

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

- 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 Eligible)**
 - 8) Select t2.micro (Free For Eligible)**
 - 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

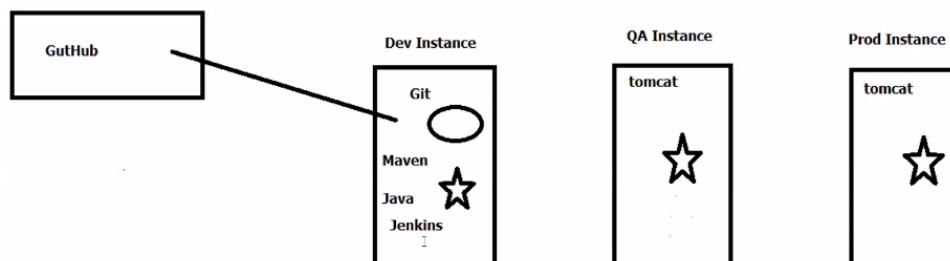
8) Just Type Yes

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 [-v--version] [--help] [-c <name>=<value>]
           [-e--exec-path[=<path>]] [--html-path] [-m--man-path] [-i--info-path]
           [-p--paginate | -P | --no-pager] [--no-replace-objects] [-bare]
           [-g--git-dir[=<path>]] [--work-tree=<path>] [-n--namespace=<name>]
           [-s--super-prefix=<path>] [-c--config-env=<name>=<envvar>]
           <commands> [<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
Java 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: "5.15.0-107-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, 2a00:7c0:0: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-ubuntu1~22.04-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.jsp.JettyJspSer...
2022-08-28 08:40:30.865+0000 [id=1] INFO o.e.j.s.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager.userName=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/,AVAILABLE}
/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=28] 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 08:40:34.767+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.Retriger#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:
21669fd14854a558414a169e10b7ad6
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

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

/home/ubuntu/.jenkins/secrets/initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password

Continue

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 password 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$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller  
2021-05-21 05:21:33.646+0000 [id=40] INFO hudson.util.Retriger#start: Performed the action check updates server successfully at the attempt #1  
2021-05-21 05:21:33.649+0000 [id=40] INFO hudson.model.AsyncPeriodicWork$doRun$0: Finished Download metadata. 23,475 ms  
2021-05-21 05:21:35.574+0000 [id=26] INFO jenkins.InitReactorRunner$1#onTained: Completed initialization  
2021-05-21 05:21:35.595+0000 [id=20] INFO hudson.WebAppMain$3#run: Jenkins is fully up and running
```

You will get the password on the step 11

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

14) Press Install Suggested Plugins

The screenshot shows the Jenkins 'Getting Started' page. On the left, there's a sidebar with links like 'Folders', 'Timestamper', 'Pipeline', 'Git', and 'LDAP'. The main area has two sections: 'Install suggested plugins' (highlighted with a blue border) and 'Select plugins to install'. The 'Install suggested plugins' section contains the text: 'Install plugins the Jenkins community finds most useful.' Below this is a large button labeled 'Install Suggested Plugins'. To the right, there's a sidebar titled 'Folders' with items like 'Trilead API', 'OWASP Markup Formatter', and 'Structs'. At the bottom of the sidebar, it says '** - required dependency'.

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

Instance Configuration

Jenkins URL:

<http://3.92.221.68:8080/>

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

2

Not now

Save and Finish

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

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

Create Sample Job in Jenkins:

Click on new item or

Jenkins

Dashboard >

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources
- New View

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job →

Set up a distributed build

Set up an agent →

Configure a cloud →

Learn more about distributed builds ↗

Build Queue ^
No builds in the queue.

Build Executor Status ^
1 idle

Enter an item name

sample » Required field

Freestyle project
 This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
 Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
 Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
 A container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate space, so you can have multiple things of the same name as long as they are in different folders.

Build tab

Dashboard > sample >

General	Source Code Management	Build Triggers	Build Environment	Build	Post-build Actions
Description <div style="border: 1px solid #ccc; height: 100px; margin-top: 10px;"></div> <p>[Plain text] Preview</p> <ul style="list-style-type: none"> <input type="checkbox"/> Discard old builds <input type="checkbox"/> GitHub project <input type="checkbox"/> This build requires lockable resources <input type="checkbox"/> This project is parameterised <input type="checkbox"/> Throttle builds <input type="checkbox"/> Disable this project <input type="checkbox"/> Execute concurrent builds if necessary <p style="text-align: right;"><small>Advanced...</small></p>					
<input type="button" value="Save"/> <input type="button" value="Apply"/>					

Click on execute Shell

Dashboard > sample >

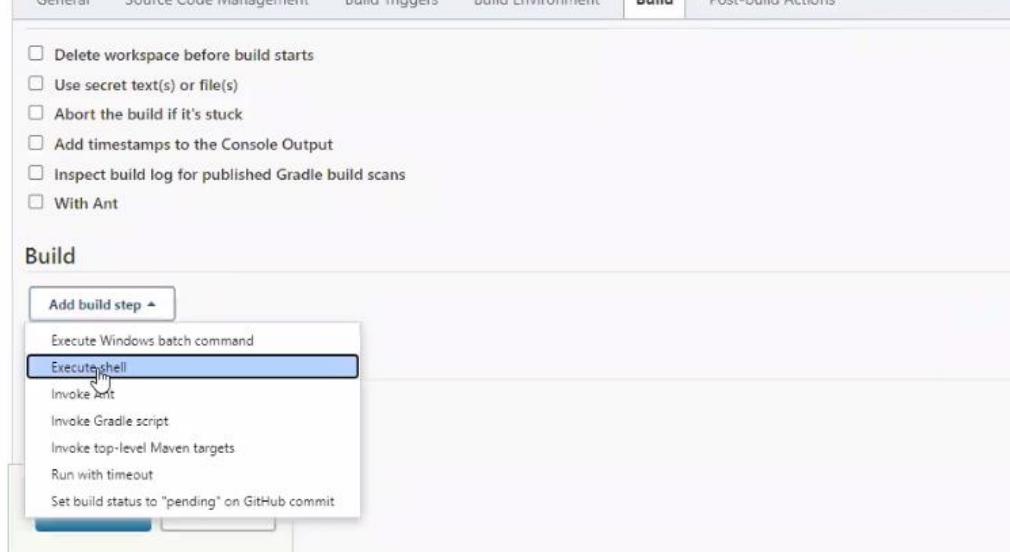
General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Delete workspace before build starts
 Use secret text(s) or file(s)
 Abort the build if it's stuck
 Add timestamps to the Console Output
 Inspect build log for published Gradle build scans
 With Ant

Build

Add build step ▾

- Execute Windows batch command
- Execute shell**
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit



In Command Box Enter echo "Hello Jenkins"

Dashboard > sample >

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Delete workspace before build starts
 Use secret text(s) or file(s)
 Abort the build if it's stuck
 Add timestamps to the Console Output
 Inspect build log for published Gradle build scans
 With Ant

Build

Execute shell

Command

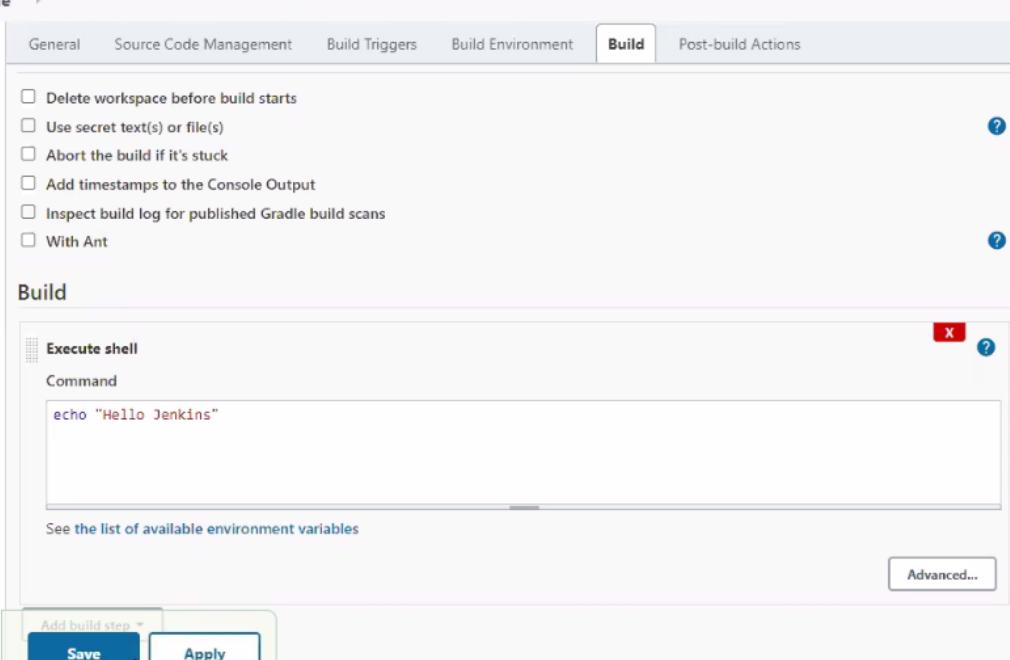
```
echo "Hello Jenkins"
```

See the list of available environment variables

Advanced...

Add build step ▾

Save **Apply**



Jenkins

Dashboard > sample

search admin log out

Project sample

Status

Changes

Workspace

Build Now

Configure

Delete Project

Rename

Build History

trend

find

Atom feed for all Atom feed for failures

add description

Disable Project

Jenkins

Dashboard

search admin log out

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

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

add description

Jenkins

Dashboard

search admin log out

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

All +

S	W	Name	Last Success	Last Failure	Last Duration
		sample	N/A	N/A	N/A

Icon: S M L

Atom feed for all Atom feed for failures Atom feed for just latest builds

Configure

Delete Project

Rename

add description

Jenkins

Dashboard >

New Item People Build History Manage Jenkins My Views Lockable Resources New View

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

Click on Console Output

Jenkins

Dashboard >

New Item People Build History Manage Jenkins My Views Lockable Resources New View

All +

S	W	Name	Last Success	Last Failure	Last Duration
		sample	10 sec - #1	N/A	0.11 sec

Icon: S M L Legend Atom feed for failures Atom feed for just latest builds

Console Output

Jenkins

Dashboard > sample > #1

Back to Project Status Changes Console Output View as plain text Edit Build Information Delete build '#1'

Console Output

Started by user admin
Running as SYSTEM
Building in workspace /home/ubuntu/.jenkins/workspace/sample
[sample] \$ /bin/sh -xe /tmp/jenkins6572208772974952260.sh
+ echo Hello Jenkins
Hello Jenkins
Finished: SUCCESS

+++++

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

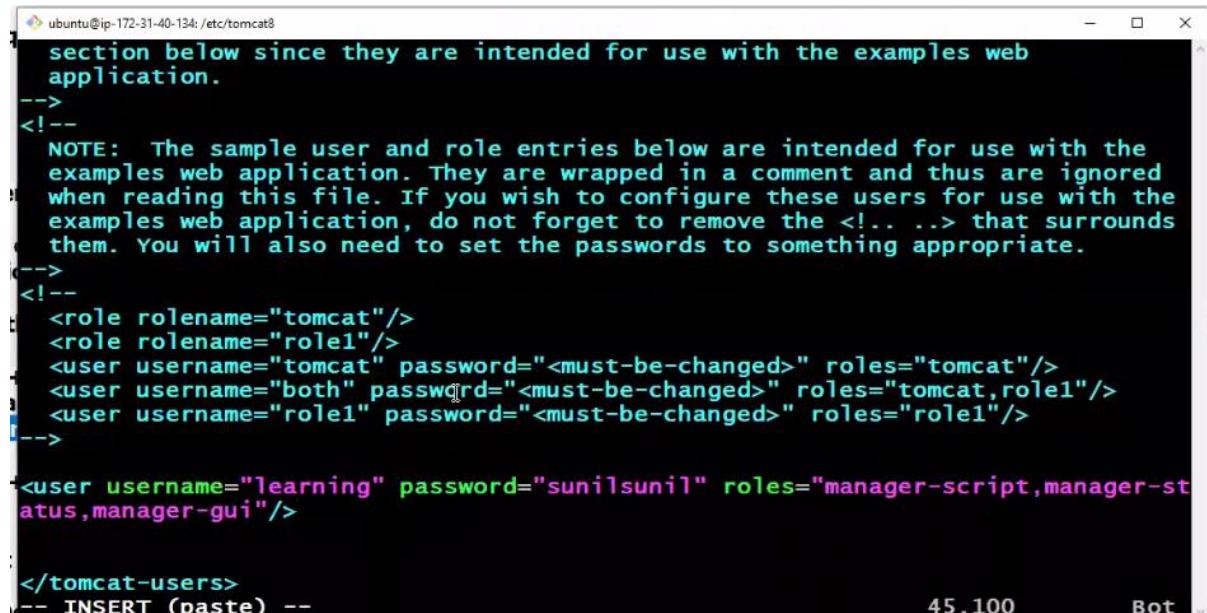
9) You will find the file tomcat-users.xml

```
ubuntu@ip-172-31-40-134:~$ cd /etc/tomcat8/
ubuntu@ip-172-31-40-134:/etc/tomcat8$ ls
Catalina                jaspic-providers.xml    server.xml
catalina.properties      logging.properties      tomcat-users.xml
context.xml              policy.d                 web.xml
ubuntu@ip-172-31-40-134:/etc/tomcat8$ 
ubuntu@ip-172-31-40-134:/etc/tomcat8$ 
ubuntu@ip-172-31-40-134:/etc/tomcat8$ |
```

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"/>
```



```
ubuntu@ip-172-31-40-134: /etc/tomcat8
section below since they are intended for use with the examples web
application.
-->
<!--
NOTE: The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!... ...> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="" roles="tomcat"/>
<user username="both" password="" roles="tomcat,role1"/>
<user username="role1" password="" roles="role1"/>
-->

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

</tomcat-users>
-- INSERT (paste) --
```

45,100

Bot

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 management

3) Select GIT

4) Enter the URL of github repository

<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

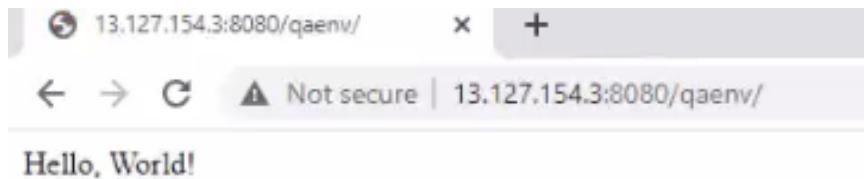
<http://172.31.34.103:8080>

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>

A screenshot of the Jenkins 'Enter an item name' dialog. The input field contains the text "testing". Below the input field, there are three options: "Freestyle project", "Pipeline", and "Multi-configuration project". The "Folder" option is highlighted with a blue border and has an "OK" button next to it. A tooltip for "Folder" explains that it is a container for nested items and useful for grouping things together.

