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:

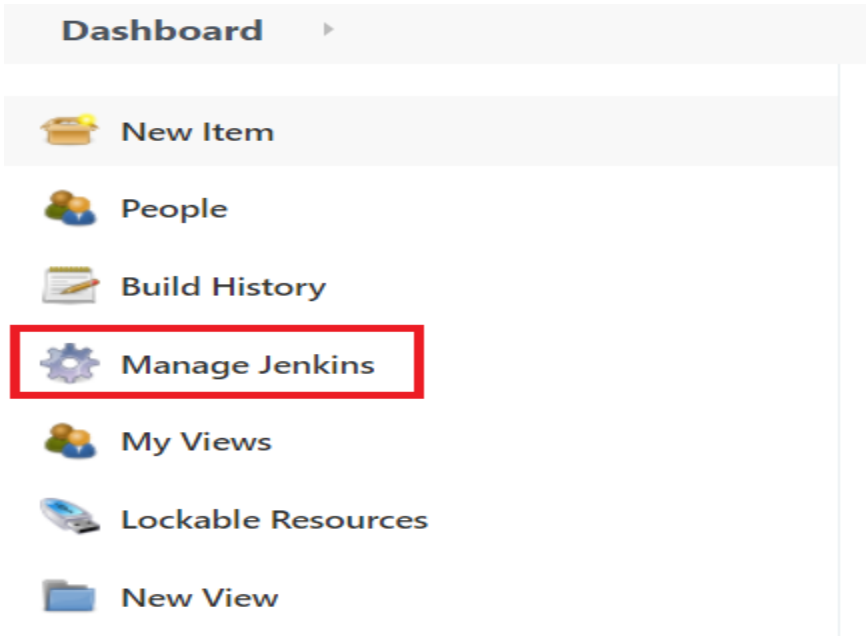
1. Creating the Pipeline Template
2. Using Template to create the main pipeline
3. Checking code into Jenkins file

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

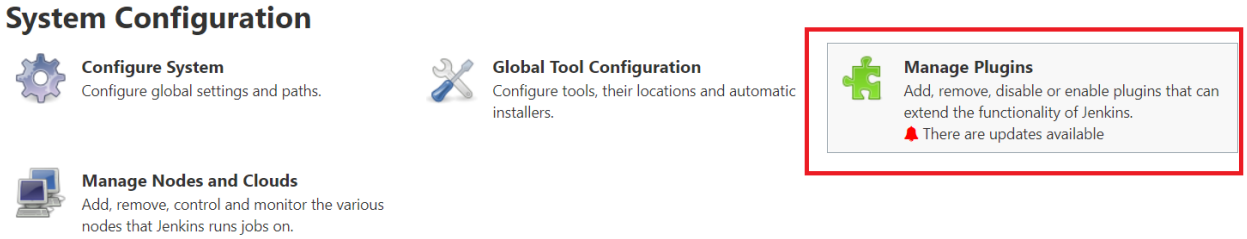
Creating the Pipeline Template:

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

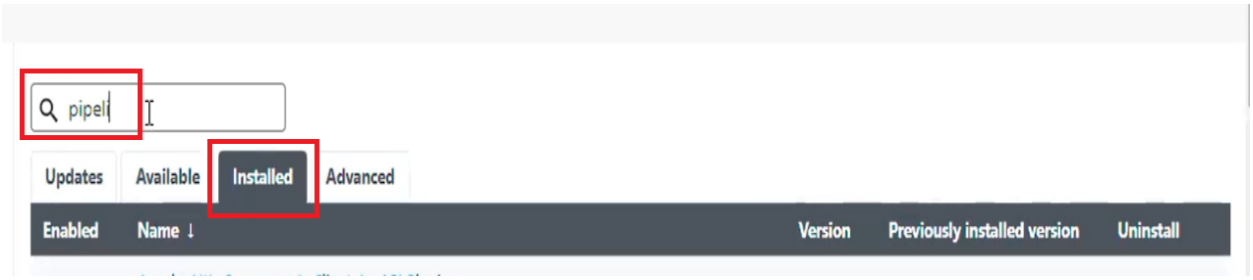
1. In the **Dashboard**, go to **Manage Jenkins**.



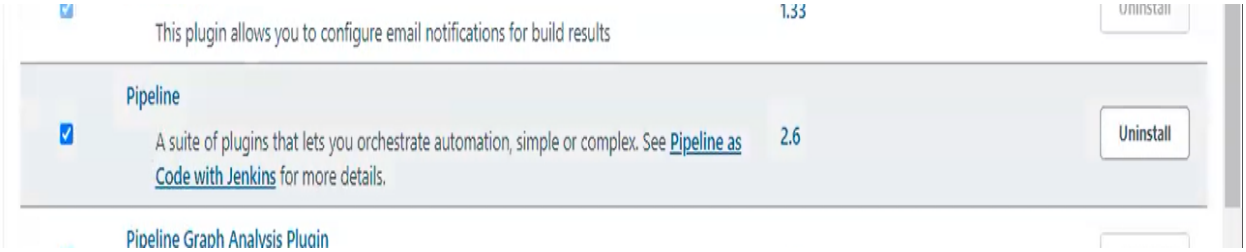
2. Then click on **Manage Plugins**.



