

Pipelines in Jenkins

This document covers the instructions to create Jenkins Pipeline and configure various stages involved within it. This document can be divided into the following three sections:

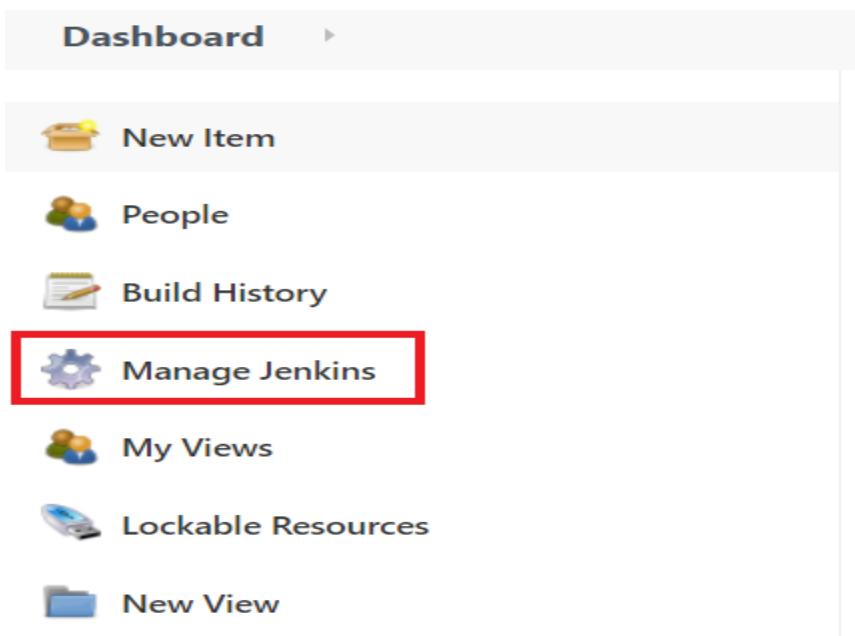
1. Creating the Pipeline Template
2. Using Template to create the main pipeline
3. Checking code into Jenkins file

Now, login to your Ubuntu instance and start Jenkins over it to proceed forward with these sections.

Creating the Pipeline Template:

So, let's see how to create the basic skeleton of the pipeline, and what are the various steps involved in it.

1. In the **Dashboard**, go to **Manage Jenkins**.



2. Then click on **Manage Plugins**.

System Configuration



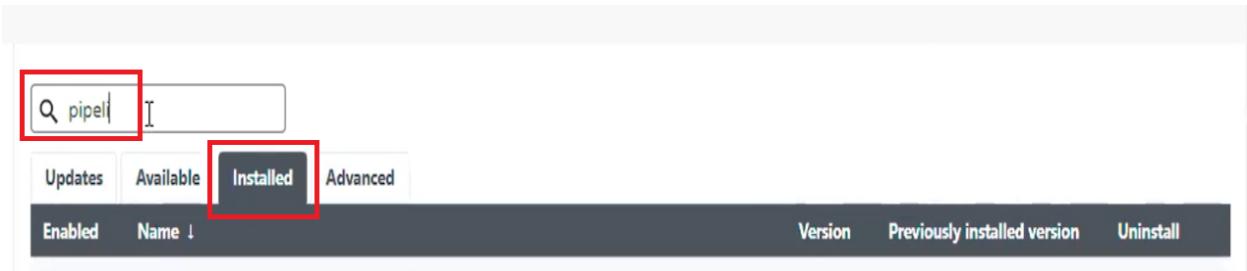
Configure System
Configure global settings and paths.

Global Tool Configuration
Configure tools, their locations and automatic installers.

Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
⚠ There are updates available

- Then, in the **Installed tab** search for **Pipeline** plugin, i.e., to check whether it is already installed or not.

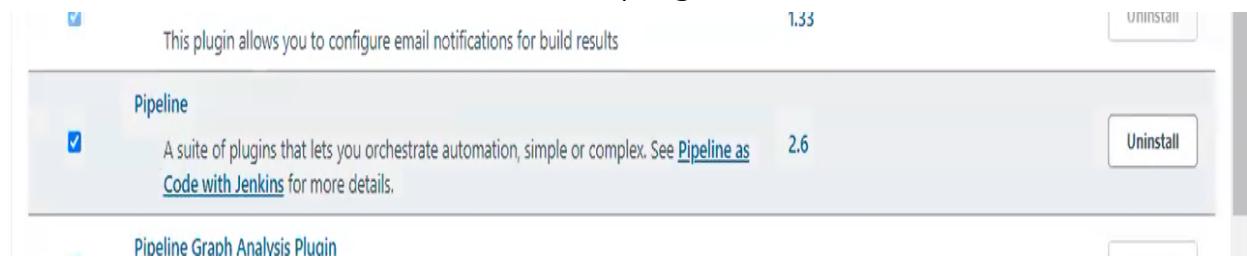


Search bar: Q pipeline

Tabs: Updates, Available, **Installed**, Advanced

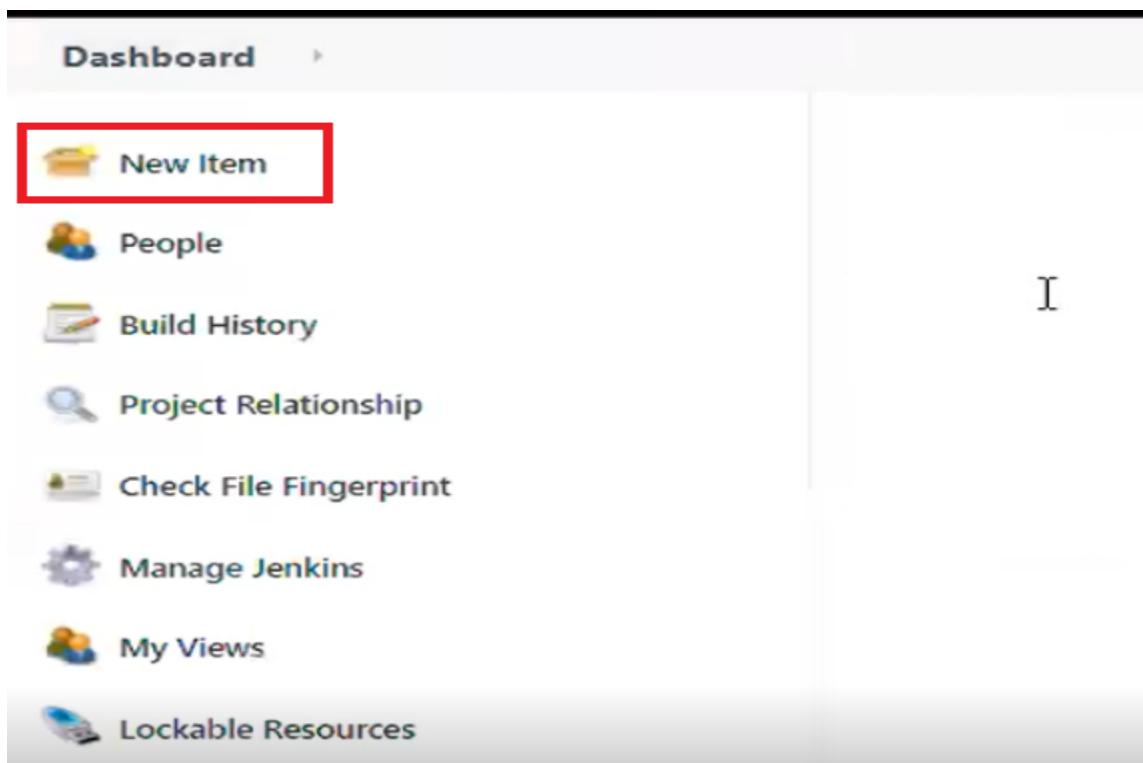
Buttons: Enabled, Name ↓, Version, Previously installed version, Uninstall

- So, scroll down and here it can be seen that the **Pipeline** plugin is already installed. If it is not installed, then install it by going to the **Available** tabs and then search for the plugin.

