipeline_Jenkinsfile.txt
 create mode 100644 pom.xml
 create mode 100644 server/pom.xml
 create mode 100644 server/src/main/java/com/example/Greeter.java
 create mode 100644 server/src/site/apt/index.apt
 create mode 100644 server/src/test/java/com/example/TestGreeter.java
 create mode 100644 webapp/pom.xml
 create mode 100644 webapp/src/main/webapp/WEB-INF/web.xml
 create mode 100644 webapp/src/main/webapp/index.jsp
 create mode 100644 webapp/src/site/apt/index.apt

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$
```

$ git log—to see commits , only a commit there.

```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$ git log --oneline
e9b922b (HEAD -> master) a
```

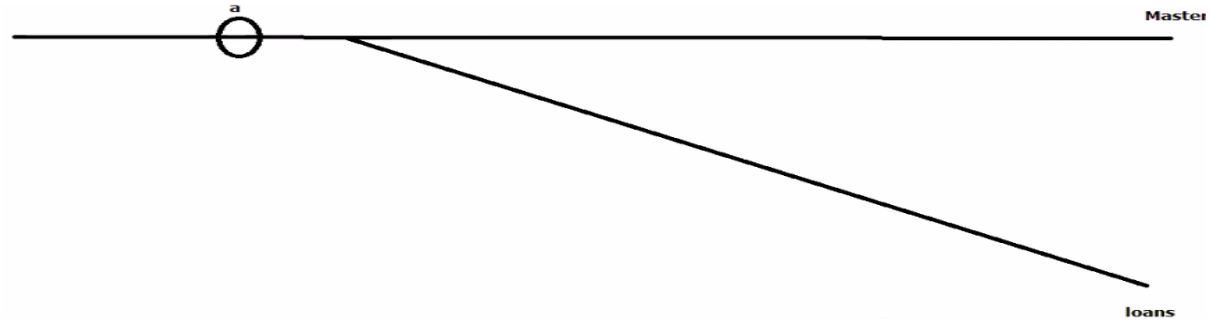
Developer creates branch

$ git checkout -b loans---- -b loans means the branch loans automatically created when you run this command without creating the Loan branch manually.

It is pointing to Loan branch .

```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$  git checkout -b loans
Switched to a new branch 'loans'

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$
```

$ git log (whenever we create a branch the commit history copied into the new branch i.e loans)



```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$ git log --oneline
e9b922b (HEAD -> loans, master) a

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$
```

$ git checkout master—go back to master branch

```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$  git checkout master
Switched to branch 'master'

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$
```

$ ls—files present in Master Branch

```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$ ls
DeclarativePipeline_Jenkinsfile  ScriptedPipeline_Jenkinsfile.txt  server/
Jenkinsfile                      git2/                             webapp/
README.md                        pom.xml

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$
```

In Above files there is Jenkins file —has the code to perform the CI CD stages.

Make changes to the Jenkins file

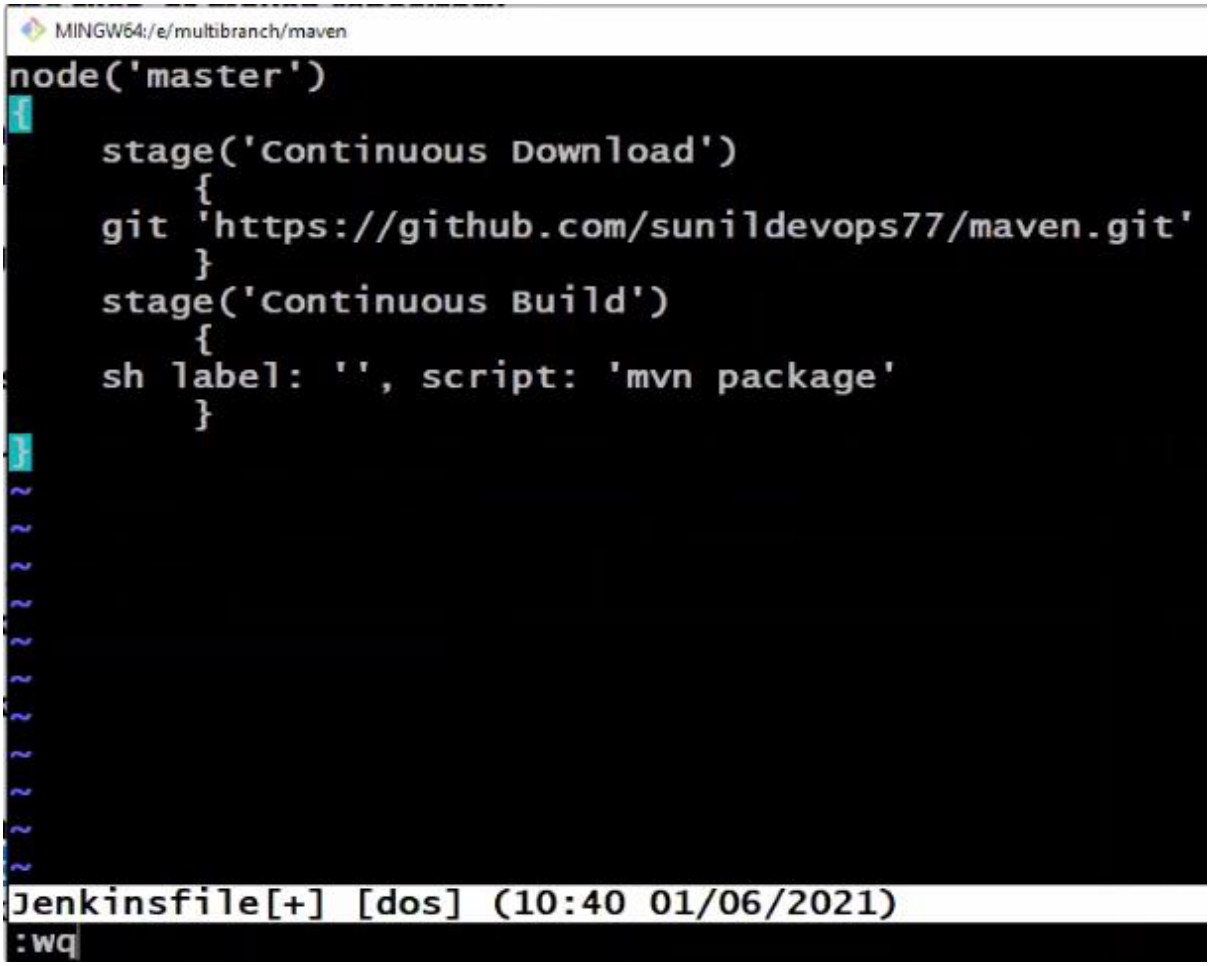
$ vim Jenkinsfile

```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$  vim Jenkinsfile
```

MINGW64:/e/multibranch/maven

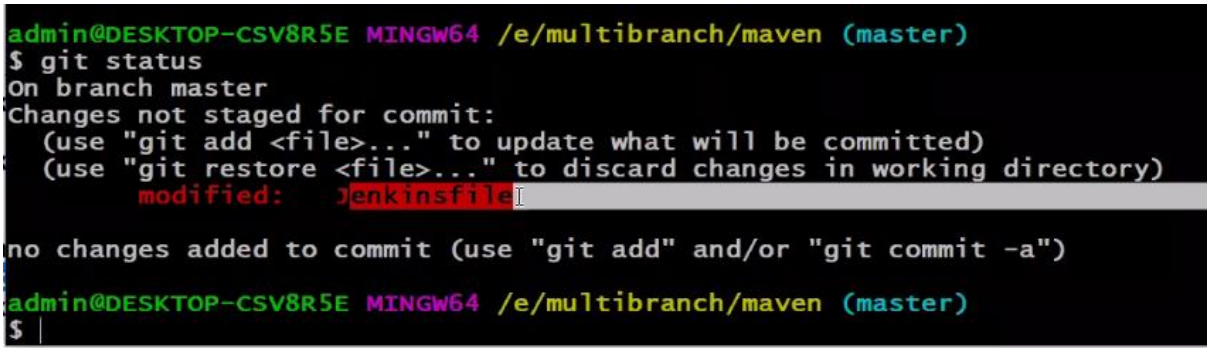
```
node('master')
{
    stage('Continuous Download')
        {
    git 'https://github.com/sunildevops77/maven.git'
        }
    stage('Continuous Build')
        {
    sh label: '', script: 'mvn package'
        }
    stage('Continuous Deployment')
        {
sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/weba
pp/target/webapp.war   ubuntu@172.31.26.217:/var/lib/tomcat8/webapps/qaenv.war'
        }
    stage('Continuous Testing')
        {
              sh label: '', script: 'echo "Testing Passed"'
        }
    stage('Continuous Delivery')
        {
@@@
```

Lets make it only two stages : save and quit :wq




```
node('master')
{
stage('ContinuousDownload_master')
      {
  git 'https://github.com/sunildevops77/maven.git'
      }
stage('Continuousbuild_master')
      {
   sh label: '', script: 'mvn package'
      }
}
:wq
```

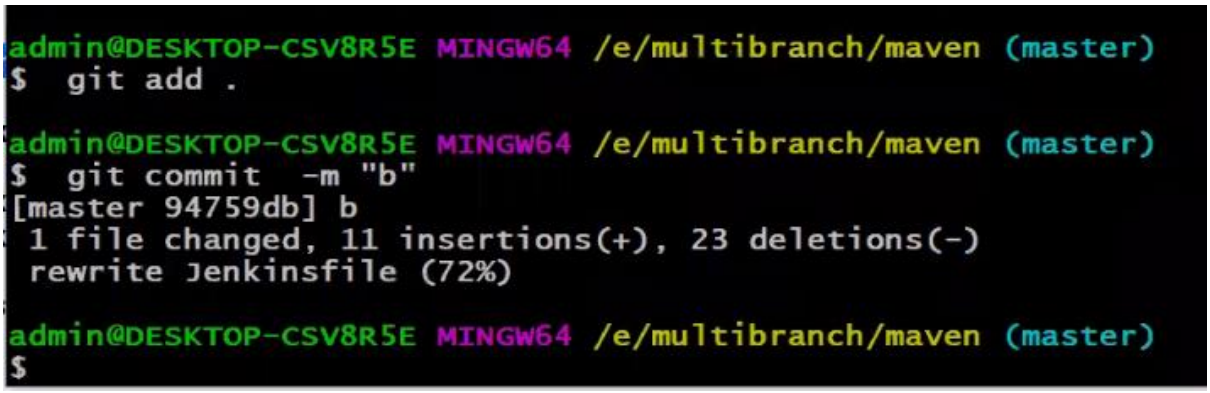
Git status- Unracked file again becz Jenins file has been modified.



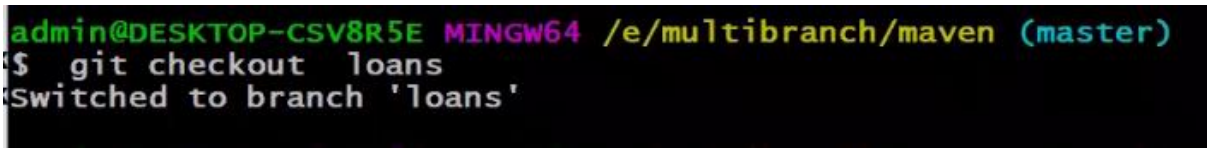
(Observation, we have done the changes in master branch)

$ git add .

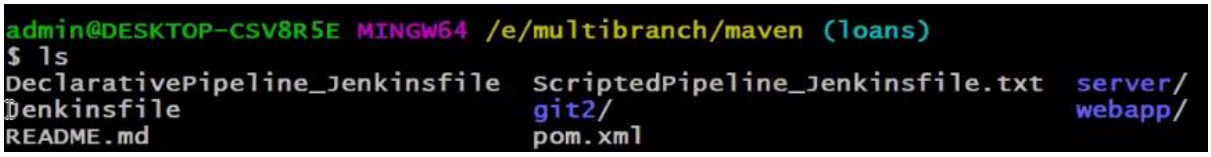
$ git commit -m "b"



$ git checkout loans--- go to Loans Branch



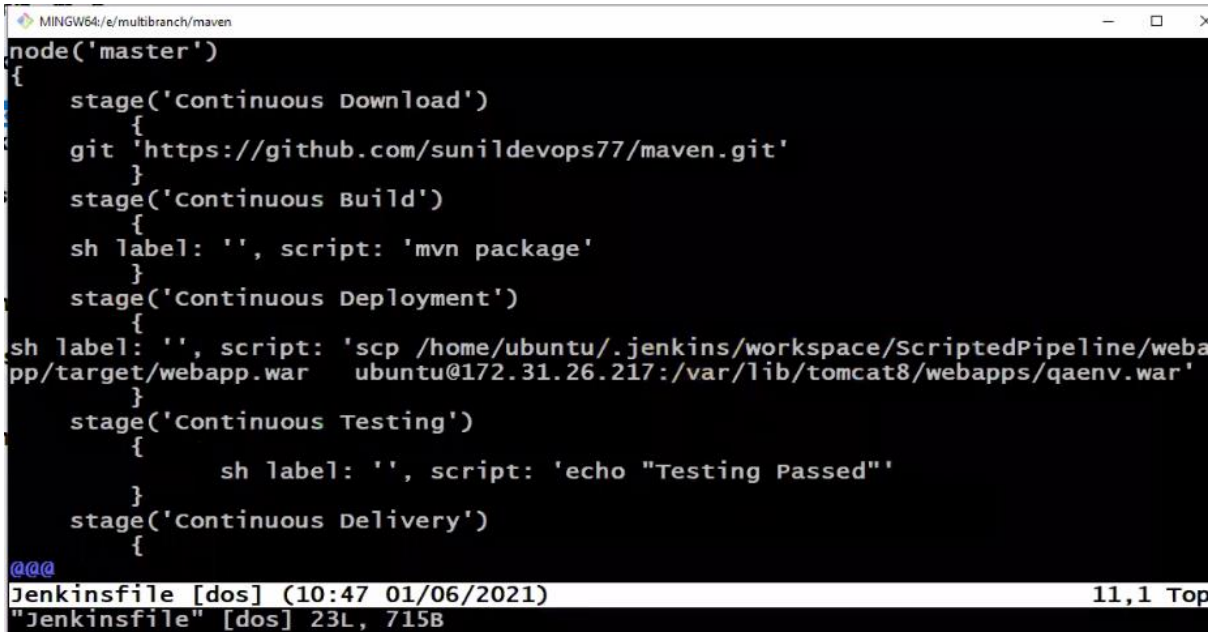
$ ls----loans branch has also have Jenkins file



$ vim Jenkinsfile --

can see all the 5 stages.(Along with the code, the developer will also create separate Jenkins file for every branch.

)



Lets make it only two stages --- stage name also followed by loans


```
node('master')
{
stage('ContinuousDownload_loans')
    {
  git 'https://github.com/sunildevops77/maven.git'
    }
stage('Continuousbuild_loans')
    {
  sh label: '', script: 'mvn package'
    }
```

}

:wq

Git status----untracked file

```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$ git status
On branch loans
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Jenkinsfile

no changes added to commit (use "git add" and/or "git commit -a")

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$
```

$ Git add .---- moved to staging area

$ git commit -m "c"---- staging area to local reposit

```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$  git add .

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$  git commit  -m "c"
[loans cb987fd] c
 1 file changed, 11 insertions(+), 23 deletions(-)
 rewrite Jenkinsfile (81%)

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$
```

Observe (master branch is having jenkins file.

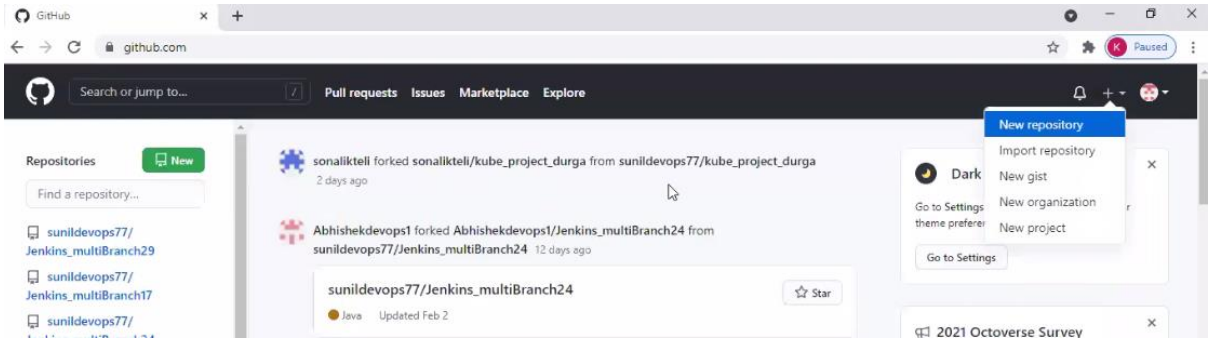
Loans branch is having jenkins file)

$ git checkout master

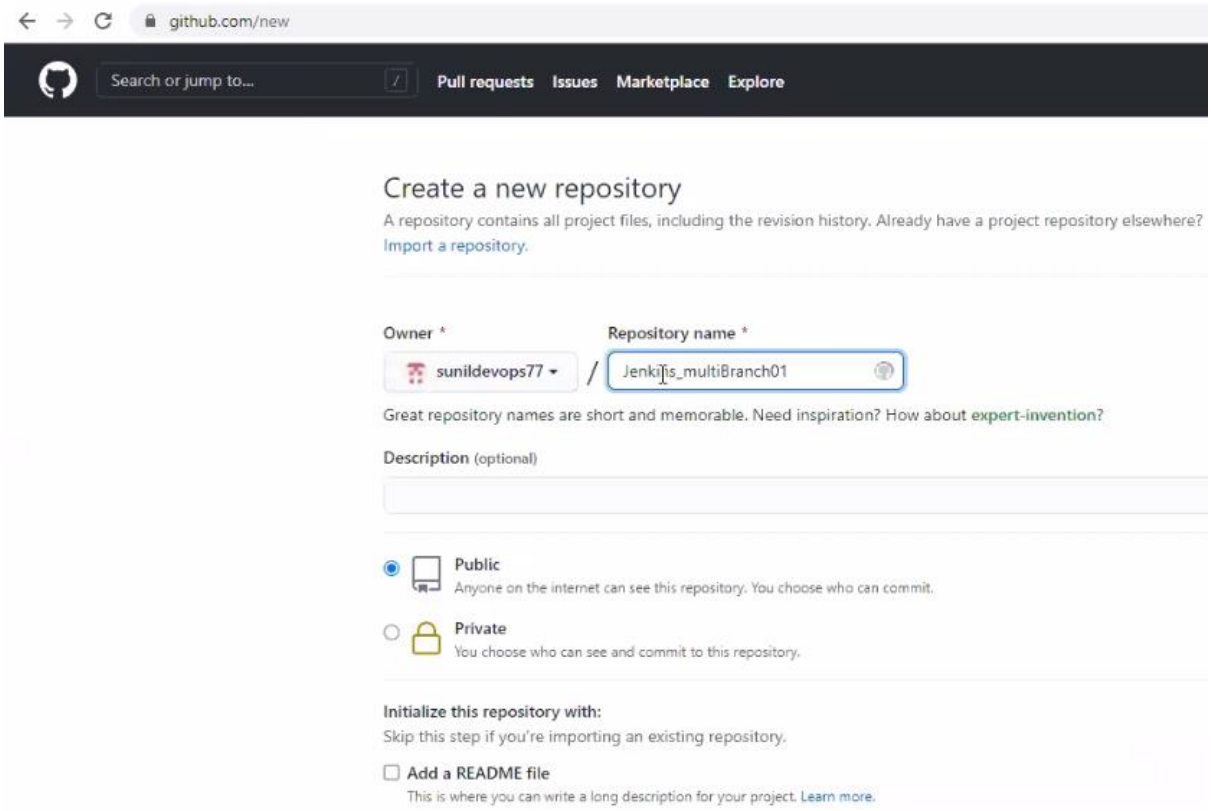
```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (loans)
$  git checkout master
Switched to branch 'master'