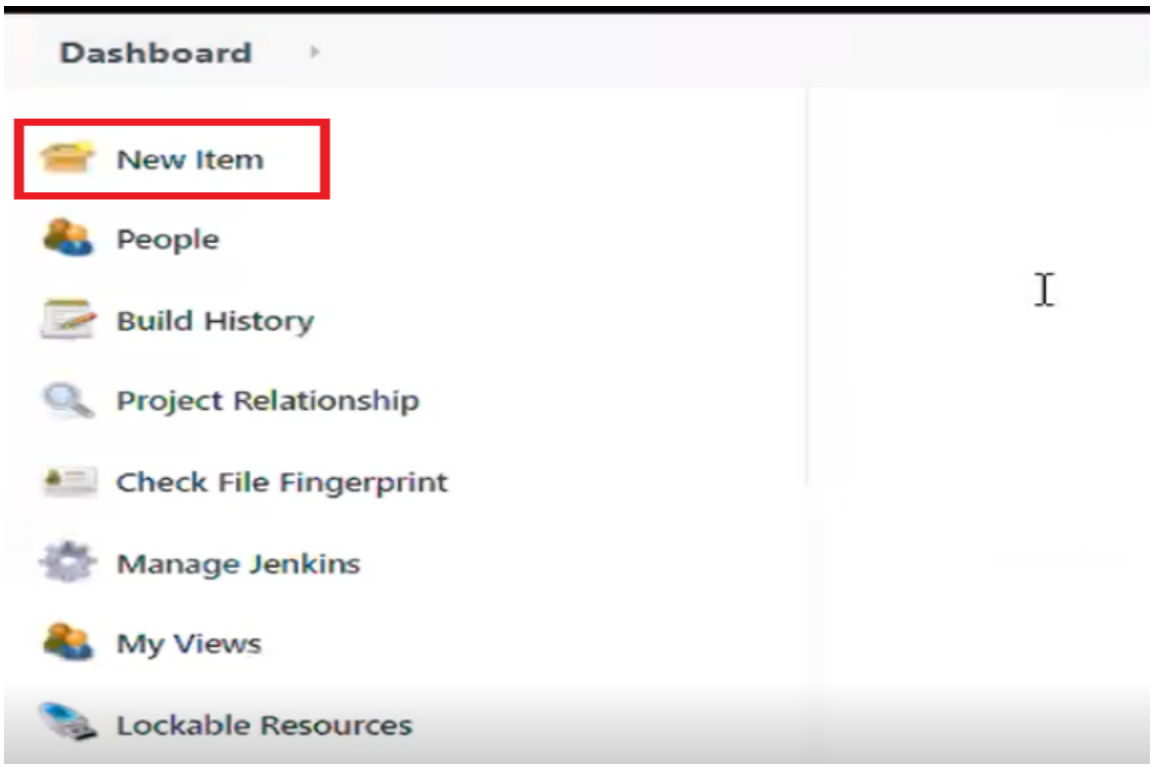
3. Then, in the **Installed tab** search for **Pipeline** plugin, i.e., to check whether it is already installed or not.



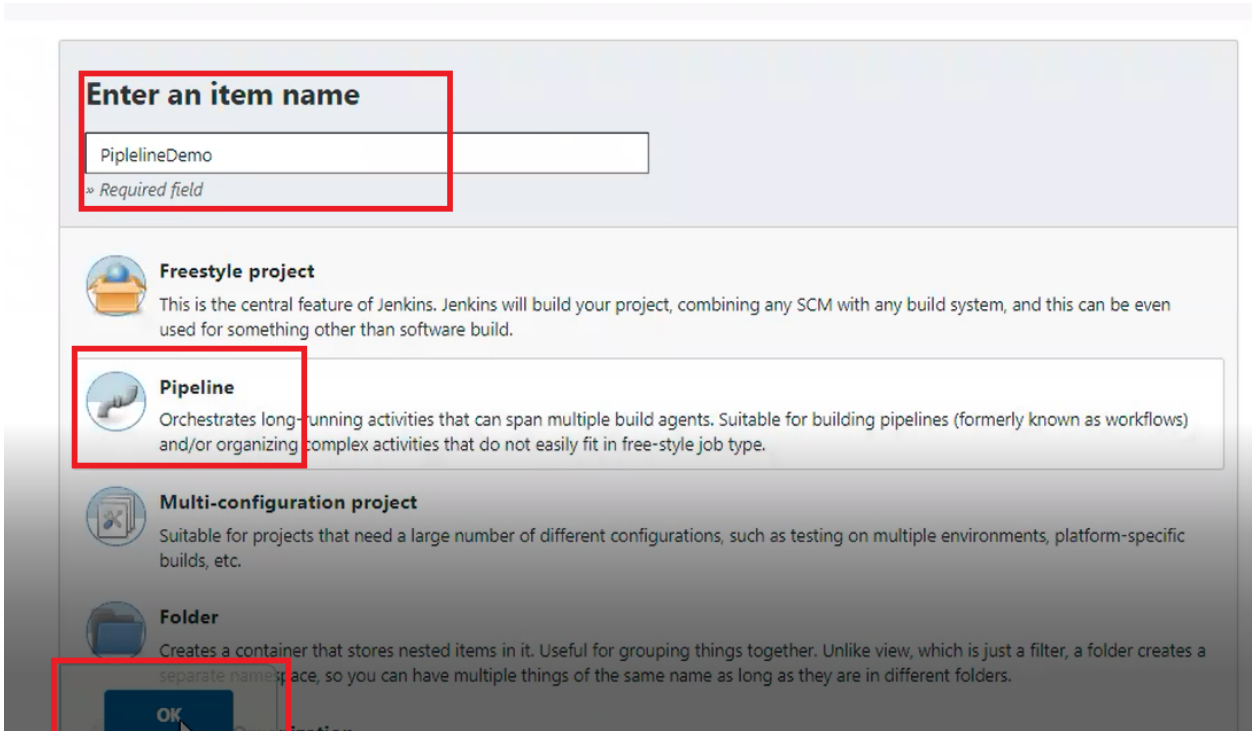
4. So, scroll down and here it can be seen that the **Pipeline** plugin is already installed. If it is not installed, then install it by going to the **Available** tabs and then search for the plugin.



5. 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.