A screenshot of the Jenkins 'Source Code Management' configuration page for the "testing" job. The "Source Code Management" tab is selected. Under "General", the "Git" option is chosen. In the "Repositories" section, the "Repository URL" is set to "https://github.com/sunildevops77/TestingNew.git". The "Branches to build" section shows a "Branch Specifier (blank for 'any')" set to "*/*master". At the bottom, there are "Save" and "Apply" buttons, along with a "Add Branch" link.

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.

The screenshot shows the Jenkins job configuration interface for a job named "testing". The "Build" tab is selected. A dropdown menu is open under the "Add build step" button, showing several options: "Execute Windows batch command", "Execute shell", "Invoke Ant", "Invoke Gradle script", "Invoke top-level Maven targets", "Run with timeout", and "Set build status to 'pending' on GitHub commit". The "Execute shell" option is highlighted with a blue selection bar.

The screenshot shows the Jenkins configuration interface for a job named 'testing'. The 'Build' tab is selected. Under 'Build', there is one step: 'Execute shell' with the command 'java -jar testing.jar'. At the bottom, there are 'Save' and 'Apply' buttons.

S	W	Name	Last Success	Last Failure	Last Duration
		development	6 min 55 sec - #3	N/A	8.4 sec
		testing	N/A	N/A	N/A

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 –

Not secure | 52.66.9.149:8080/job/development/configure

Dashboard > development >

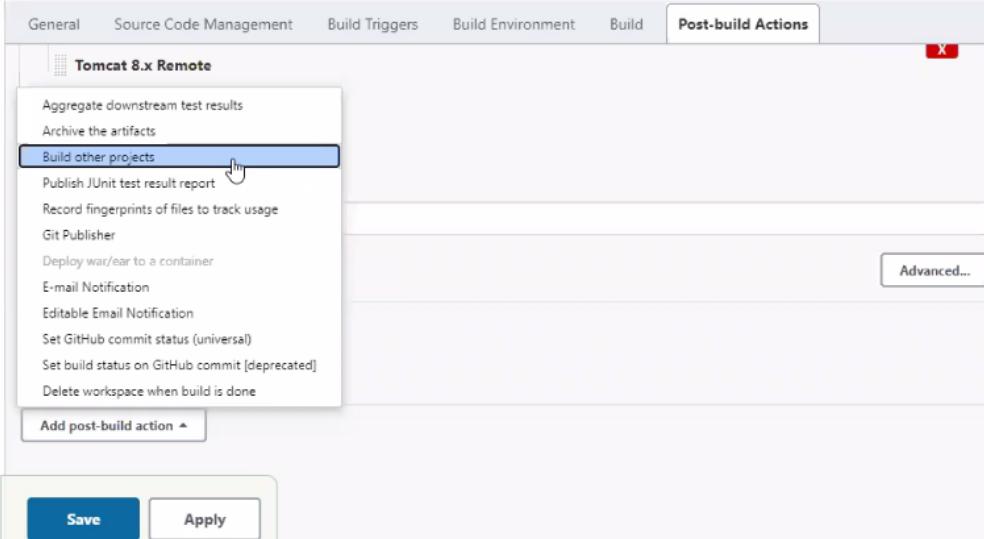
General Source Code Management Build Triggers Build Environment Build Post-build Actions

Tomcat 8.x Remote

- Aggregate downstream test results
- Archive the artifacts
- Build other projects**
- Publish JUnit test result report
- Record fingerprints of files to track usage
- Git Publisher
- Deploy war/ear to a container
- E-mail Notification
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Delete workspace when build is done

Add post-build action ▾

Save Apply Advanced...



Projects to build - testing (name of the job)

Not secure | 52.66.9.149:8080/job/development/configure

Dashboard > development >

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Add build step ▾

Post-build Actions

Build other projects

Projects to build

testing

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

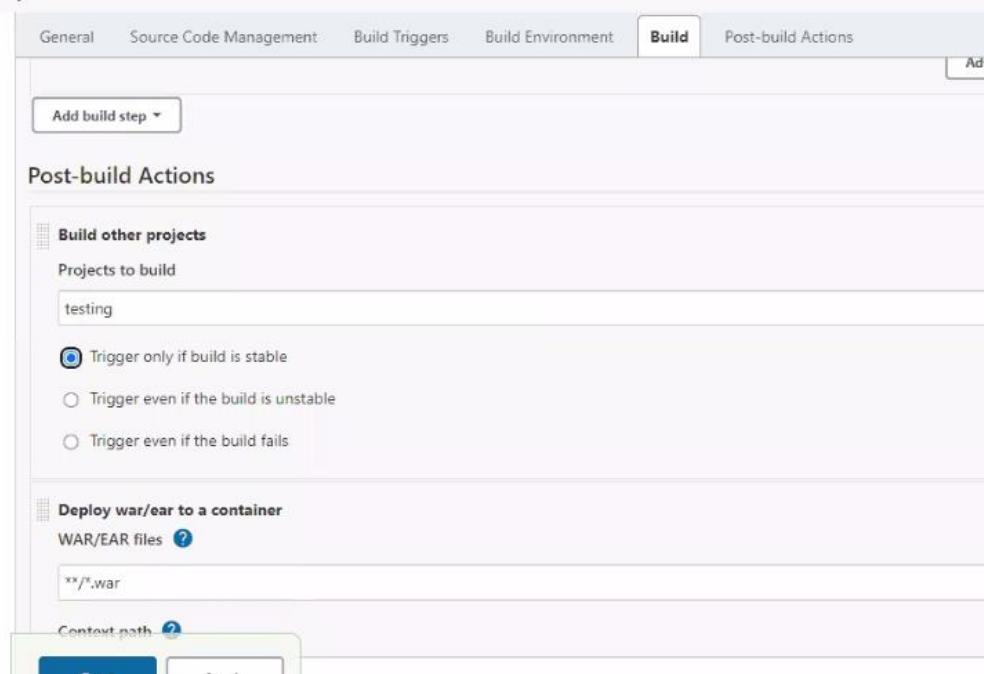
Deploy war/ear to a container

WAR/EAR files ?

**/*.war

Context path ?

Save Apply



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**

Not secure | 52.66.9.149:8080/job/testing/configure

Dashboard > testing >

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Execute shell

Command

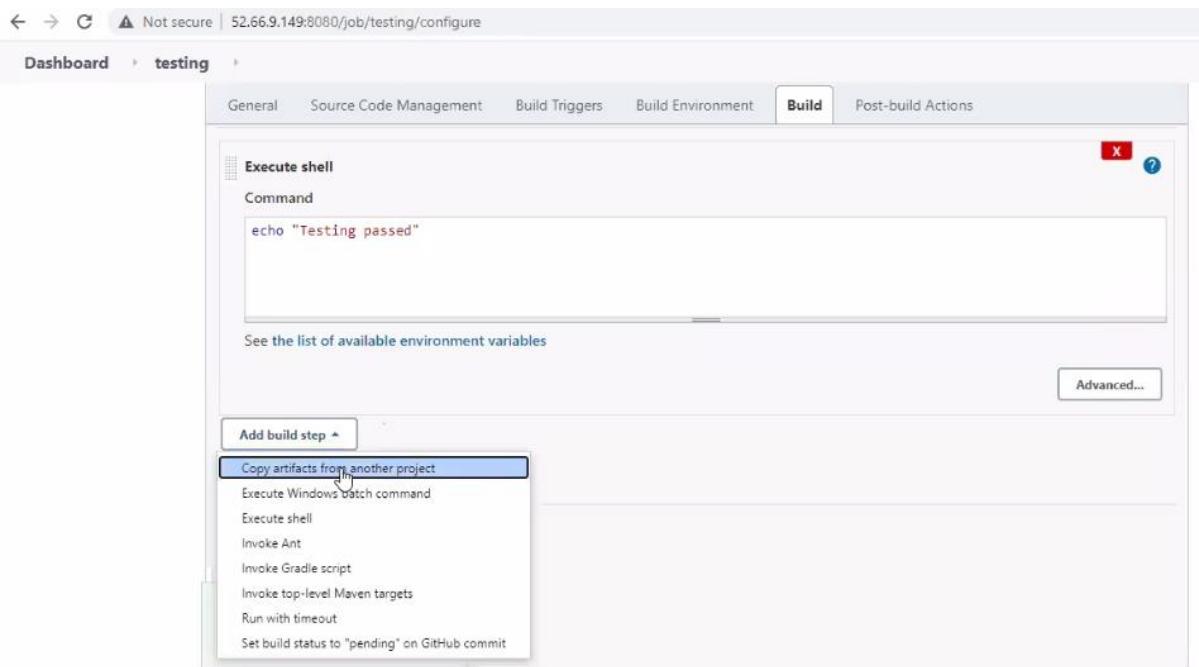
```
echo "Testing passed"
```

See the list of available environment variables

Add build step ▾

- Copy artifacts from another project** (selected)
- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit

Advanced...



13) Enter Development as project name

Dashboard > testing >

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Execute shell

Command

```
echo "Testing passed"
```

See the list of available environment variables

Copy artifacts from another project

Project name ?

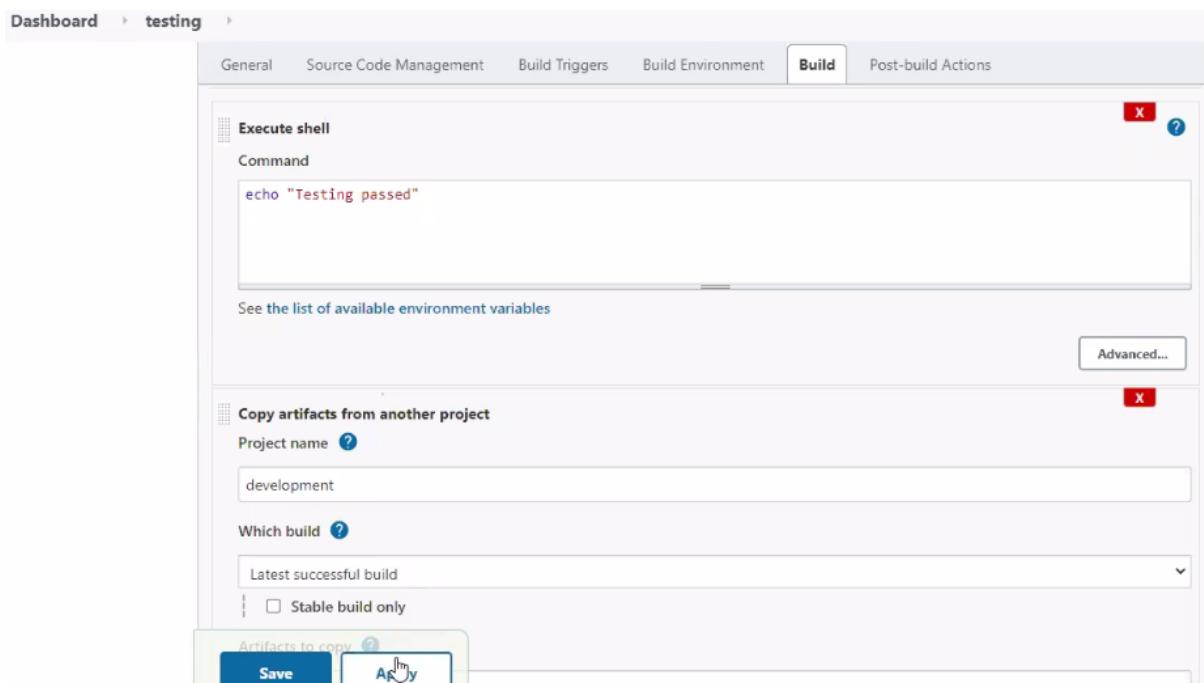
Which build ?