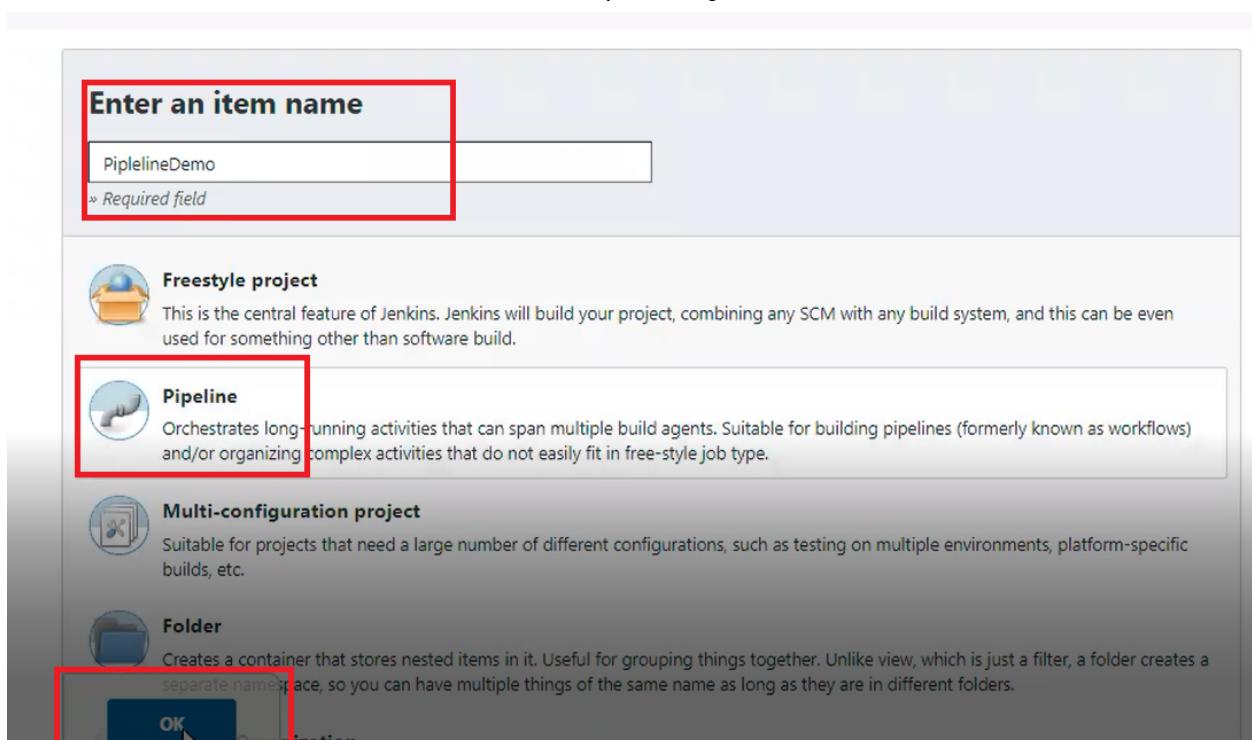


Plugin	Description	Version	Action
Email Notifier	This plugin allows you to configure email notifications for build results	1.33	Uninstall
Pipeline	A suite of plugins that lets you orchestrate automation, simple or complex. See Pipeline as Code with Jenkins for more details.	2.6	Uninstall
Pipeline Graph Analysis Plugin			

- Now, go to the **Dashboard** and select the **New Item** option.



6. Now, enter the project name, and select **Pipeline project** this time.
7. After that, click on **OK** to create the Pipeline job.



Enter an item name

PipelineDemo
* 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.

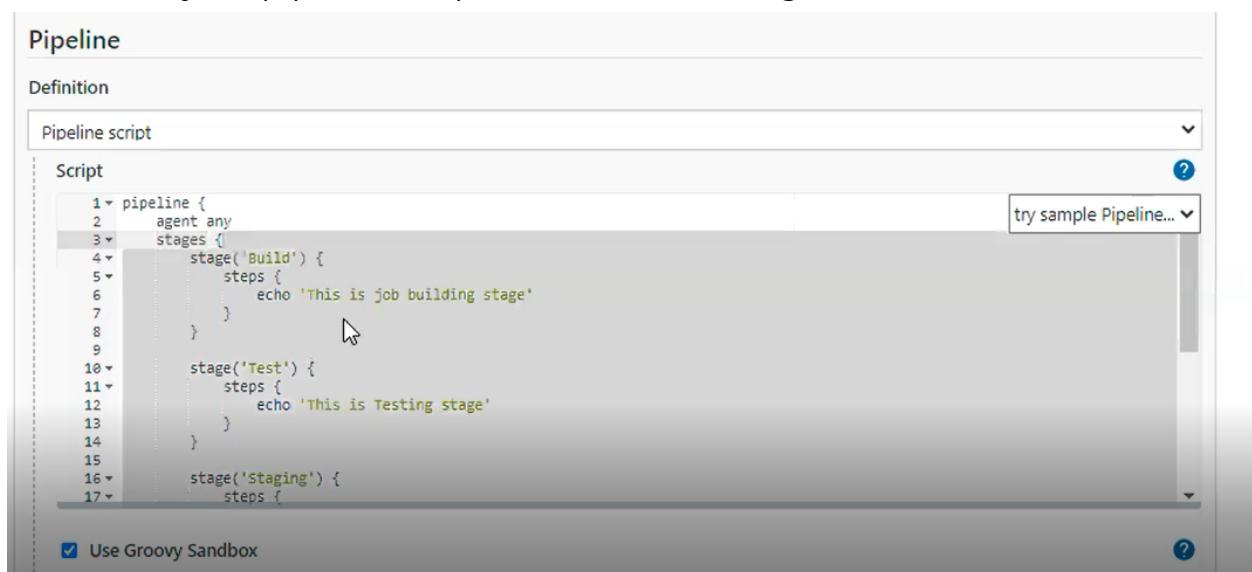
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
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.

OK

8. After that, you will be able to see the **Pipeline** option.
9. Then, in the **Pipeline** definition, select **Pipeline Script** as shown in the figure below.