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$
```

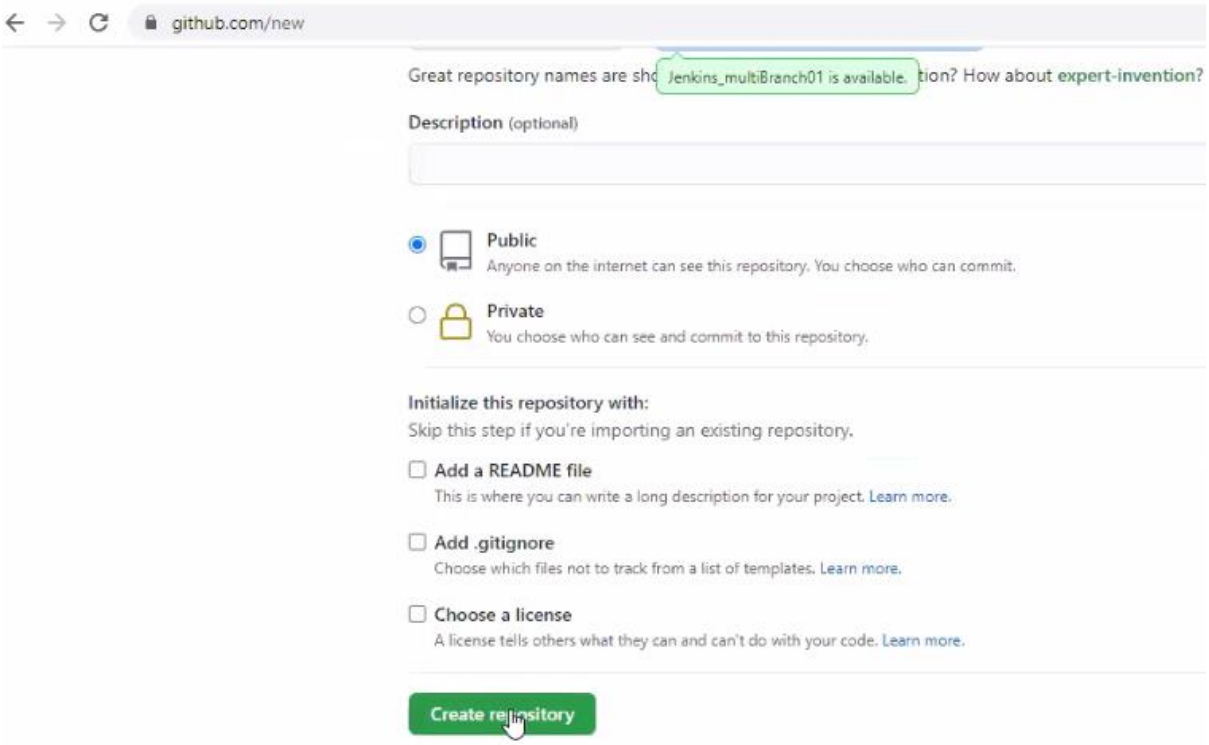
We saved in local repo then we go to git hub for uploading

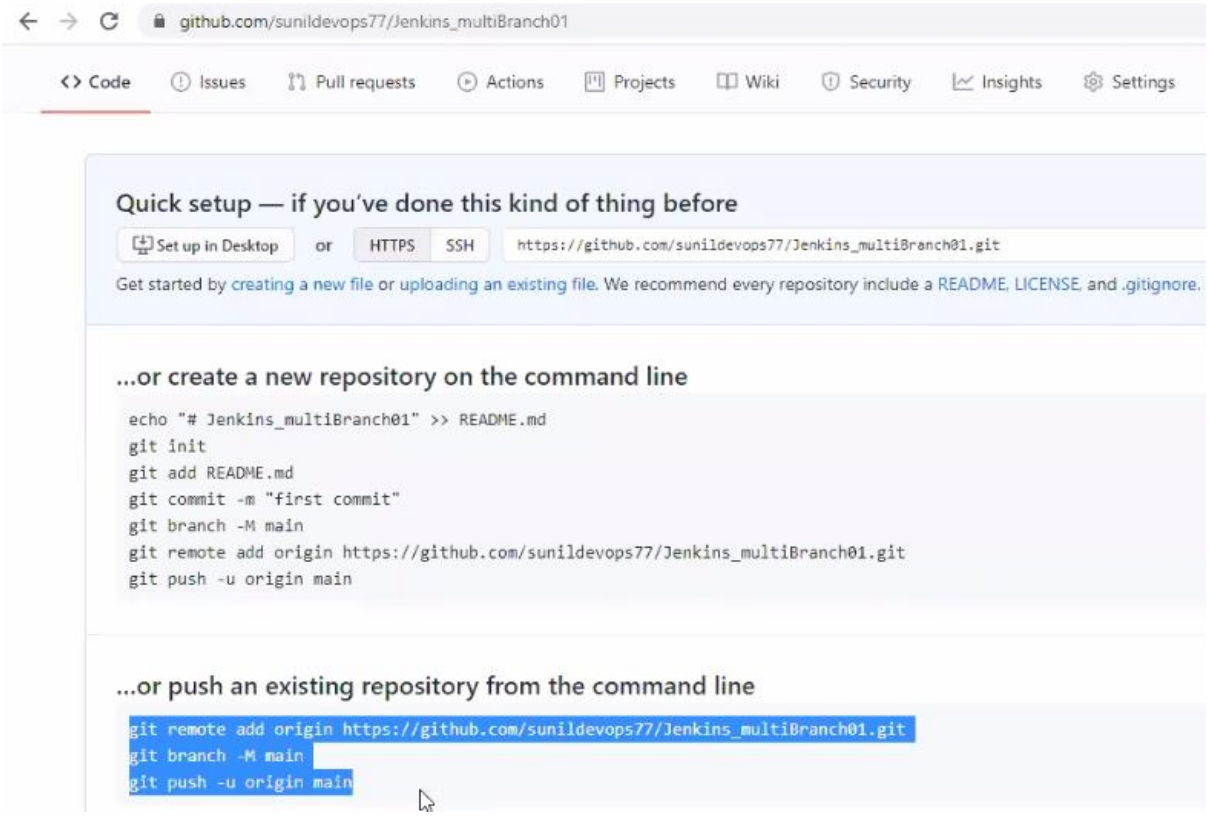
Create new repository in github



Repository name - Jenkins_multiBranch24







Copy above code and paste in terminal

$ git remote add origin https://github.com/sunildevops77/Jenkins_multiBranch.git

This will link Local rep to Remote repo

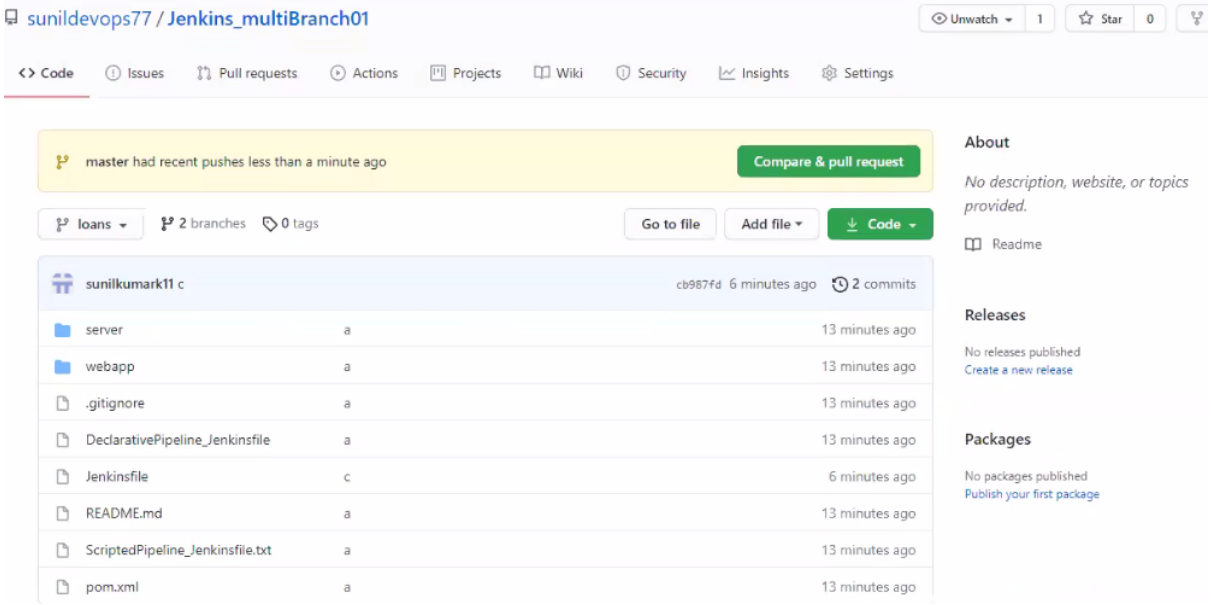
```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$ git remote add origin https://github.com/sunildevops77/Jenkins_multiBranch01.g
it

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$
```

$ git push -u origin --all (as we want to push all branches , here like both master and loan)

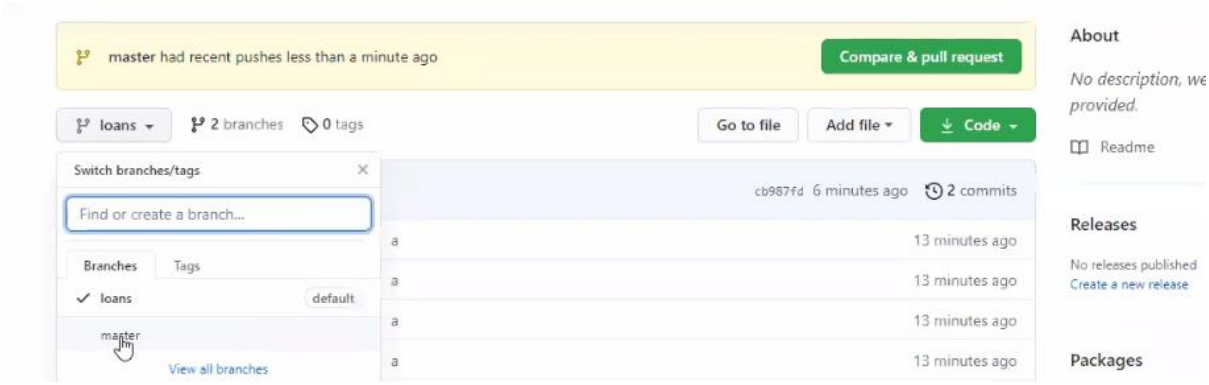
```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$ git push -u origin --all
Enumerating objects: 41, done.
Counting objects: 100% (41/41), done.
Delta compression using up to 4 threads
Compressing objects: 100% (23/23), done.
Writing objects: 100% (41/41), 5.48 KiB | 311.00 KiB/s, done.
Total 41 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), done.
To https://github.com/sunildevops77/Jenkins_multiBranch01.git
 * [new branch]      loans -> loans
 * [new branch]      master -> master
Branch 'loans' set up to track remote branch 'loans' from 'origin'.
Branch 'master' set up to track remote branch 'master' from 'origin'.