Latest successful build

Stable build only

Artifacts to copy 

Save **Apply**



14) For Deployment **Go to Post build actions section**

15) Click on **add post build action**

The screenshot shows the Jenkins configuration interface for the 'testing' job. The 'Post-build Actions' tab is selected. A context menu is open over the 'Deploy war/ear to a container' action, with the option highlighted. To the right of the menu, a checkbox labeled 'Fingerprint Artifacts' is checked. At the bottom of the screen, there are 'Save' and 'Apply' buttons.

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**

The screenshot shows the Jenkins configuration interface for the 'testing' job. The 'Post-build Actions' tab is selected. Under the 'Deploy war/ear to a container' section, the 'WAR/EAR files' field contains '**/*.war'. The 'Context path' field contains 'prodenv'. In the 'Containers' section, the 'Add Container' button is highlighted with a cursor. Below it is a checkbox for 'Deploy on failure'. At the bottom of the screen, there are 'Save' and 'Apply' buttons.

20) Select **tomcat 8**

21) Select your Credentials

22) Enter private ip:8080 of the prod server

<http://172.31.39.130:8080>

The screenshot shows the Jenkins interface for a job named 'testing'. The 'Post-build Actions' tab is active. Under the 'Containers' section, a 'Tomcat 8.x Remote' container is defined with a credential named 'learning/*****' and a Tomcat URL of 'http://172.31.40.134:8080'. There are buttons for 'Save' and 'Apply' at the bottom.

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

The screenshot shows the Jenkins interface with the 'Nodes' page selected. On the left sidebar, there are links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', 'Configure Clouds', and 'Node Monitoring'. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (master: 1 Idle, 2 Idle; Myslave: 0 Idle). The main content area displays a table of nodes:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	4.92 GB	0 B	4.92 GB	0ms
	Myslave	Linux (amd64)	In sync	N/A	N/A	N/A	N/A

At the bottom right of the table, there is a blue button labeled 'Refresh status'.

Configure job to run on slave

14) Select Testing Job

The screenshot shows the Jenkins interface. At the top, there's a search bar and navigation links for 'Dashboard' and 'All'. Below the header, a modal window titled 'Enter an item name' has 'sample' entered in the field, with a note '» Required field'. The main content area displays four project types: 'Freestyle project', 'Pipeline', 'Multi-configuration project', and 'Folder'. The 'Folder' option is selected, and its description states: 'This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.' Below this, the 'sample' job configuration is shown under the 'Build' tab. It includes build steps like 'Execute shell' with the command 'echo "Hello"', and buttons for 'Save' and 'Apply'.

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 slave with slave hardware)

18) Enter Label Expression (Slave_lab)

Dashboard > sample

General

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

Label Expression

Slave_lab

Label Slave_lab matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Source Code Management

None
 Git

Build Triggers

Save **Apply** (from scripts)

Advanced...

Apply ---> Save

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

Jenkins

Dashboard > sample > #1

Console Output

Started by user admin
Running as SYSTEM
Building remotely on `MySlave \Slave_lab` in workspace /home/ubuntu/workspace/workspace/sample
[sample] \$ /bin/sh -xe /tmp/jenkins5510569761258436607.sh
+ echo Hello
Hello
Finished: SUCCESS

View as plain text

Edit Build Information

Delete build '#1'

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

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

+++++

Dashboard > Plugin Manager

Back to Dashboard | Manage Jenkins

Updates Available Installed Advanced

Install	Name	Version	Release Date
<input type="checkbox"/>	Git Parameter	0.9.13	9 months
<input type="checkbox"/>	Build Pipeline	1.5.8	3 years

Install without restart | Download now and install after restart | Update information obtained: 1 min 4 sec ago | Check now

Ex:

Create new item --- Scripted Pipeline

Select pipeline –OK

Dashboard > All

Enter an item name

ScriptedPipeline » Required field

Freestyle project This is the central feature of Jenkins. Jenkins will build your project, combining any SCM used for something other than software build.

Maven project Build a maven project. Jenkins takes advantage of your POM files and drastically reduces configuration.

Pipeline Orchestrates long-running activities that can span multiple build agents. Suitable for building and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project Suitable for projects that need a large number of different configurations, such as testing.

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

Dashboard > ScriptedPipeline > Pipeline Syntax

 Back

-  [Snippet Generator](#)
-  [Declarative Directive Generator](#)
-  [Declarative Online Documentation](#)
-  [Steps Reference](#)
-  [Global Variables Reference](#)
-  [Online Documentation](#)
-  [Examples Reference](#)
-  [IntelliJ IDEA GDSDL](#)

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 

Generate Pipeline Script

ScriptedPipeline Config [Jenkins] Pipeline Syntax Snippet Generator +

Not secure | 65.0.184.6:8080/job/

Jenkins

Dashboard > ScriptedPipeline > Pipe

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

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator page. The left sidebar contains links to various Jenkins documentation and tools:

- Snippet Generator (selected)
- Declarative Directive Generator
- Declarative Online Documentation
- Steps Reference
- Global Variables Reference
- Online Documentation
- Examples Reference
- IntelliJ IDEA GDSL

The main content area has the following sections:

Overview

This **Snippet Generator** will help you learn the Pipeline Script. To use it, click **Generate Pipeline Script**, and you will see the whole statement generated for you to copy and paste the whole statement into your script, or pick up just the part you need (leaving them at default values.)

Steps

Sample Step

node: Allocate node

node

Label

master

Label master matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Generate Pipeline Script

leaving them at default values.)

Steps

Sample Step

node: Allocate node

node

Label

master

Label master matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Generate Pipeline Script

```
node('master') {  
    // some block  
}
```

Global Variables

Sample step - node: Allocate node

label - master

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

The screenshot shows the JFrog Pipeline Syntax Snippet Generator interface. On the left, there's a sidebar with various links. The main area has two tabs: 'Overview' and 'Steps'. In the 'Overview' tab, there's a note about the Snippet Generator, a 'Generate Pipeline Script' button, and a code editor showing Groovy pipeline syntax. In the 'Steps' tab, there's a 'Sample Step' section with a 'stage: Stage' dropdown menu containing 'stage', 'Stage Name', and 'Continuous Download'. A 'Generate Pipeline Script' button is also present here. The code editor shows the generated Groovy script.

```
node('master') {
    stage('Continuous Download') {
        // some block
    }
}
```

In pipeline syntax

Sample step - stage: Stage

Stage name - Continuous Download

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

The screenshot shows the Jenkins Pipeline Syntax tool. On the left, there's a sidebar with links like 'Dashboard', 'ScriptedPipeline', 'Pipeline Syntax', 'Global Variables Reference', 'Online Documentation', 'Examples Reference', and 'IntelliJ IDEA GDSL'. The main area has tabs for 'git: Git' and 'Untitled - Notepad'. In the 'git: Git' tab, fields include 'Repository URL' (set to 'https://github.com/sunildevops77/maven.git'), 'Branch' (set to 'master'), and 'Credentials' (set to '- none -'). Below these are checkboxes for 'Include in polling?' and 'Include in changelog?'. A 'Generate Pipeline Script' button is present, and the generated script is shown in the 'Untitled - Notepad' tab:

```

node('master')
{
    stage('Continuous Download')
    {
        git
        [
            url: 'https://github.com/sunildevops77/maven.git'
        ]
    }
}

```

At the bottom of the 'Untitled - Notepad' tab, status information is displayed: 'Ln 5, Col 55', '100%', 'Windows (CRLF)', and 'UTF-8'.

In Jenkins, scripted pipeline paste the code

Apply --- Save --> Run the job

The screenshot shows the Jenkins Pipeline configuration screen. At the top, there are tabs for 'General', 'Build Triggers', 'Advanced Project Options', and 'Pipeline'. The 'Pipeline' tab is selected. Under the 'Definition' section, there's a 'Pipeline script' tab. The 'Script' field contains the Groovy code from the previous screenshot:

```

node('master')
{
    stage('Continuous Download')
    {
        git
        [
            url: 'https://github.com/sunildevops77/maven.git'
        ]
    }
}

```

Below the script, there's a checkbox for 'Use Groovy Sandbox'. At the bottom of the 'Pipeline' tab, there are 'Save' and 'Apply' buttons. The 'Save' button is highlighted with a cursor.

The screenshot shows the Jenkins dashboard. At the top, there's a search bar and a user menu for 'admin'. Below the dashboard, there's a list of jobs. One job, 'ScriptedPipeline', is highlighted with a yellow sun icon. The job details show: Name: 'ScriptedPipeline', Last Success: N/A, Last Failure: N/A, Last Duration: N/A. There are also links for 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

Build now and check the status below in pipeline

Dashboard > ScriptedPipeline >

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Build History trend ^

find

#1 31-May-2021 05:28

Atom feed for all Atom feed for failures

Recent Changes

Stage View

Average stage times:
(Average full run time: ~3s)

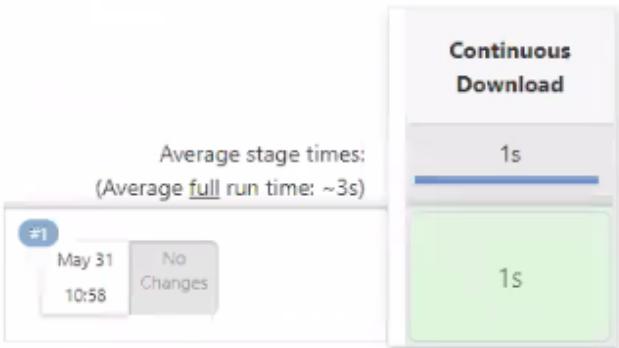
#1 May 31 10:58 No Changes

Continuous Download

1s

1s

Permalinks



++++++

2nd stage

We need to run 'mvn package' command.

This command can be executed as a shell script

In pipeline syntax:

Dashboard > ScriptedPipeline >

General Build Triggers Advanced Project Options Pipeline

Definition

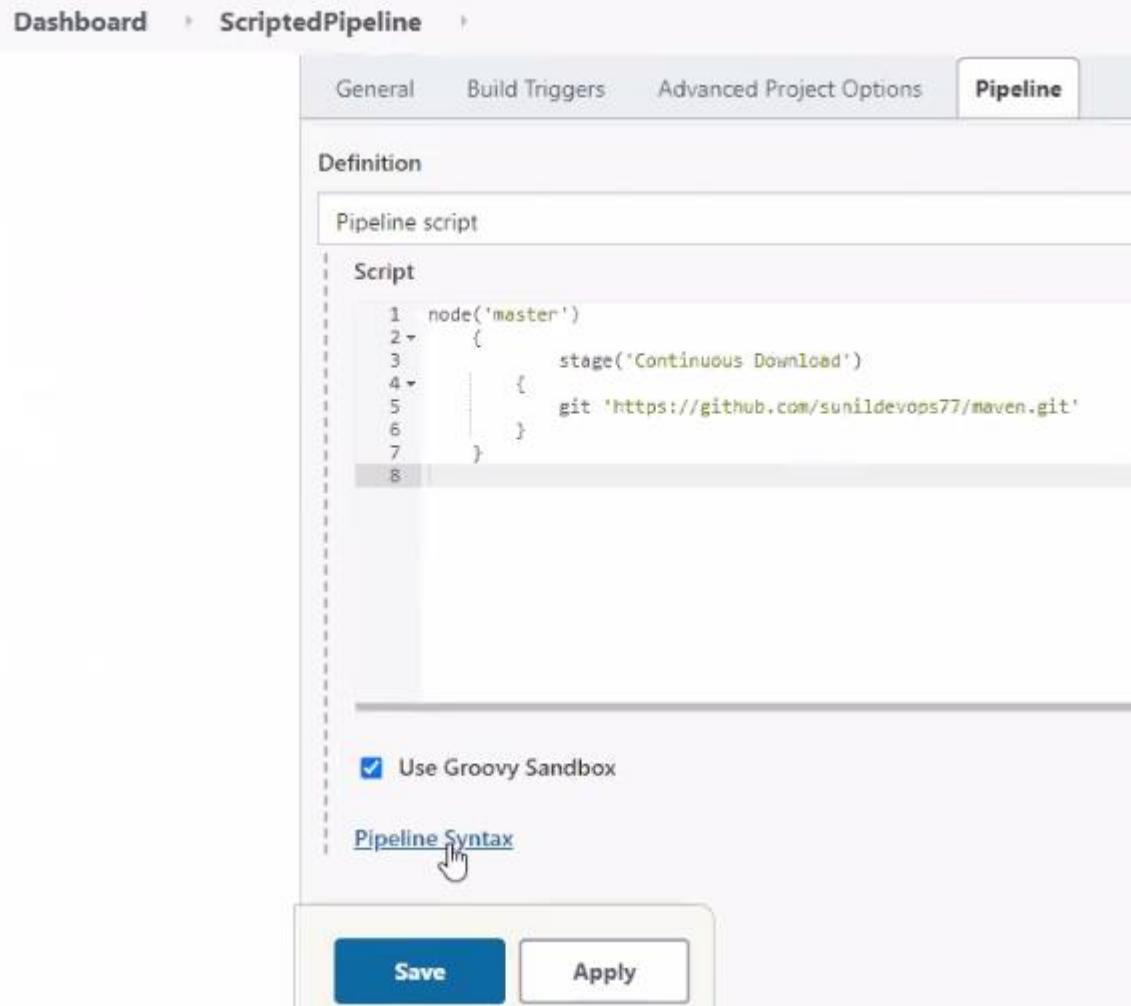
Pipeline script

```
1 node('master')
2 {
3     stage('Continuous Download')
4     {
5         git 'https://github.com/sunildevops77/maven.git'
6     }
7 }
```

Use Groovy Sandbox

[Pipeline Syntax](#)

[Save](#) [Apply](#)



Sample step - sh: Shell Script

Stage name - mvn package

Dashboard > ScriptedPipeline > Pipeline Syntax

Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL

leaving them at default values.)