10. Then write your pipeline script as shown in the figure below.



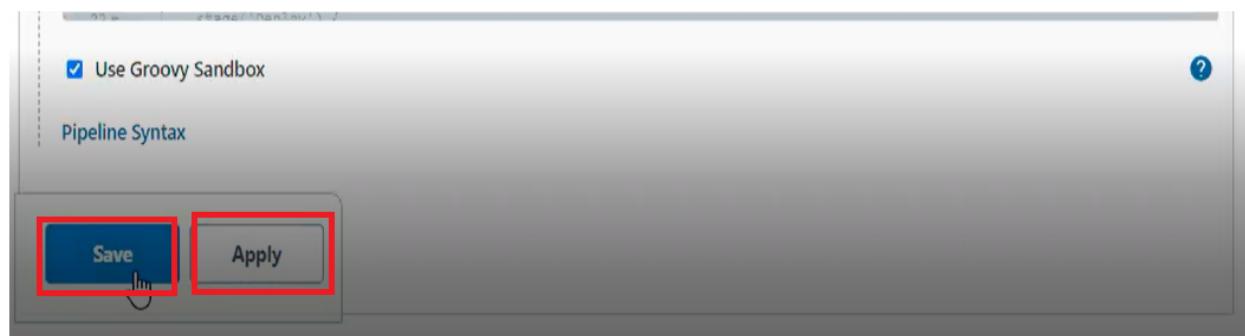
```

Pipeline
Definition
Pipeline script
try sample Pipeline... ▾
Script
1> pipeline {
2>   agent any
3>   stages {
4>     stage('Build') {
5>       steps {
6>         echo 'This is job building stage'
7>       }
8>     }
9>
10>    stage('Test') {
11>      steps {
12>        echo 'This is Testing stage'
13>      }
14>    }
15>
16>    stage('Staging') {
17>      steps {

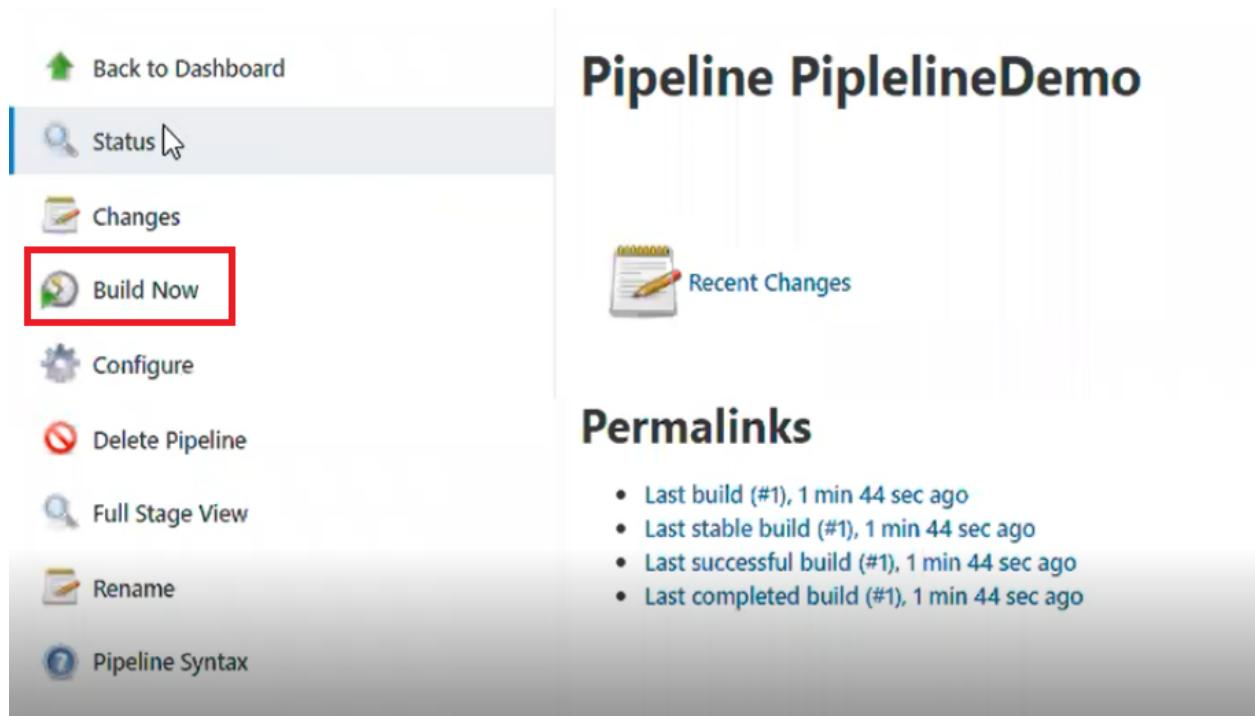
```

Use Groovy Sandbox

11. Click on **Apply** and then **Save**.



12. Then, click on **Build Now**.



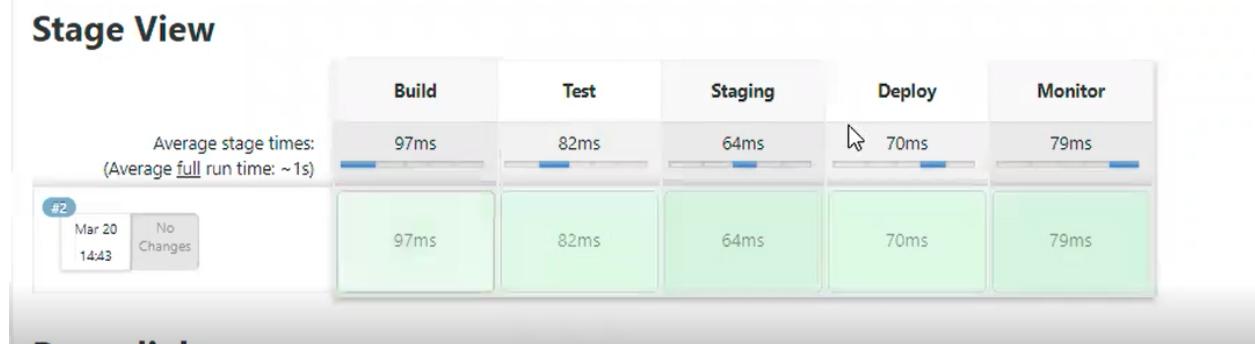
Pipeline PipelineDemo

Recent Changes

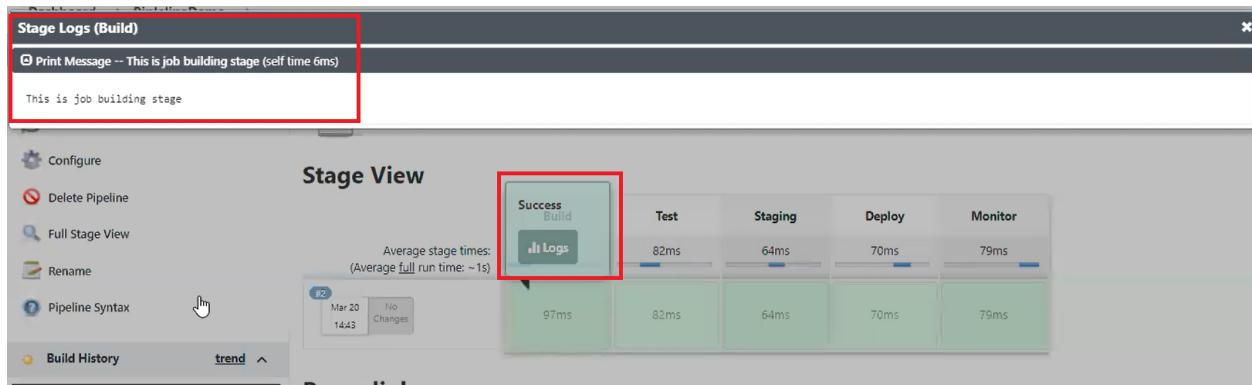
- Last build (#1), 1 min 44 sec ago
- Last stable build (#1), 1 min 44 sec ago
- Last successful build (#1), 1 min 44 sec ago
- Last completed build (#1), 1 min 44 sec ago

Permalinks

13. When the pipeline runs, you can see all the stages in the **Stage View** of the pipeline as shown in the figure below.