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$
```

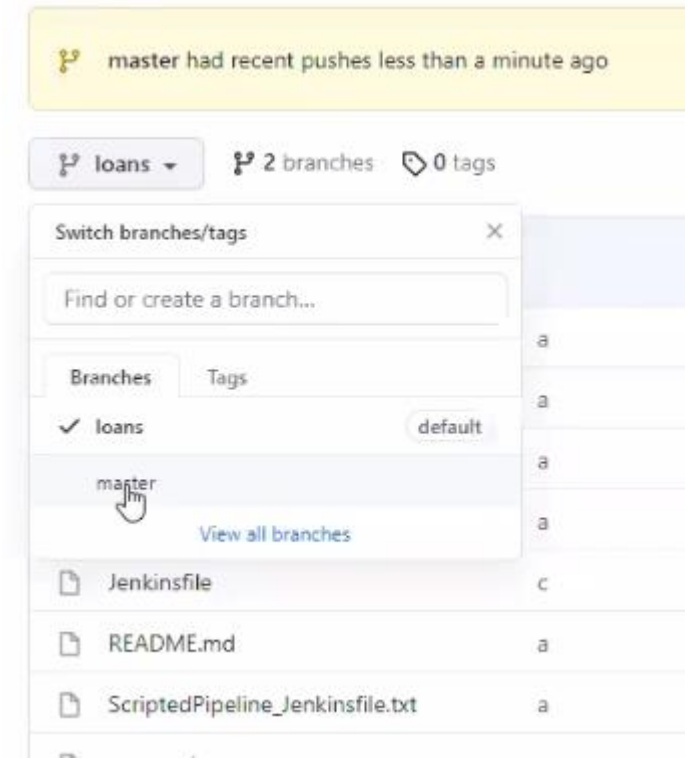
Now the code uploaded to the GITHUB Repo,,,,,,,,refresh that we can see

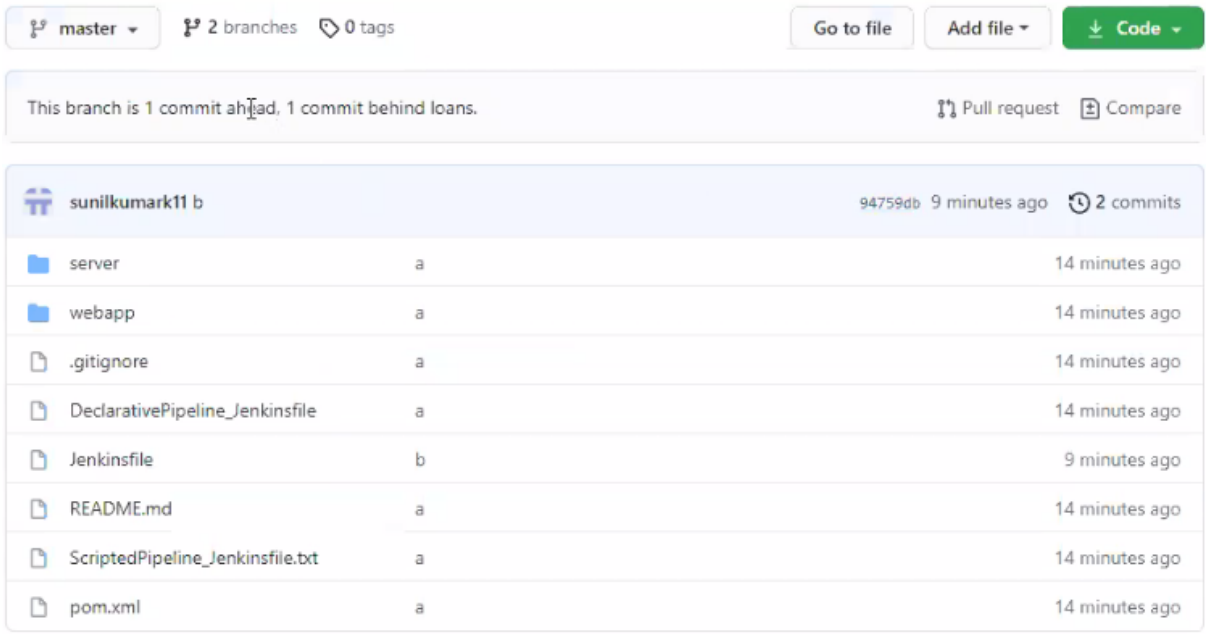
(Check the remote repository)

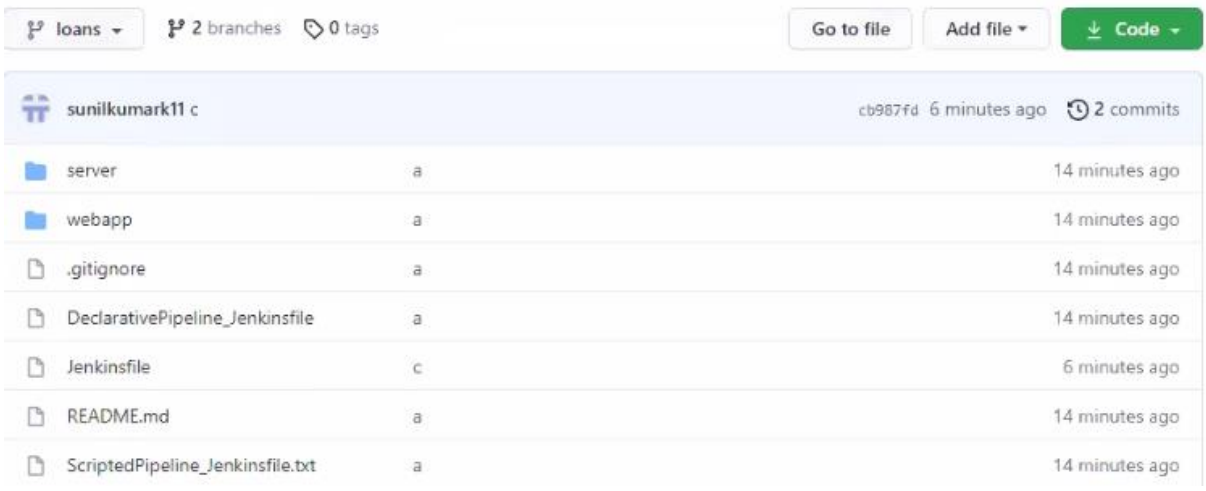


Can see 2 brancges


master had recent pushes less than a minute ago
Compare & pull request
loans
2 branches
0 tags
Go to file
Add file
Code
Switch branches/tags
Find or create a branch...
Branches
Tags
loans
default
master
View all branches
cb987fd 6 minutes ago
2 commits
a 13 minutes ago
a 13 minutes ago
a 13 minutes ago
a 13 minutes ago
About
No description, we
provided.
Readme
Releases
No releases published
Create a new release
Packages


master had recent pushes less than a minute ago
loans
2 branches
0 tags
Switch branches/tags
Find or create a branch...
Branches
Tags
loans
default
master
View all branches
Jenkinsfile
README.md
ScriptedPipeline_Jenkinsfile.txt


master
2 branches
0 tags
Go to file
Add file
Code
This branch is 1 commit ahead, 1 commit behind loans.
Pull request
Compare
sunilkumark11 b
94759db 9 minutes ago
2 commits
server a 14 minutes ago
webapp a 14 minutes ago
.gitignore a 14 minutes ago
DeclarativePipeline_Jenkinsfile a 14 minutes ago
Jenkinsfile b 9 minutes ago
README.md a 14 minutes ago
ScriptedPipeline_Jenkinsfile.txt a 14 minutes ago
pom.xml a 14 minutes ago

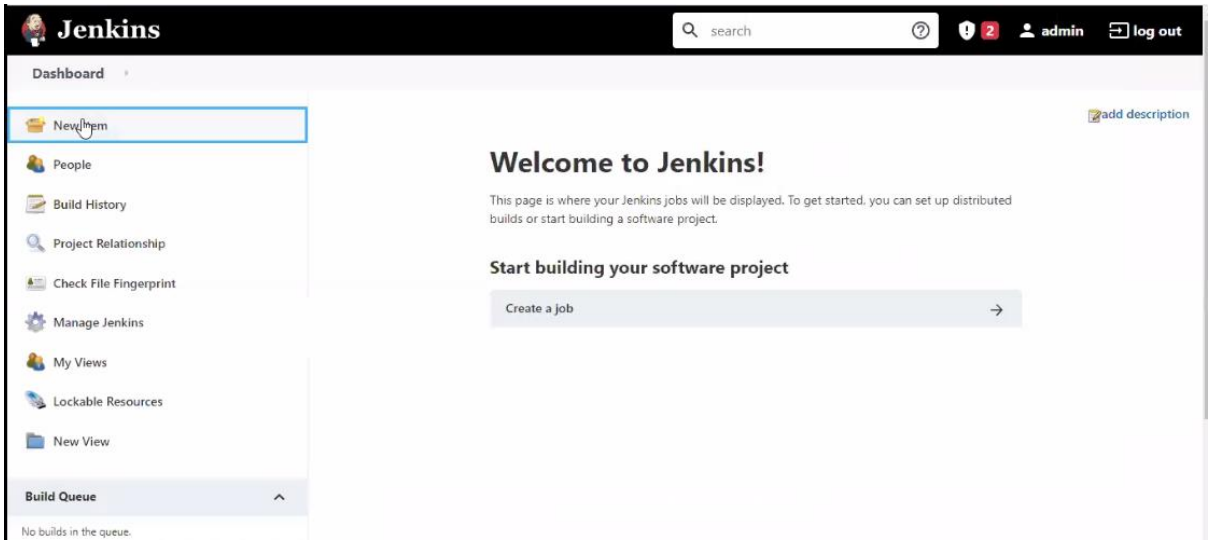


+++This is developers activity+++++++++

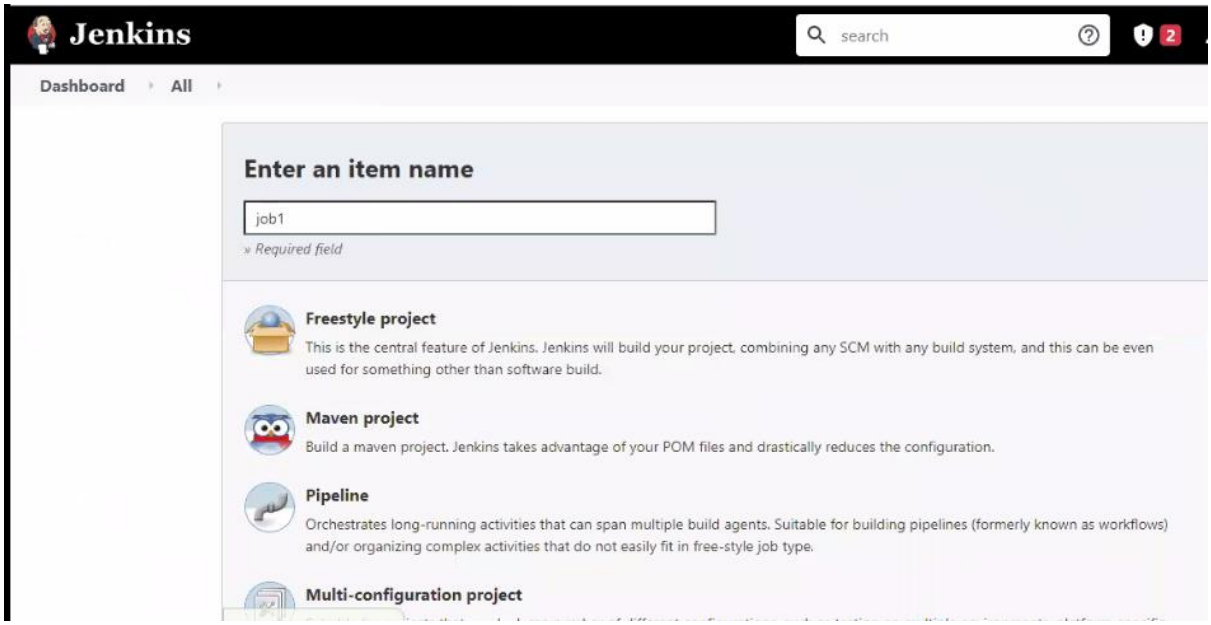
Will create a Jenkins job to process both Branches.

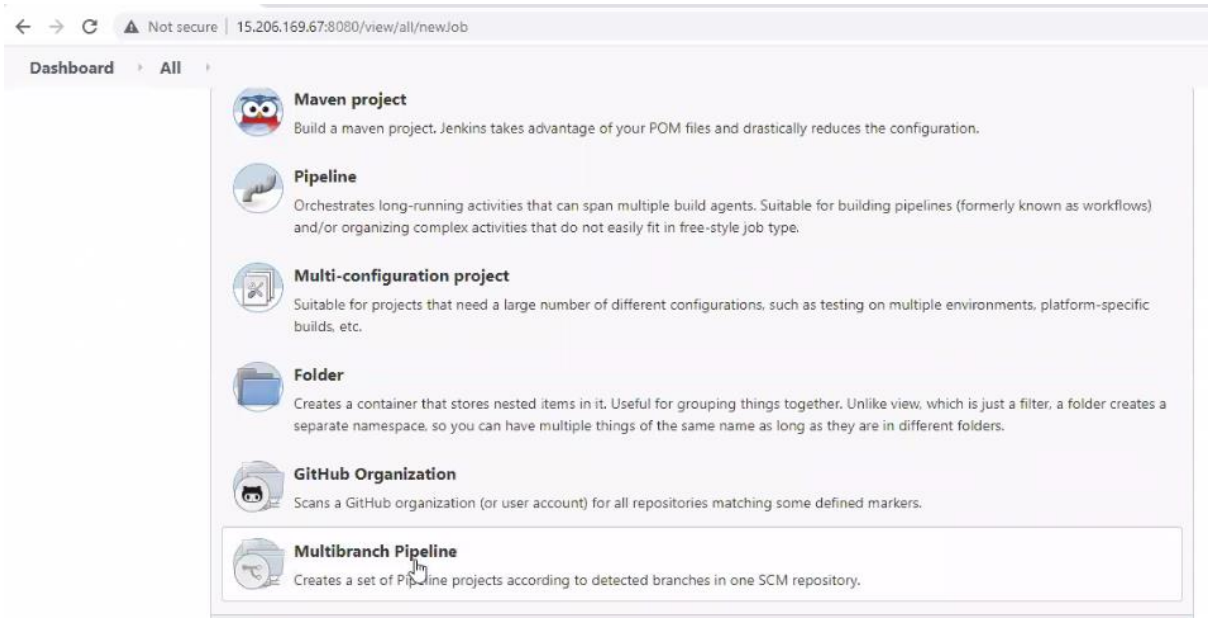
Login to jenkins

New item – MultiBranchPipeline

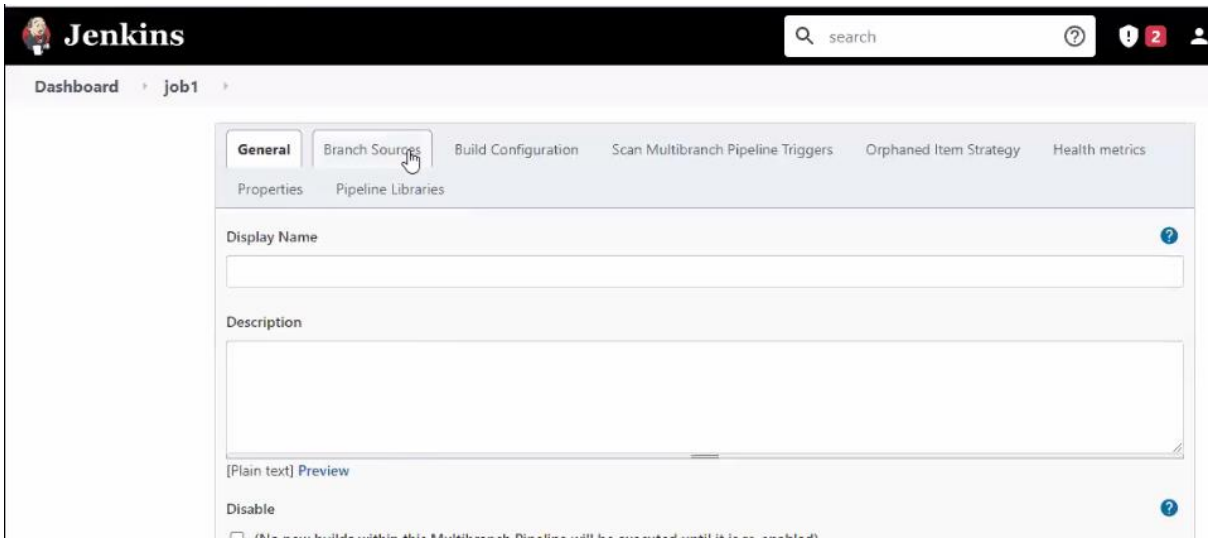


Gve the jb name-Select multibranch Pipeline gi



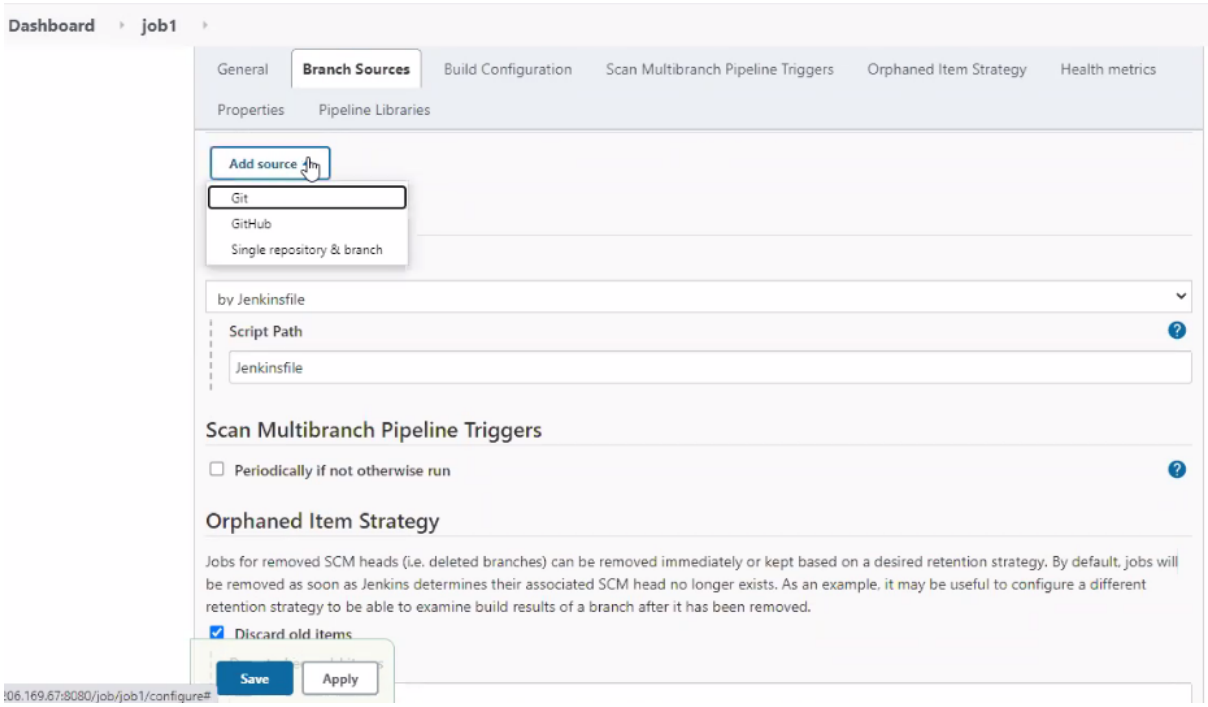


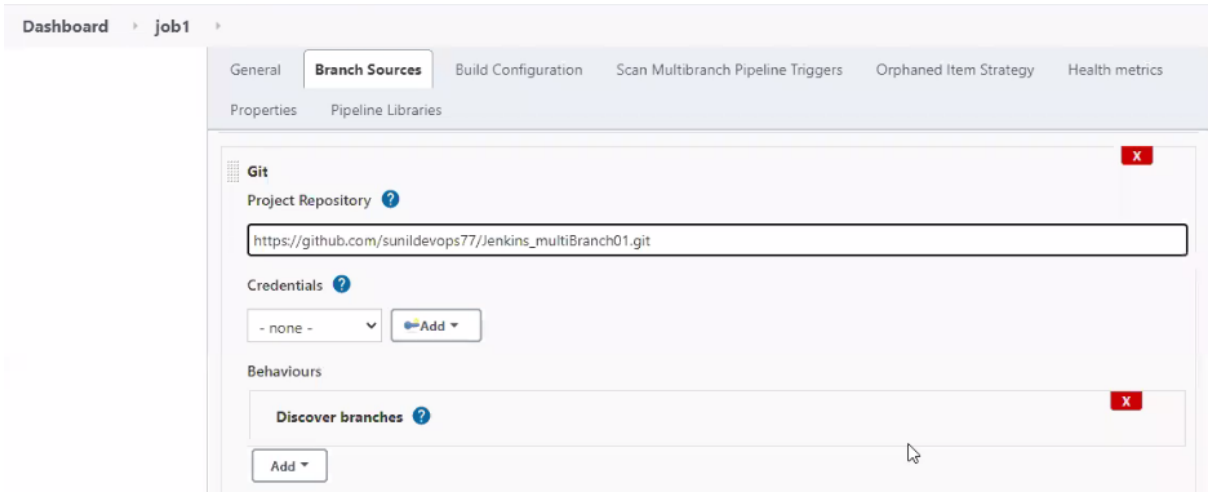
Branch Sources



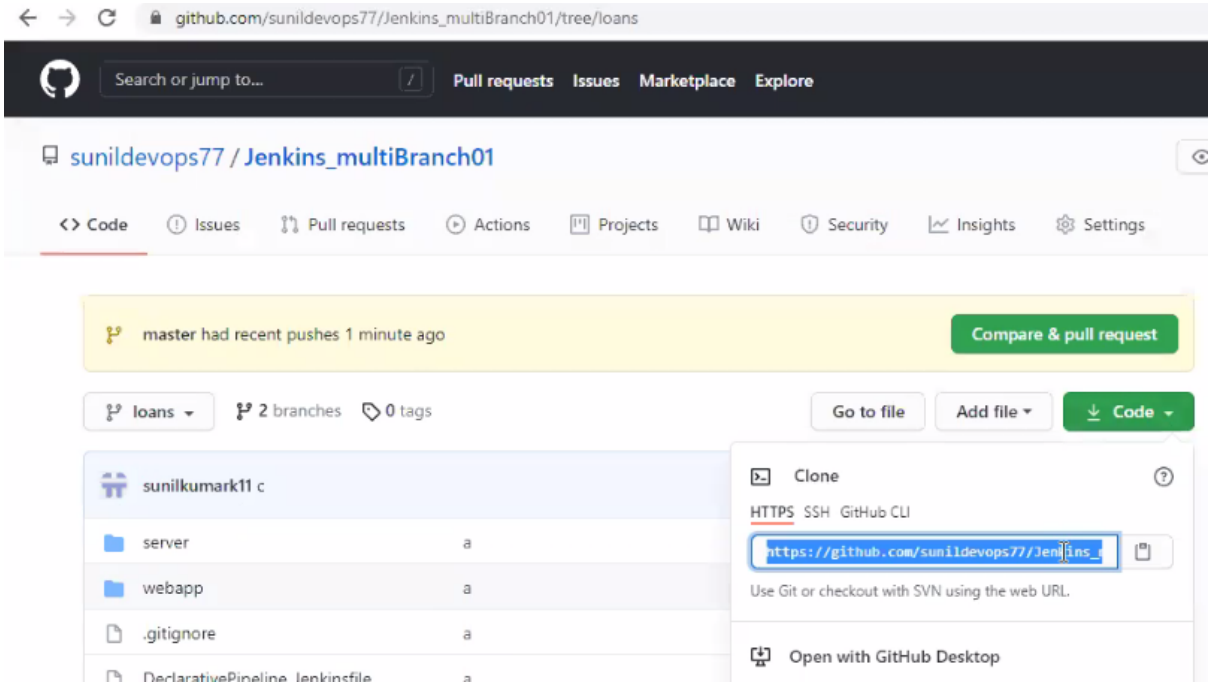
Add source

Git





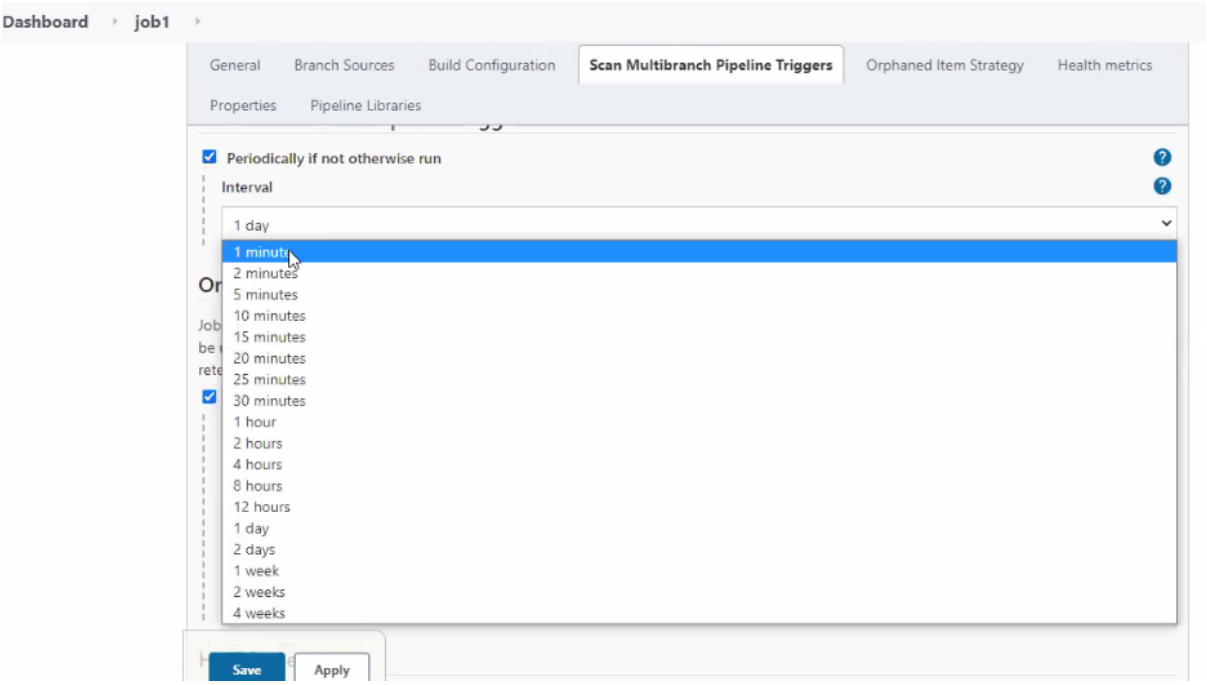