Steps

Sample Step

```
sh: Shell Script
```

```
sh
```

Shell Script

```
mvn package
```

Generate Pipeline Script

```
sh 'mvn package'
```

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

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

Dashboard > ScriptedPipeline >

General Build Triggers Advanced Project Options Pipeline

Definition

Pipeline script

```
1 node('master')
2 {
3     stage('Continuous Download')
4     {
5         git 'https://github.com/sunildevops77/maven.git'
6     }
7     stage('Continuous Build')
8     {
9         sh 'mvn package'
10    }
11}
12
```

Use Groovy Sandbox

Pipeline Syntax

Save **Apply**

The screenshot shows the Jenkins Pipeline configuration page for a project named "ScriptedPipeline". The "Pipeline" tab is selected. The "Definition" section contains a "Pipeline script" block. Inside the script, a "node('master')" block is defined. This block contains two nested "stage" blocks: "Continuous Download" and "Continuous Build". The "Continuous Download" stage performs a "git" operation to clone a repository from a GitHub URL. The "Continuous Build" stage runs a "sh 'mvn package'" command. A "Use Groovy Sandbox" checkbox is checked. At the bottom, there are "Save" and "Apply" buttons, with "Apply" being highlighted by a mouse cursor.

Save and run.

The screenshot shows the Jenkins Pipeline ScriptedPipeline interface. On the left, there's a sidebar with various options: Back to Dashboard, Status, Changes, Build Now (which is selected and highlighted in blue), Configure, Delete Pipeline, Full Stage View, Rename, Pipeline Syntax, and Build History. The Build History section shows a search bar with 'find' and a list item '#2 31-May-2021 05:32'. To the right, the main area is titled 'Pipeline ScriptedPipeline' and 'Stage View'. It displays two stages: 'Continuous Download' (964ms) and 'Continuous Build' (2s). Below these are two boxes labeled '#2' and '#1', each containing a timestamp (May 31) and a 'No Changes' message. A summary at the top states 'Average stage times: (Average full run time: ~3s)'.

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

```
ubuntu@ip-172-31-41-7:~$ cat > file1
sdfdsfdsf
dsfgfdsgdfsgfd
dsfgfdgfps
'dfgfdgdfs
ubuntu@ip-172-31-41-7:~$ 
ubuntu@ip-172-31-41-7:~$ 
ubuntu@ip-172-31-41-7:~$ ls
file1 jenkins.war
ubuntu@ip-172-31-41-7:~$ scp file1 ubuntu@172.31.47.36:/tmp/file2
file1                                         100%   46    78.5KB/s  00:00
ubuntu@ip-172-31-41-7:~$ |
```

Lets check for the file, by connecting to qa server

```
$ ssh ubuntu@172.31.47.36
```

```
$ cd /tmp
```

```
$ ls
```

```
$ cat file2
```

```
$ exit
```

```
ubuntu@ip-172-31-47-36:~$ ls
ubuntu@ip-172-31-47-36:~$ cd /tmp
ubuntu@ip-172-31-47-36:/tmp$ 
ubuntu@ip-172-31-47-36:/tmp$ ls
file2
hsperfdata_tomcat8
systemd-private-d1304f8348484869916e17c7563ac921-systemd-resolved.service-ac5
systemd-private-d1304f8348484869916e17c7563ac921-systemd-timesyncd.service-Gh
g
tomcat8-tomcat8-tmp
ubuntu@ip-172-31-47-36:/tmp$ cat file2
sdfdsfdsf
dsfgfdsgdfsgfd
dsfgfdgfps
dfgfdgdfs
ubuntu@ip-172-31-47-36:/tmp$ exit
logout
Connection to 172.31.47.36 closed.
ubuntu@ip-172-31-41-7:~$ |
```

+++++

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

Get the location of war file from log

Dashboard > ScriptedPipeline > #2

```
[1m--- [0;32m[maven-surefire-plugin:2.11:test@m @ [1m(default-test)m @ [36mwebapp@0;1m ---@[m
[1;34mINFO@n] No tests to run.
[1;34mINFO@n] Surefire report directory: /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/surefire-reports

-----
T E S T S
-----

Results :

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

[1;34mINFO@n]
[1m--- [0;32m[maven-war-plugin:2.2:war@m @ [1m(default-war)m @ [36mwebapp@0;1m ---@[m
[1;34mINFO@n] Packaging webapp
[1;34mINFO@n] Assembling webapp [webapp] in [/home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp]
[1;34mINFO@n] Processing war project
[1;34mINFO@n] Copying webapp resources [/home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/src/main/webapp]
[1;34mINFO@n] Webapp assembled in [33 ms]
[1;34mINFO@n] Building war: /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war
[1;34mINFO@n] WEB-INF/web.xml already added, skipping
[1;34mINFO@n] [1m-----@[m
ref+authentifat information required for Maven Default + download@1m
-----@[m
```

\$ 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

Dashboard > ScriptedPipeline > Pipeline Syntax

- [Declarative Online Documentation](#)
- [Steps Reference](#)
- [Global Variables Reference](#)
- [Online Documentation](#)
- [Examples Reference](#)
- [IntelliJ IDEA GDSL](#)

leaving them at default values.)

Steps

Sample Step

sh: Shell Script

sh

Shell Script

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

Advanced...

Generate Pipeline Script

```
sh "scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.47.36:/var/lib/tomcat8/webapps/qaenv.war"
```

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

The screenshot shows the Jenkins Pipeline configuration interface. At the top, there's a navigation bar with 'Dashboard' and 'ScriptedPipeline'. Below it, tabs for 'General', 'Build Triggers', 'Advanced Project Options', and 'Pipeline' are visible, with 'Pipeline' being the active tab. The main area is titled 'Definition' and contains a 'Pipeline script' section. A large text area displays the Groovy script provided above. Below the script, there's a checkbox labeled 'Use Groovy Sandbox' which is checked. At the bottom of the screen, there are two buttons: 'Save' and 'Apply'.

```

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
"""
    }
}

```

Apply --- save -- run

Deployment fails

Dashboard > ScriptedPipeline

- Status
- Changes
- Build Now**
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Stage View

Average stage times:
(Average full run time: ~6s)

Continuous Download	Continuous Build	Continuous Deployment
870ms	7s	880ms
683ms	6s	880ms failed
493ms	8s	
1s		

Build History

- find
- #3 31-May-2021 05:48
- #2 31-May-2021 05:32
- #1 31-May-2021 05:28

Observe the log file (permissions denied)

- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Stage View

Average stage times:
(Average full run time: ~6s)

Continuous Download	Continuous Build
870ms	7s
683ms	6s
493ms	8s
1s	

Failed with the following error(s)

Shell Script script returned exit code 1
See stage logs for more detail.

Logs

880ms
failed

Build History

- find
- #3 31-May-2021 05:48
- #2 31-May-2021 05:32
- #1 31-May-2021 05:28

Stage Logs (Continuous Deployment)

```
Shell Script -- scp /home/ubuntu/jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.47.36:/var/lib/tomcat8/webapps/qaenv.war (self time 851ms)

+ scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.47.36:/var/lib/tomcat8/webapps/qaenv.war
scp: /var/lib/tomcat8/webapps/qaenv.war: Permission 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$ sudo service ssh force-reload
ubuntu@ip-172-31-47-36:/etc/ssh$ cd /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$ |
```

Now run the job

The screenshot shows a CI pipeline interface with a sidebar on the left and a main 'Stage View' area on the right.

Left Sidebar:

- Status
- Changes
- Build Now** (highlighted)
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Right Area:

Stage View

Average stage times:
(Average full run time: ~6s)

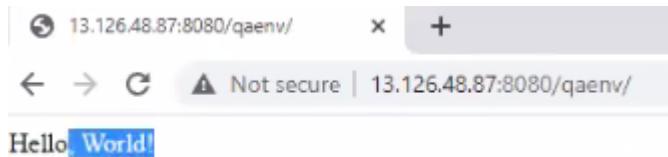
Continuous Download	Continuous Build	Continuous Deployment
763ms	7s	863ms
443ms	6s	847ms
683ms	6s	880ms (failed)
493ms	8s	

Build History:

#	Date	Status
#4	31-May-2021 05:50	Success
#3	31-May-2021 05:48	Failure
#2	31-May-2021 05:32	Success
#1	31-May-2021 05:28	Success

+++++=

Connect qa server and check



++++++

4th Stage: Continuous testing

In pipeline -- add a new stage

Shell script -- echo "Tesing Passed"

Steps

Sample Step

sh: Shell Script

sh

Shell Script

echo "Tesing Passed"

?

Advanced...

Generate Pipeline Script

sh echo "Tesing Passed"

I

```
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

Dashboard > ScriptedPipeline >

General Build Triggers Advanced Project Options Pipeline

Definition

Pipeline script

```

1  Script
2  {
3      git 'https://github.com/sunildevops77/maven.git'
4  }
5  stage('Continuous Build')
6  {
7      sh 'mvn package'
8  }
9  stage('Continuous Deployment')
10 {
11     sh '''scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.47.36:/var/www/html'''
12 }
13 stage('Continuous Testing')
14 {
15     sh 'echo "Testing Passed"'
16 }
17
18 }
19
20 }
```

try sample Pipeline... ?

Use Groovy Sandbox ?

Pipeline Syntax

Save **Apply** ?

Apply -- save-- run

Dashboard > ScriptedPipeline >

Back to Dashboard Status Changes Build Now Configure Delete Pipeline Full Stage View Rename Pipeline Syntax Build History trend ^

find #5 31-May-2021 05:53

Pipeline ScriptedPipeline

Stage View

Continuous Download	Continuous Build	Continuous Deployment	Continuous Testing
706ms	7s	859ms	296ms
476ms	6s	852ms	296ms
443ms	6s	847ms	

Average stage times: (Average full run time: ~7s)

#5 May 31 11:23 No Changes

#4 May 31 11:20 No Changes

+++++
+++++

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 "Testing 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

The screenshot shows the Jenkins Pipeline configuration page. At the top, there's a green bar with a checkmark icon and the word 'Saved'. Below it, the tabs are General, Build Triggers, Advanced Project Options, and Pipeline, with Pipeline being the active tab. The Pipeline section contains a 'Definition' box labeled 'Pipeline script'. Inside the script editor, the Groovy code for the pipeline is pasted. A checkbox labeled 'Use Groovy Sandbox' is checked. At the bottom of the editor are 'Save' and 'Apply' buttons.

```

script {
    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 "Testing 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'"
    }
}

```

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
```

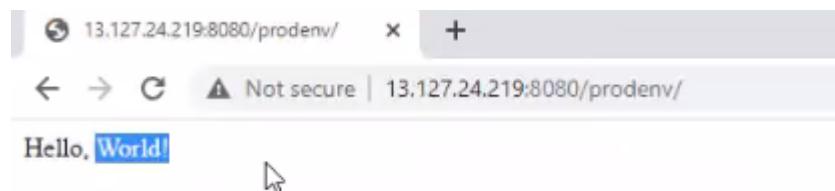
(Observation: tomcat8 directory -- others is not having write permissions)

```
$ 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 contain 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.

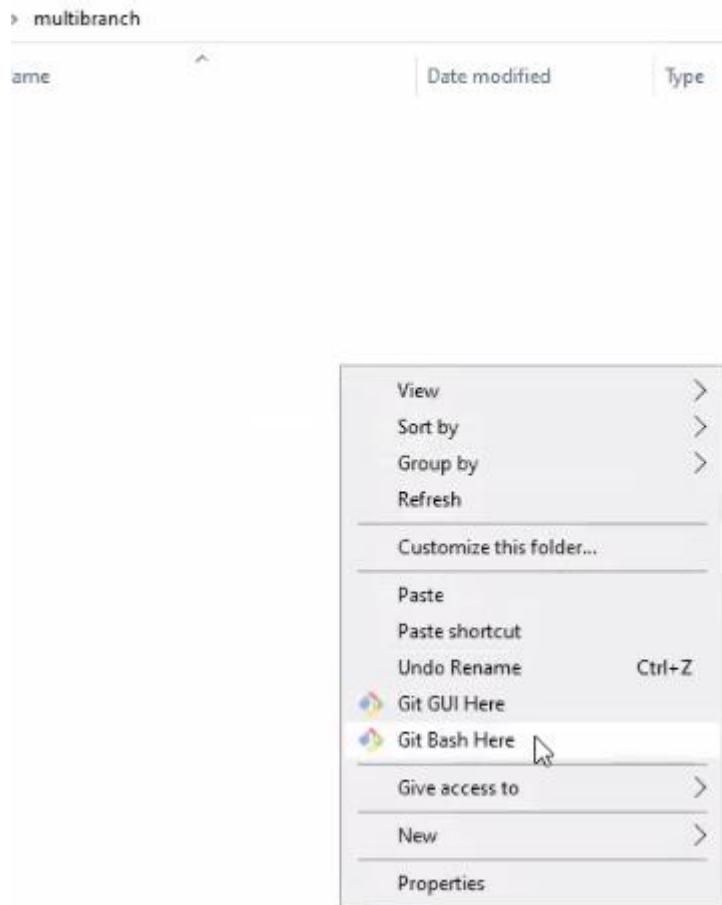
All these branches along with Jenkins file will be uploaded by into the github repository.

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

This PC > Local Disk (E) >				
	Name	Date modified	Type	Size
stch	client32	14-02-2017 05:25 PM	File folder	
itch	CORELDRAW-X7	07-12-2020 05:09 PM	File folder	
M Batch	Kubernetes	29-09-2020 11:59 AM	File folder	
	RHEL5.7_VM	19-05-2021 05:06 PM	File folder	
	vmware15	01-07-2020 04:43 PM	File folder	
	multibran	01-06-2021 10:38 AM	File folder	

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



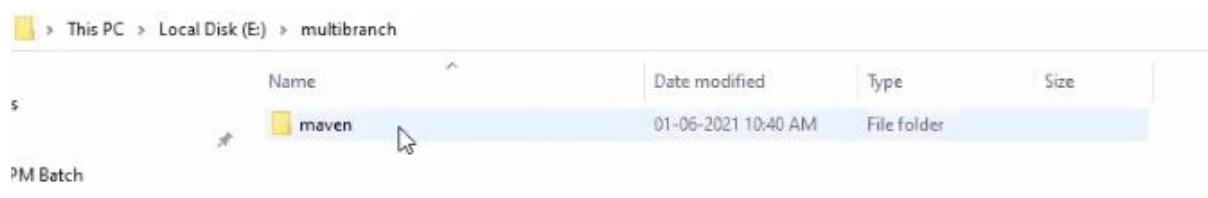
```
$ mkdir multibranch
```