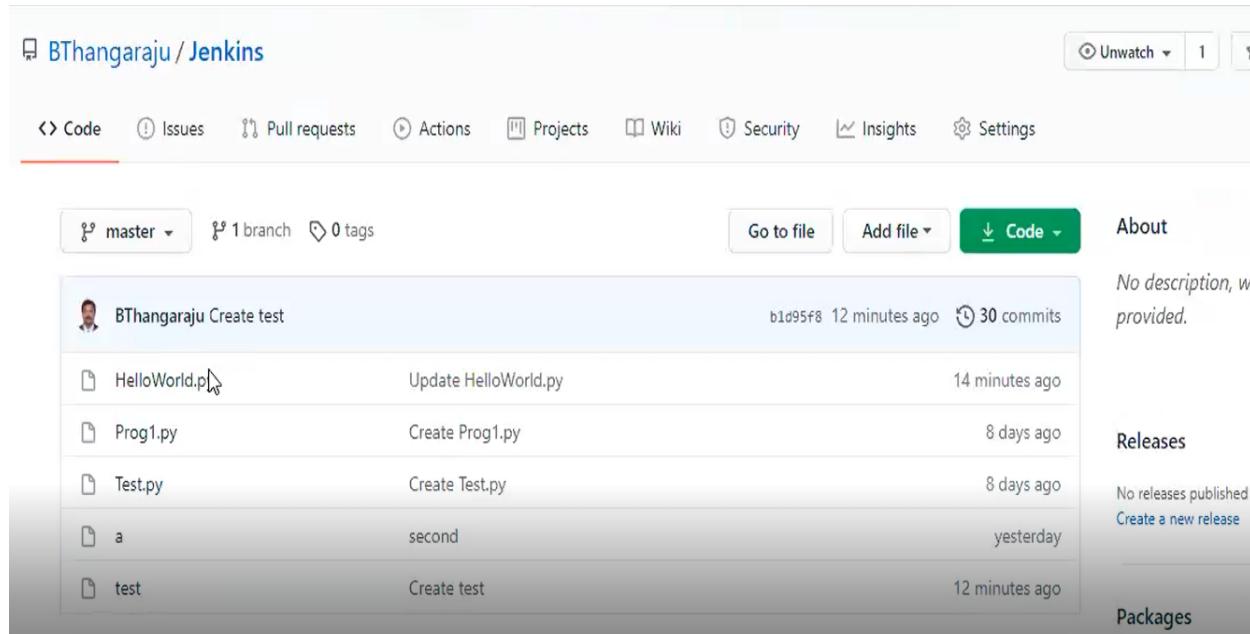
14. You can also see the logs of any particular stage as shown.



15. Now, you need to create the main pipeline

Using Template to Create Main Pipeline:

Till now, you have looked at the basic structure of a pipeline. But, let's use it to accomplish some real-world tasks. So, you are going to work with two main files: **Prog1.py** and **Test.py**. First, **Prog1.py** needs to be executed and then tested with **Test.py**. So, let's see the steps involved in it.



The screenshot shows a GitHub repository page for 'BThangaraju / Jenkins'. The 'Code' tab is selected. It displays a list of recent commits:

- BThangaraju Create test (b1d95f8, 12 minutes ago)
- HelloWorld.py (Update HelloWorld.py, 14 minutes ago)
- Prog1.py (Create Prog1.py, 8 days ago)
- Test.py (Create Test.py, 8 days ago)
- a (second, yesterday)
- test (Create test, 12 minutes ago)

On the right side, there are sections for 'About' (No description, we provided.), 'Releases' (No releases published, Create a new release), and 'Packages'.

1. Go to the **Configure** section, and in the Pipeline section select **Pipeline script** in the **Definition**.
2. Write your modified pipeline code as shown in the figure below.

General Build Triggers **Advanced Project Options** Pipeline Advanced...

Pipeline

Definition

Pipeline script

```

 1 pipeline {
 2   agent any
 3   stages {
 4     stage('Clone Git') {
 5       steps {
 6         git 'https://github.com/BThangaraju/Jenkins.git'
 7       }
 8     }
 9     stage('Build Code') {
10       steps {
11         sh "chmod u+x Prog1.py"
12         sh "./Prog1.py"
13       }
14     }
15     stage('Test Code') {
16       steps {
17         sh "chmod u+x Test.py"
18       }
19     }
20   }
21 }
22 }
```

Use Groovy Sandbox

try sample Pipeline... ?

Pipeline Syntax

- Once you finish writing your pipeline, click **Apply** and then **Save**, as shown in the figure below.

```

 7   }
 8 }
 9
10 stage('Build Code') {
11   steps {
12     sh "chmod u+x Prog1.py"
13     sh "./Prog1.py"
14   }
15 stage('Test Code') {
16   steps {
17     sh "chmod u+x Test.py"
18     sh "./Test.py"
19   }
20 }
21 }
22 }
```

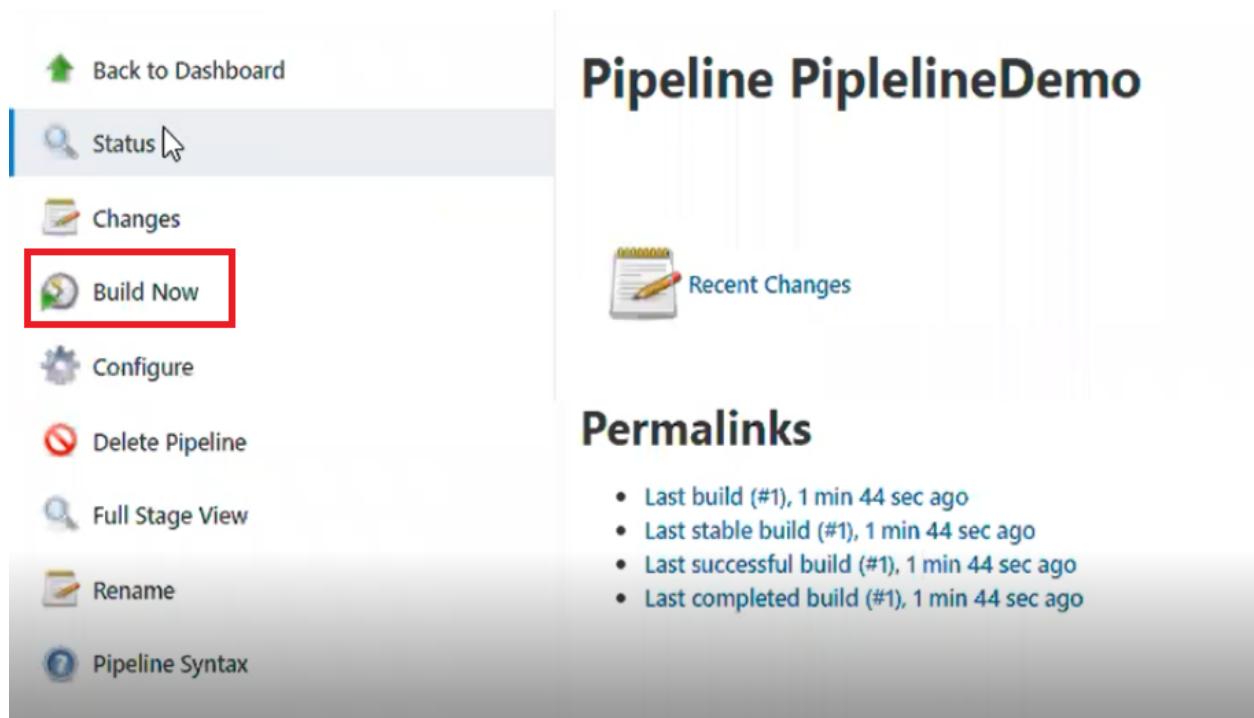
Use Groovy Sandbox

try sample Pipeline... ?

Pipeline Syntax

Save **Apply**

- Then, click on **Build Now**.