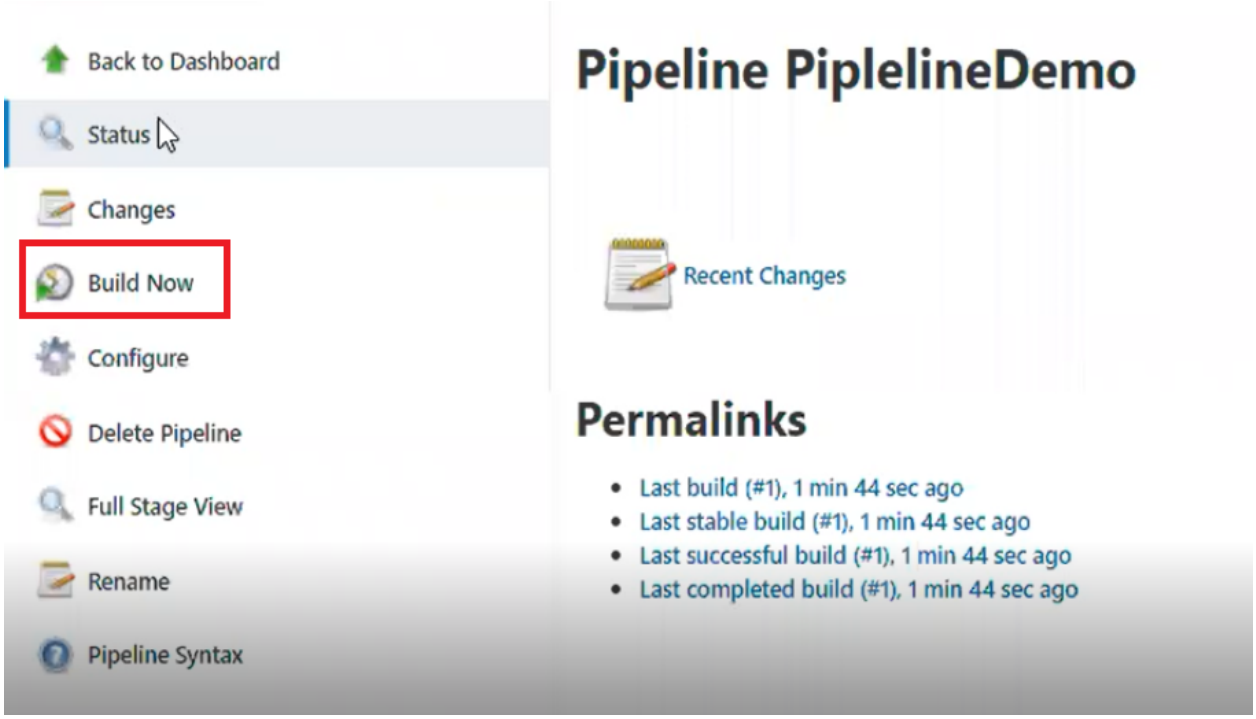


8. After that, you will be able to see the **Pipeline** option.
9. Then, in the **Pipeline** definition, select **Pipeline Script** as shown in the figure below.

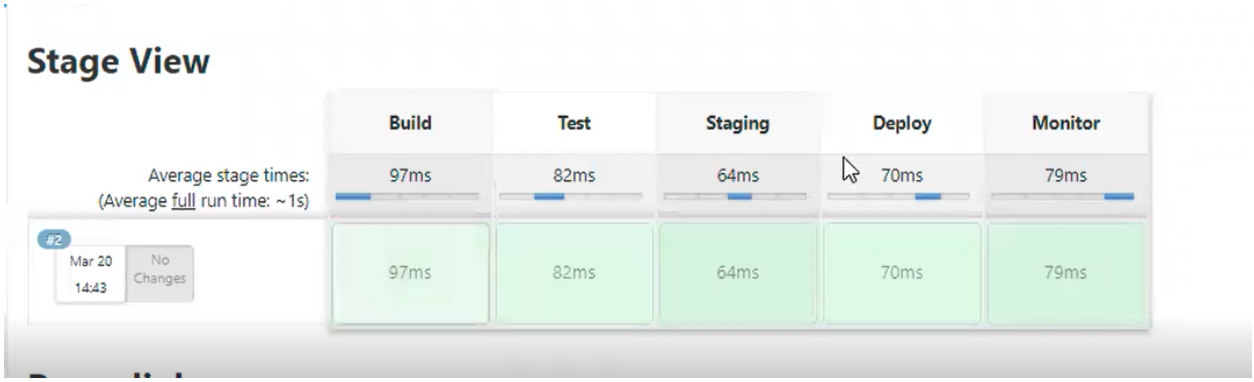
10. Then write your pipeline script as shown in the figure below.

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

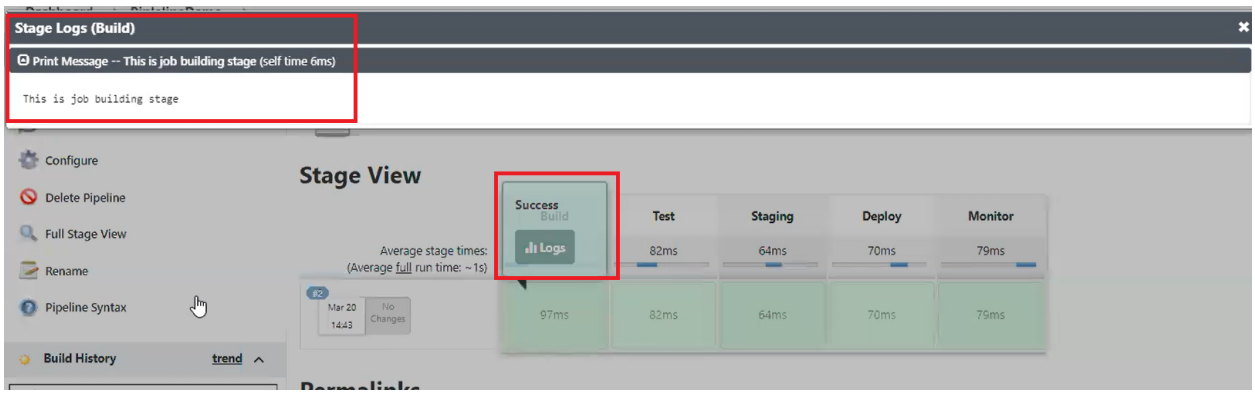
12. Then, click on **Build Now**.