```
$ cd multibranch
```

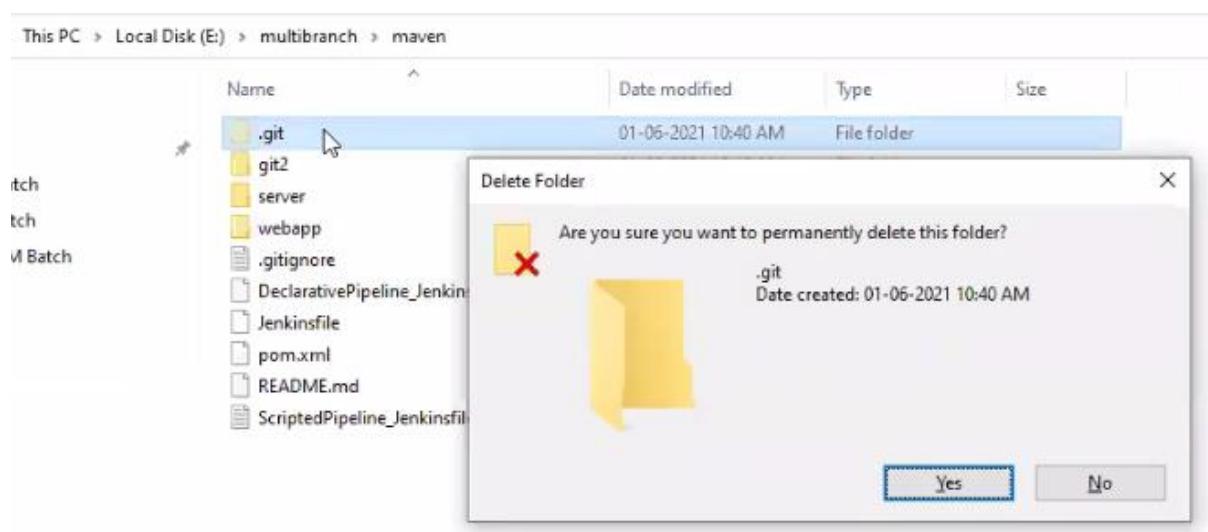
Download the files of maven repository—code is downloaded and Maven folder is created.

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

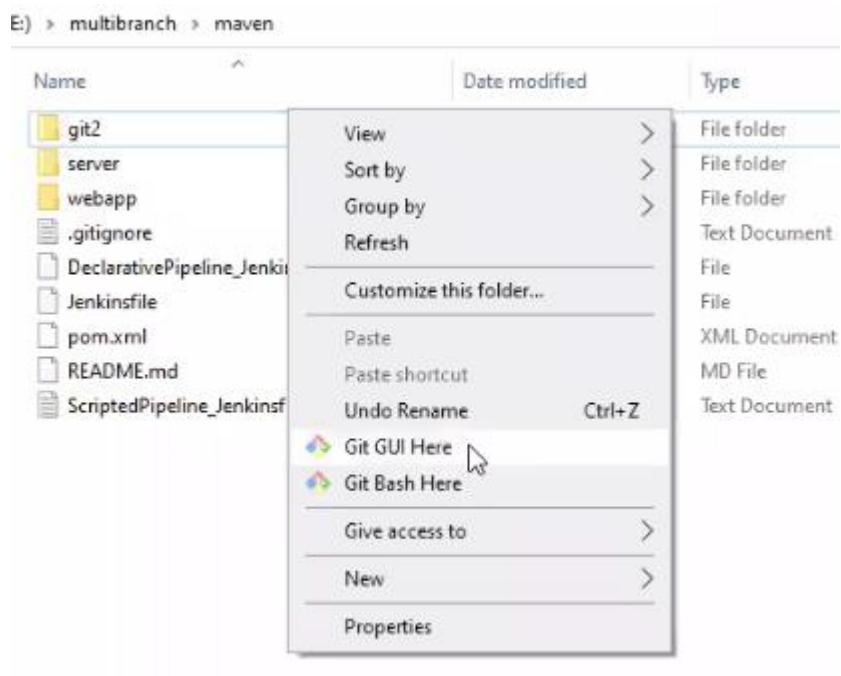
```
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch
$ git clone https://github.com/sunildevops77/maven.git
cloning into 'maven'...
remote: Enumerating objects: 321, done.
remote: Counting objects: 100% (321/321), 37.93 KiB | 882.00 KiB/s, done.
remote: Total 321 (delta 0), reused 0 (delta 0), pack-reused 321
Receiving objects: 100% (321/321), 37.93 KiB | 882.00 KiB/s, done.
Resolving deltas: 100% (148/148), done.
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch
$
```



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)

```
MINGW64:/e/multibranch/maven
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven
$ git init
Initialized empty Git repository in E:/multibranch/maven/.git/
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$
```

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.

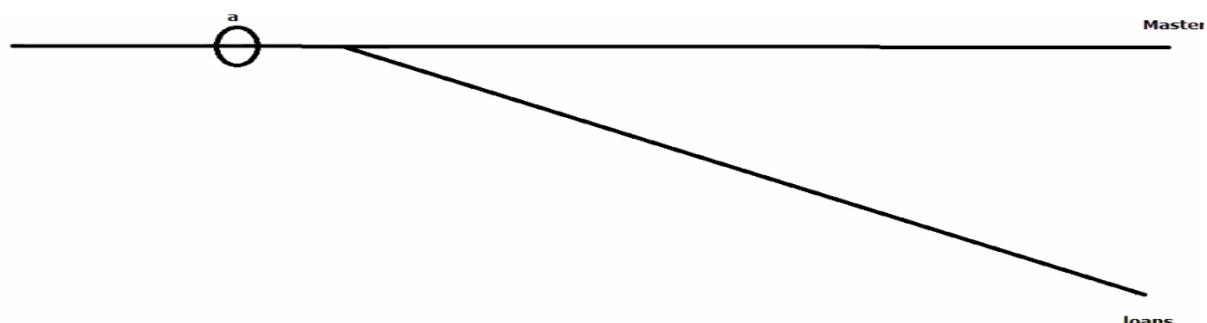
It is pointing to Loan branch .

```

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

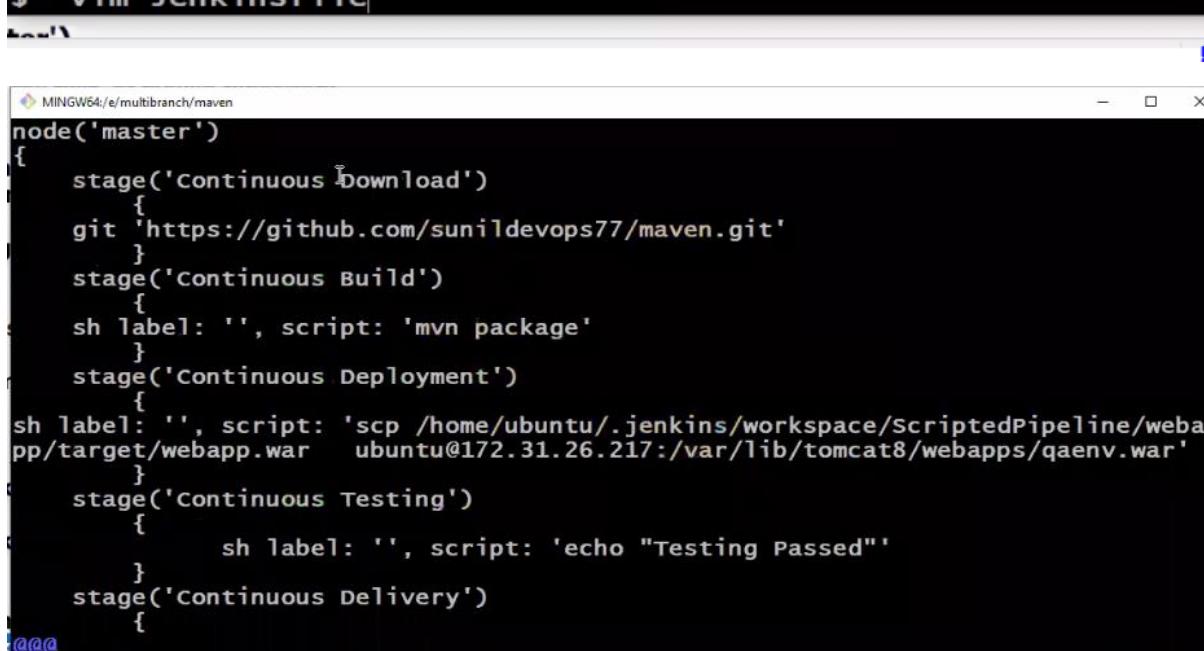
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$ |
```

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|
```



```
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@172.31.26.217:/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

```
MINGW64:/e/multibranch/maven
node('master')
{
    stage('Continuous Download')
    {
        git 'https://github.com/sunildevops77/maven.git'
    }
    stage('Continuous Build')
    {
        sh label: '', script: 'mvn package'
    }
}
Jenkinsfile[+] [dos] (10:40 01/06/2021)
: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 Jenkins file has been modified.

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

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

```
$ git add .
```

```
$ git commit -m "b"
```

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

admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$ git commit -m "b"
[master 94759db] b
 1 file changed, 11 insertions(+), 23 deletions(-)
 rewrite Jenkinsfile (72%)

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

```
$ git checkout loans--- go to Loans Branch
```

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

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

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

```
$ vim Jenkinsfile --
```

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

```
)
```

```
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/webapp/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')
    {
}
@@@ Jenkinsfile [dos] (10:47 01/06/2021) 11,1 Top
"Jenkinsfile" [dos] 23L, 715B
```

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

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

```
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 reposi

```
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)

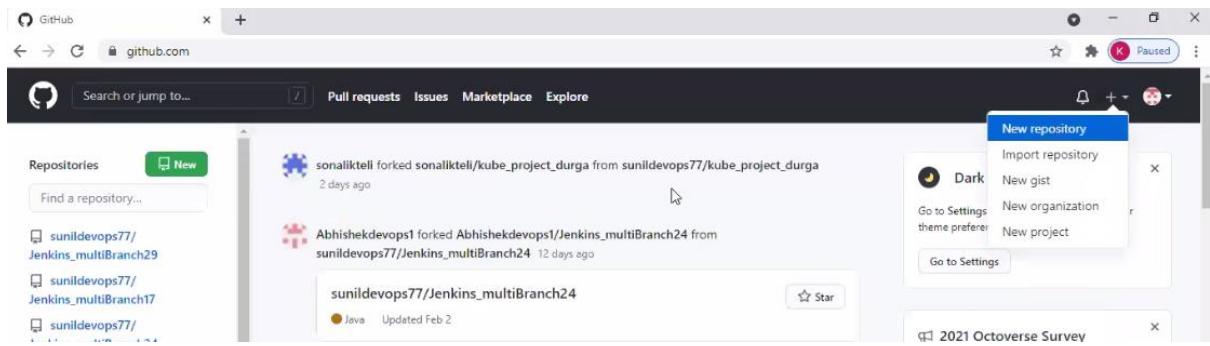
\$ git checkout master

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

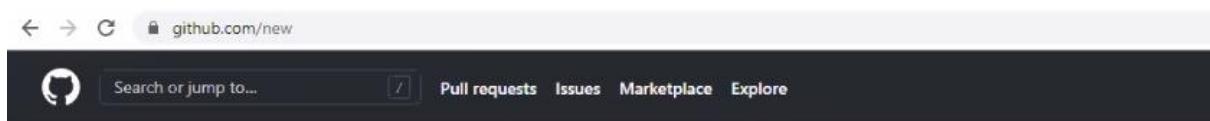
admin@DESKTOP-CSV8R5E MINGW64 /e/multibranch/maven (master)
$
```

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

Create new repository in github



Repository name - Jenkins_multiBranch24



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner * Repository name *

sunildevops77 / Jenkins_multiBranch01 [?](#)

Great repository names are short and memorable. Need inspiration? How about [expert-invention](#)?

Description (optional)

 Public Anyone on the internet can see this repository. You choose who can commit.

 Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more](#).

[←](#) [→](#) [C](#) [github.com/new](#)

Great repository names are short and descriptive. Jenkins_multiBranch01 is available. What's in a name? How about [expert-invention?](#)

Description (optional)

 **Public**
Anyone on the internet can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. [Learn more](#).

Add .gitignore
Choose which files not to track from a list of templates. [Learn more](#).

Choose a license
A license tells others what they can and can't do with your code. [Learn more](#).

Create repository

[←](#) [→](#) [C](#) [github.com/sunildevops77/Jenkins_multiBranch01](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) https://github.com/sunildevops77/Jenkins_multiBranch01.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

...or create a new repository on the command line

```
echo "# Jenkins_multiBranch01" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/sunildevops77/Jenkins_multiBranch01.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/sunildevops77/Jenkins_multiBranch01.git
git branch -M main
git push -u origin main
```

Copy above code and paste in terminal

```
$ git remote add origin https://github.com/sunildevops77/Jenkins_multiBranch01.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.git

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)

sunildevops77 / Jenkins_multibranch01

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master had recent pushes less than a minute ago

Compare & pull request

Go to file Add file Code

2 branches 0 tags

sunilkumark11 c

cb987fd 6 minutes ago 2 commits

server	a	13 minutes ago
webapp	a	13 minutes ago
.gitignore	a	13 minutes ago
DeclarativePipeline_Jenkinsfile	a	13 minutes ago
Jenkinsfile	c	6 minutes ago
README.md	a	13 minutes ago
ScriptedPipeline_Jenkinsfile.txt	a	13 minutes ago
pom.xml	a	13 minutes ago

About
No description, website, or topics provided.