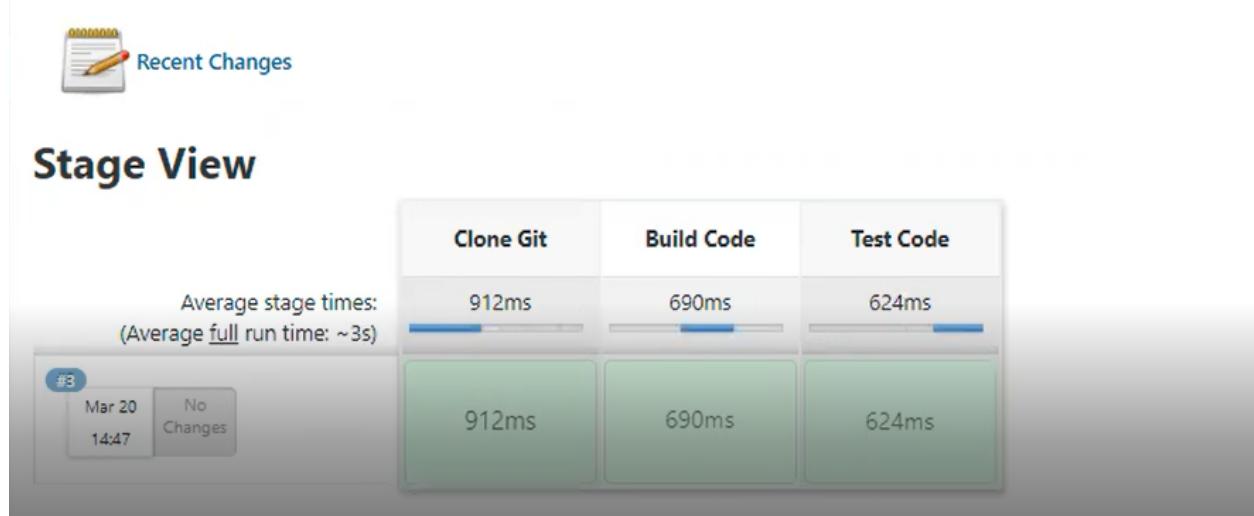
Pipeline PipelineDemo

Recent Changes

- Last build (#1), 1 min 44 sec ago
- Last stable build (#1), 1 min 44 sec ago
- Last successful build (#1), 1 min 44 sec ago
- Last completed build (#1), 1 min 44 sec ago

- After that, you can see the different stages of the pipeline occurring one after another.

Pipeline PipelineDemo



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

Clone Git	Build Code	Test Code
912ms	690ms	624ms

#3 Mar 20 14:47 No Changes

912ms 690ms 624ms

- After that, you can go and check the **Console Output**.

```

[Pipeline] {
[Pipeline] stage
[Pipeline] { (Clone Git)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/BThangaraju/Jenkins.git
> git init /var/lib/jenkins/workspace/PipelineDemo # timeout=10
Fetching upstream changes from https://github.com/BThangaraju/Jenkins.git
> git --version # timeout=10
> git --version # 'git version 2.17.1'
> git fetch --tags --progress -- https://github.com/BThangaraju/Jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/BThangaraju/Jenkins.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision b1d95f8d85483328be6200db8b833b54684b7307 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f b1d95f8d85483328be6200db8b833b54684b7307 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git checkout -b master b1d95f8d85483328be6200db8b833b54684b7307 # timeout=10
Commit message: "Create test"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build Code)
[Pipeline] sh
+ chmod u+x Prog1.py

```

7. In the **Console Output**, you can see that your program executed and passed all the test cases successfully.

```

> git branch -a -v --no-abbrev # timeout=10
> git checkout -b master b1d95f8d85483328be6200db8b833b54684b7307 # timeout=10
Commit message: "Create test"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build Code)
[Pipeline] sh
+ chmod u+x Prog1.py
[Pipeline] sh
+ ./Prog1.py
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test Code)
[Pipeline] sh
+ chmod u+x Test.py
[Pipeline] sh
+ ./Test.py
.

-----
Ran 1 test in 0.000s

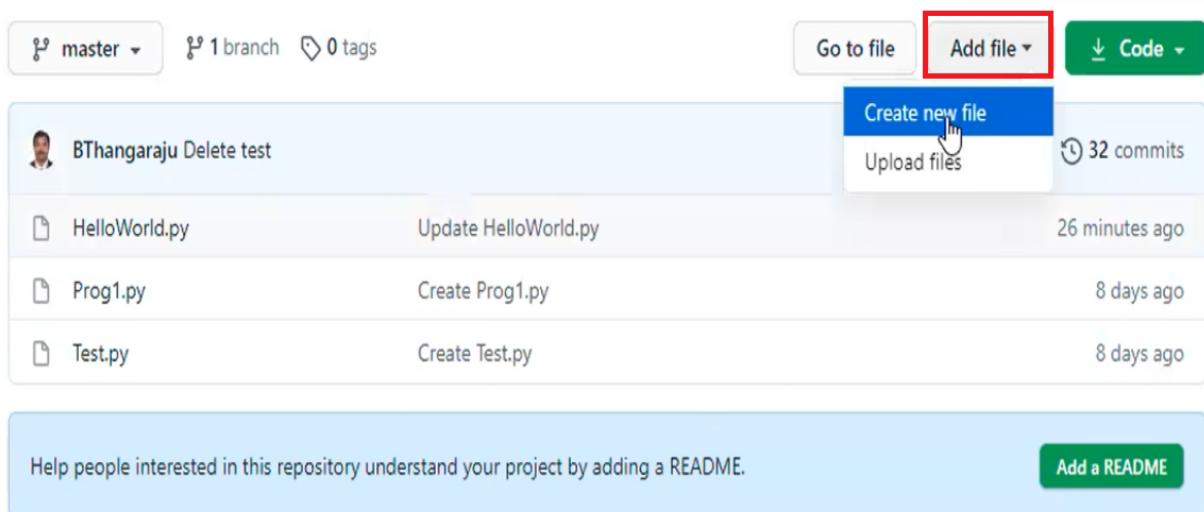
OK
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline

```

Checking the code into Jenkins file:

So, let's see how to create a Jenkins file, and check it out in GitHub. Here, you would only be using the project shown above. Let's see what are the various steps involved.

1. Sign in to your GitHub account and go to the **repository**.
2. After that, click on **Add file**.
3. Then, select **Create new file**.