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



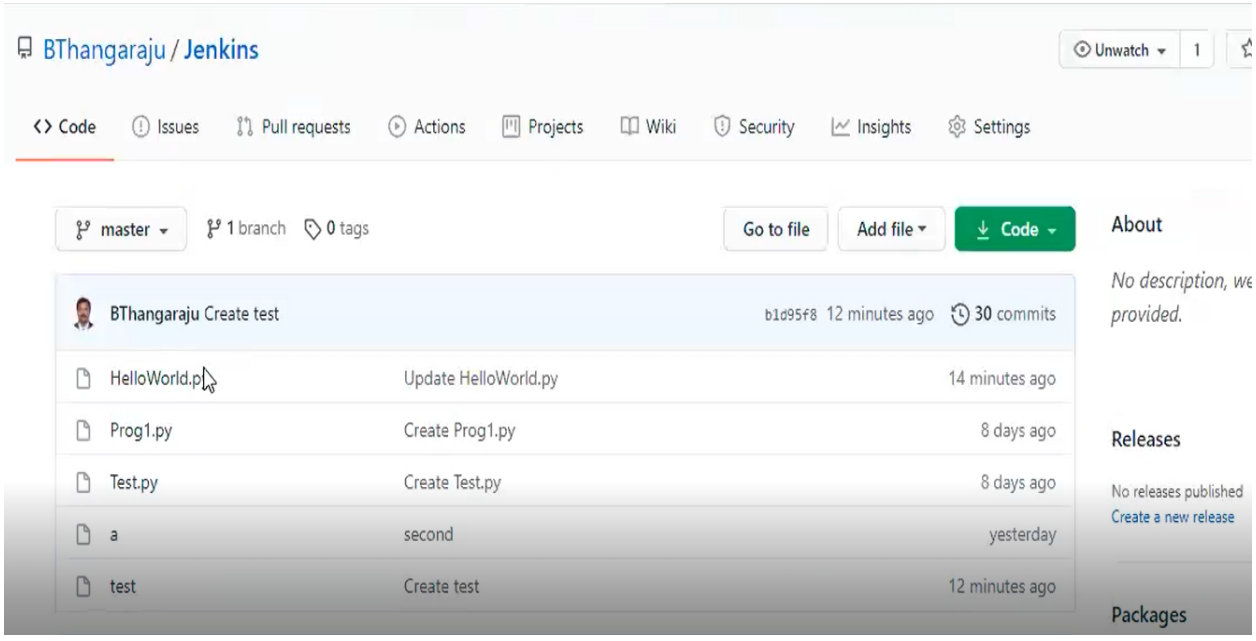
14. You can also see the logs of any particular stage as shown.



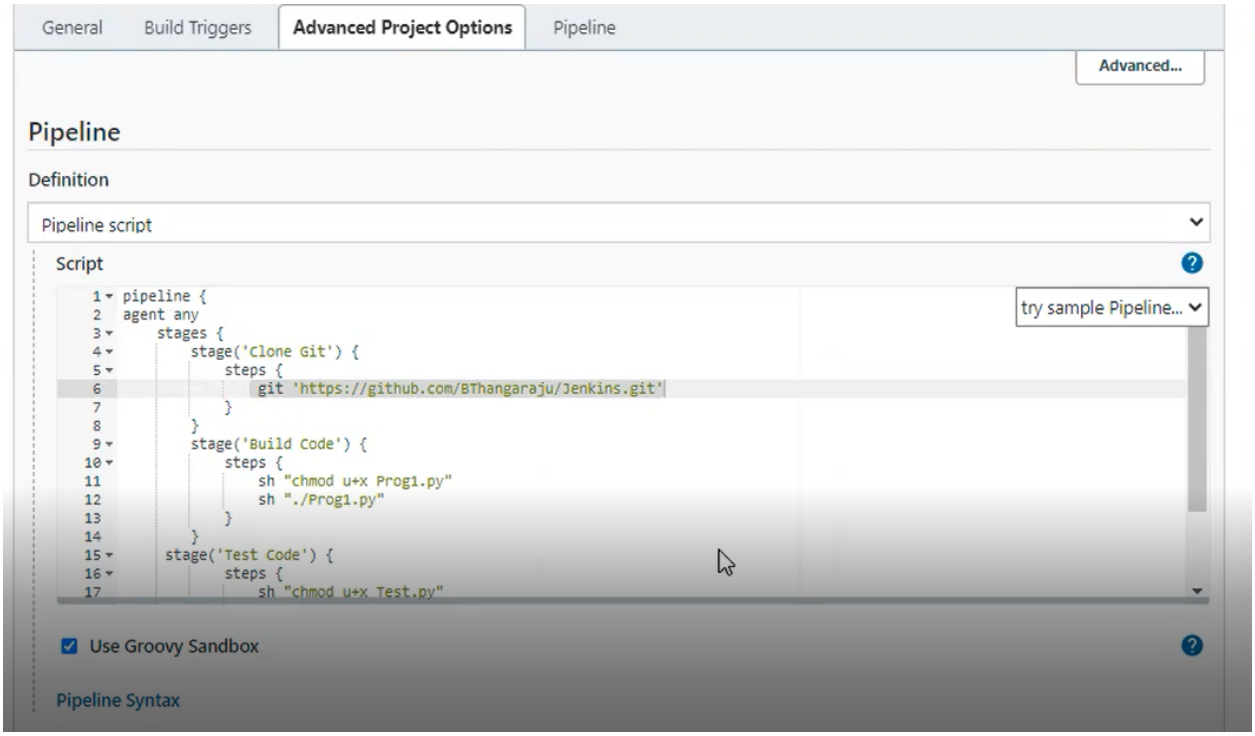
15. Now, you need to create the main pipeline

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.