Readme

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Can see 2 brancges

master had recent pushes less than a minute ago

Compare & pull request

loans 2 branches 0 tags

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 c

README.md a

ScriptedPipeline_Jenkinsfile.txt a

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

	sunilkumark11 c	cb987fd · 6 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	c	6 minutes ago
	README.md	a	14 minutes ago
	ScriptedPipeline_Jenkinsfile.txt	a	14 minutes ago

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

Will create a Jenkins job to process both Branches.

Login to jenkins

New item – MultiBranchPipeline

The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with various links: Dashboard, New Item (which is highlighted with a blue border), People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Lockable Resources, and New View. Below this is a 'Build Queue' section with the message 'No builds in the queue.' In the center, there is a 'Welcome to Jenkins!' message and a 'Start building your software project' section with a 'Create a job' button. At the top right, there is a search bar, a user icon, and a 'log out' link.

Gve the jb name-Select multibranch Pipeline gi

Jenkins

Dashboard > All >

Enter an item name

job1
* Required field

 **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

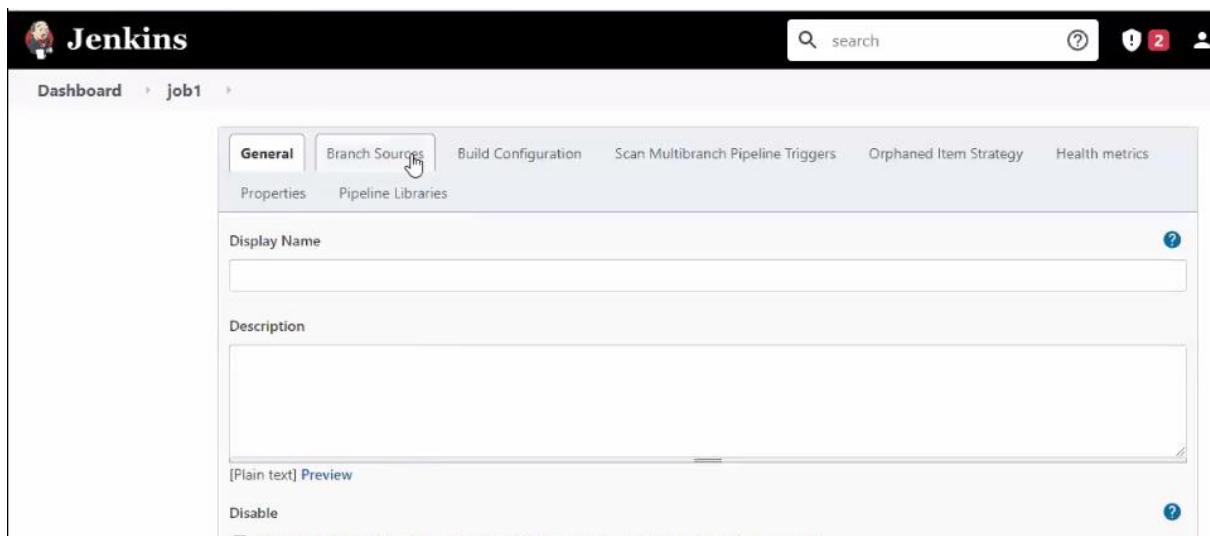
 **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

 **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

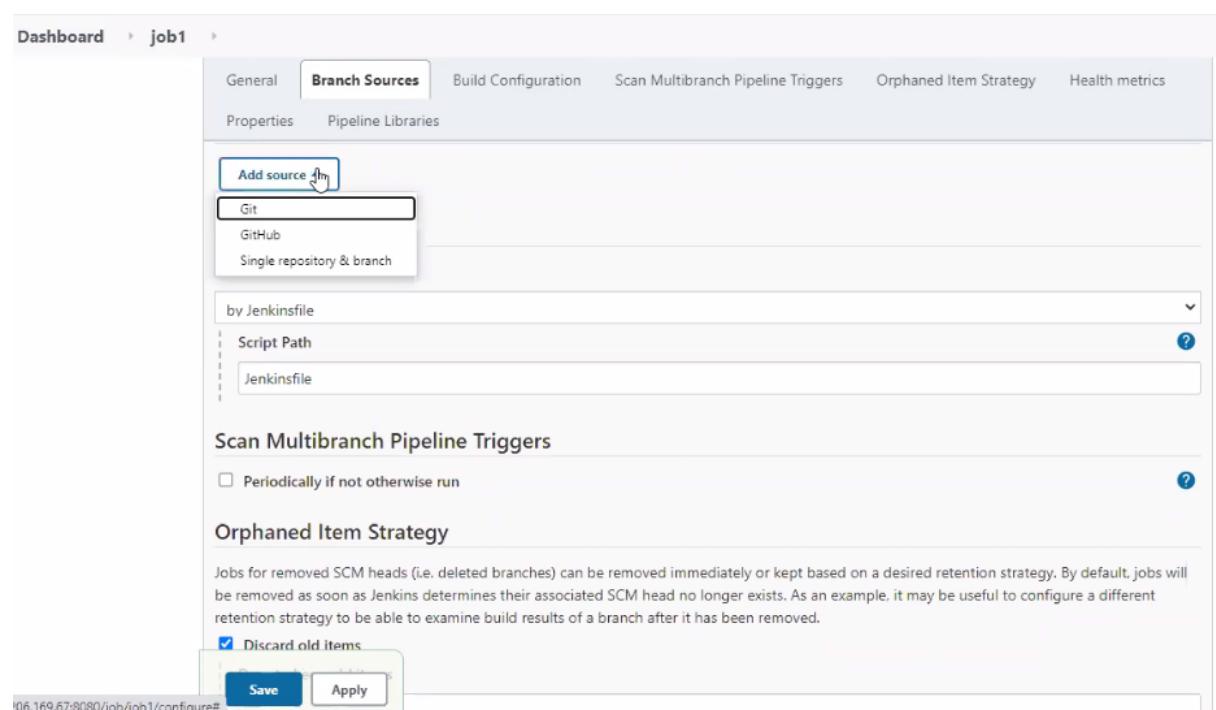
Branch Sources



The screenshot shows the Jenkins configuration page for a job named "job1". The "General" tab is selected. The "Branch Sources" tab is highlighted with a mouse cursor. Other tabs include "Build Configuration", "Scan Multibranch Pipeline Triggers", "Orphaned Item Strategy", and "Health metrics". The "Properties" and "Pipeline Libraries" sections are also visible.

Add source

Git



The screenshot shows the Jenkins configuration page for "job1" under the "Branch Sources" tab. The "Add source" button is highlighted with a mouse cursor. A dropdown menu is open, showing options: "Git", "GitHub", and "Single repository & branch". Below the dropdown, there are fields for "by Jenkinsfile" (set to "Script Path") and "Script Path" (containing "Jenkinsfile"). The "Scan Multibranch Pipeline Triggers" section includes a checkbox for "Periodically if not otherwise run". The "Orphaned Item Strategy" section contains a note about removing deleted SCM heads and a checkbox for "Discard old items" which is checked. At the bottom are "Save" and "Apply" buttons.

The screenshot shows the Jenkins job configuration for 'job1'. The 'Branch Sources' tab is selected. Under the 'Git' section, the 'Project Repository' field contains the URL https://github.com/sunildevops77/Jenkins_multiBranch01.git. The 'Behaviours' section contains a single item: 'Discover branches'.

Project Repository -- https://github.com/sunildevops77/Jenkins_multiBranch01.git

The screenshot shows the GitHub repository page for 'sunildevops77/Jenkins_multiBranch01'. The 'Code' tab is selected. The repository has 2 branches and 0 tags. The 'loans' branch is currently selected. On the right, there is a 'Clone' button with the HTTPS URL https://github.com/sunildevops77/Jenkins_multiBranch01.

Scan multiline pipeline triggers

Check periodically if not otherwise

Interval - 1 minute

Dashboard > job1 >

General Branch Sources Build Configuration Scan Multibranch Pipeline Triggers Orphaned Item Strategy Health metrics

Properties Pipeline Libraries

Periodically if not otherwise run

Interval

1 day

1 minute

2 minutes

5 minutes

10 minutes

15 minutes

20 minutes

25 minutes

30 minutes

1 hour

2 hours

4 hours

8 hours

12 hours

1 day

2 days

1 week

2 weeks

4 weeks

Save Apply

Apply --- Save then back to Dashboard

By this time it will be started.

← → C Not secure | 15.206.169.67:8080

Dashboard

- Manage Jenkins
- My Views
- Lockable Resources
- New View

Build Queue

No builds in the queue.

Build Executor Status

Node	Job	Build #	Status
master	job1 » master	#1	(Continuous Build)
master	job1 » loans	#1	(Continuous Build_loans)
Myslave	(1 launching...)		

This job will check github every minute.

Jenkins

Dashboard

- New Item
- People
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins

All	+	S	W	Name	Last Success	Last Failure	Last Duration	Icon: S M L	Legend	Atom feed for all	Atom feed for failures	Atom feed for just latest builds
				job1	5.8 sec - log	N/A	2.4 sec					

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

The screenshot shows the Jenkins dashboard for a multibranch pipeline named 'job1'. On the left sidebar, there are several options: Up, Status, Configure, Scan Multibranch Pipeline Now, Scan Multibranch Pipeline Log, Multibranch Pipeline Events, Delete Multibranch Pipeline, People, Build History, and Project Relationship. The main content area is titled 'job1' and displays a table for 'Branches (2)'. The table has columns: S, W, Name, Last Success, Last Failure, and Last Duration. It lists two branches: 'loans' and 'master'. Both branches have a blue circle icon (Status), a yellow sun icon (Working), and a '26 sec - #1' entry under 'Last Success'. Under 'Last Failure', it says 'N/A'. The 'Last Duration' is '22 sec'. There are three green circular icons with arrows at the bottom right of the table. A legend below the table indicates: 'Icon: S M L', 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'. A tooltip 'Icon: S M L' is shown over the first column.

Click on master branch

This screenshot is similar to the previous one, but the 'master' branch is now selected. The 'master' row in the 'Branches (2)' table has a mouse cursor hovering over it. The rest of the interface remains the same, including the sidebar options and the table structure.

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

The screenshot shows the Jenkins dashboard for the 'master' branch of 'job1'. The sidebar includes: Up, Status, Changes, Build Now, View Configuration, Full Stage View, Pipeline Syntax, Build History (selected), trend, find, and Atom feeds for all, failures, and latest builds. The main content area is titled 'Pipeline master' with the full project name 'job1/master'. It features a 'Recent Changes' section with a notebook icon. Below that is the 'Stage View' section, which displays two stages: 'Continuous Download' (1s) and 'Continuous Build' (17s). A note states 'Average stage times: (Average full run time: ~22s)'. At the bottom, there is a summary for build '#1' from Jun 01 at 11:01, noting 'No Changes'. The 'Permalinks' section at the bottom right lists the last build as '#1, 1 min 13 sec ago'.