4. Give a name to your file.
5. Then write the code into the Jenkins file.



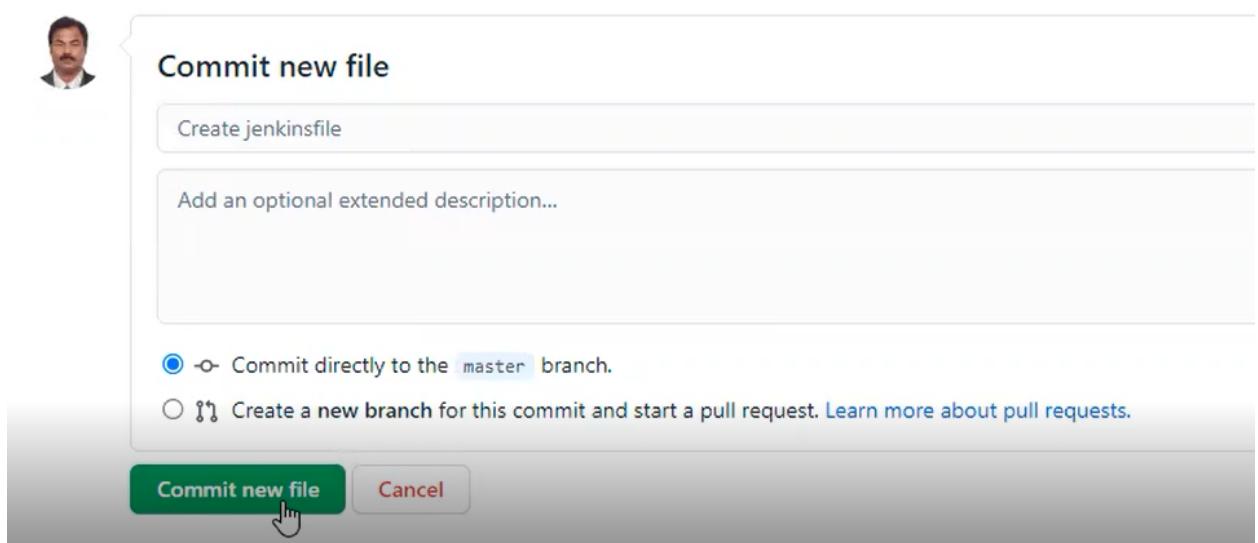
Jenkins / Jenkinsfile in master

<> Edit file Preview changes

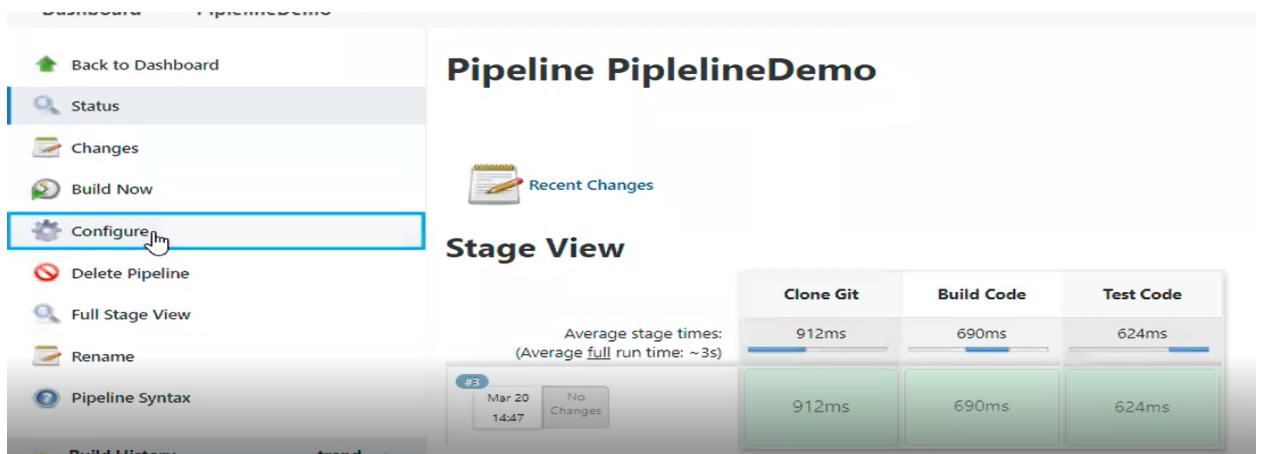
```

1 pipeline {
2   agent any
3   stages {
4     stage('Clone Git') {
5       steps {
6         git 'https://github.com/BThangaraju/Jenkins.git'
7       }
8     }
9     stage('Build Code') {
10    steps {
11      sh "chmod u+x Prog1.py"
12      sh "./Prog1.py"
13    }
14  }
15  stage('Test Code') {
16    steps {
17      sh "chmod u+x Test.py"
18      sh "./Test.py"
19    }
20  }
21 }
22 }
23
  
```

- After that, scroll down and click on the **Commit new file** option.

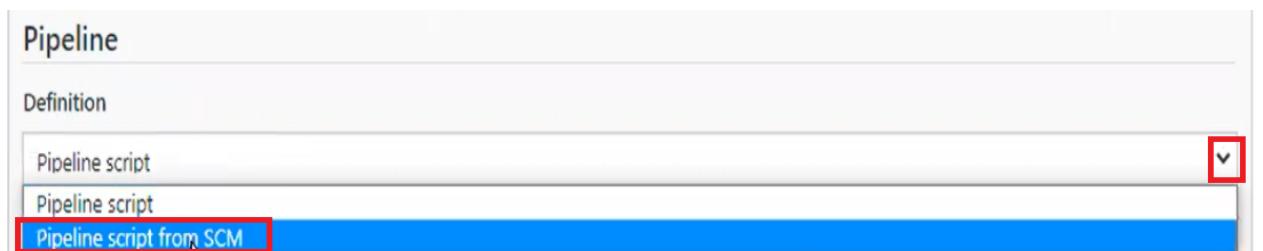


- After that, return to the Jenkins pipeline job that you created and click on the **Configure** section as shown.



The screenshot shows the Jenkins Pipeline PipelineDemo stage view. On the left, there's a sidebar with options like Back to Dashboard, Status, Changes, Build Now, Configure (which is highlighted with a blue border), Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area displays the stage view with three stages: Clone Git (912ms), Build Code (690ms), and Test Code (624ms). A summary bar at the top indicates average stage times of 912ms, 690ms, and 624ms respectively, with a total run time of ~3s. A build history card for #3 is shown, indicating Mar 20 14:47 and No Changes.

8. In the **Pipeline** section, click on the **Definition**.
9. After that, select the **Pipeline script from SCM** option.



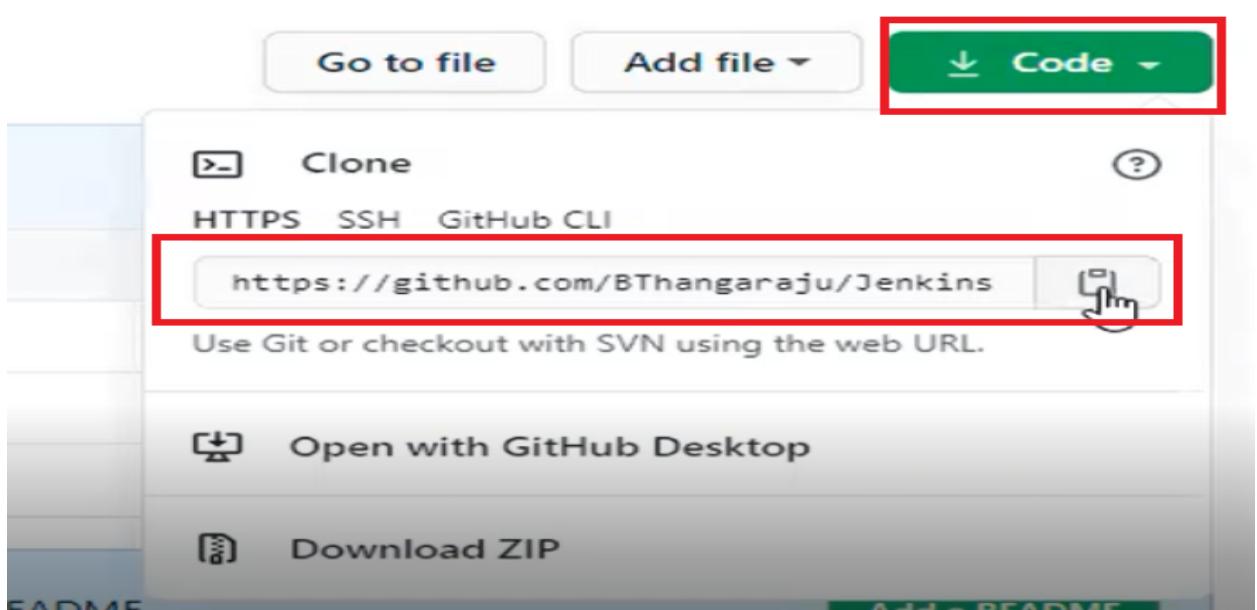
The screenshot shows the Pipeline definition screen. Under the Definition section, there is a dropdown menu with three options: Pipeline script, Pipeline script, and Pipeline script from SCM. The 'Pipeline script from SCM' option is highlighted with a red box and a blue selection bar.