Project Repository -- https://github.com/sunildevops77/Jenkins_multiBranch24.git



Scan multiline pipeline triggers

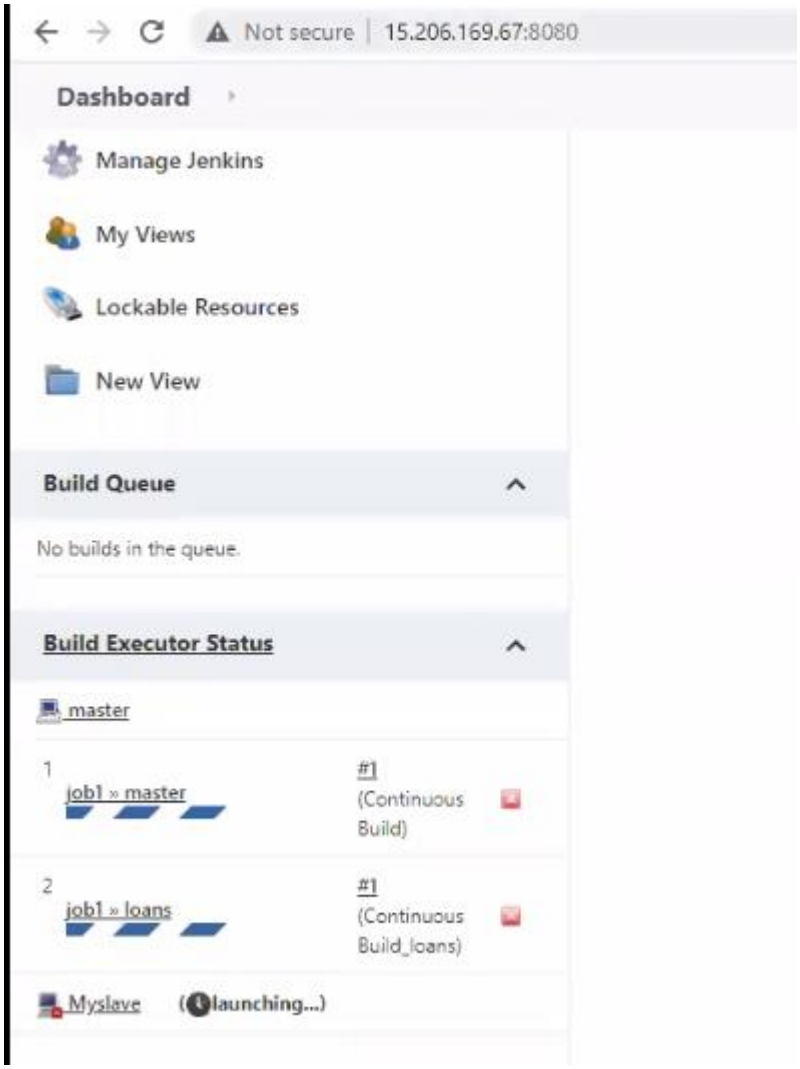
Check periodically if not otherwise

Interval - 1 minute

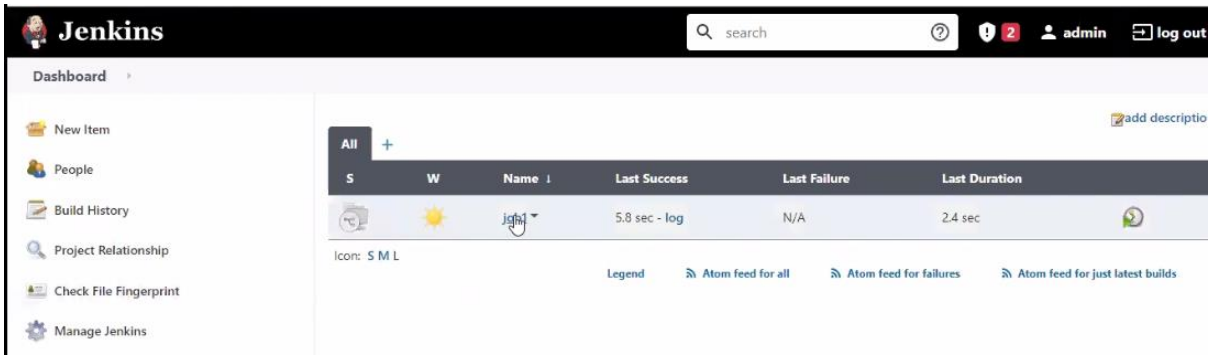


Apply --- Save then backto Dashboard

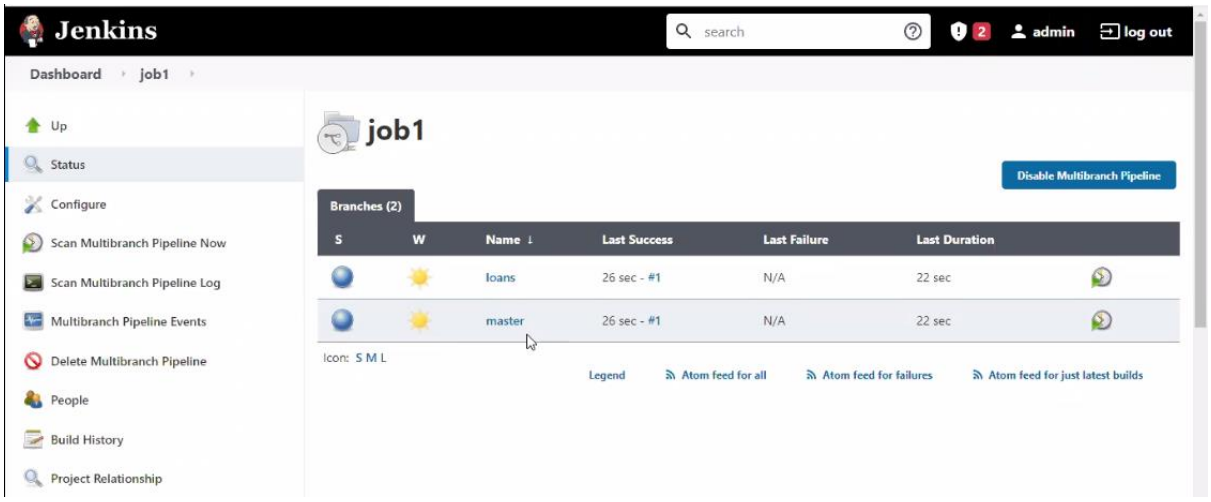
By this time it will be started.



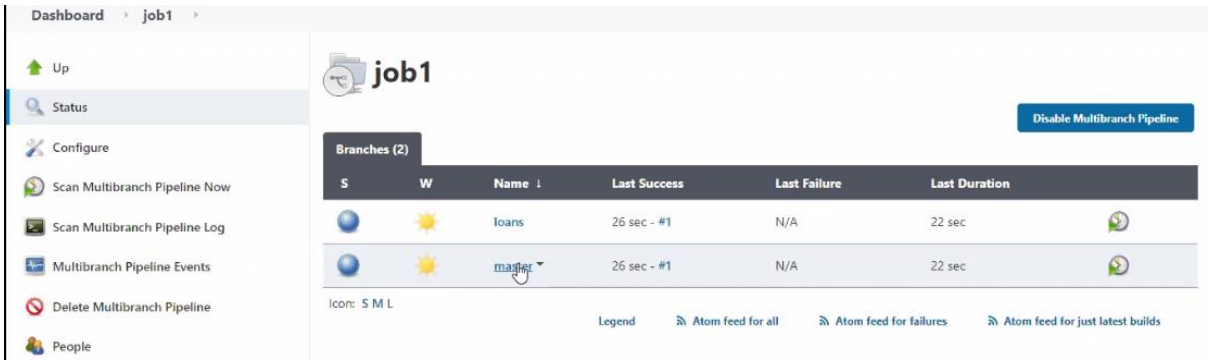
This job will check github every minute.



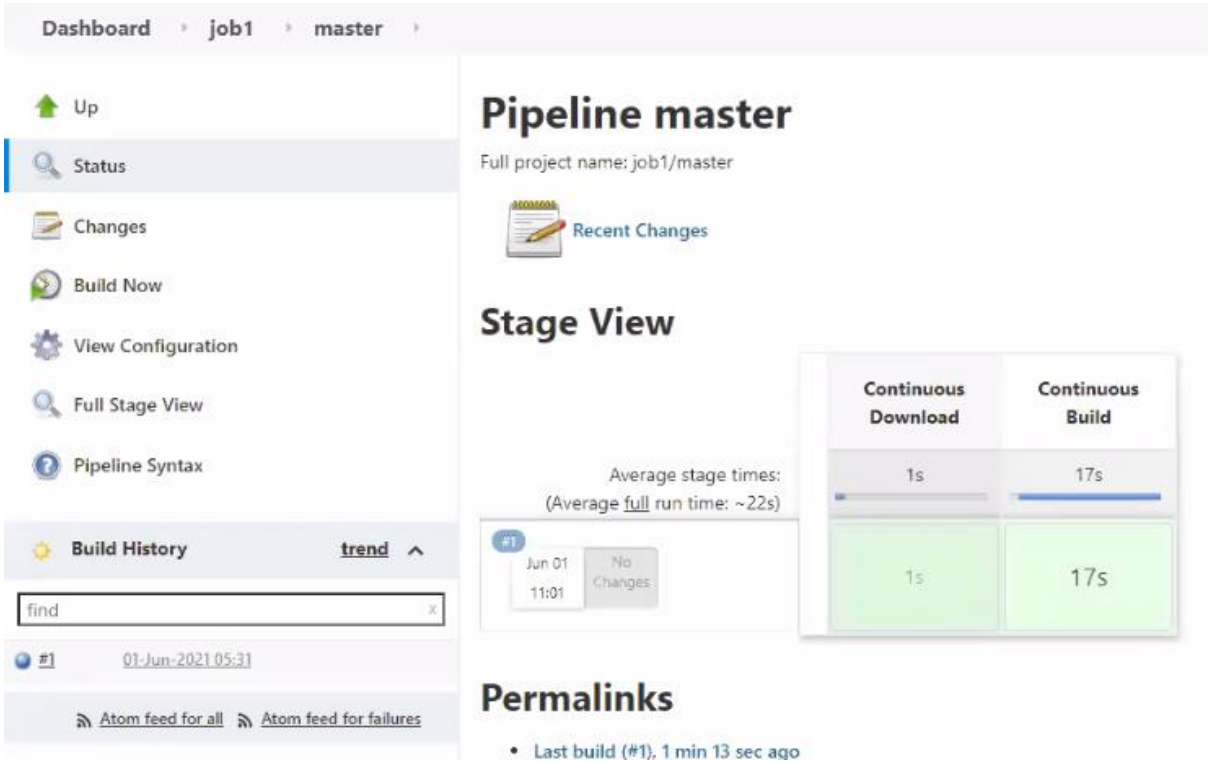
Created one job in jenkins which Processing 2 branches bcz 2 barnches. Present in the github repo

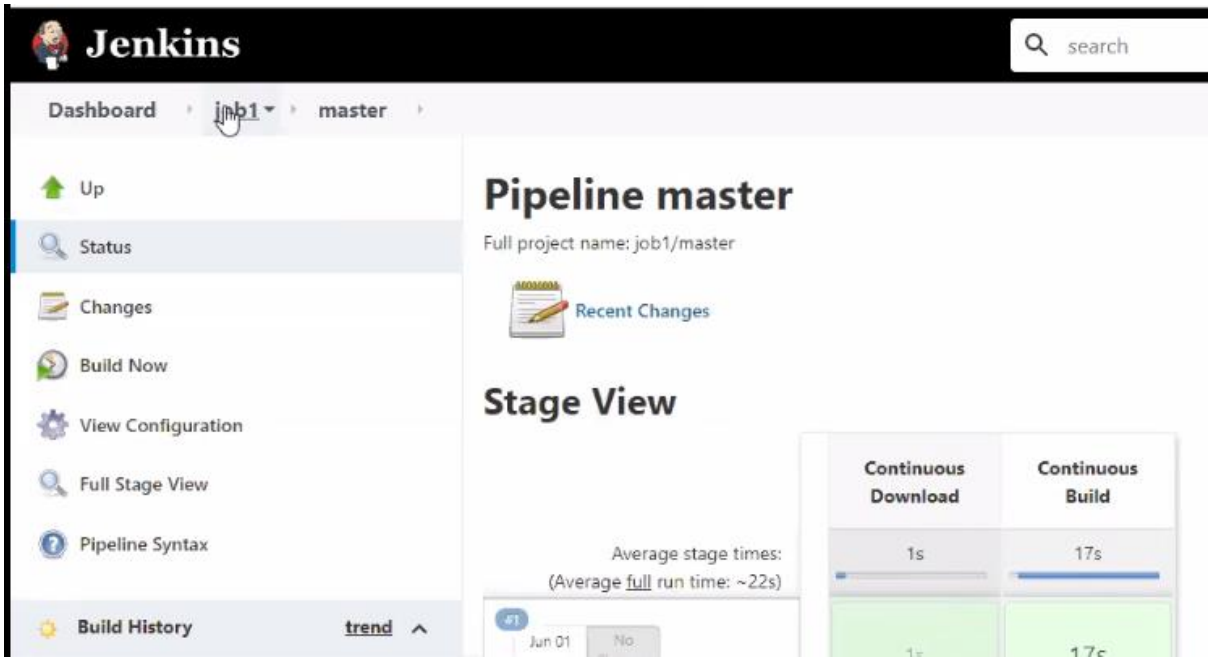


Click on master branch



Performing only 2 stages as jenkins file master has only 2 stages present in github

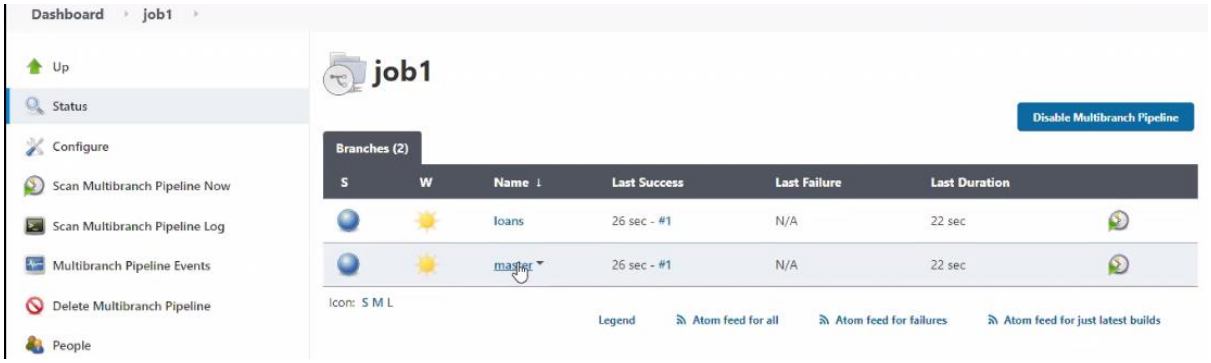


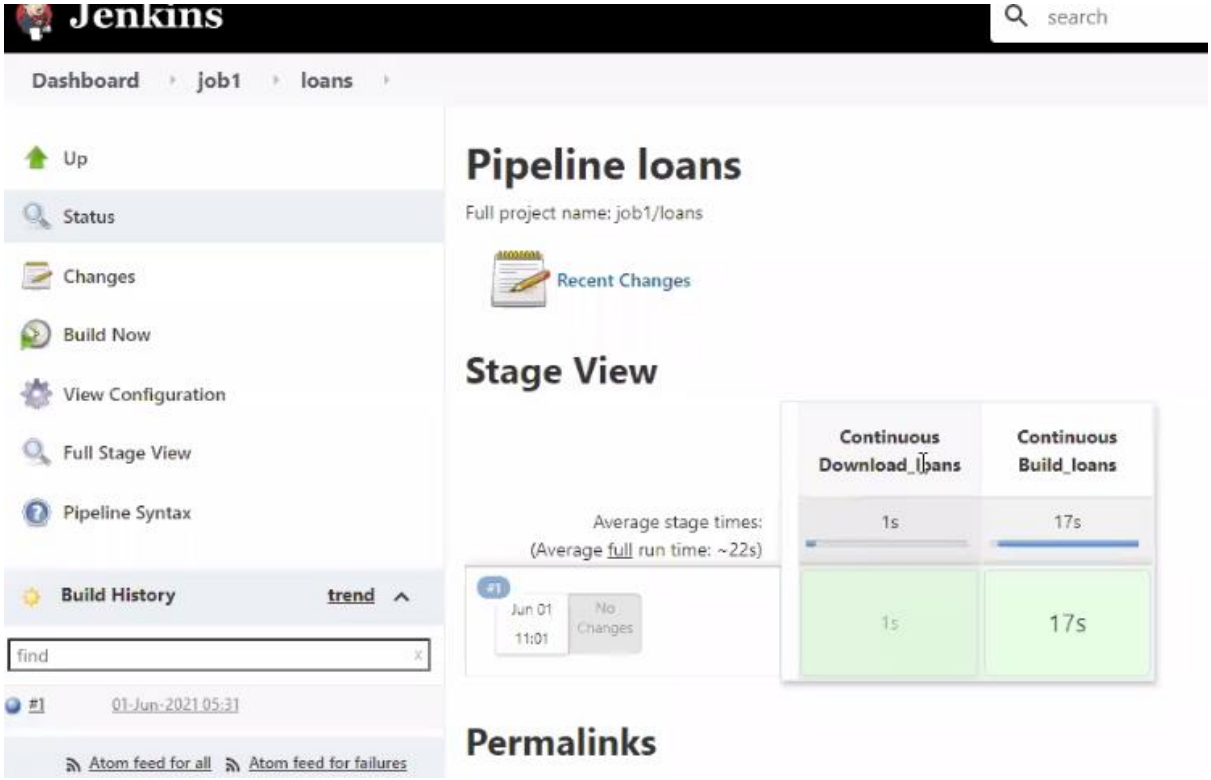


Select multibranch pipeline

You will find two branches

Select loans , we can see two stages

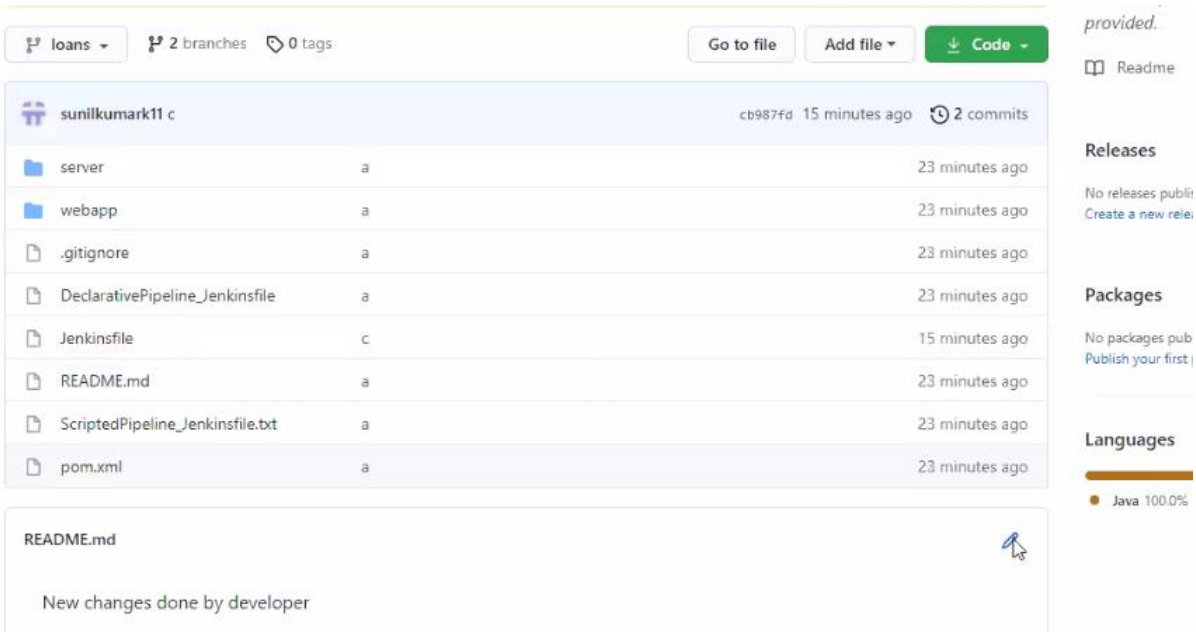




Select master, we can see two stages

Let's say, developer will make changes and push to the repository

Or we can make changes in README.md Loan in GITHUB site directly

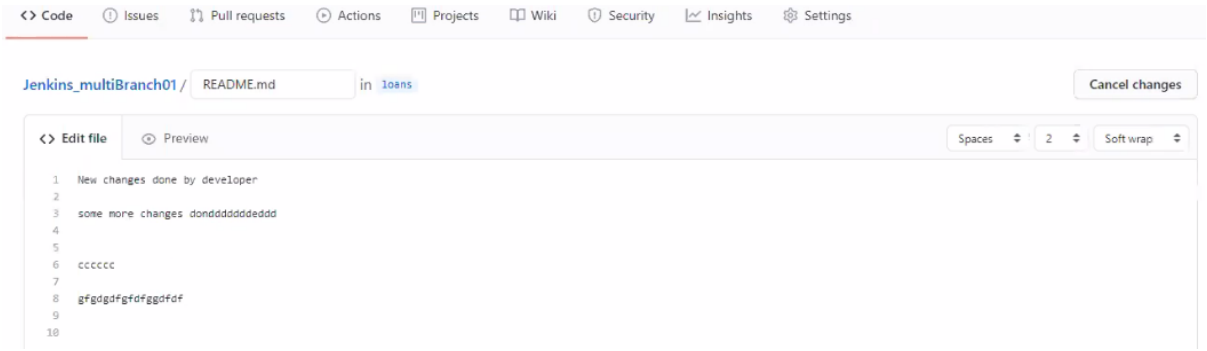


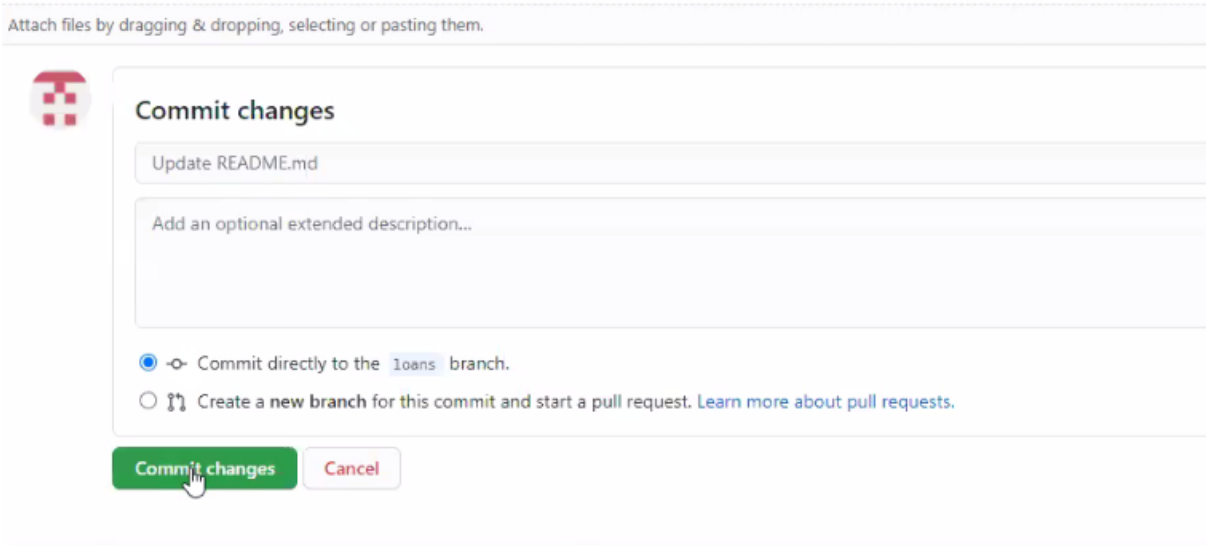
Assuming Developer is making some changes in source code and then committing changes but its not good we need to change in source code and then git push to github.

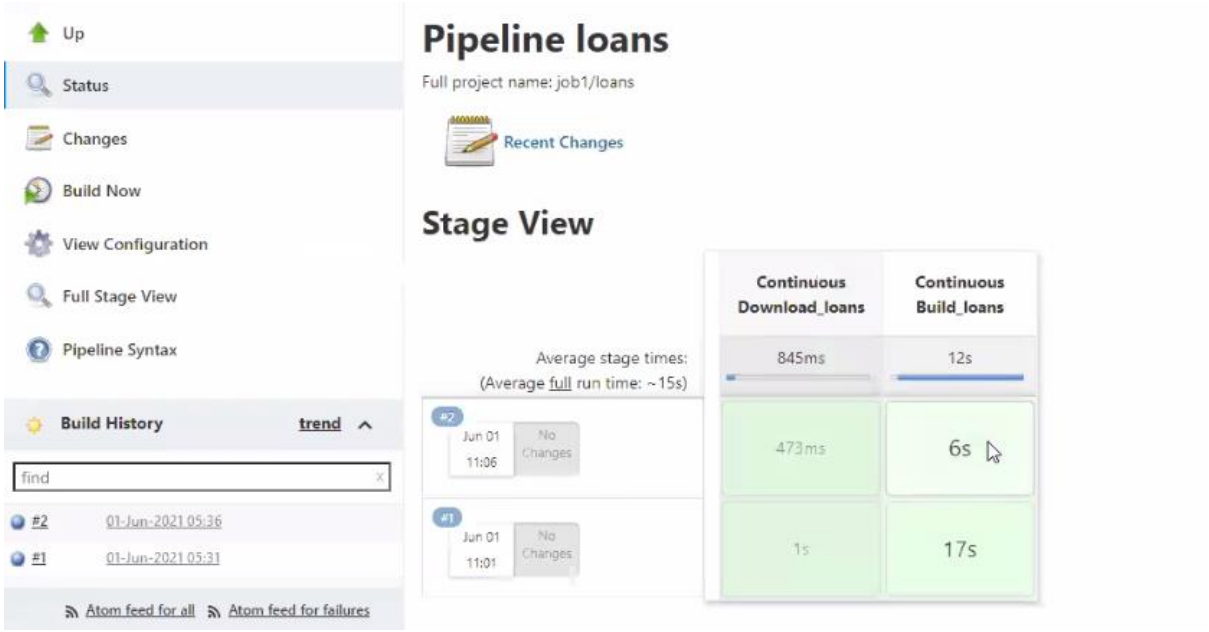
Code has been updated in loans branch.

So particular branch source code has been updated except master branch then it will go to the Jenkins to auto execute each stage for every minute which we had setup interval as above.

The Jenkins will check for every 1 minute frequency to check the repository if any code modified or not.







$ 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

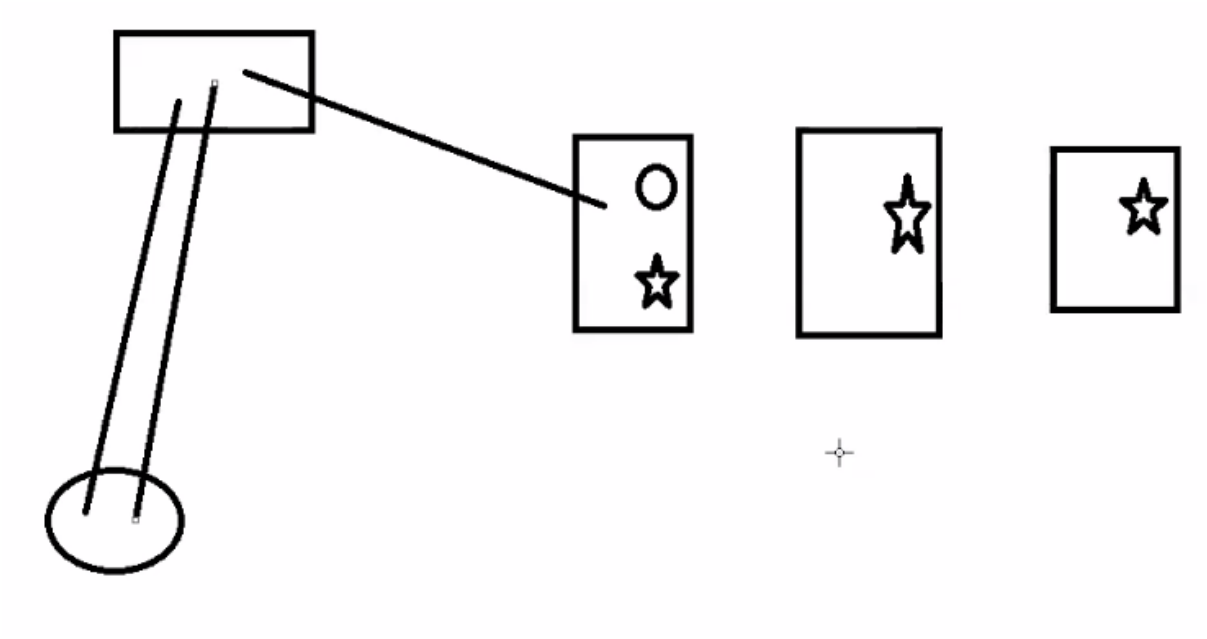
To push all the branches

$ git push -u origin --all