The screenshot shows the Jenkins Pipeline master dashboard for the project 'job1/master'. The left sidebar includes links for Up, Status, Changes, Build Now, View Configuration, Full Stage View, Pipeline Syntax, and Build History. The main area displays the 'Pipeline master' title, the full project name 'job1/master', and a 'Stage View' section. This view shows average stage times of 1s for download and 17s for build. Below this is a table of recent builds:

#	Jun 01	No
#1	Jun 01	No

Select multibranch pipeline

You will find two branches

Select loans , we can see two stages

The screenshot shows the Jenkins job1 dashboard. The left sidebar includes links for Up, Status, Configure, Scan Multibranch Pipeline Now, Scan Multibranch Pipeline Log, Multibranch Pipeline Events, Delete Multibranch Pipeline, and People. The main area displays the 'job1' title and a 'Branches (2)' table:

S	W	Name	Last Success	Last Failure	Last Duration
●	☀	loans	26 sec - #1	N/A	22 sec
●	☀	master	26 sec - #1	N/A	22 sec

Icons S M L are shown below the table, along with a legend for Atom feed links.

The screenshot shows the Jenkins interface for the 'Pipeline loans' job. On the left, there's a sidebar with links like 'Up', 'Status', 'Changes', 'Build Now', 'View Configuration', 'Full Stage View', 'Pipeline Syntax', and 'Build History'. The 'Build History' section shows a single entry for 'Jun 01 11:01' with 'No Changes'. The main content area has a title 'Pipeline loans' and a subtitle 'Full project name: job1/loans'. It features a 'Recent Changes' section with a pencil icon and a 'Stage View' section with two stages: 'Continuous Download_loans' (1s) and 'Continuous Build_loans' (17s). Below these are 'Permalinks' for Atom feeds.

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

This screenshot shows a GitHub repository named 'loans'. At the top, it displays '2 branches' and '0 tags'. The main area lists recent commits by 'sunilkumark11 c' with commit hash 'cb987fd' and a timestamp of '15 minutes ago'. The commits include changes to 'server', 'webapp', '.gitignore', 'DeclarativePipeline_Jenkinsfile', 'Jenkinsfile', 'README.md', 'ScriptedPipeline_Jenkinsfile.txt', and 'pom.xml'. To the right, there are sections for 'provided.', 'Readme', 'Releases', 'Packages', and 'Languages'. A note at the bottom states 'New changes done by developer'.

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.

The screenshot shows a GitHub commit dialog for a file named README.md in the loans branch of a repository named Jenkins_multiBranch01. The file contains the following content:

```
1 New changes done by developer
2
3 some more changes donddddddddd
4
5
6 cccccc
7
8 gfggdgfdgfdgdf
9
10
```

Below the file content, there is a note: "Attach files by dragging & dropping, selecting or pasting them." The dialog has a title "Commit changes" and two buttons: "Commit changes" (highlighted with a mouse cursor) and "Cancel". There are also fields for "Update README.md" and "Add an optional extended description...".

At the bottom of the dialog, there are two radio button options:

- Commit directly to the loans branch.
- Create a new branch for this commit and start a pull request. Learn more about pull requests.

The screenshot shows the Jenkins Pipeline loans dashboard. On the left, there's a sidebar with links: Up, Status (which is selected), Changes, Build Now, View Configuration, Full Stage View, Pipeline Syntax, and Build History. The Build History section shows two recent builds: #2 (01-Jun-2021 05:36) and #1 (01-Jun-2021 05:31), both with 'No Changes' noted. Below the sidebar is a 'Recent Changes' section with a notebook icon.

The main area is titled 'Pipeline loans' with the full project name 'job1/loans'. It features a 'Stage View' section with a table comparing 'Continuous Download_loans' (845ms) and 'Continuous Build_loans' (12s). Below this are two tables of stages: one for build #2 (Jun 01 11:06) and one for build #1 (Jun 01 11:01), both showing 'No Changes'.

```
$ 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
```

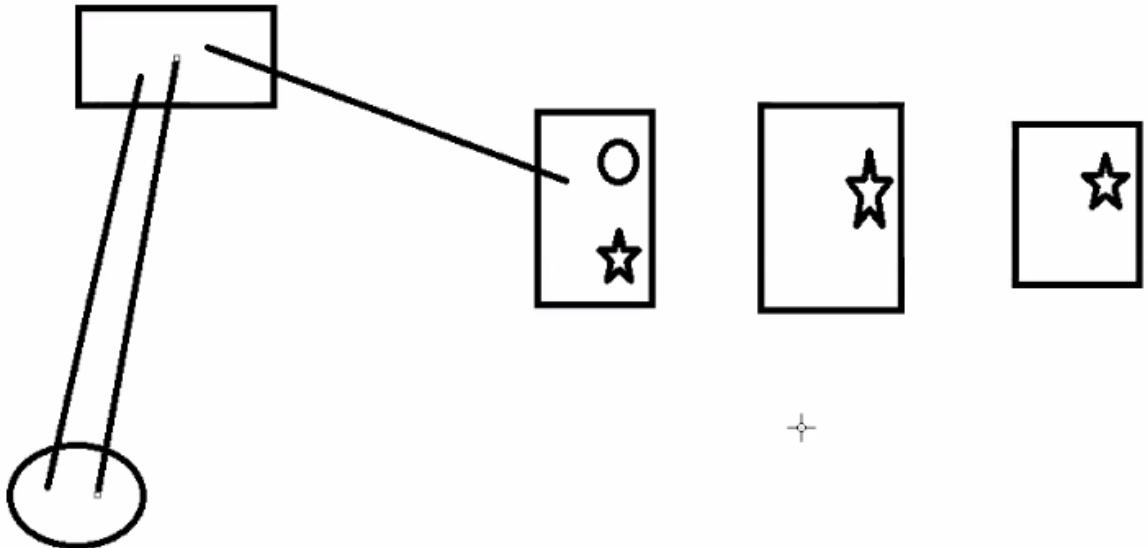
To push all the branches

```
$ 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 to 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**

The screenshot shows the Jenkins dashboard with a sidebar on the left containing links like 'New Item', 'People', 'Build History', etc. The main area displays a list of jobs. A context menu is open over the job named 'job1'. The menu items include:

- Configure
- Scan Multibranch Pipeline Now
- Scan Multibranch Pipeline Log
- Multibranch Pipeline Events
- Delete Multibranch Pipeline
- People
- Build History
- Project Relationship
- Check File Fingerprint
- Rename
- Pipeline Syntax

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

The screenshot shows the Jenkins dashboard with the following sections:

- Dashboard** (selected)
- Build History**
- Project Relationship**
- Check File Fingerprint**
- Manage Jenkins** (selected)
- My Views**
- Lockable Resources**
- New View**

System Configuration

- Configure System**: Configure global settings and paths.
- Global Tool Configuration**: Configure tools, their locations and automatic installers.
- Manage Nodes and Clouds**: Add, remove, control and monitor the nodes under that build over labels.
- Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. A red warning icon indicates updates are available.

Warnings (red box):

- No implementation of access control for builds is present. It is recommended that you install [Authorize Project Plugin](#) or another plugin implementing the [QueueItemAuthenticator](#) extension point.

Go to plugin manager | **Configure which of these warnings are shown**

Email Notification

The screenshot shows the "Email Notification" configuration page with the following fields:

- Content Token Reference**
- E-mail Notification**
- SMTP server**: smtp.gmail.com
- Default user e-mail suffix**: (empty field)
- Test configuration by sending test e-mail**:
- Save** | **Apply**
- Default Triggers...**
- Advanced...** (button with a notepad icon)

SMTP Server - smtp.gmail.com

Click on Advance button (with notepad icon)

E-mail Notification

SMTP server

Default user e-mail suffix

Use SMTP Authentication

User Name

Password

Use SSL

Use TLS

SMTP Port

USE SMTP Authentication

Username - sunildevops77@gmail.com

Password - password for the above email

use SSL

SMTP Port - 465

Dashboard > configuration

Use SSL

Use TLS

SMTP Port

Reply-To Address

Charset

Test configuration by sending test e-mail

Test e-mail recipient

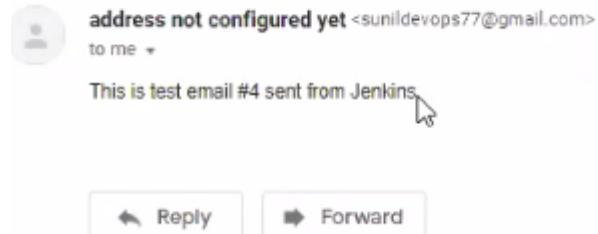
Test Configuration by sending e-mail.

Test email Recipient - sunildevops77@gmail.com

(We should get success message)

The screenshot shows the Jenkins configuration page for an email plugin. The 'Test configuration by sending test e-mail' checkbox is checked, and the recipient is set to 'sunildevops77@gmail.com'. A message box indicates 'Email was successfully sent'. At the bottom, there are 'Save' and 'Apply' buttons.

Test email #4 Inbox X



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

The screenshot shows the Jenkins interface with a search dialog box. The title of the dialog is "Enter an item name". Inside the dialog, the word "job" is typed into a search input field. Below the input field, a list of suggestions is displayed: "job1", "Job1", "job5", and "job`1". The background of the Jenkins dashboard is visible, showing the "Dashboard" and "All" navigation links.

The screenshot shows a "Select project type" dialog box. At the top, it says "Enter an item name" and has a field containing "job2" with a note "» Required field". Below this, there are four options: "Freestyle project" (selected), "Maven project", "Pipeline", and "Multi-configuration project". Each option has a small icon and a brief description. A blue "OK" button is at the bottom left of the dialog.

- Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific

Build tab→add build step→Execute shell

Dashboard > job2 >

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Delete workspace before build starts
 Use secret text(s) or file(s)
 Abort the build if it's stuck
 Add timestamps to the Console Output
 Inspect build log for published Gradle build scans
 With Ant

Build

Add build step ▾

- Conditional step (single)
- Conditional steps (multiple)
- Copy artifacts from another project
- Execute Windows batch command
- Execute shell**
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit
- Trigger/call builds on other projects

Give echo statement → apply → save

Dashboard > job2 >

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Delete workspace before build starts
 Use secret text(s) or file(s)
 Abort the build if it's stuck
 Add timestamps to the Console Output
 Inspect build log for published Gradle build scans
 With Ant

Build

Execute shell

Command

```
echo "Hello"
```

See the list of available environment variables

Advanced...

Add build step ▾

Save **Apply**

Post build actions → add post build → email notification

The screenshot shows the Jenkins job configuration interface for a job named 'job2'. The 'Post-build Actions' tab is selected. A context menu is open over the 'E-mail Notification' option, which is highlighted with a blue background. The menu lists several actions: Aggregate downstream test results, Archive the artifacts, Build other projects, Publish JUnit test result report, Publish Javadoc, Record fingerprints of files to track usage, Git Publisher, Build other projects (manual step), Deploy war/ear to a container, E-mail Notification (highlighted), Editable Email Notification, Set GitHub commit status (universal), Set build status on GitHub commit [deprecated], Trigger parameterized build on other projects, and Delete workspace when build is done. At the bottom of the menu is a 'Save' button.

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

The screenshot shows the 'Post-build Actions' configuration page. It features an 'E-mail Notification' section with a 'Recipients' input field containing 'sunildevops77@gmail.com'. Below the input field is a note: 'Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.' There are two checkboxes: 'Send e-mail for every unstable build' (checked) and 'Send separate e-mails to individuals who broke the build' (unchecked). At the bottom are 'Save' and 'Apply' buttons.

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

Jenkins

Dashboard >

New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Lockable Resources

All +

S	W	Name	Last Success	Last Failure	Last Duration
Icon: S M L	job2	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

Dashboard > job2 >

Back to Dashboard

Status

Changes

Workspace

Build Now

Configure

Delete Project

Rename

Build History

trend ^

find X

Workspace

Recent Changes

Project job2

Permalinks

Build now → refresh the browser → status → job executed

job2 [Jenkins] x +

Not secure | 65.2.143.56:8080/job/job2/

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

Wantedly make an error, we get email triggered.

Dashboard > job2 >

[BACK TO DASHBOARD](#)

[Status](#) [Changes](#) [Workspace](#) [Build Now](#) [Configuration](#) [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 [?](#)

[Save](#) [Apply](#) [Advanced...](#)

Dashboard > job2 >

- General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Execute shell

Command

