Practice for Lesson 4: Integrating Jenkins Pipelines with Jobs

Practices for Lesson 4

Overview

In these practices, you will trigger the Jenkins Jobs from one job to another automatically in the Jenkins Dashboard, create the pipeline using Groovy script in Jenkins instance and learn to integrate the GitHub source code to the Jenkins pipeline, and further delegate a pipeline Job using Agent in Jenkins Instance.

Practice 4-3: Integrate GitHub as Source Code on Jenkins

Overview

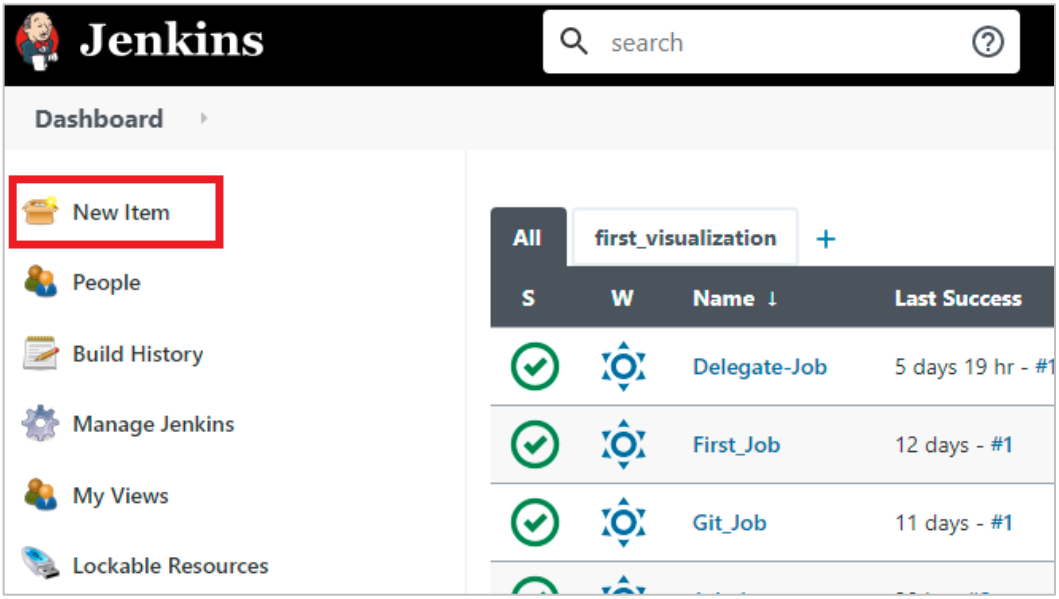
In this practice, you will learn how to integrate the GitHub source code to the Jenkins pipeline and execute the code.

Assumptions

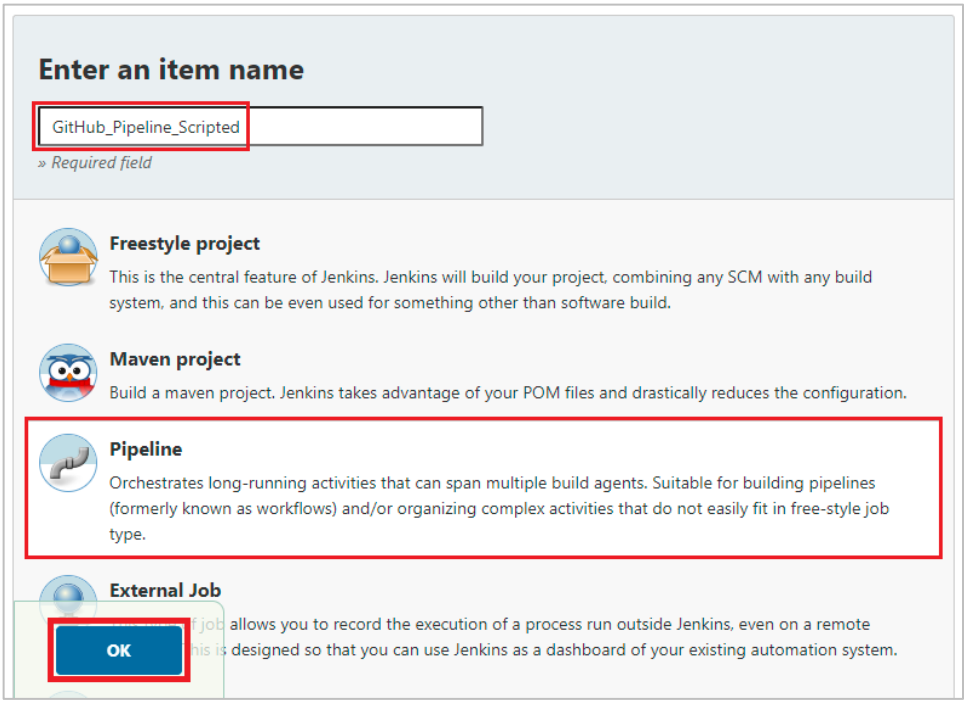
You should have completed the Practice of Lesson 4-2.