10. Then in the **SCM** option select **Git**.

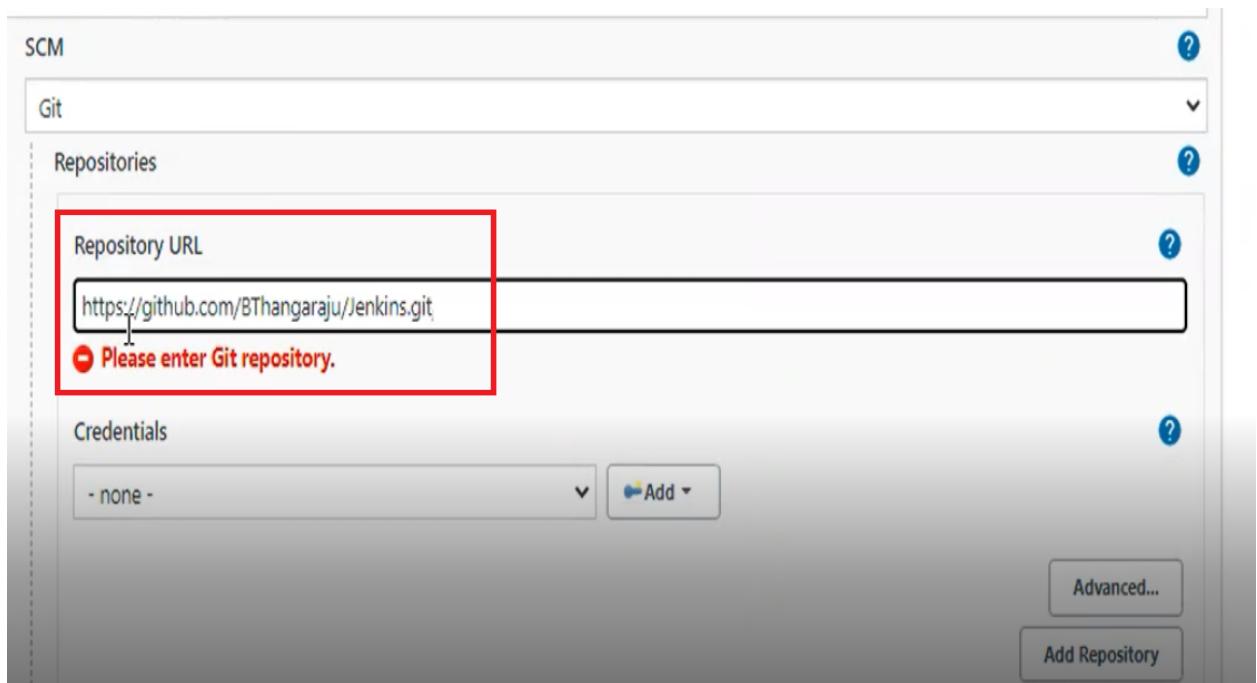


The screenshot shows the Pipeline definition screen again. Under the Definition section, the SCM dropdown menu is open, showing options: SCM, None, None, Git, Jenkinsfile, and Jenkinsfile. The 'Git' option is highlighted with a red box and a blue selection bar.

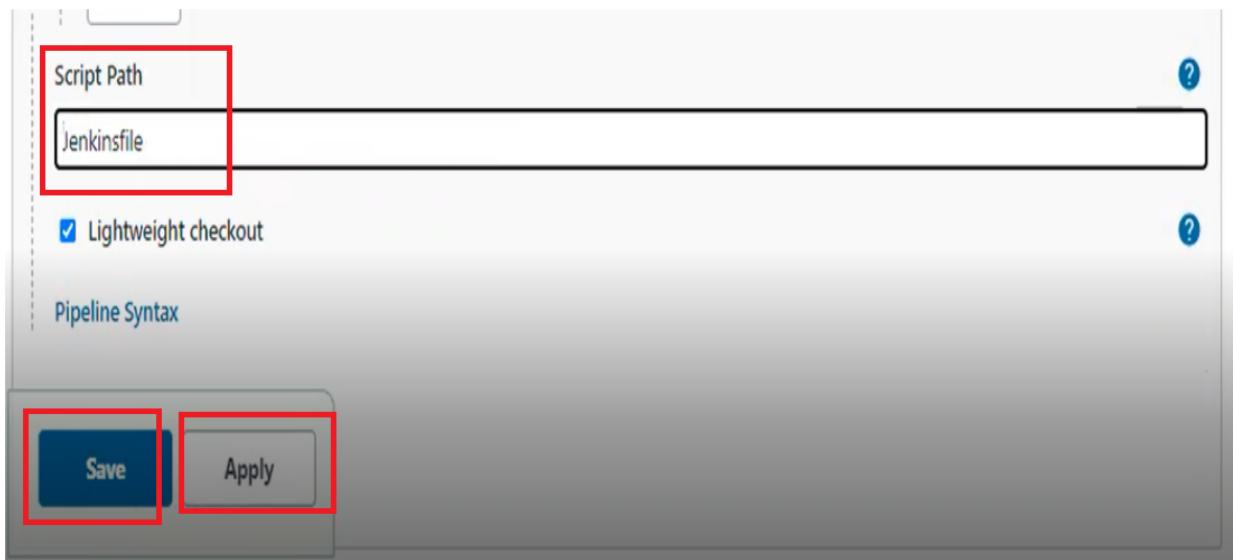
11. Then, go to your GitHub repository.
12. Then, click on **Code** and copy the URL as shown below.



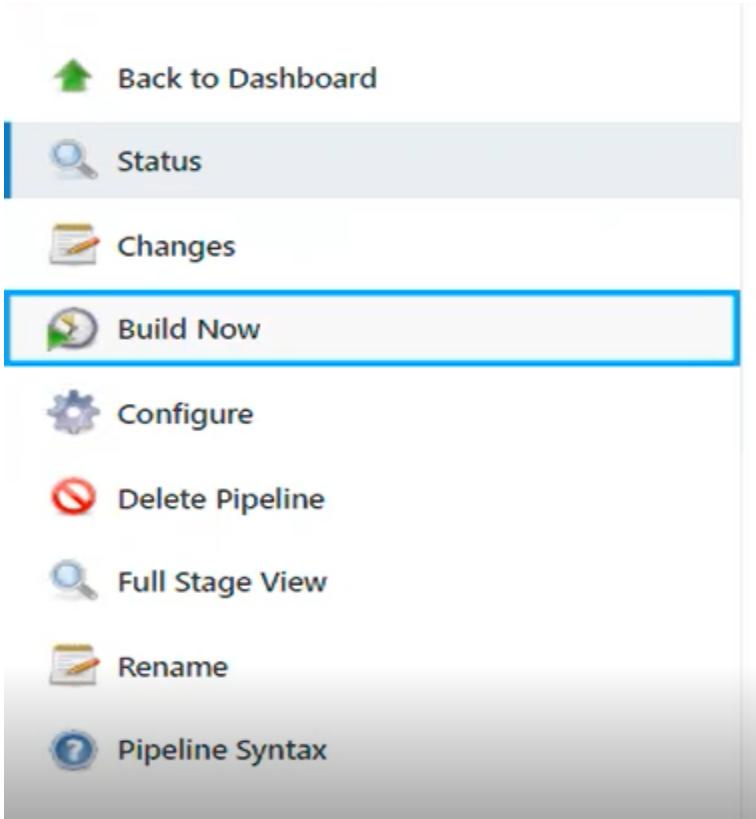
13. Paste this URL in the **Repository URL** section as shown below.