Observation: Job will start automatically.

Whenever the developer update/modified any code in any branch, the file will be overwritten then jenkins will pull the code from git hub to trigger jobs and perform all the stages automatically as per the interval. like Dev t0 QA

Dev to Prod.



++

2nd June

++

Email Integration

In case, if a job fails, we need to send notification to specific dev team through specific mails.

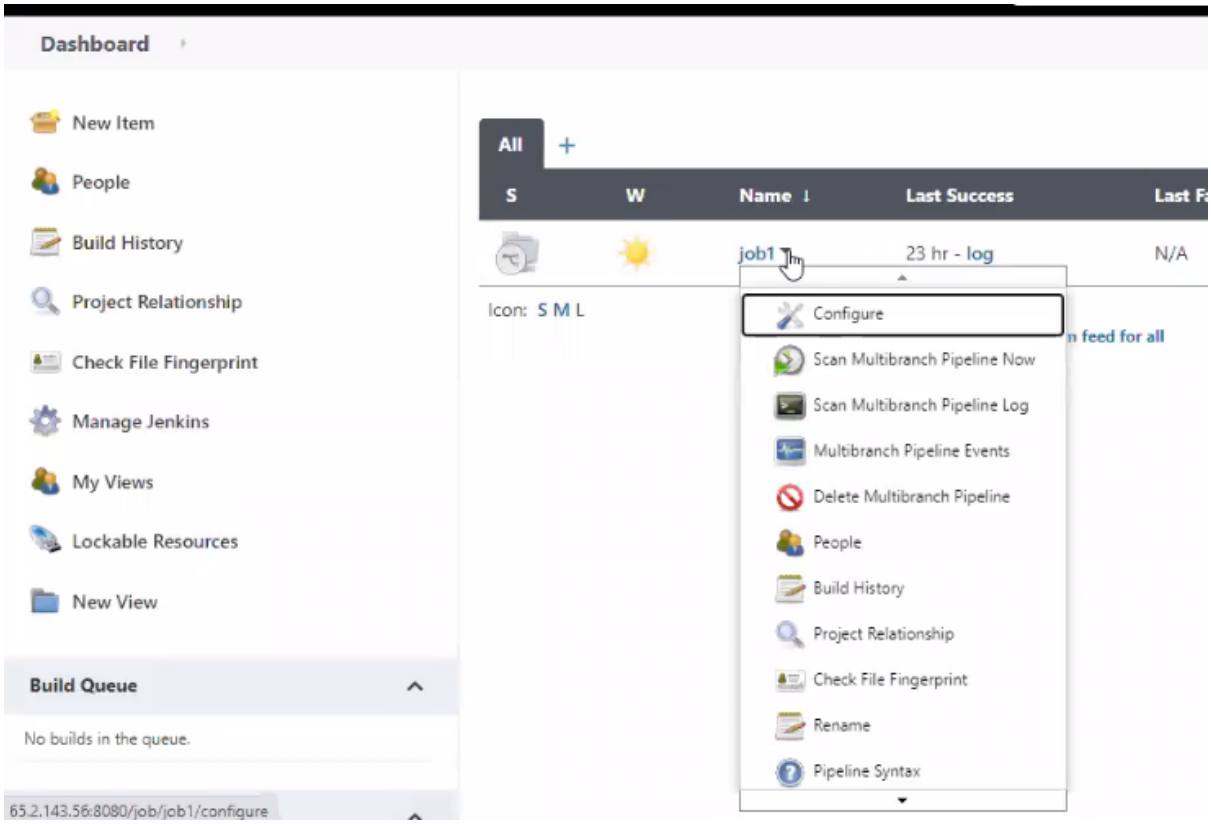
For that we need to integrate Jenkins to smtp server.

1.connect dev instance

2.start java

3.access Jenkins through google

4.delete old job



We are now integrating jenkins with gmail smtp server.

Search in google "gmail smtp server"

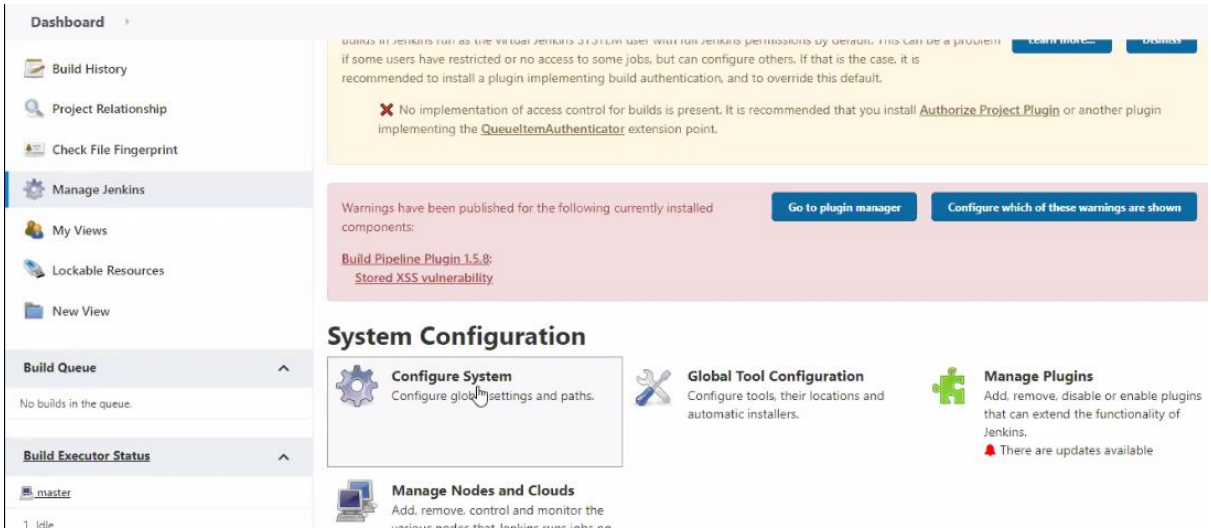
SMTP server: smtp.gmail.com

Port: 465

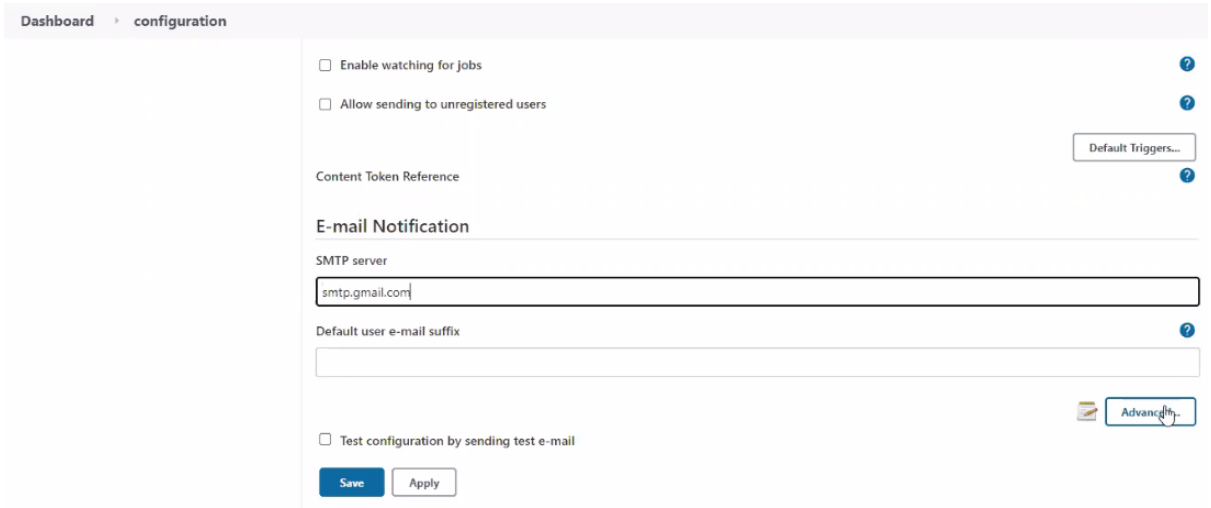
sunildevops77@gmail.com

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

Manage Jenkins --- Configure System



Email Notification



SMTP Server - smtp.gmail.com

Click on Advance button (with notepad icon)

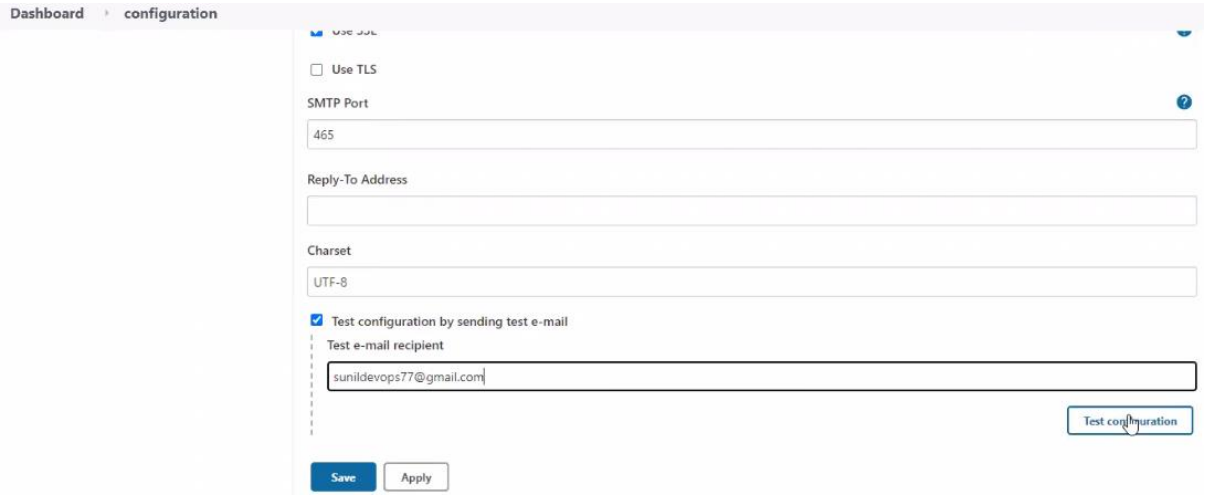
USE SMTP Authentication

Username - sunildevops77@gmail.com

Password - password for the above email

use SSL

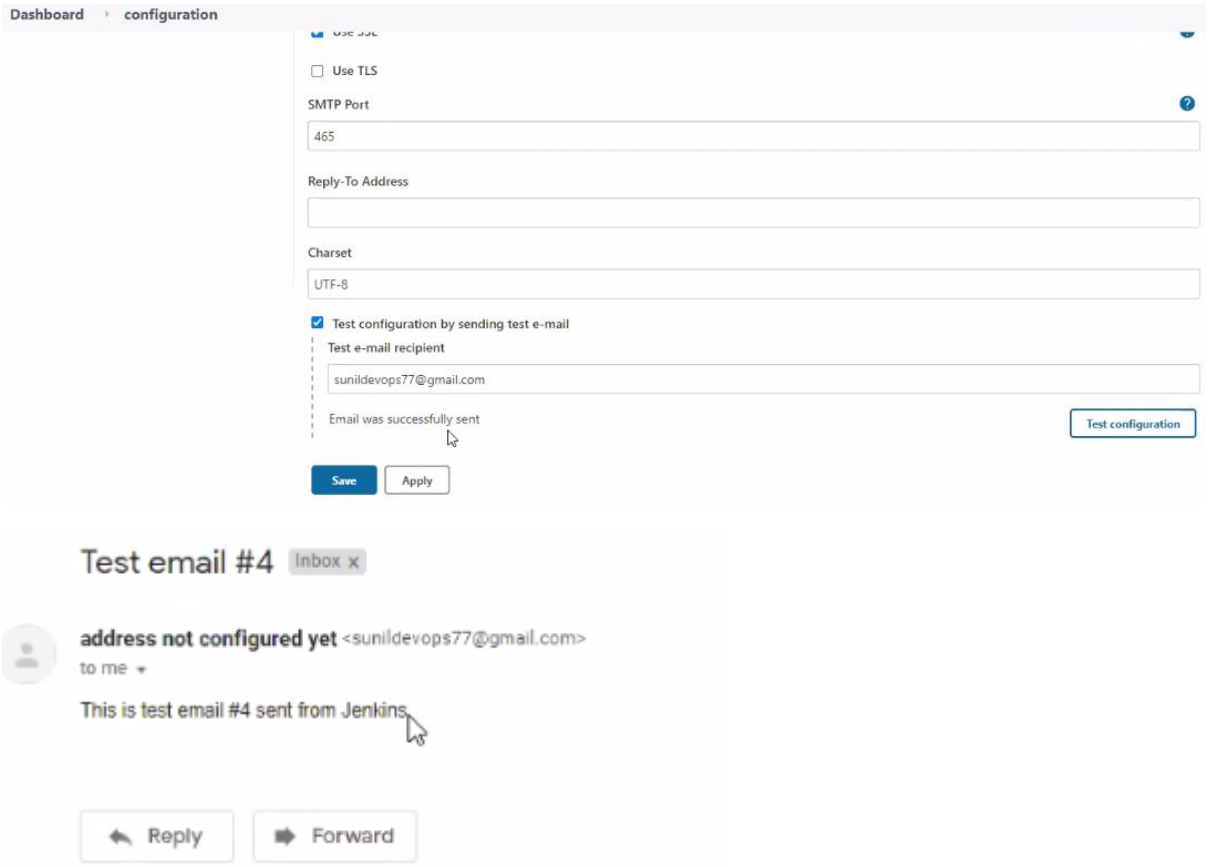
SMTP Port - 465



Test Configuration by sending e-mail.

Test email Receipent - sunildevops77@gmail.com

(We should get success message)



Then Apply and Save.

Gmail Settings to get email from jenkins

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

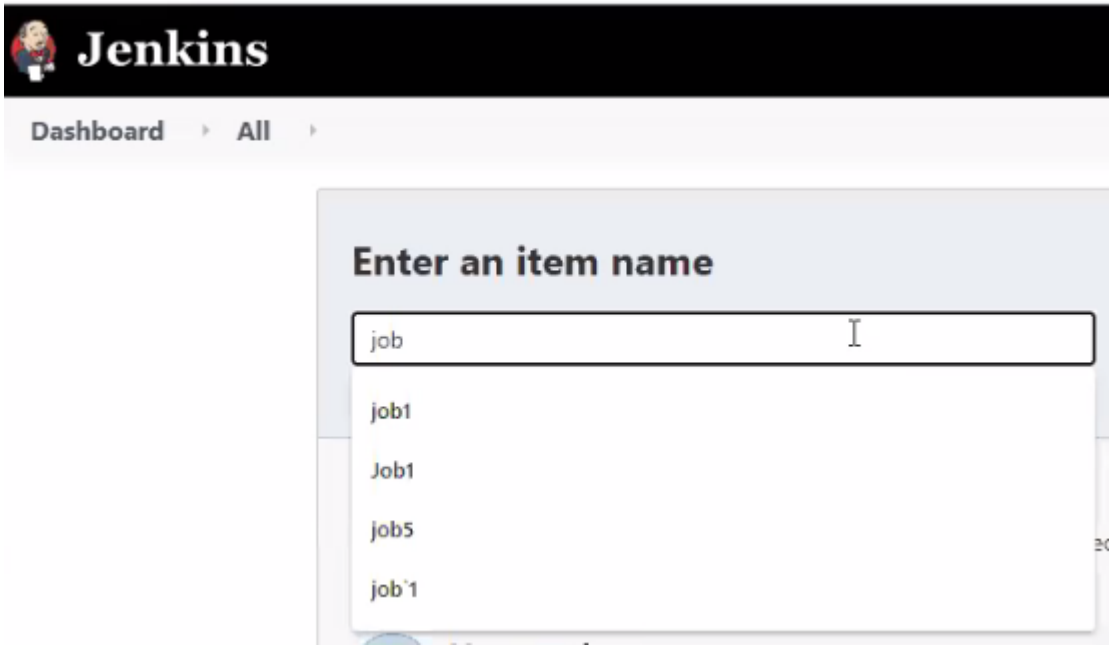
2) "Disable captcha gmail"

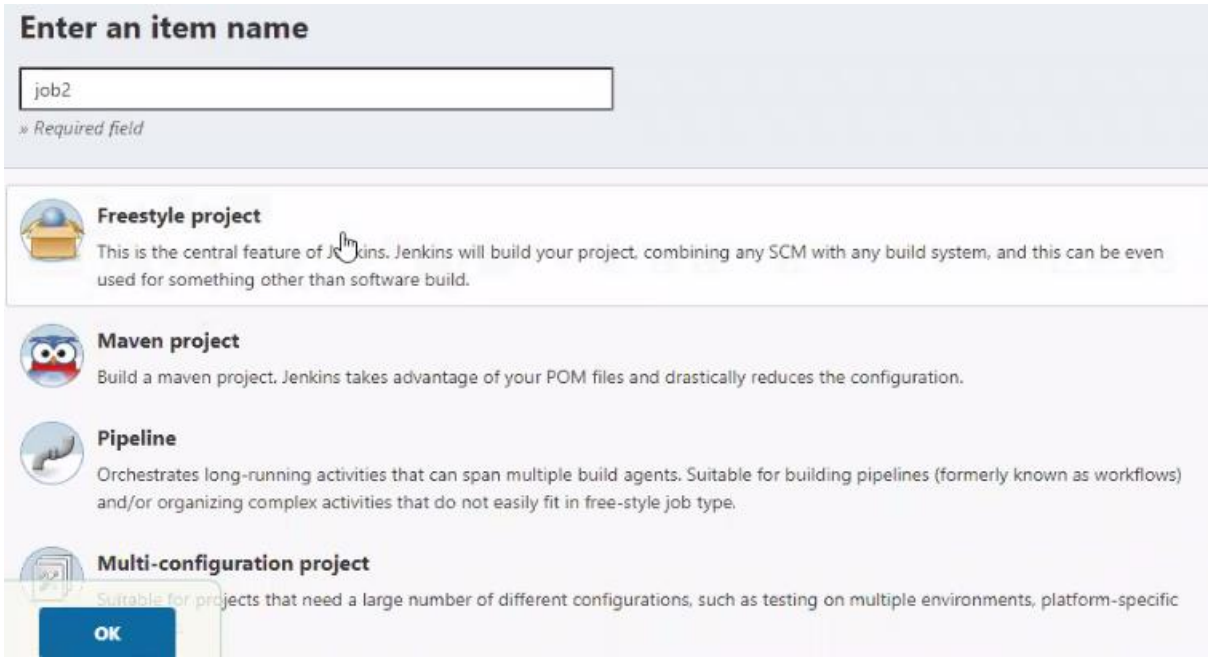
Search in google "Disable captcha gmail" -- Continue

++

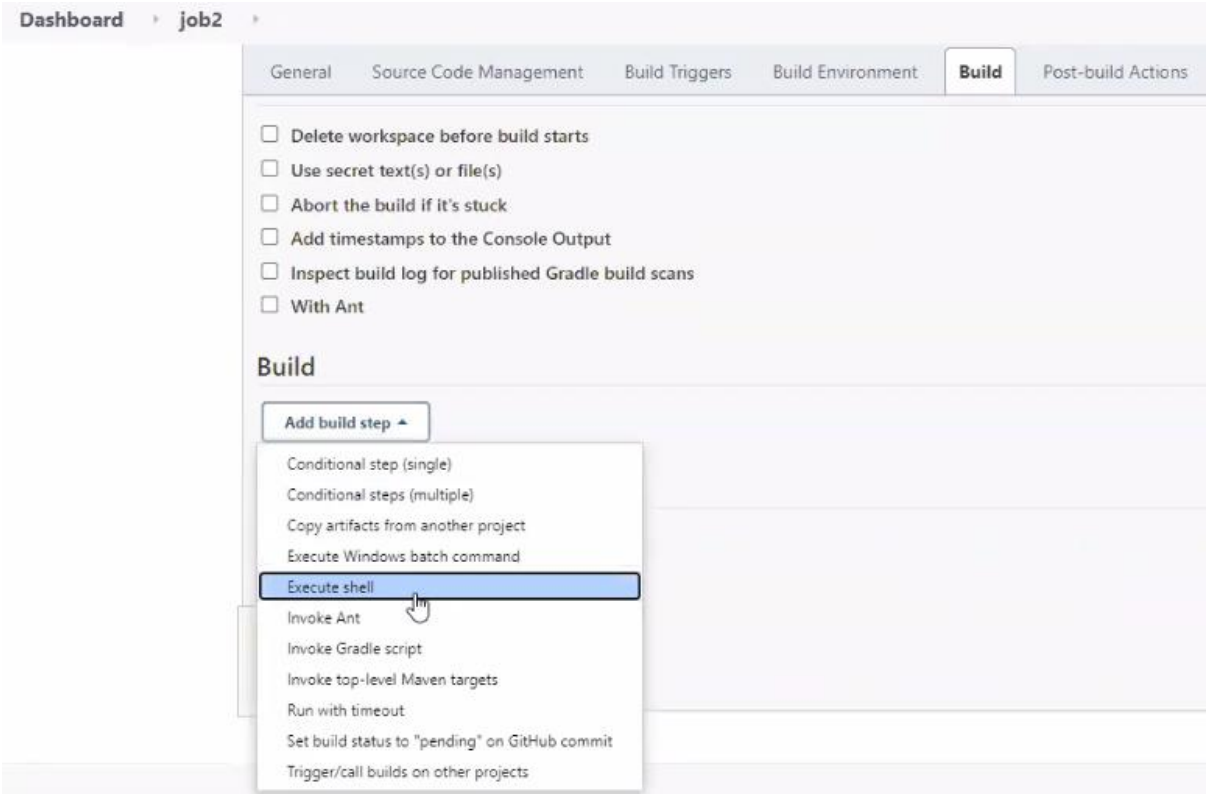
Its time to test

Enter job name ans select frees style→ok

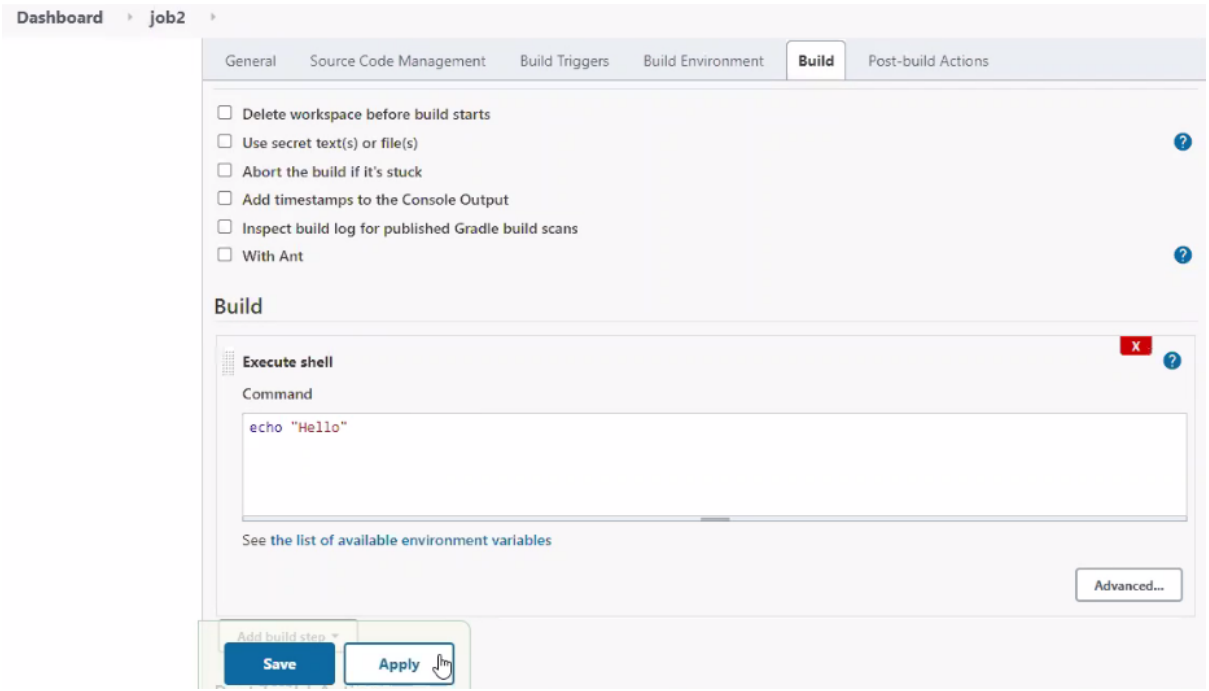




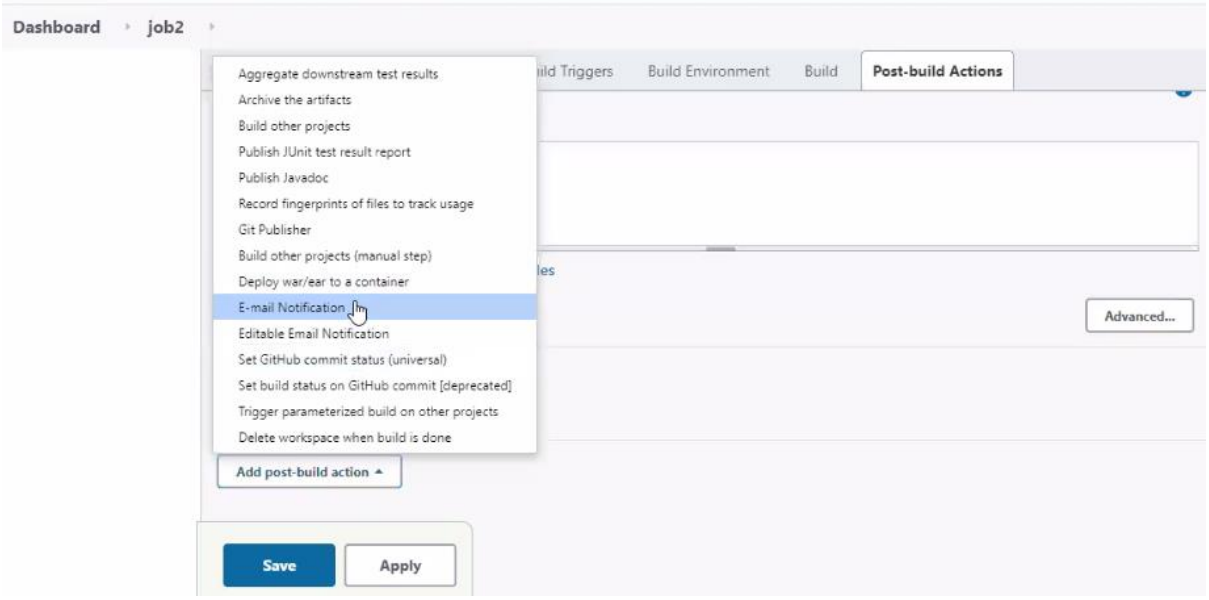
Build tab→add build step→Execute shell



Give echo statement→apply→save



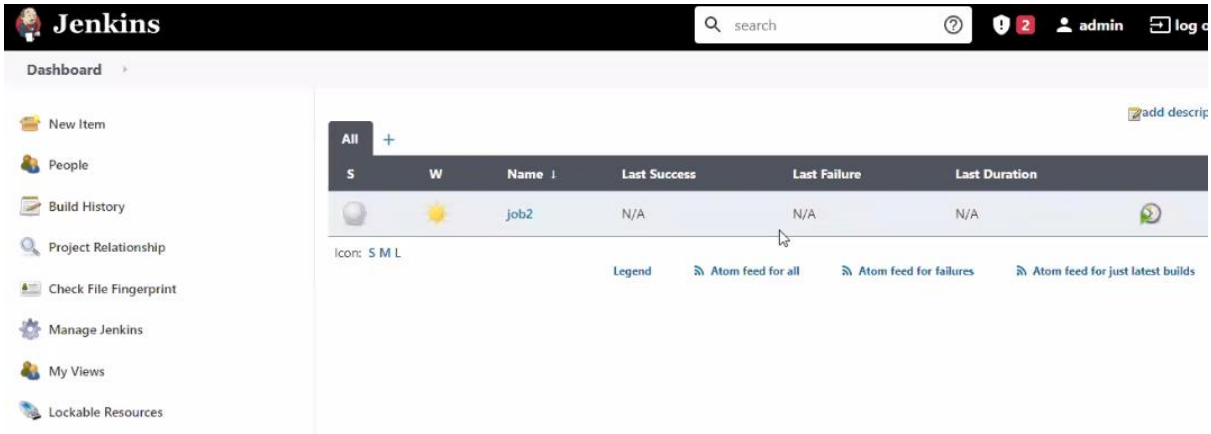