Tasks

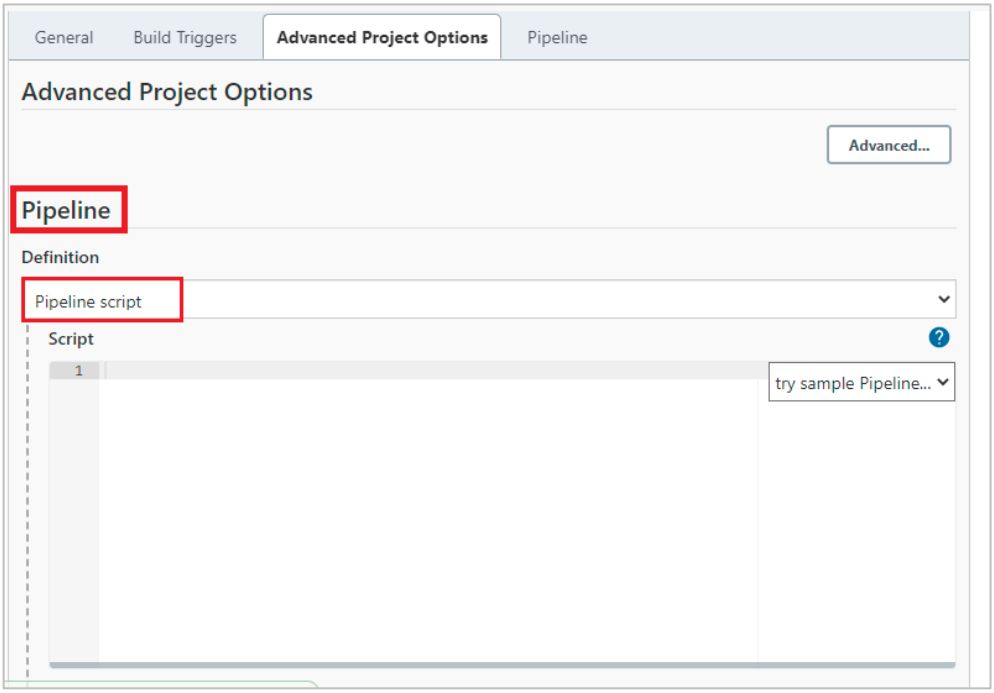
1. Create a Pipeline in Jenkins instance to integrate with GitHub.
   a. In the Jenkins Dashboard, navigate to main menu and select **New Item** to create a **Groovy** Scripted Pipeline as shown below.



   b. Provide the **name** for the GitHub Pipeline, further select **Pipeline** and click **OK** as shown below.



c. Scroll down to **Pipeline** and select **Pipeline script** under **Definition** as shown below.