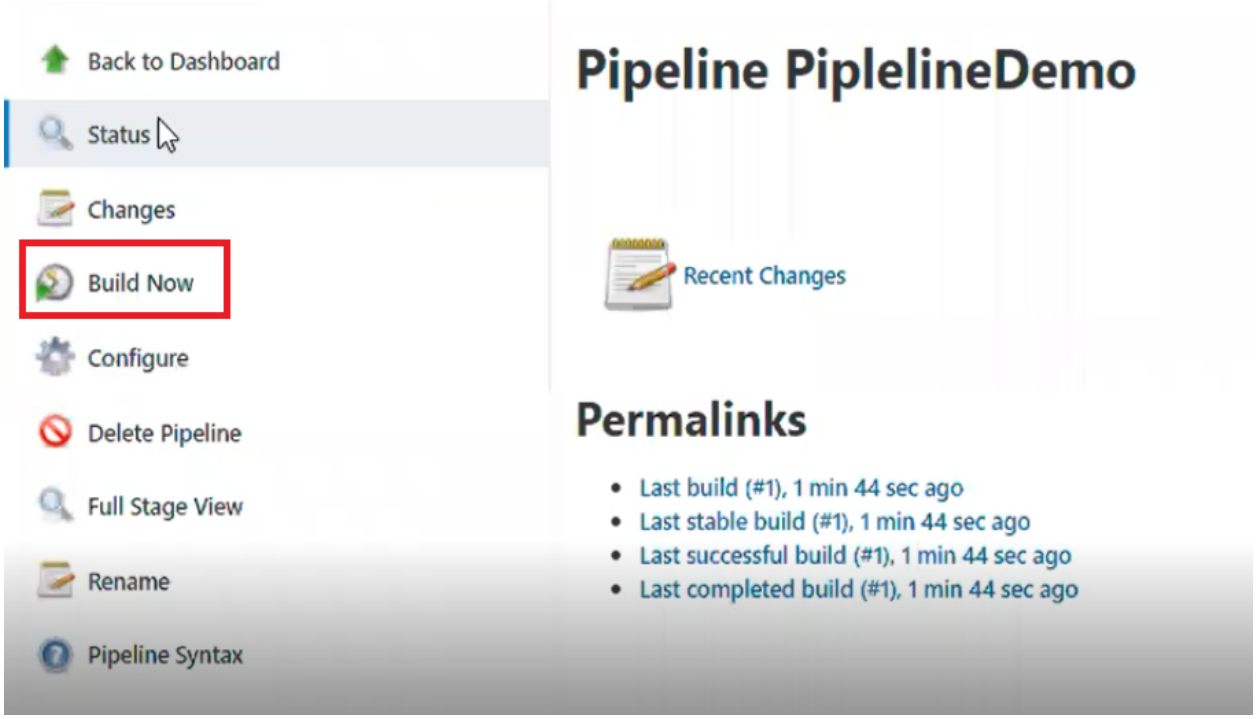


1. Go to the **Configure** section, and in the Pipeline section select **Pipeline script** in the **Definition**.
2. Write your modified pipeline code as shown in the figure below.

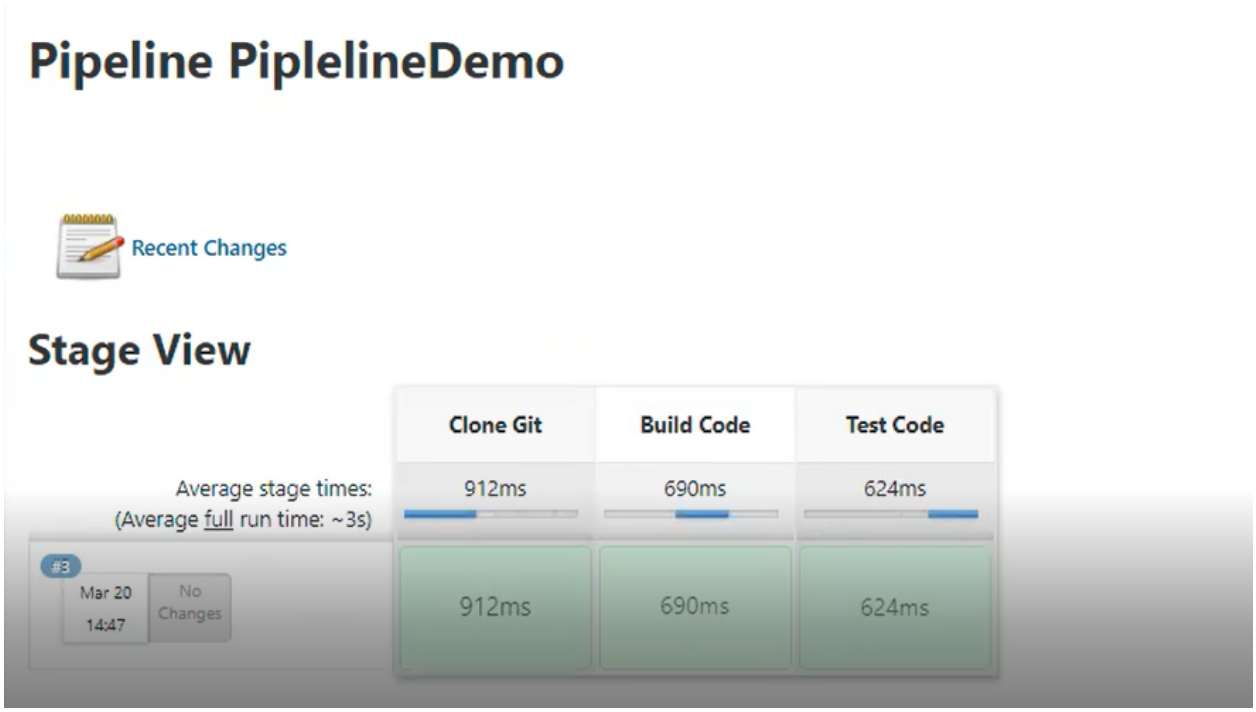


3. Once you finish writing your pipeline, click **Apply** and then **Save,** as shown in the figure below.

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



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



6. 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/PiplelineDemo # 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.