Post build actions→add post build→email notification

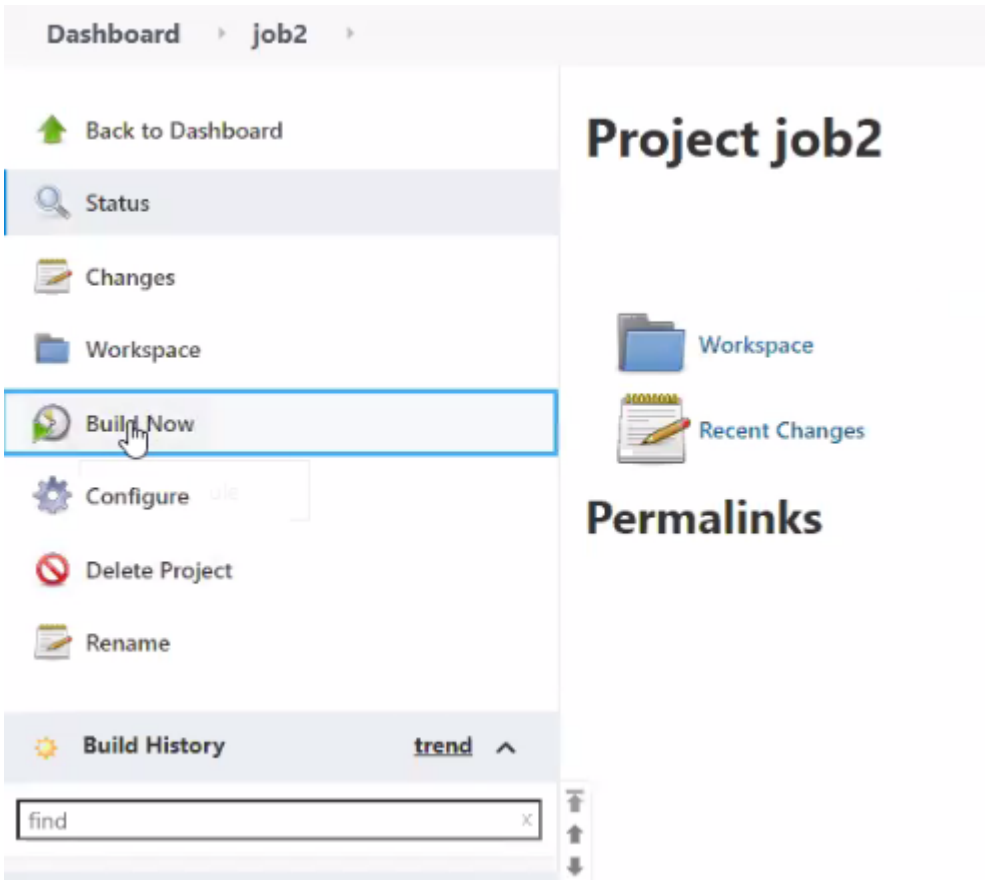


Create a new job in jenkins ---- with a simple echo statement.-- Link email in new job --- It will run successfully.

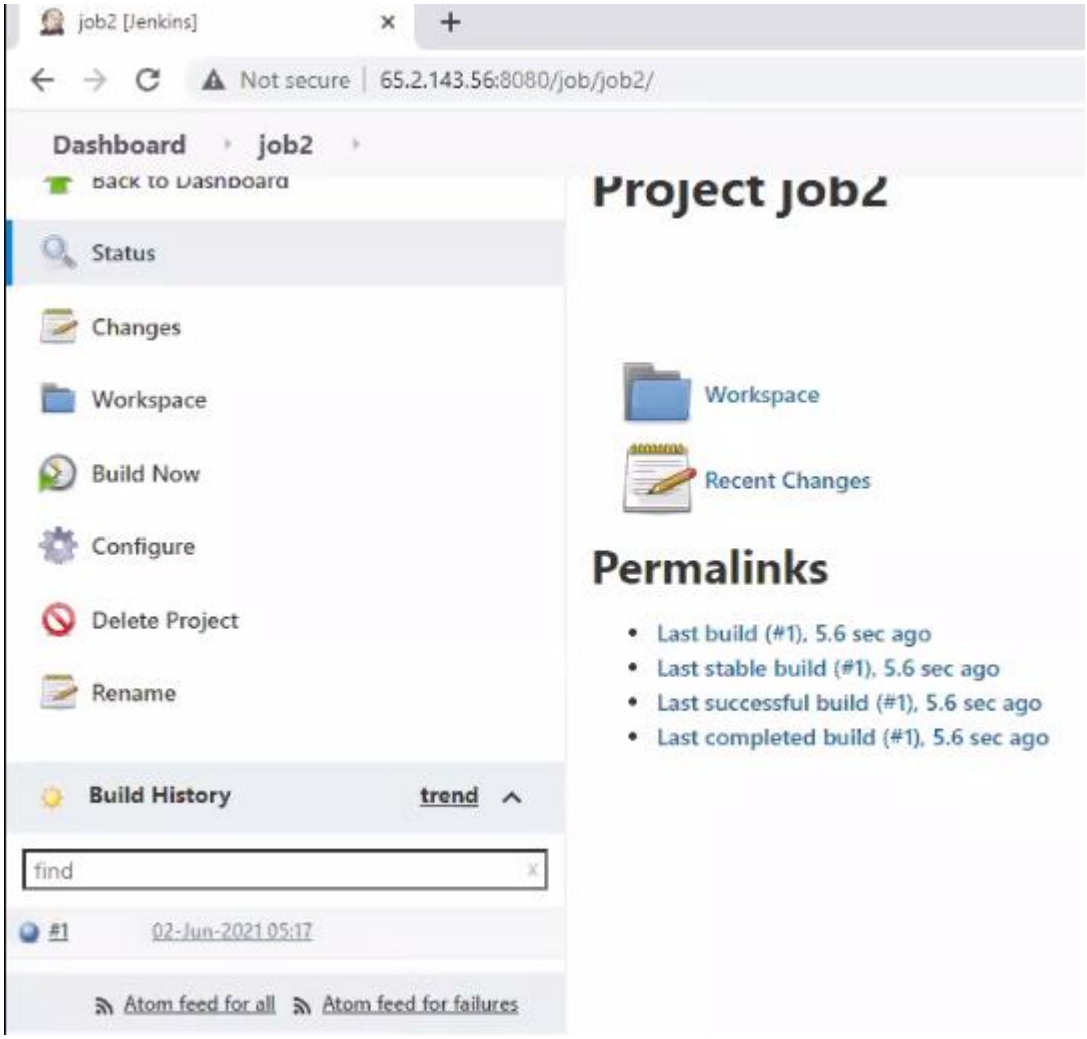
Post Build Actions Tab, we have email notification

To run the job→back dashboard→ click on job name

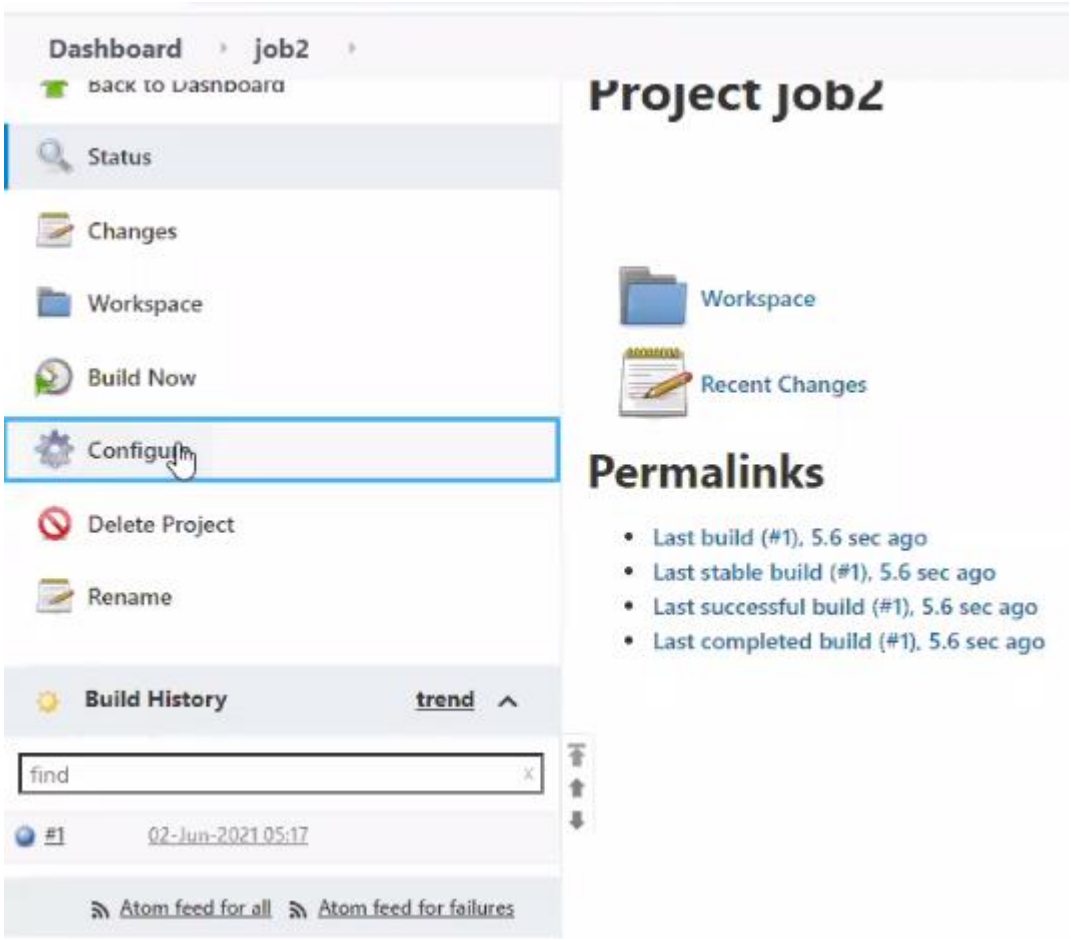


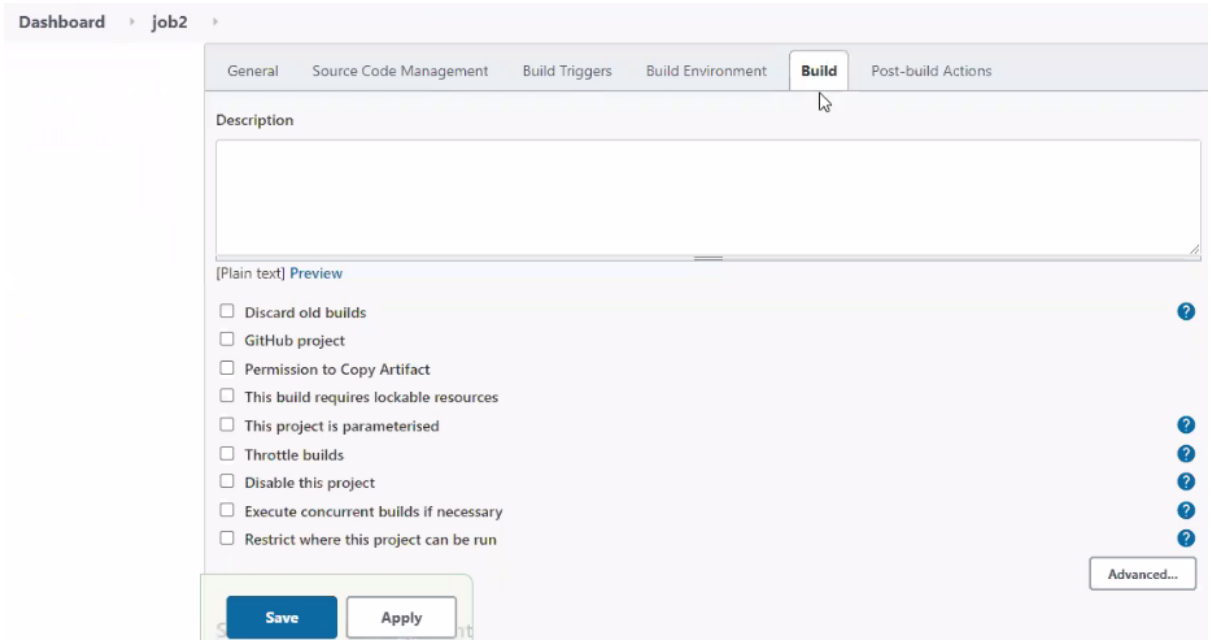


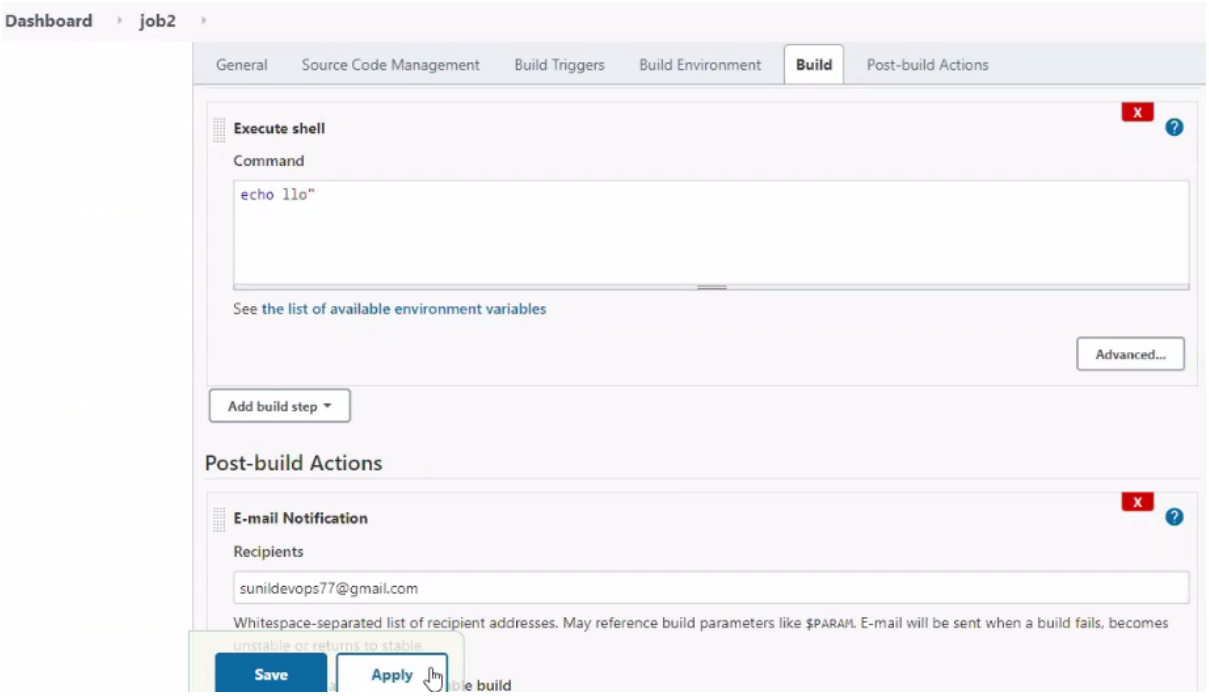
Build now→ refersh th bowser→ status→ job executed



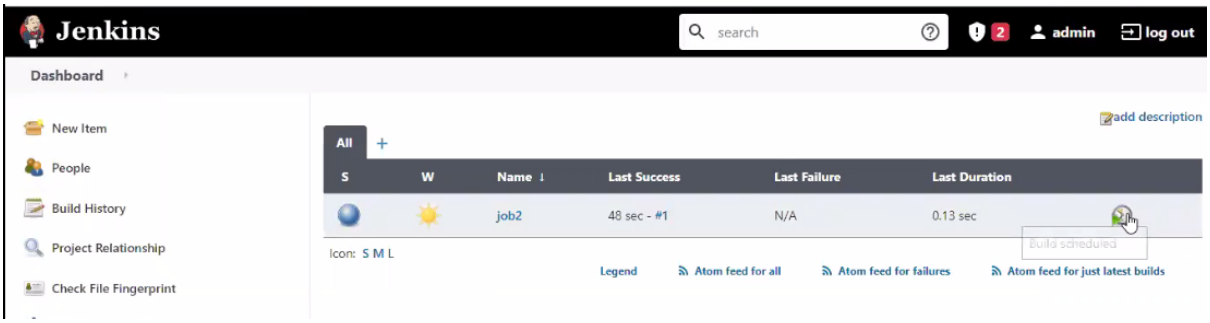
Wantedly make an error, we get email triggered.

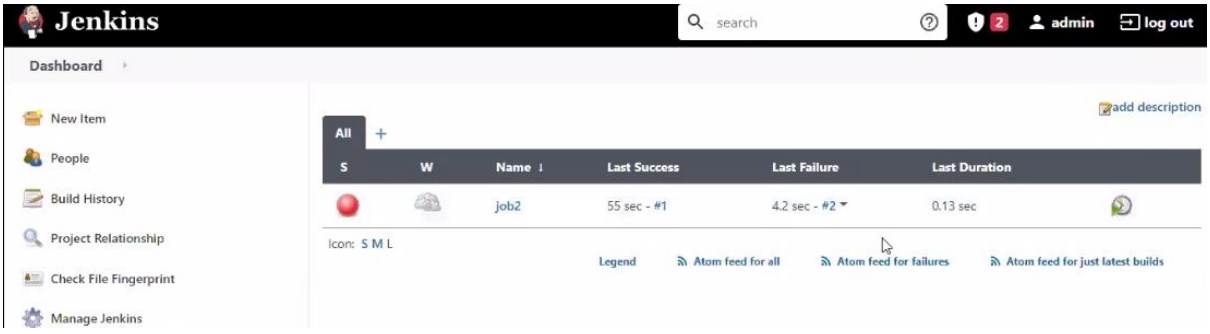

Dashboard
job2
Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Project
Rename
Build History
trend
find
#1
02-Jun-2021 05:17
Atom feed for all
Atom feed for failures
Project job2
Workspace
Recent Changes
Permalinks
Last build (#1), 5.6 sec ago
Last stable build (#1), 5.6 sec ago
Last successful build (#1), 5.6 sec ago
Last completed build (#1), 5.6 sec ago


Dashboard
job2
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
[Plain text] Preview
Discard old builds
GitHub project
Permission to Copy Artifact
This build requires lockable resources
This project is parameterised
Throttle builds
Disable this project
Execute concurrent builds if necessary
Restrict where this project can be run
Advanced...
Save
Apply



Run it then refresh browser and job failed showing in red colour so we got auto mail notification





Give right command in shell and back to job got executed normally.

++

Build Periodically

To build the job periodically

We use cron job format

If we want to run job periodically then we give five values -- 30 21 * * *

We need to give 5 values 30 21 * * *

1st value: Min - 0-59

2nd--hour - 0-23

3rd—dom(days in month) - 1-31

4th--month - 1-12

5th—dow (days of week) - 0-6

Ex: Lets say you want to run the job every day at 9:30 PM then as per above naming format will give