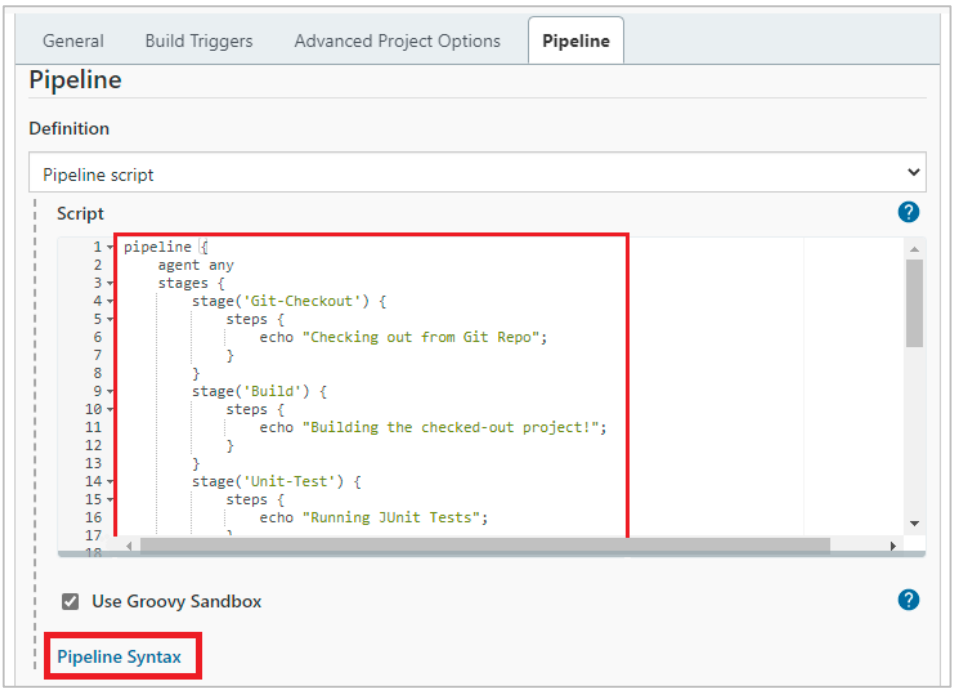


d. Copy the Groovy script provided below and paste it in the **Script** block as shown below. Select the checkbox of **Use Groovy Sandbox**.

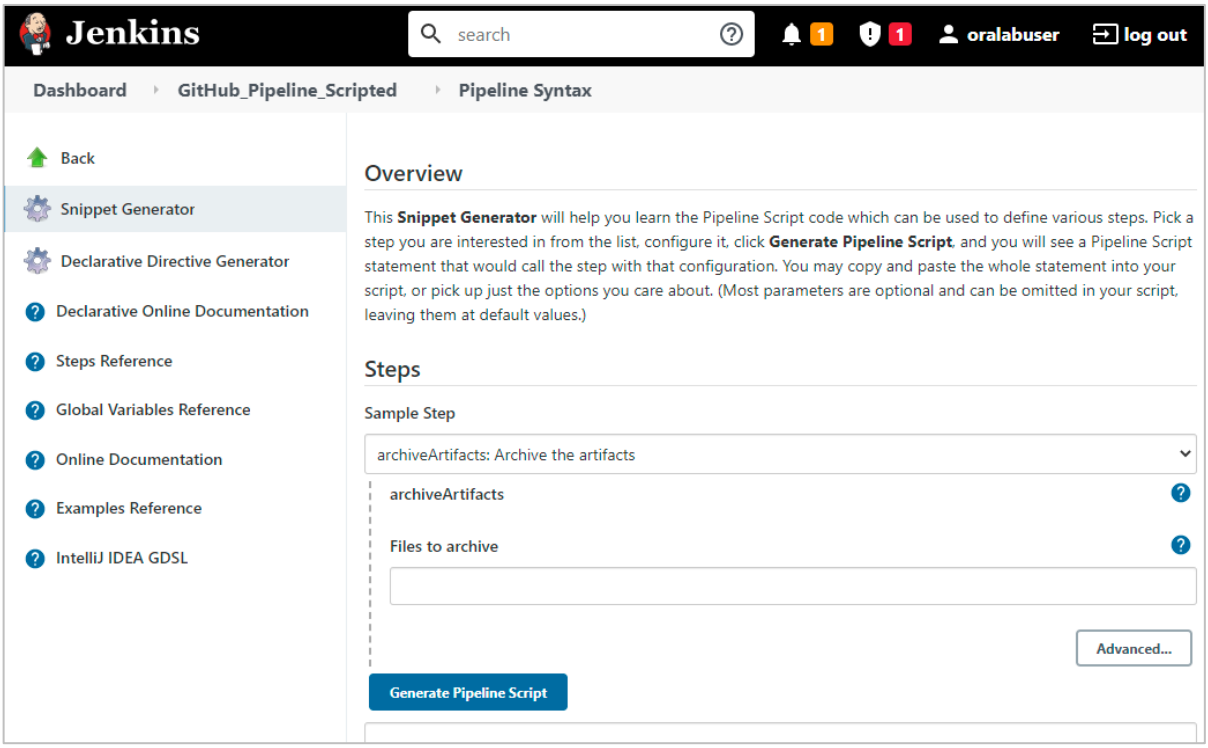
```
pipeline {
      agent any
      stages {
            stage('Git-Checkout') {
                  steps {
```

```
                        echo "Checking out from Git Repo";
                  }
            }
            stage('Build') {
                  steps {
                        echo "Building the checked-out project!";
                  }
            }
            stage('Unit-Test') {
                  steps {
                        echo "Running JUnit Tests";
                  }
            }
            stage('Quality-Gate') {
                  steps {
                        echo "Verifying Quality Gates";
                  }
            }
            stage('Deploy') {
                  steps {
                        echo "Deploying to Stage Environment for more
tests!";
                  }
            }
      }
      post {
            always {
                  echo 'This will always run'
            }
            success {
                  echo 'This will run only if successful'
            }
            failure {
                  echo 'This will run only if failed'
            }
            unstable {
                  echo 'This will run only if the run was marked as
unstable'
            }
            changed {
                  echo 'This will run only if the state of the pipeline
has changed'
```

```
                    echo 'For example, if the Pipeline was previously
failing but is now successful'
            }
        }
}
```