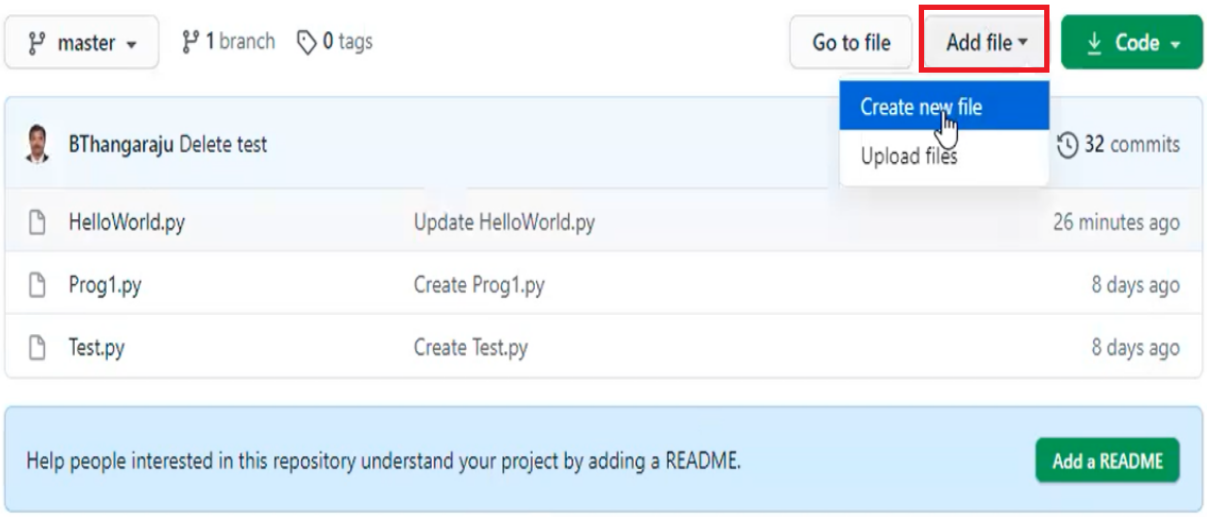
```
 > git branch -a -v --no-abbrev # timeout=10
 > git checkout -b master b1d95f8d85483328be6200db8b833b54684b7307 # timeout=10
Commit message: "Create test"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build Code)
[Pipeline] sh
+ chmod u+x Prog1.py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

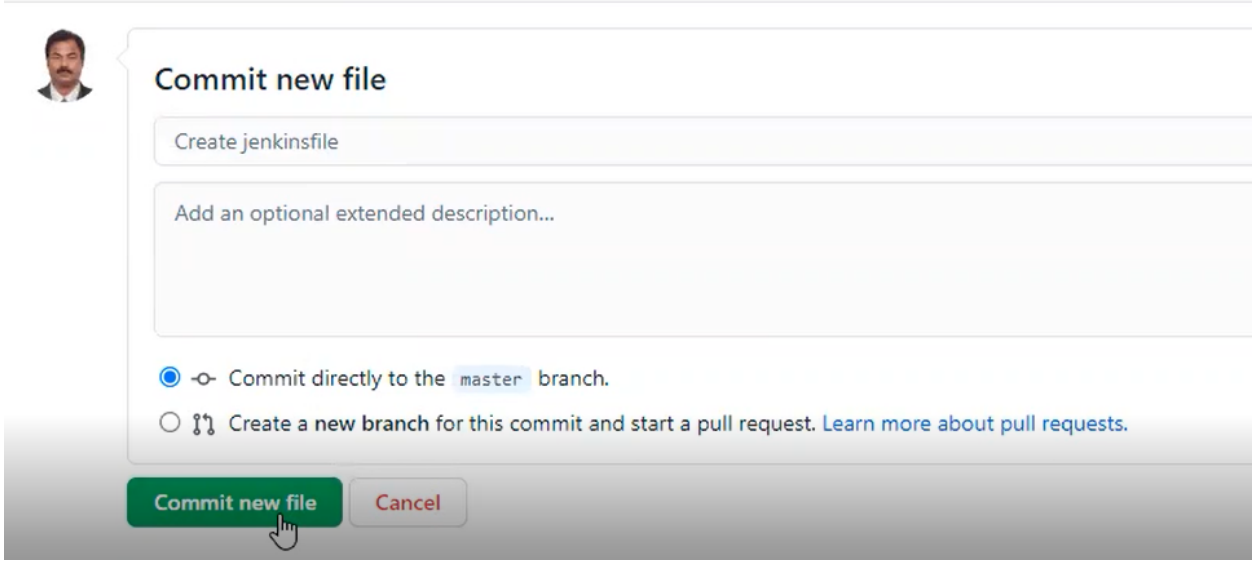
So, let's see how to create a Jenkins file, and check it out in GitHub. Here, you would only be using the project shown above. Let's see what are the various steps involved.