30 21 * * *

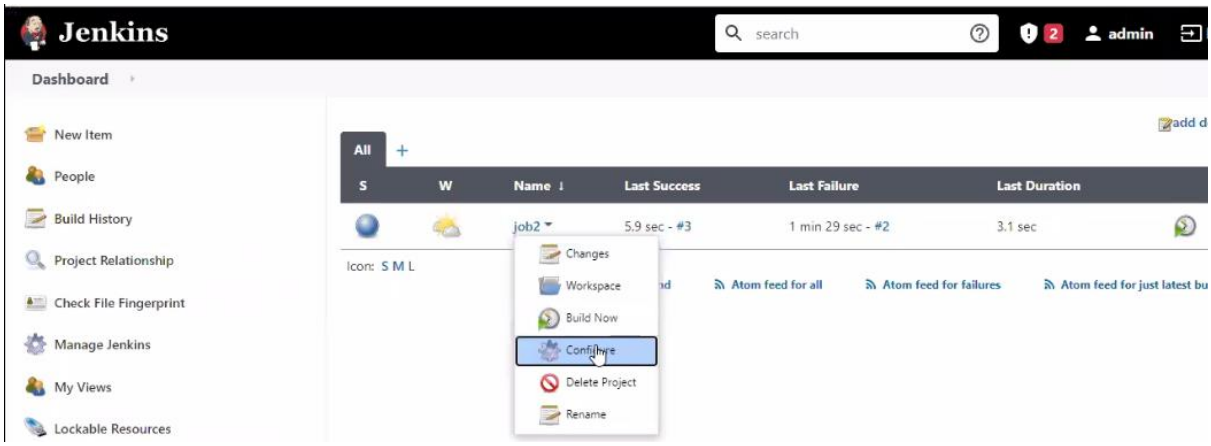
Ex2: 1=mon; 5=fri

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

30 21 * * 1-5

Ex3:To run the job every minute

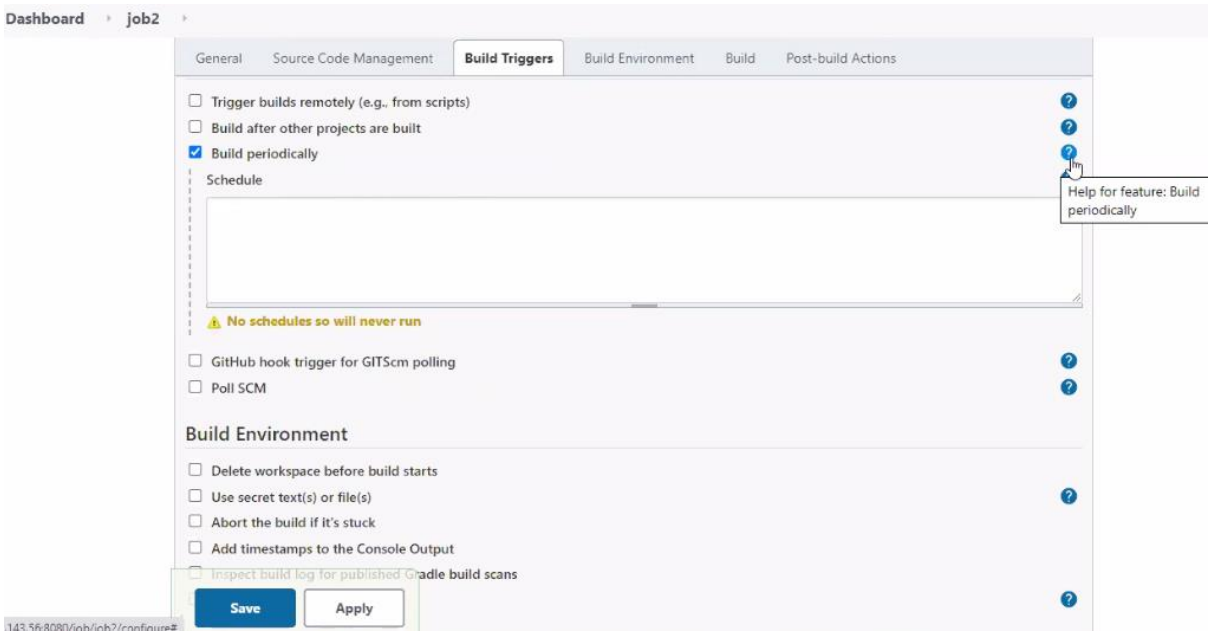
*** * * * ***



Under scoure code management tab

Build Triggers --- Build periodically

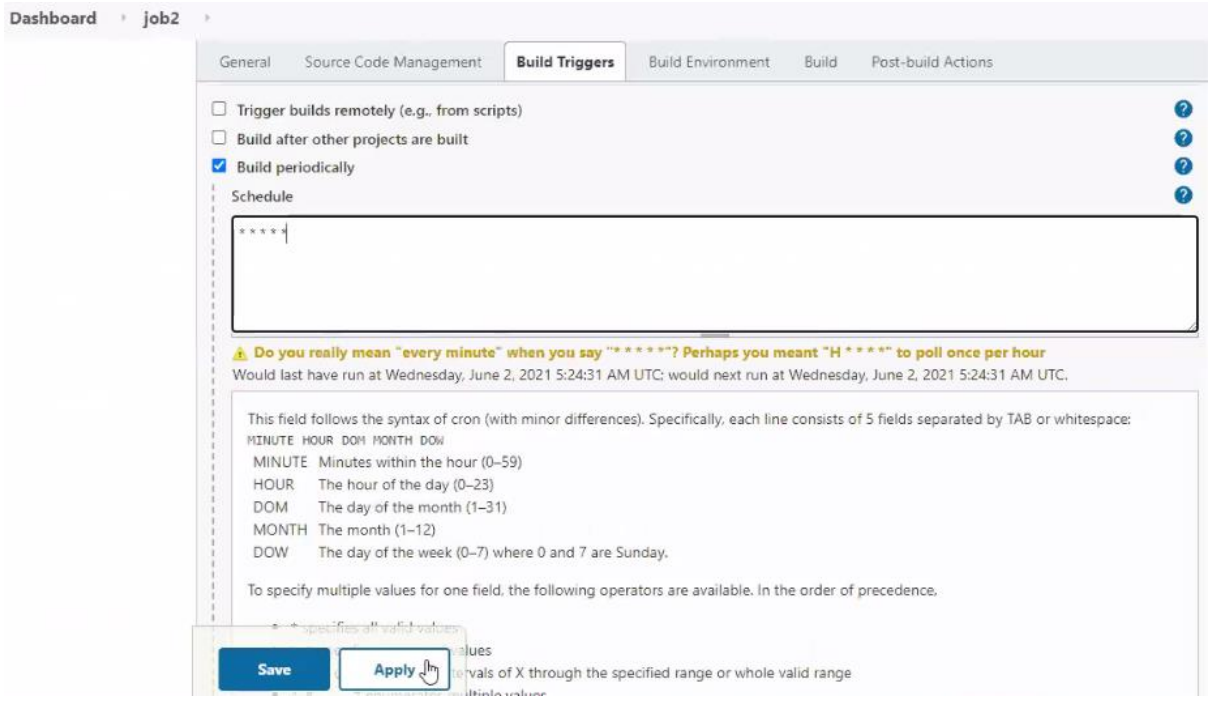
Schedule -



Click on Question mark besides Schedule for help

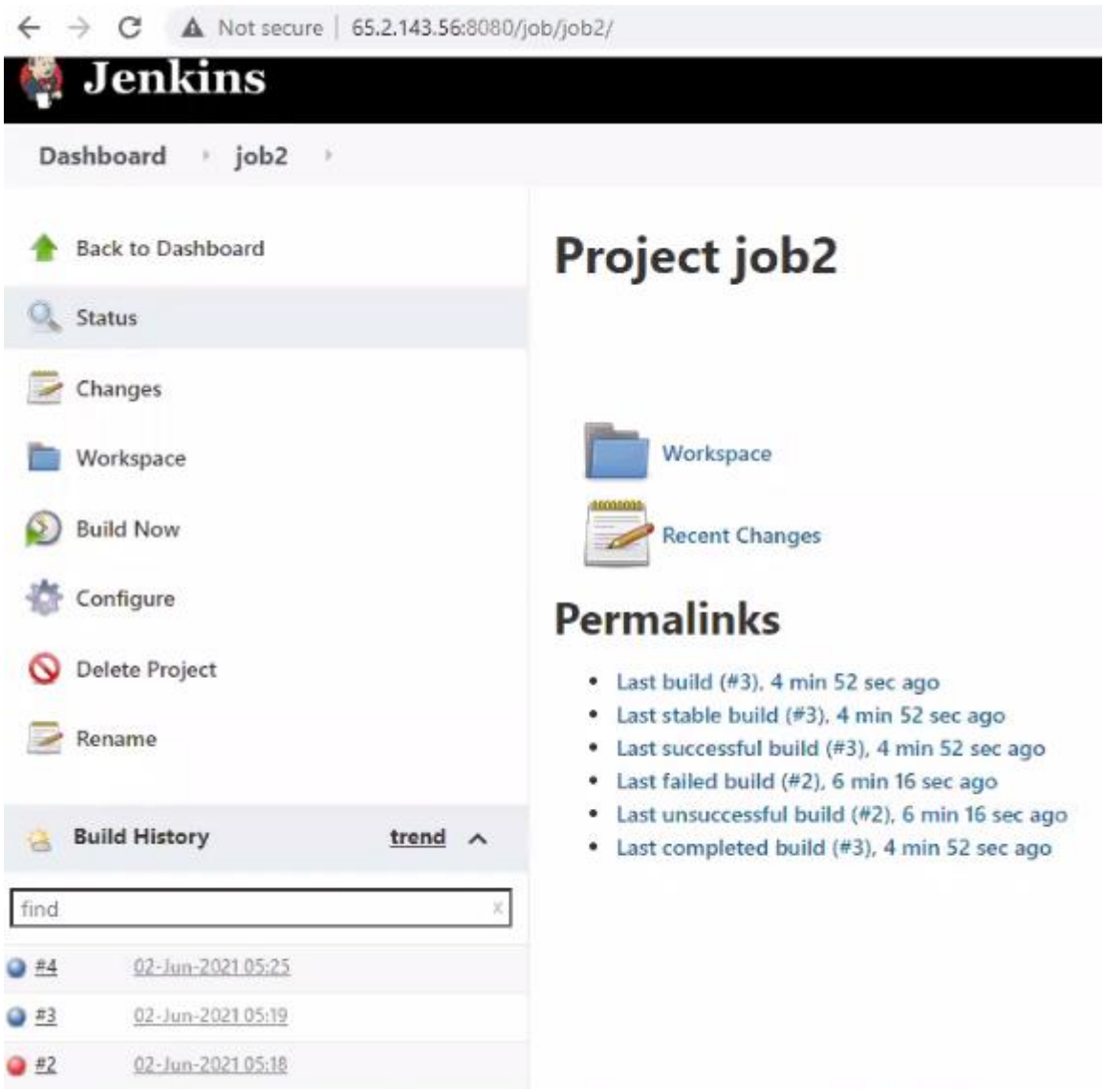
Give ****

Apply--save



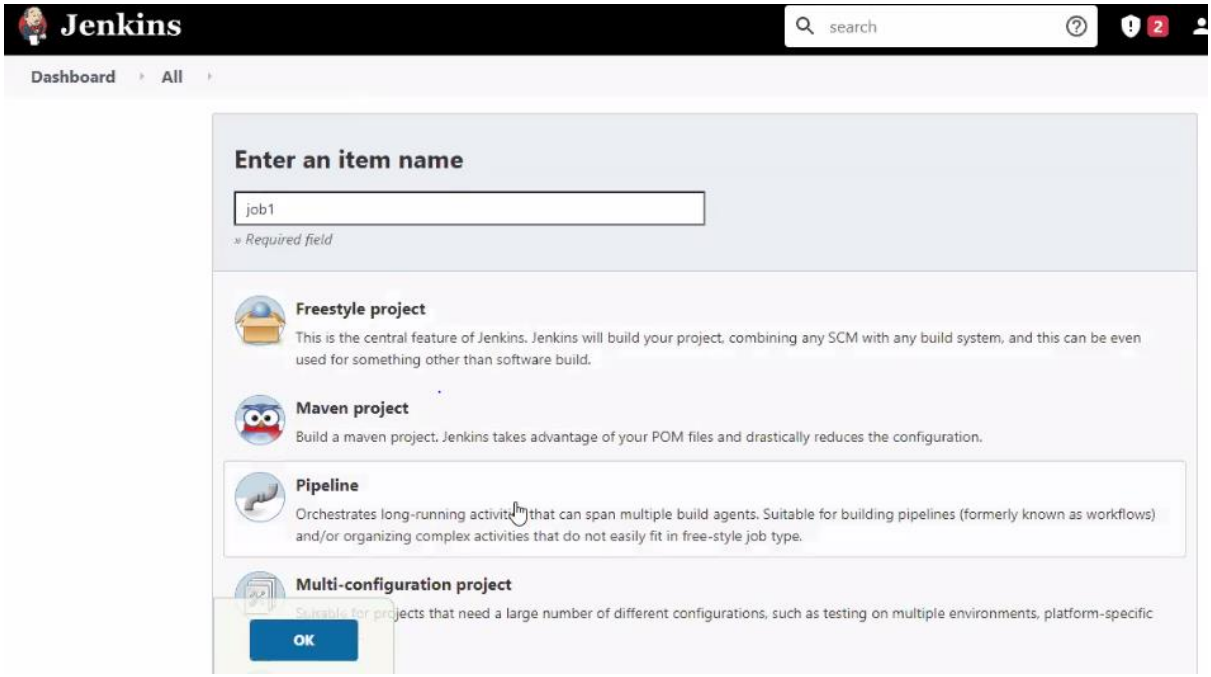
Go back to dashboard→click on job→observe job has been executing automatically for evry minute.

Also, Trigger the job based on changes done in source code management.

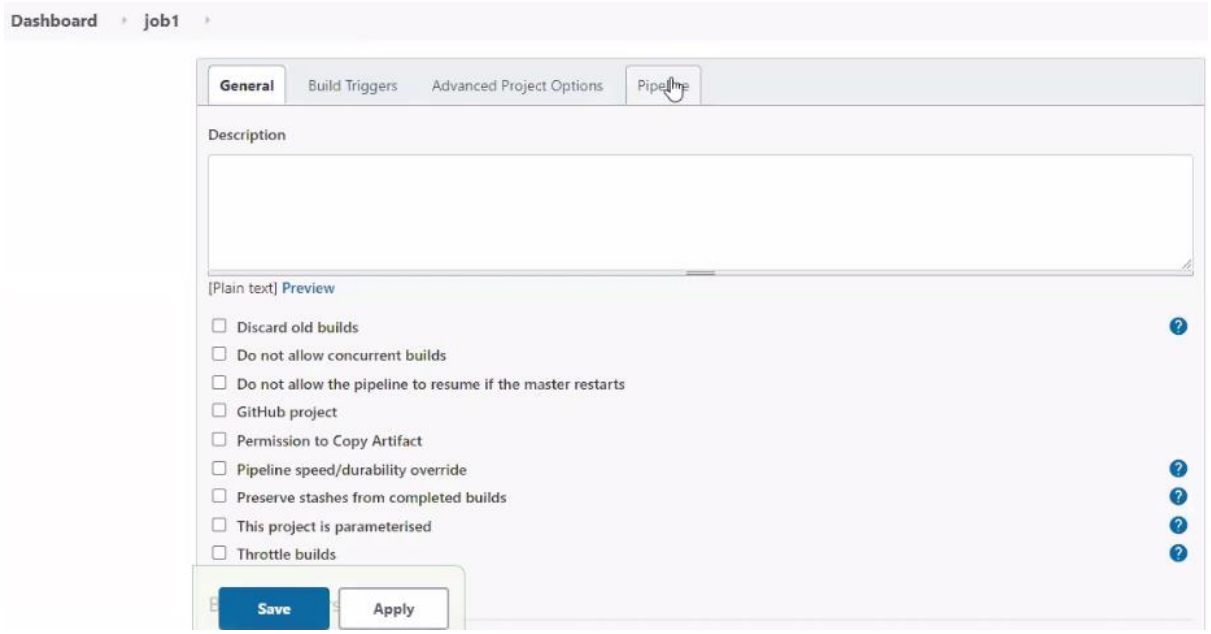


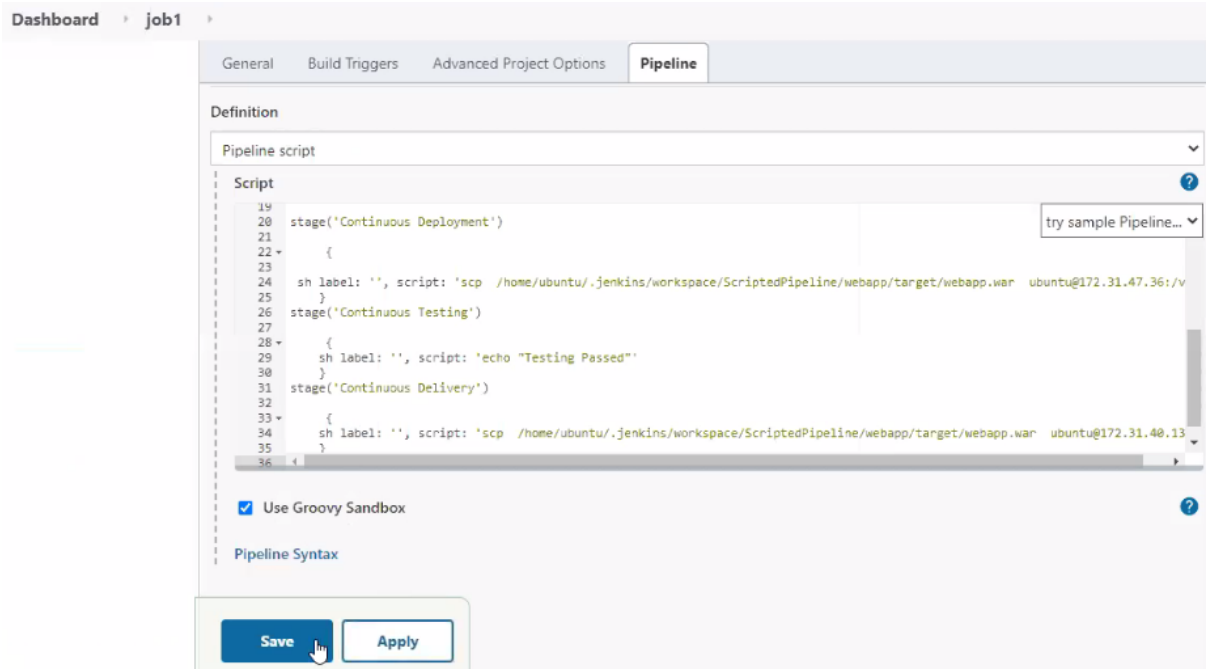
Waiting for the approval from delivery head

Newitem→job→pipelinejob→ok

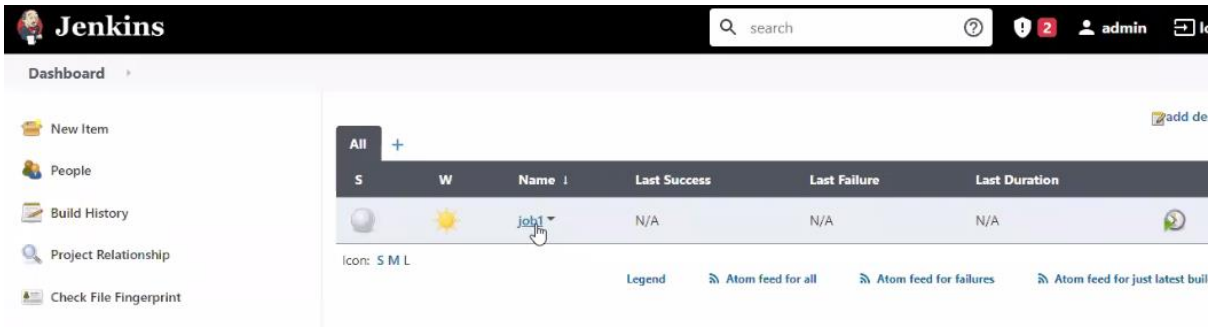


Go to pipeline scroll down and copy and paste the source code →apply--save

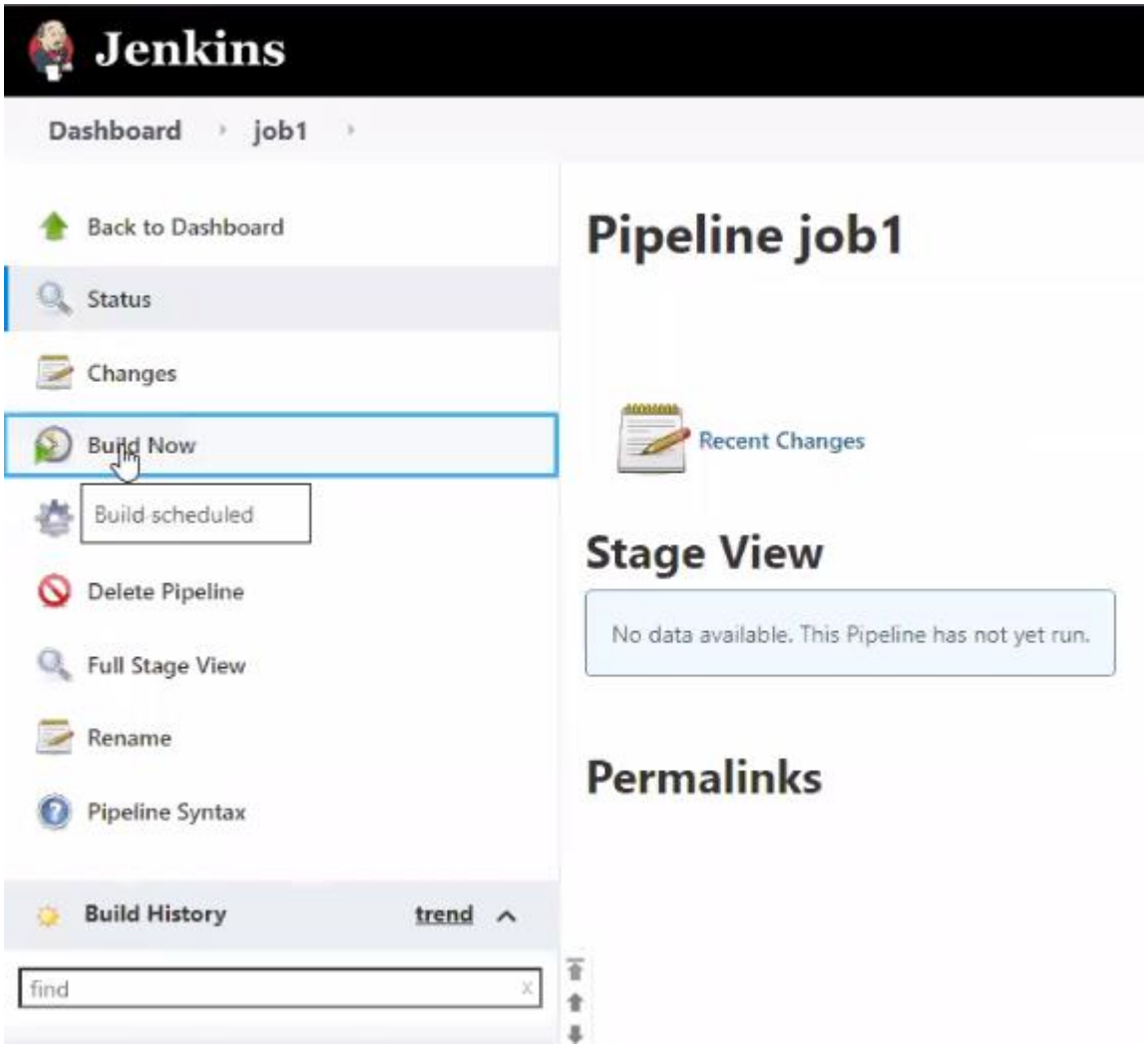


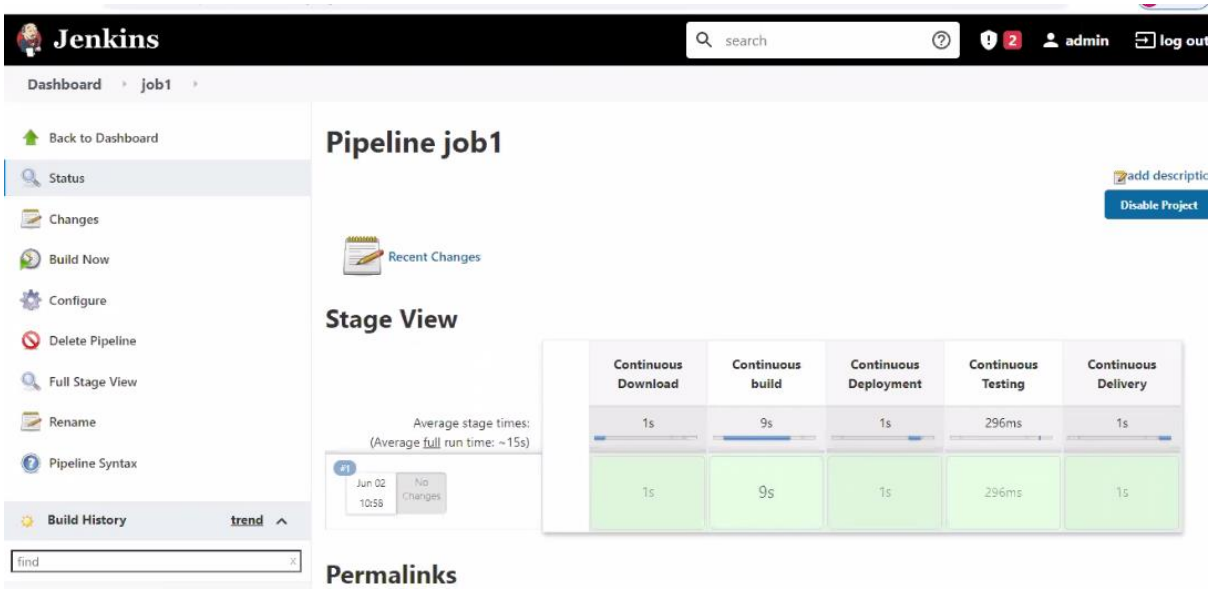


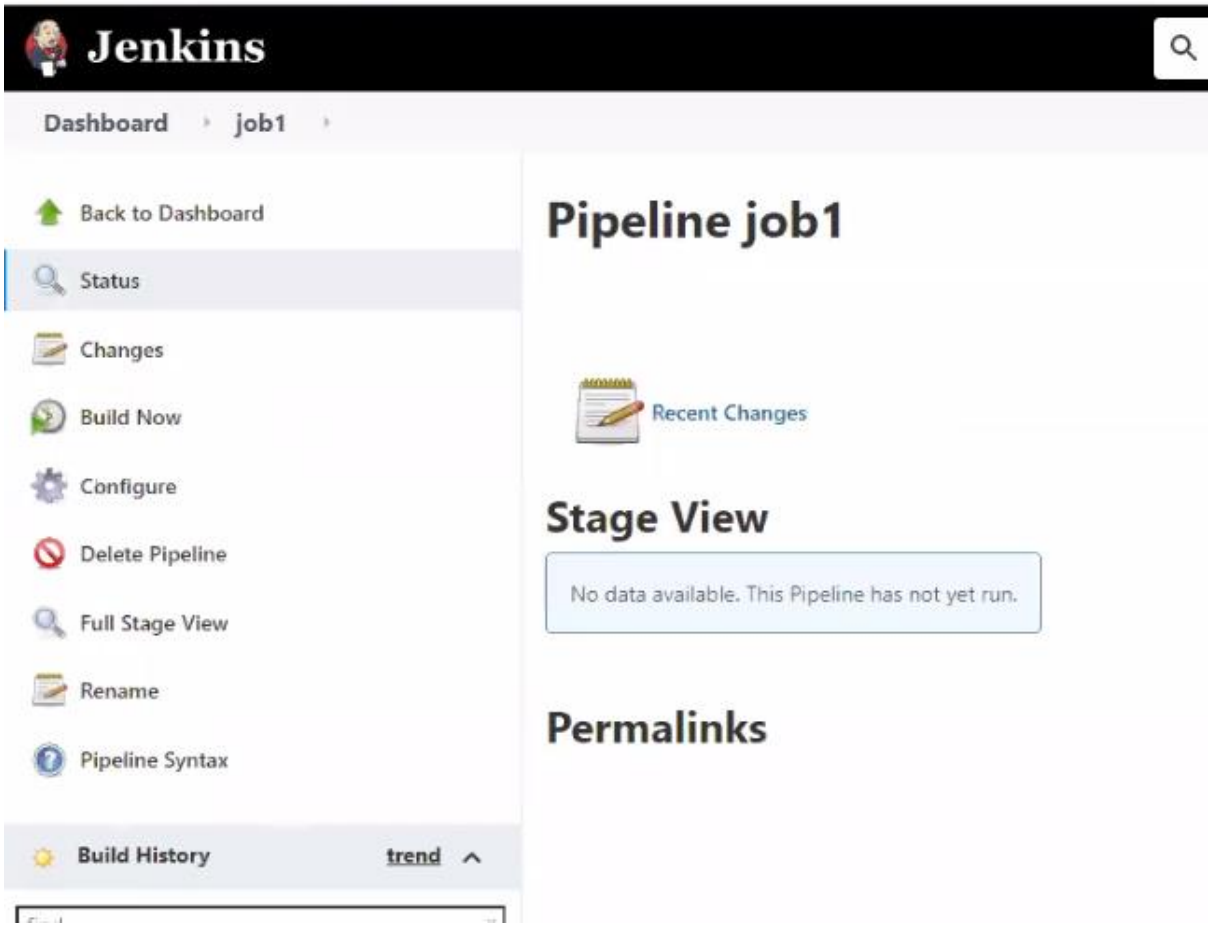
Back to dashboard→click on job→



Build now to run the job

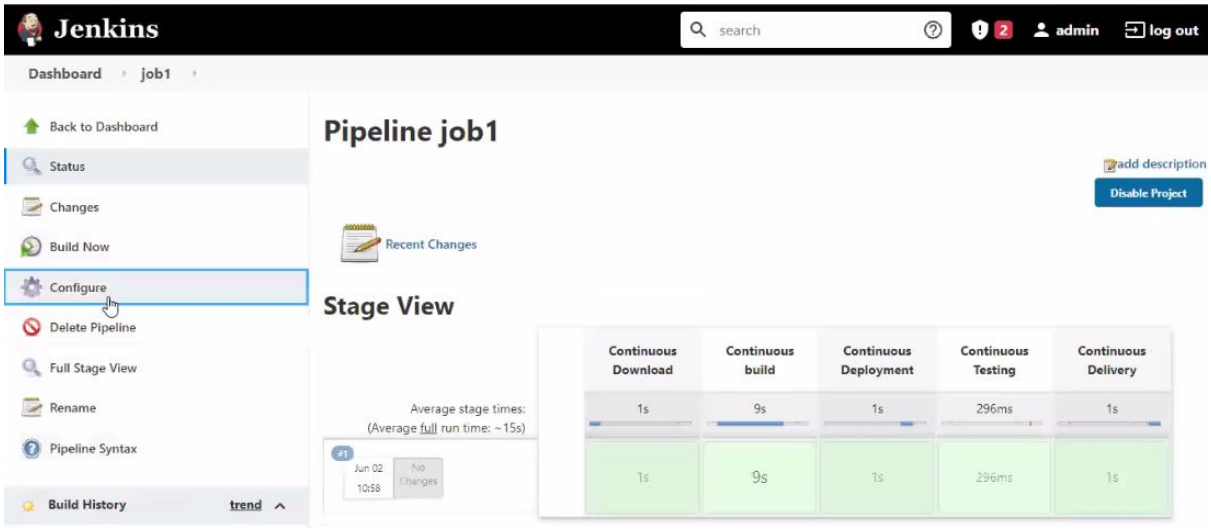

Jenkins
Dashboard
job1
Back to Dashboard
Status
Changes
Build Now
Build scheduled
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History
trend
find
Pipeline job1
Recent Changes
Stage View
No data available. This Pipeline has not yet run.
Permalinks

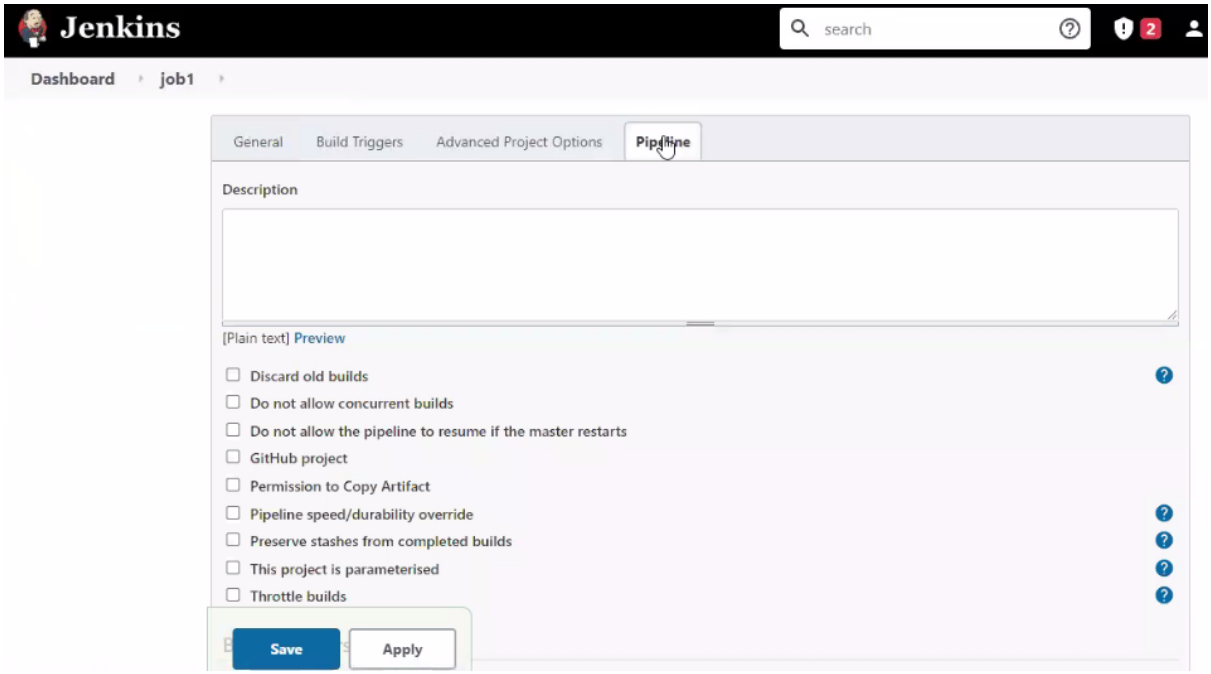

Jenkins
search
admin
log out
Dashboard
job1
Back to Dashboard
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History
trend
find
Pipeline job1
Disable Project
Recent Changes
Stage View
Continuous Download
Continuous build
Continuous Deployment
Continuous Testing
Continuous Delivery
Average stage times:
(Average full run time: ~15s)
1s
9s
1s
296ms
1s
Jun 02
10:58
No Changes
Permalinks



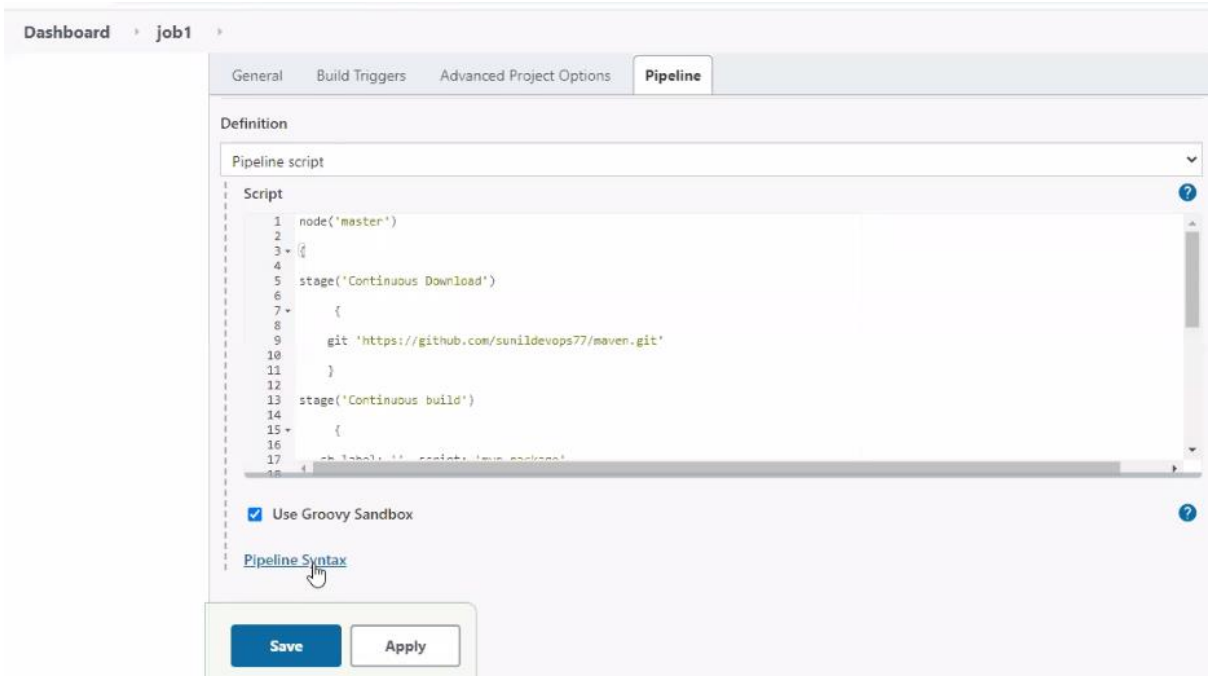