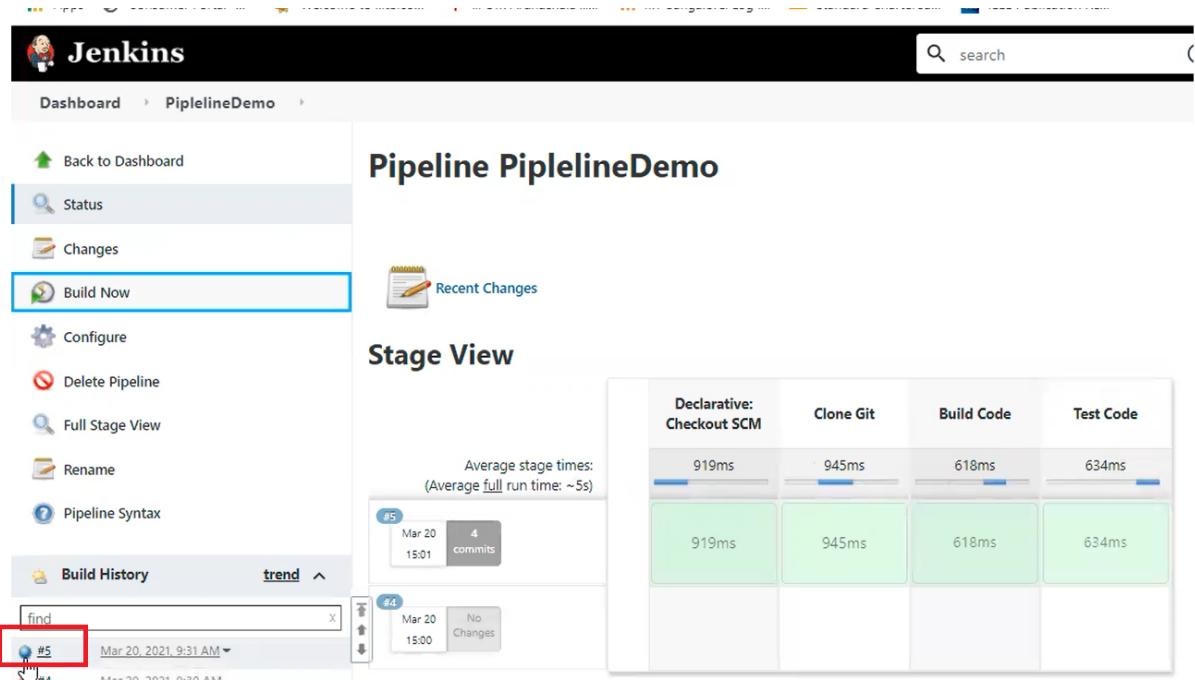
14. Then, scroll down and provide **Script Path** with respect to the GitHub repository.



15. Click on **Apply** and then **Save** buttons.
16. Then, click on the **Build Now** option.

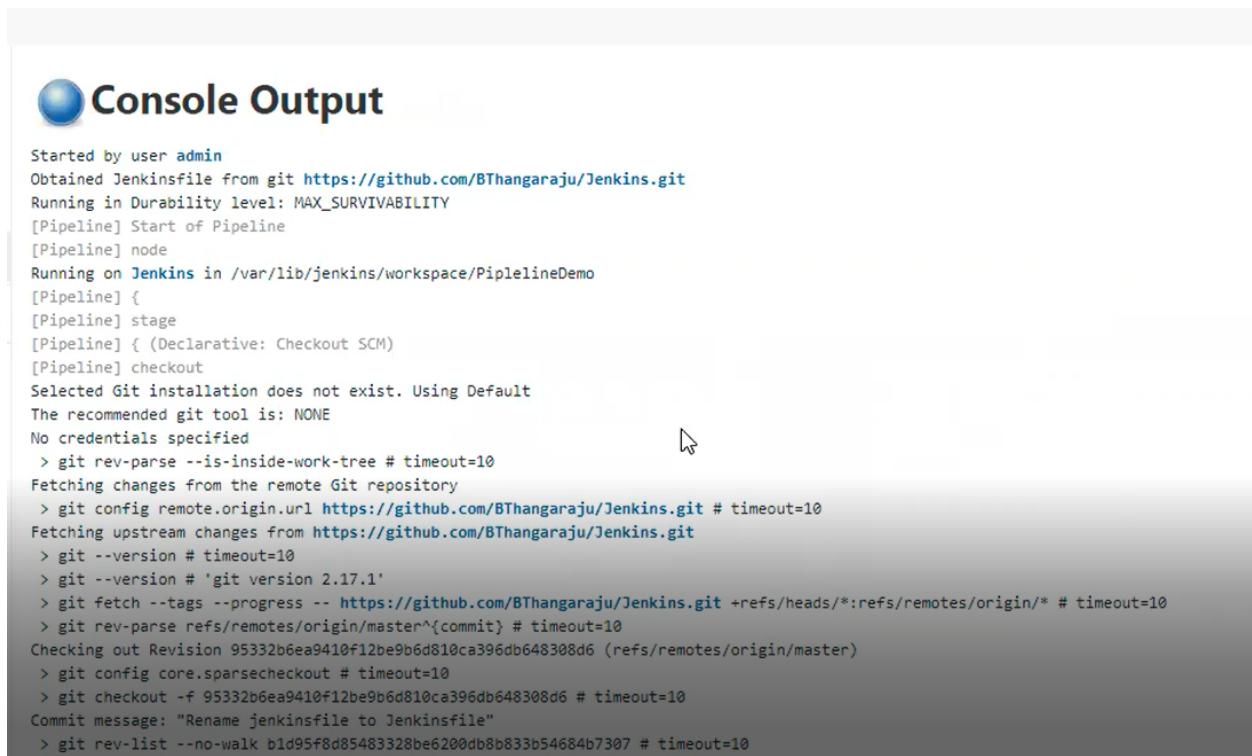


17. Then you can see that your project was successfully built, and the **Stage View** is as shown in the figure below.



The screenshot shows the Jenkins Pipeline Demo dashboard. On the left sidebar, under the 'PipelineDemo' section, the 'Build Now' button is highlighted with a blue border. The main area displays the 'Stage View' for the Pipeline. It includes a summary card with average stage times (919ms, 945ms, 618ms, 634ms) and four stages: Declarative: Checkout SCM, Clone Git, Build Code, and Test Code. Below this is a timeline showing two builds: #5 (Mar 20, 15:01, 4 commits) and #4 (Mar 20, 15:00, No Changes). The 'Build History' section at the bottom shows build #5 with a red box around its number.

- After that, click on the build number in the **Build History** to see the **Console Output**.



The screenshot shows the Jenkins Console Output for build #5. The output log is as follows:

```

Started by user admin
Obtained Jenkinsfile from git https://github.com/BThangaraju/Jenkins.git
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/PipelineDemo
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/BThangaraju/Jenkins.git # timeout=10
Fetching upstream changes from https://github.com/BThangaraju/Jenkins.git
> git --version # timeout=10
> git --version # 'git version 2.17.1'
> git fetch --tags --progress -- https://github.com/BThangaraju/Jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 95332b6ea9410f12be9b6d810ca396db648308d6 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 95332b6ea9410f12be9b6d810ca396db648308d6 # timeout=10
Commit message: "Rename jenkinsfile to Jenkinsfile"
> git rev-list --no-walk b1d95f8d85483328be6200db8b833b54684b7307 # timeout=10
  
```