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**.

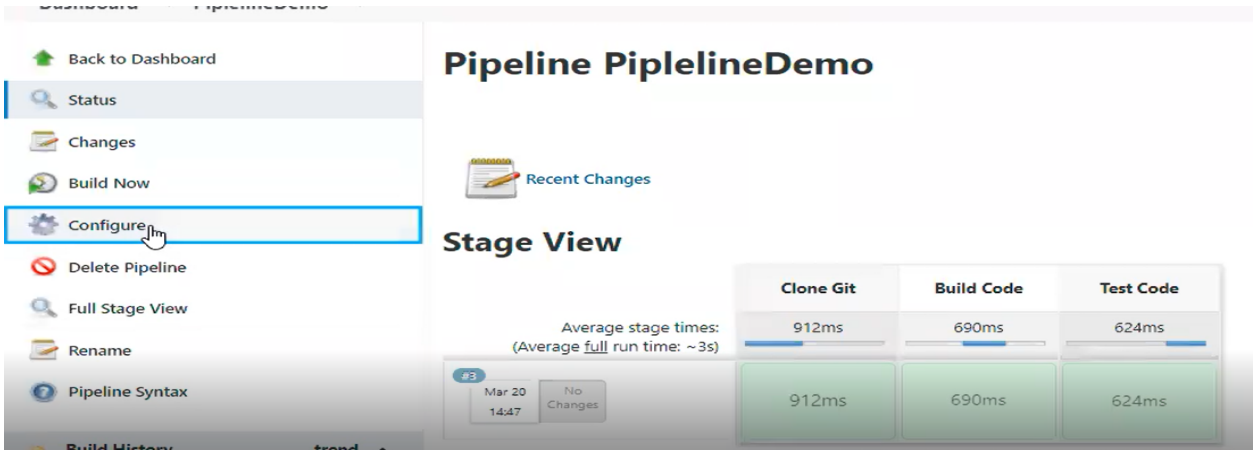


4. Give a name to your file.
5. Then write the code into the Jenkins file.

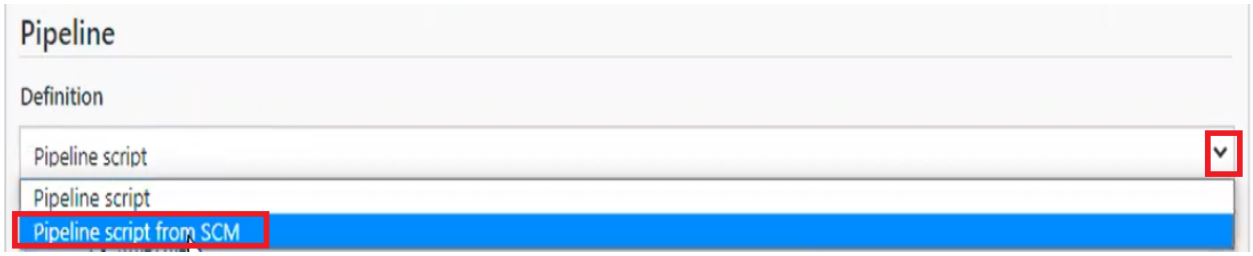
6. After that, scroll down and click on the **Commit new file** option.



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

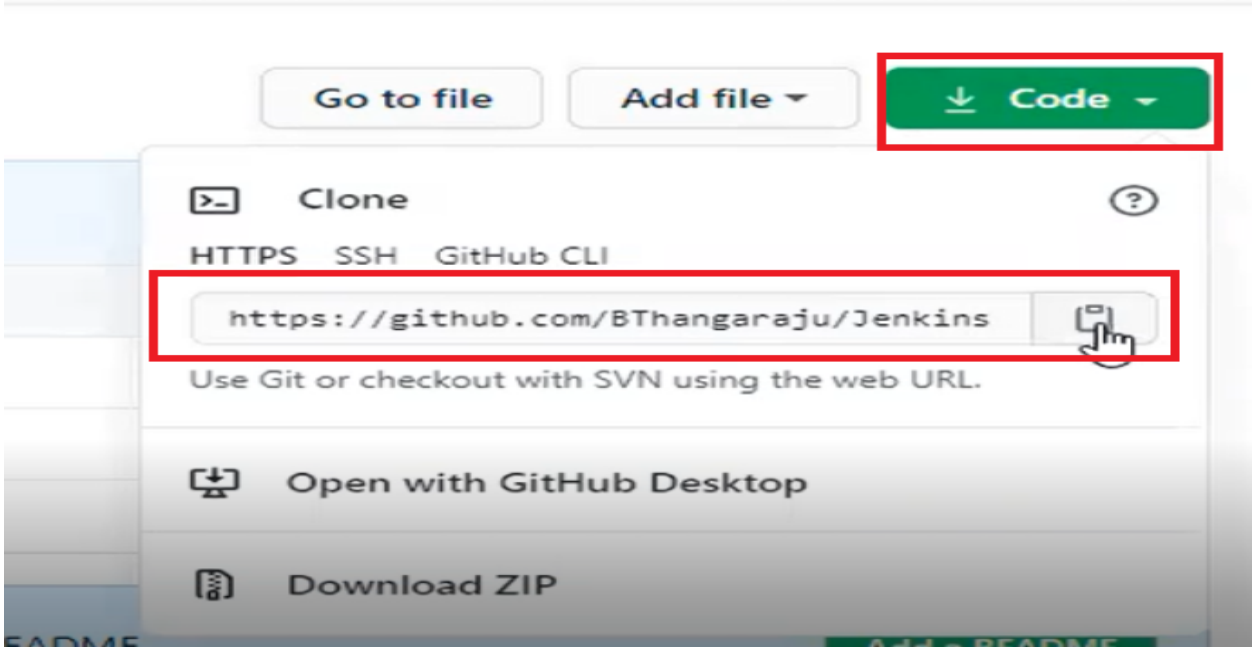


8. In the **Pipeline** section, click on the **Definition**.
9. After that, select the **Pipeline script from SCM** option.