```
node('master')

{

stage('Continuous Download')

        {

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

        }

stage('Continuous build')

        {
```

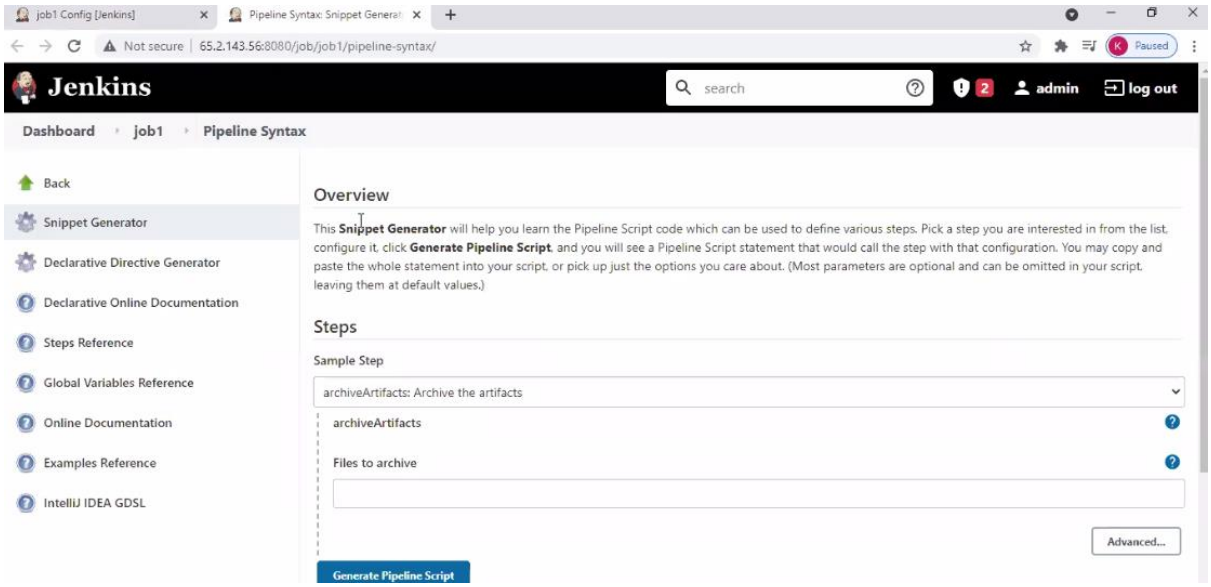
```
   sh label: '', script: 'mvn package'
            }

stage('Continuous Deployment')

            {

 sh label: '', script: 'scp
/home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war
ubuntu@172.31.47.36:/var/lib/tomcat8/webapps/qaenv.war'
            }
stage('Continuous Testing')

            {
            sh label: '', script: 'echo "Testing Passed"'
            }
stage('Continuous Delivery')

            {

            input 'Waiting for Approval'

            sh label: '', script: 'scp
/home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war
ubuntu@172.31.40.134:/var/lib/tomcat8/webapps/prodenv.war'
            }
}

In
```



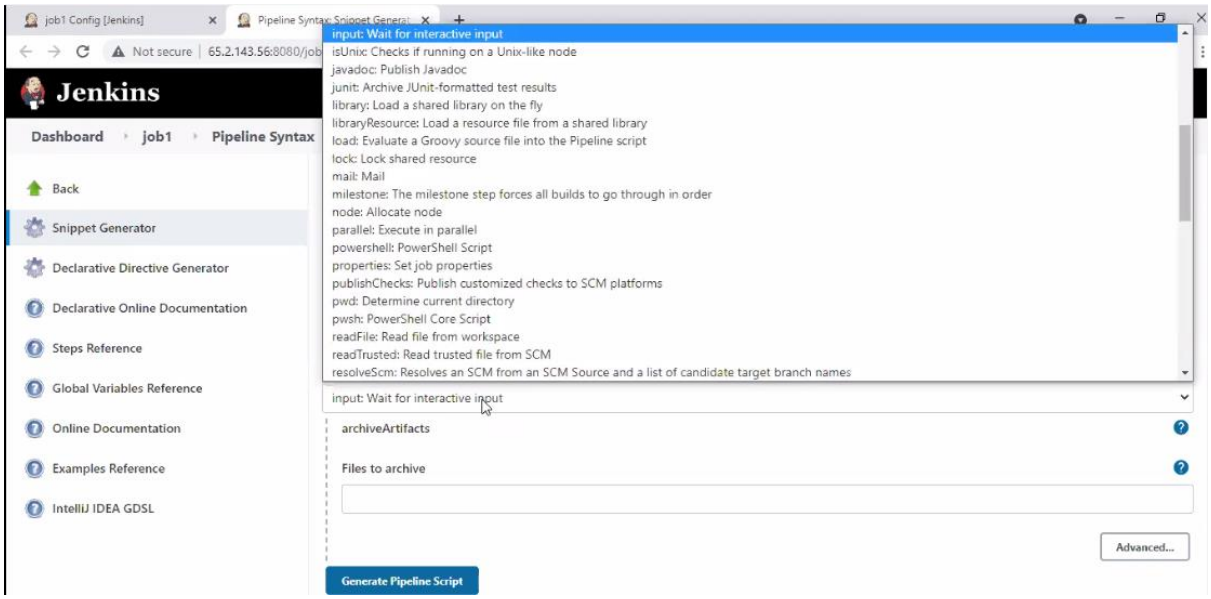


Scroll down and click on Pipeline syntax it navigates to another page for generating code

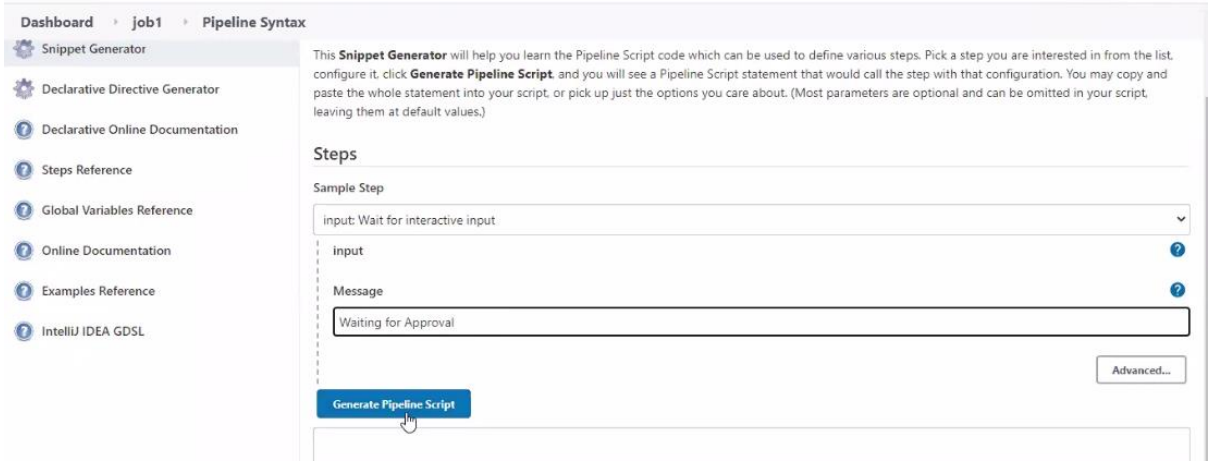




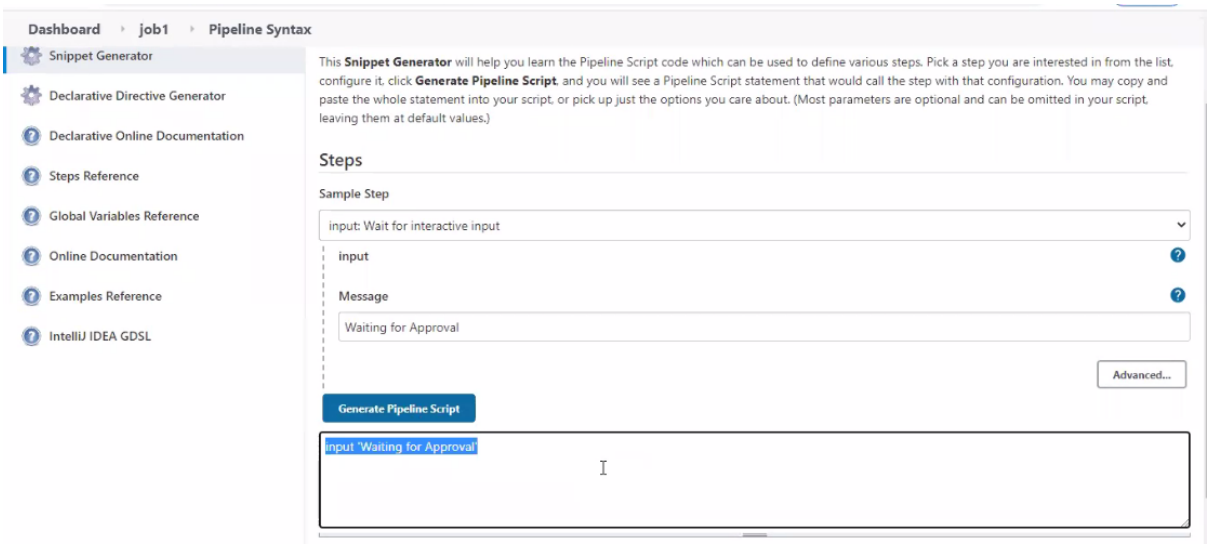
Select Wait for interactive input



Message: enter as waiting for approval



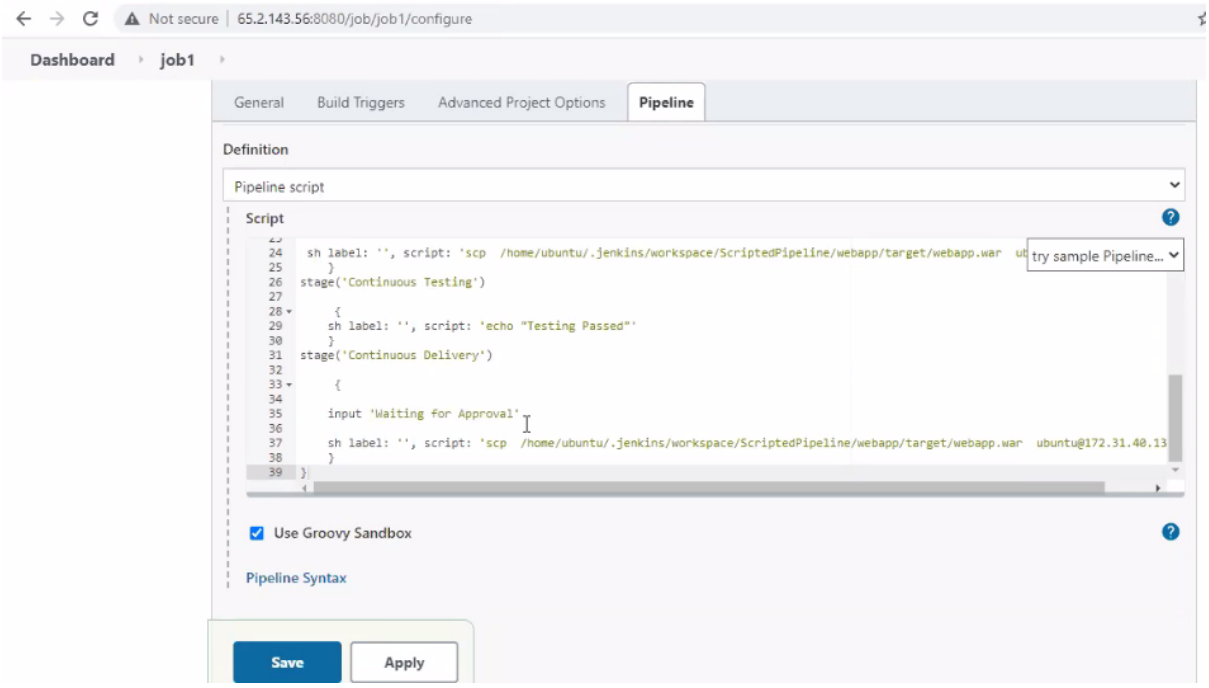
Got script as Input “waiting for approval”



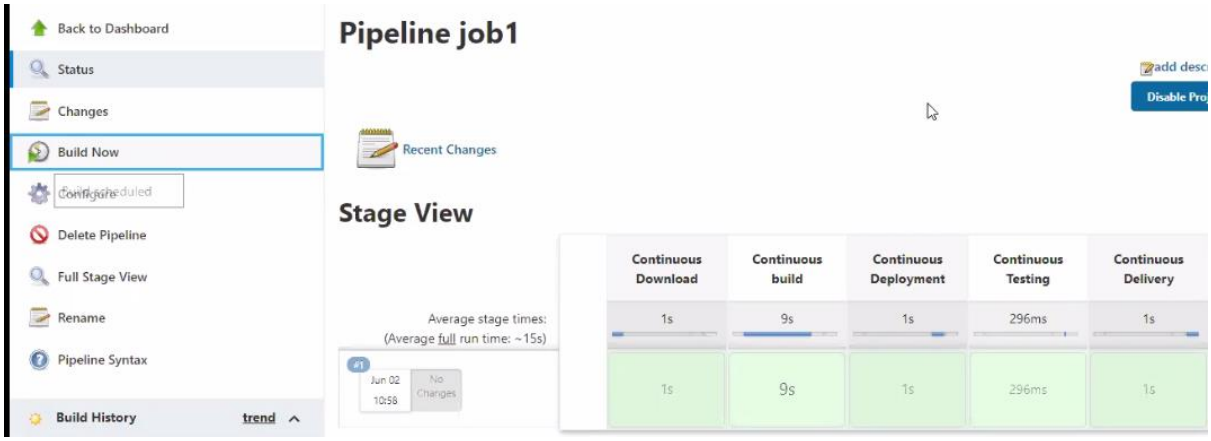
Should update groovy code in continuous delivery for asking approval to move forward 5th stage

```
stage('Continuous Delivery')
        {
        input 'Waiting for Approval'
        sh label: '', script: 'scp  /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war
ubuntu@172.31.40.134:/var/lib/tomcat8/webapps/prodenv.war'
        }
}
```

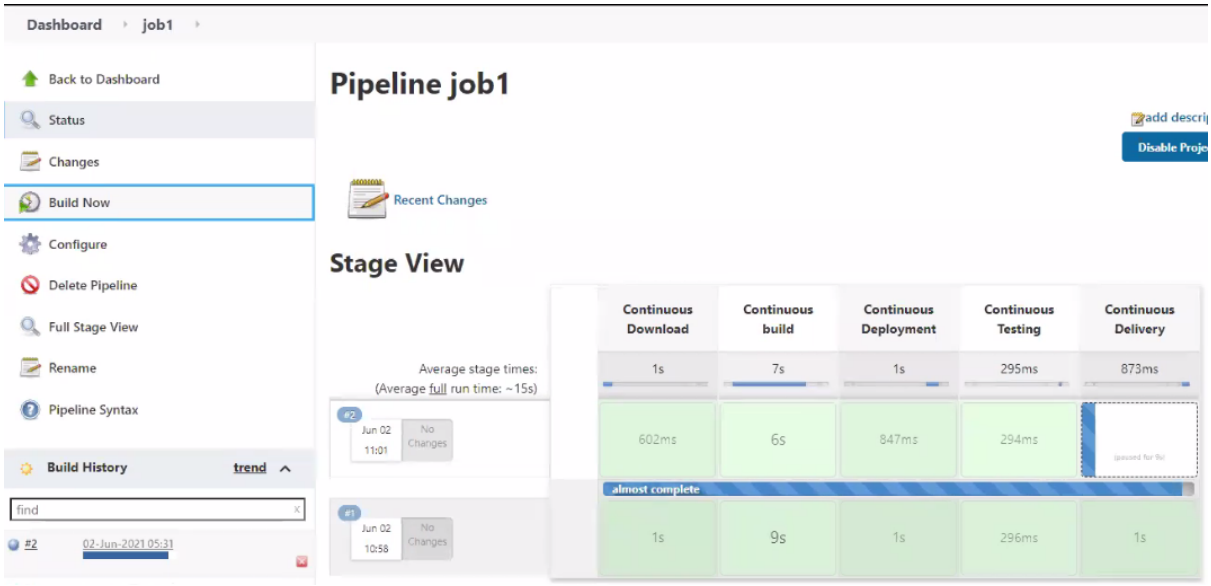
Update the code in pipeline--? Apply--save



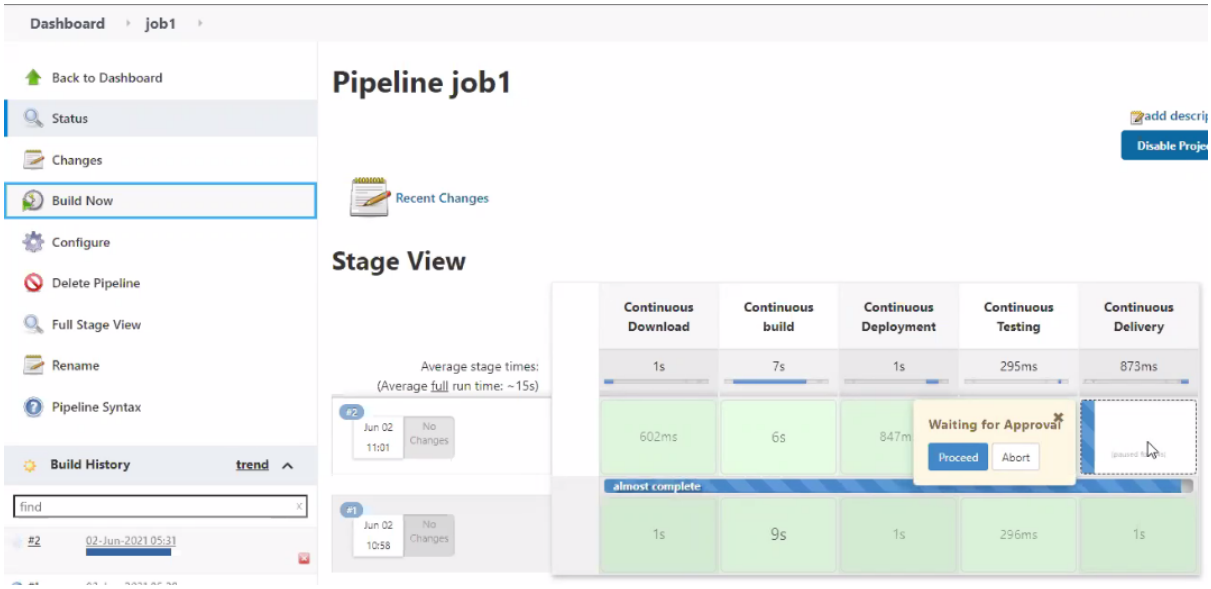
Go back to dashboard->click job--?click build now→



It was paused on stage 5 as seen below



Drag the mouse on 5th step can see interactive input asking approval



Once you click on Approve button it gets into prod