```
echo llo"
```

See the [list of available environment variables](#)

[Advanced...](#)

[Add build step ▾](#)

Post-build Actions

E-mail Notification

Recipients

Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

Save **Apply** **Unstable build**

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

Jenkins

Dashboard >

- New Item People Build History Project Relationship Check File Fingerprint

All	W	Name	Last Success	Last Failure	Last Duration
		job2	48 sec - #1	N/A	0.13 sec

Icon: S M L

Legend: Atom feed for all Atom feed for failures Atom feed for just latest builds

[Builds scheduled](#)

Jenkins

Dashboard >

- New Item People Build History Project Relationship Check File Fingerprint Manage Jenkins

All	W	Name	Last Success	Last Failure	Last Duration
		job2	55 sec - #1	4.2 sec - #2	0.13 sec

Icon: S M L

Legend: Atom feed for all Atom feed for failures Atom feed for just latest builds

Primary Social 2 new Nagalounika Jampani via Li...

Promotions 50 new Thomas Loepp Thiessen, Erica ...

me me

Build failed in Jenkins: job2 #2 - See <<http://52.66.160.163:8080/job/job2/2/display/redirect>> Changes: -----

Test email #4 - This is test email #4 sent from Jenkins 10:45 AM

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 * * *

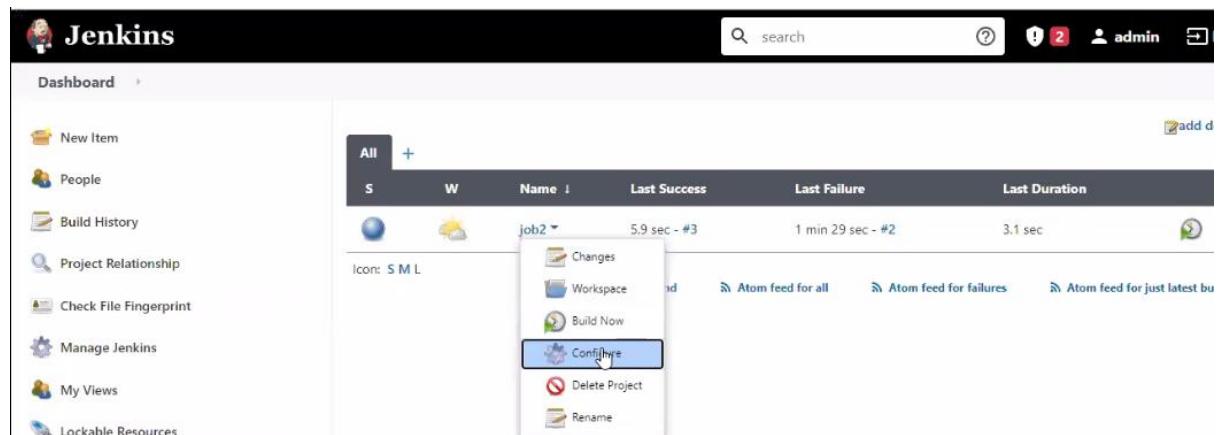
Ex2: 1=mon; 5=fri

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 scour code management tab

Build Triggers --- Build periodically

Schedule -

The screenshot shows the Jenkins job configuration interface for a job named 'job2'. The 'Build Triggers' tab is active. The 'Build periodically' checkbox is checked. A tooltip 'Help for feature: Build periodically' is displayed when hovering over the question mark icon next to the checkbox. Other trigger options like 'Trigger builds remotely' and 'Build after other projects are built' are available but unchecked.

Click on Question mark besides Schedule for help

Give ****

Apply--save

The screenshot shows the Jenkins job configuration interface for a job named 'job2'. The 'Build Triggers' tab is active. The 'Build periodically' checkbox is checked. The 'Schedule' field contains '*****'. A tooltip 'Do you really mean "every minute" when you say "*****"? Perhaps you meant "H * * * *" to poll once per hour' is displayed when hovering over the field. Below the field, it says 'Would last have run at Wednesday, June 2, 2021 5:24:31 AM UTC; would next run at Wednesday, June 2, 2021 5:24:31 AM UTC.' A detailed cron syntax explanation follows:

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:
MINUTE Minutes within the hour (0-59)
HOUR The hour of the day (0-23)
DOM The day of the month (1-31)
MONTH The month (1-12)
DOW The day of the week (0-7) where 0 and 7 are Sunday.

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

Save Apply

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.

The screenshot shows the Jenkins interface for the 'job2' project. The top navigation bar includes links for 'Dashboard', 'job2', 'Logout', and 'Help'. Below the header, there's a sidebar with links: 'Back to Dashboard', 'Status' (which is selected), 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', and 'Rename'. A 'Build History' section displays three builds: #4 (green circle, Jun 2021 05:25), #3 (green circle, Jun 2021 05:19), and #2 (red circle, Jun 2021 05:18). To the right, the main content area is titled 'Project job2'. It features sections for 'Recent Changes' (with a blue folder icon) and 'Permalinks' (with a notepad icon). The 'Permalinks' section lists the following items:

- Last build (#3), 4 min 52 sec ago
- Last stable build (#3), 4 min 52 sec ago
- Last successful build (#3), 4 min 52 sec ago
- Last failed build (#2), 6 min 16 sec ago
- Last unsuccessful build (#2), 6 min 16 sec ago
- Last completed build (#3), 4 min 52 sec ago

Waiting for the approval from delivery head

New item → job → pipeline job → ok

The screenshot shows the Jenkins dashboard with a search bar and notification icons. A modal window titled "Enter an item name" is open, containing a text input field with "job1" and a note "» Required field". Below the input field, there are four project type options: "Freestyle project", "Maven project", "Pipeline", and "Multi-configuration project". Each option has a brief description and a corresponding icon. The "Pipeline" option is highlighted with a light gray background.

job1
» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific

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

The screenshot shows the Jenkins pipeline configuration screen for the "job1" project. The top navigation bar includes "Dashboard", "job1", and tabs for "General", "Build Triggers", "Advanced Project Options", and "Pipeline". The "Pipeline" tab is selected. The main area contains a "Description" text input field and a "Plain text" preview area. Below these are several configuration options with help icons: "Discard old builds", "Do not allow concurrent builds", "Do not allow the pipeline to resume if the master restarts", "GitHub project", "Permission to Copy Artifact", "Pipeline speed/durability override", "Preserve stashes from completed builds", "This project is parameterised", and "Throttle builds". At the bottom are "Save" and "Apply" buttons.

General Build Triggers Advanced Project Options Pipeline

Description

[Plain text] Preview

Discard old builds ?
 Do not allow concurrent builds ?
 Do not allow the pipeline to resume if the master restarts ?
 GitHub project ?
 Permission to Copy Artifact ?
 Pipeline speed/durability override ?
 Preserve stashes from completed builds ?
 This project is parameterised ?
 Throttle builds ?

Save Apply

Dashboard > job1 >

General Build Triggers Advanced Project Options Pipeline

Definition

Pipeline script

```

19
20 stage('Continuous Deployment')
21
22 {
23
24 sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.47.36:/v'
25 }
26 stage('Continuous Testing')
27
28 {
29 sh label: '', script: 'echo "Testing Passed"'
30 }
31 stage('Continuous Delivery')
32
33 {
34 sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/ScriptedPipeline/webapp/target/webapp.war ubuntu@172.31.40.13'
35 }
36

```

try sample Pipeline... ?

Use Groovy Sandbox ?

Pipeline Syntax

Save **Apply**

Back to dashboard → click on job→

Jenkins

Dashboard

- New Item
- People
- Build History
- Project Relationship
- Check File Fingerprint

All	W	Name	Last Success	Last Failure	Last Duration
S	W	job1	N/A	N/A	N/A

Icon: S M L

Legend:  Atom feed for all  Atom feed for failures  Atom feed for just latest build

Build now to run the job

Jenkins

Dashboard > job1

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Build Now](#) (highlighted)

[Build scheduled](#)

[Delete Pipeline](#)

[Full Stage View](#)

[Rename](#)

[Pipeline Syntax](#)

Build History trend ^

find

Pipeline job1

Stage View

No data available. This Pipeline has not yet run.

Permalinks

Jenkins

Dashboard > job1

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Build Now](#)

[Configure](#)

[Delete Pipeline](#)

[Full Stage View](#)

[Rename](#)

[Pipeline Syntax](#)

Build History trend ^

find

Pipeline job1

Stage View

Average stage times:
(Average full run time: ~15s)

Continuous Download	Continuous build	Continuous Deployment	Continuous Testing	Continuous Delivery
1s	9s	1s	296ms	1s
1s	9s	1s	296ms	1s

#1 Jun 02 10:58 No Changes

Permalinks

The screenshot shows the Jenkins interface for a pipeline job named 'job1'. The left sidebar contains the following items:

- Back to Dashboard
- Status (selected)
- Changes
- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Below the sidebar, there is a 'Build History' section with a single entry:

```
node('master')
```

Pipeline job1



Stage View

No data available. This Pipeline has not yet run.

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

Jenkins

Dashboard > job1 >

Back to Dashboard Status Changes Build Now Configure Delete Pipeline Full Stage View Rename Pipeline Syntax Build History trend ▾

Pipeline job1

Recent Changes

Add description Disable Project

Stage View

Average stage times:
(Average full run time: ~15s)

Continuous Download	Continuous build	Continuous Deployment	Continuous Testing	Continuous Delivery
1s	9s	1s	296ms	1s
1s	9s	1s	296ms	1s

Jun 02 10:58 No Changes

This screenshot shows the Jenkins Pipeline job1 interface. On the left, there's a sidebar with links like Back to Dashboard, Status, Changes, Build Now, Configure (which is selected and highlighted in blue), Delete Pipeline, Full Stage View, Rename, Pipeline Syntax, and Build History. Below the sidebar is a 'trend' dropdown set to 'trend'. The main area is titled 'Pipeline job1' and contains a 'Recent Changes' section with a date and time stamp ('Jun 02 10:58') and a note ('No Changes'). To the right is a 'Stage View' section with a table showing average stage times for five stages: Continuous Download (1s), Continuous build (9s), Continuous Deployment (1s), Continuous Testing (296ms), and Continuous Delivery (1s). The table has three rows of data.

Jenkins

Dashboard > job1 >

General Build Triggers Advanced Project Options Pipeline

Description

[Plain text] Preview

Discard old builds ?
 Do not allow concurrent builds ?
 Do not allow the pipeline to resume if the master restarts ?
 GitHub project ?
 Permission to Copy Artifact ?
 Pipeline speed/durability override ?
 Preserve stashes from completed builds ?
 This project is parameterised ?
 Throttle builds ?

Save Apply

This screenshot shows the Jenkins Pipeline job1 configuration page. It features tabs for General, Build Triggers, Advanced Project Options, and Pipeline (which is selected and highlighted in blue). The Pipeline tab contains a 'Description' field with a rich text editor and a preview link. Below the editor is a list of pipeline settings with checkboxes: 'Discard old builds', 'Do not allow concurrent builds', 'Do not allow the pipeline to resume if the master restarts', 'GitHub project', 'Permission to Copy Artifact', 'Pipeline speed/durability override', 'Preserve stashes from completed builds', 'This project is parameterised', and 'Throttle builds'. At the bottom are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins Pipeline configuration page for a job named 'job1'. The 'Pipeline' tab is selected. The 'Definition' section contains a 'Pipeline script' editor. The script content is:

```

1 node('master')
2
3 {
4
5   stage('Continuous Download')
6
7   {
8     git 'https://github.com/sunildevops77/maven.git'
9   }
10
11 }
12
13 stage('Continuous build')
14
15 {
16   // This is a comment
17 }
18

```

Use Groovy Sandbox

[Pipeline Syntax](#)

Save **Apply**

The screenshot shows the Jenkins Snippet Generator interface. The title bar says 'job1 Config [Jenkins] Pipeline Syntax: Snippet Generator'. The main content area is titled 'Pipeline Syntax'.

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

Generate Pipeline Script

Select Wait for interactive input

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. On the left, there's a sidebar with links like Dashboard, Snippet Generator (which is selected), Declarative Directive Generator, etc. The main area shows a list of Pipeline steps. One step, 'input: Wait for interactive input', is highlighted with a blue background. Below it, there are two configuration fields: 'archiveArtifacts' and 'Files to archive'. At the bottom right is a 'Generate Pipeline Script' button.

Message: enter as waiting for approval

This screenshot shows the same Jenkins Pipeline Syntax Snippet Generator interface as above, but with a different step selected. A 'Sample Step' section shows the configuration for the 'input: Wait for interactive input' step. It includes fields for 'Input' (set to 'Waiting for Approval') and 'Message'. The 'Generate Pipeline Script' button at the bottom is highlighted with a blue background.

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→

Pipeline job1

Stage View

Average stage times:
(Average full run time: ~15s)

Continuous Download	Continuous build	Continuous Deployment	Continuous Testing	Continuous Delivery
1s	9s	1s	296ms	1s
1s	9s	1s	296ms	1s

Build History trend ▲

It was paused on stage 5 as seen below

Dashboard > job1 >

Pipeline job1

Stage View

Average stage times:
(Average full run time: ~15s)

Continuous Download	Continuous build	Continuous Deployment	Continuous Testing	Continuous Delivery
1s	7s	1s	295ms	873ms
602ms	6s	847ms	294ms	almost complete (Paused for 8s)
1s	9s	1s	296ms	1s

Build History trend ▲

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

Dashboard > job1 >

[Back to Dashboard](#)

- [Status](#)
- [Changes](#)
- [Build Now](#)
- [Configure](#)
- [Delete Pipeline](#)
- [Full Stage View](#)
- [Rename](#)
- [Pipeline Syntax](#)

Build History trend ^

#	Date	Time	Status
#2	Jun 02	10:58	No Changes

Average stage times:
(Average full run time: ~15s)

Stage View

Continuous Download	Continuous build	Continuous Deployment	Continuous Testing	Continuous Delivery
1s	7s	1s	295ms	873ms
602ms	6s	847ms	296ms	1s
almost complete				
1s	9s	1s	296ms	1s

Waiting for Approval

[Proceed](#) [Abort](#)

Once you click on Approve button it gets into prod

Dashboard > job1 >

[Back to Dashboard](#)

- [Status](#)
- [Changes](#)
- [Build Now](#)
- [Configure](#)
- [Delete Pipeline](#)
- [Full Stage View](#)
- [Rename](#)
- [Pipeline Syntax](#)

Build History trend ^

#	Date	Time	Status
#2	Jun 02	11:01	No Changes
#1	Jun 02	10:58	No Changes

Average stage times:
(Average full run time: ~27s)

Stage View

Continuous Download	Continuous build	Continuous Deployment	Continuous Testing	Continuous Delivery
1s	7s	1s	295ms	1s
602ms	6s	847ms	296ms	949ms (queued for 28s)
1s	9s	1s	296ms	1s