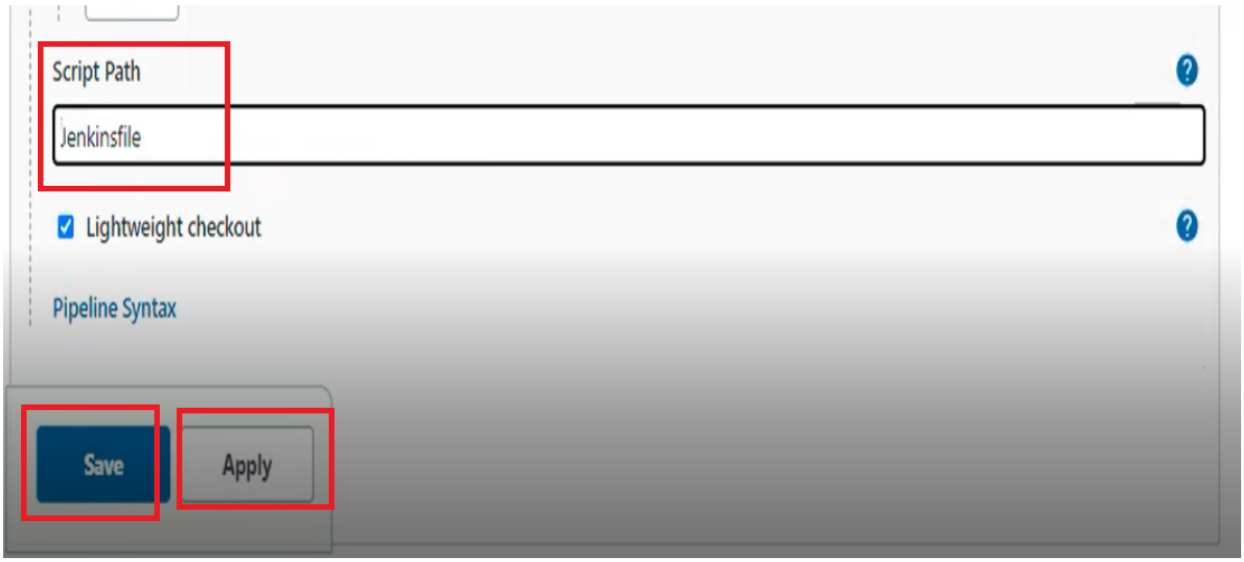
10. Then in the **SCM** option select **Git**.

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

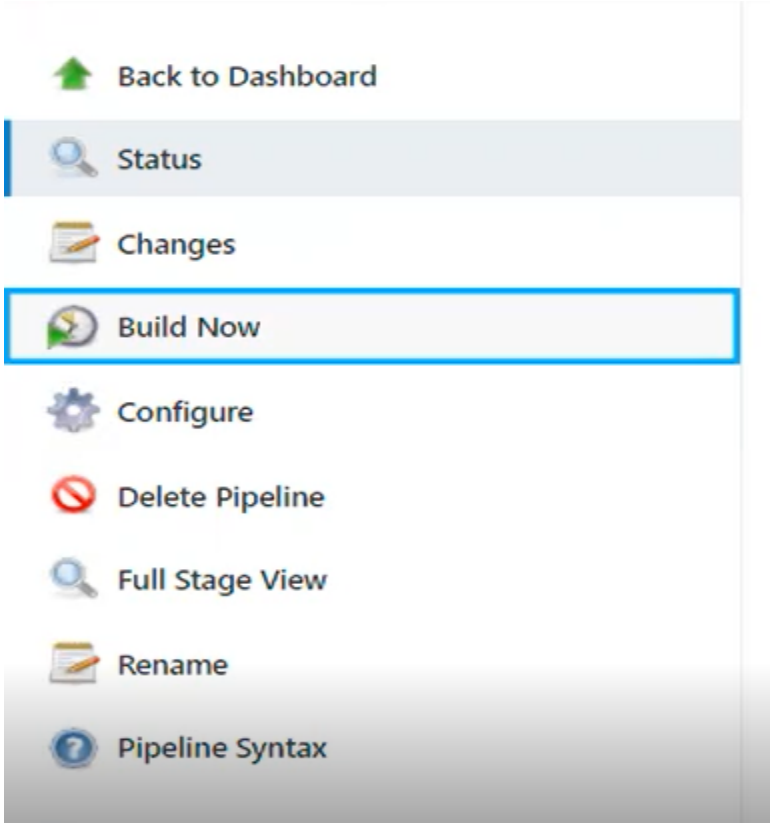


13. Paste this URL in the **Repository URL** section as shown below.

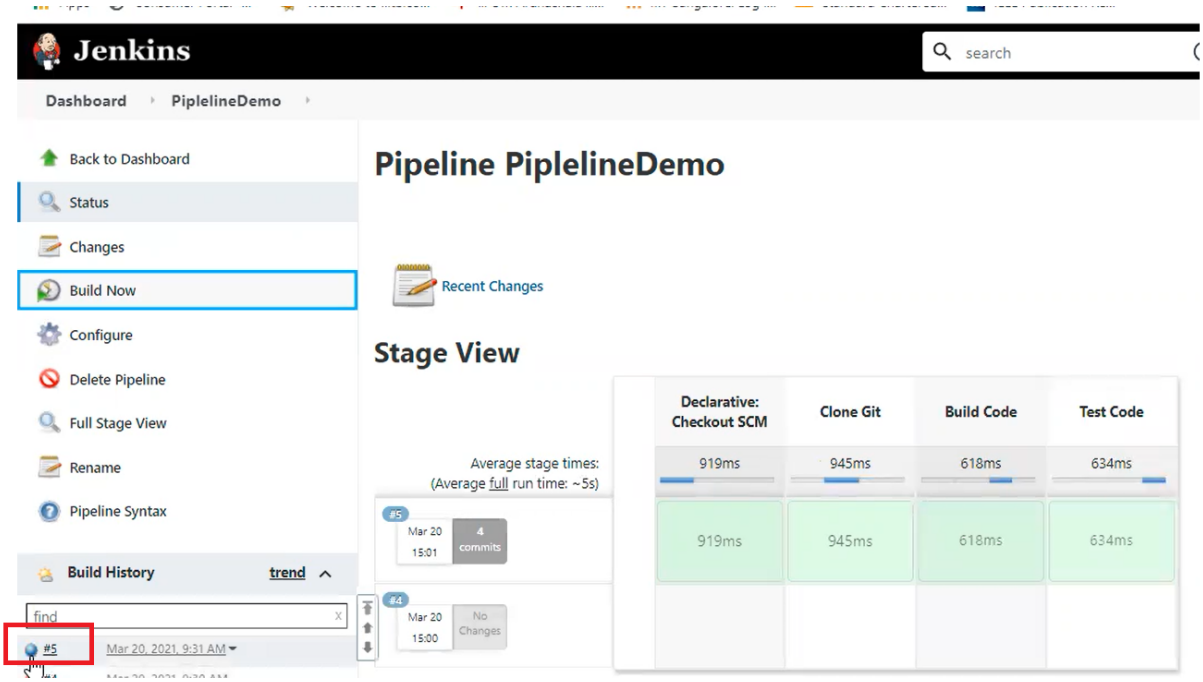
14. Then, scroll down and provide **Script Path** with respect to the GitHub repository.



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.



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