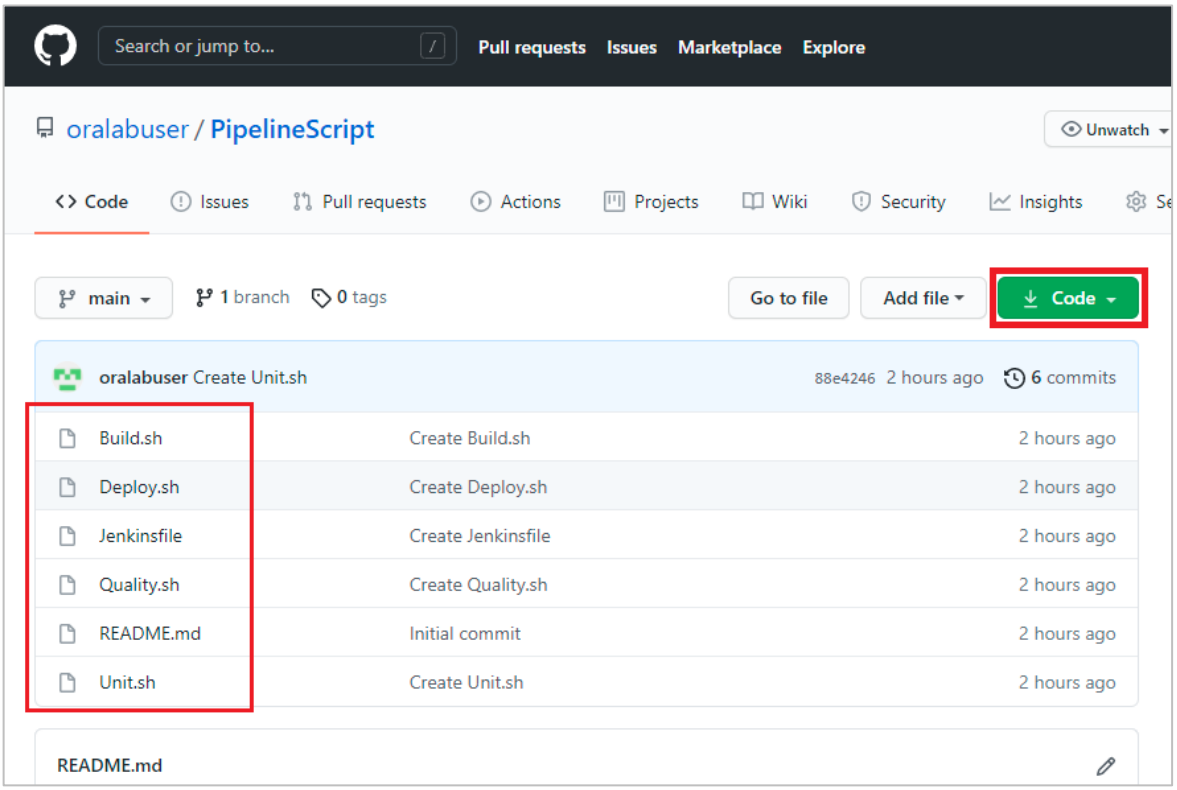


e. Click on **Pipeline Syntax** as shown above to get the **Pipeline Syntax Snippet Generator** page as shown below.

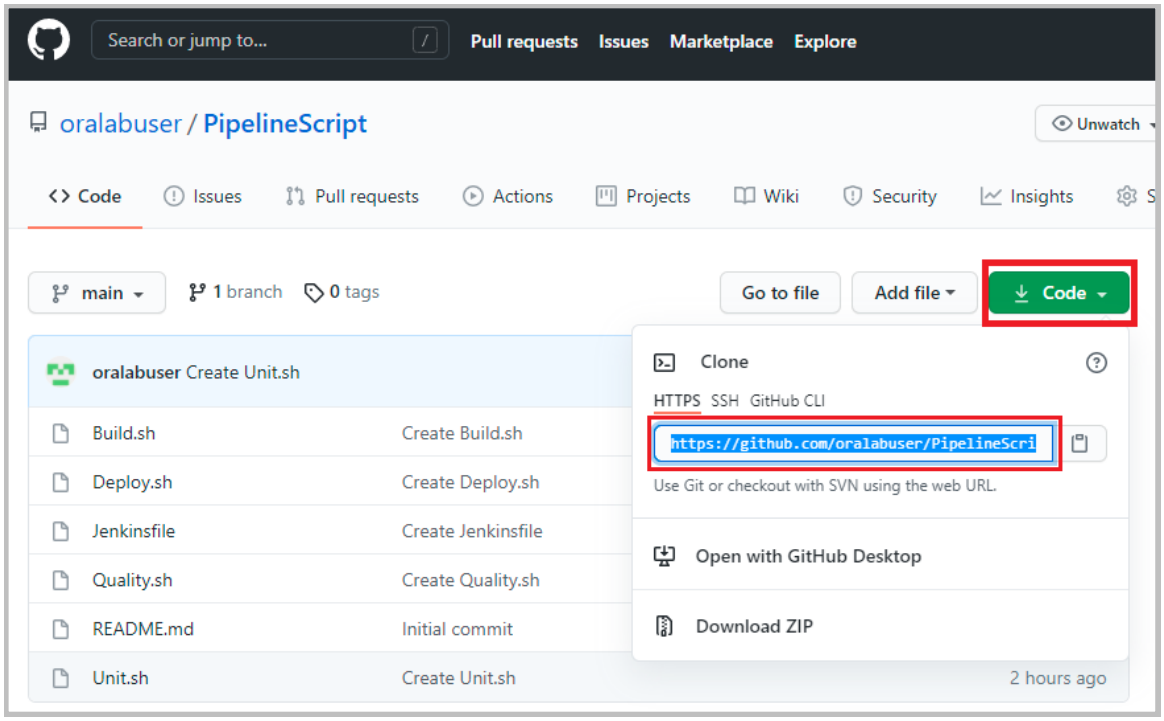


f. The GitHub link is provided below which is in the public domain, consisting of the **.sh** program files to be executed.

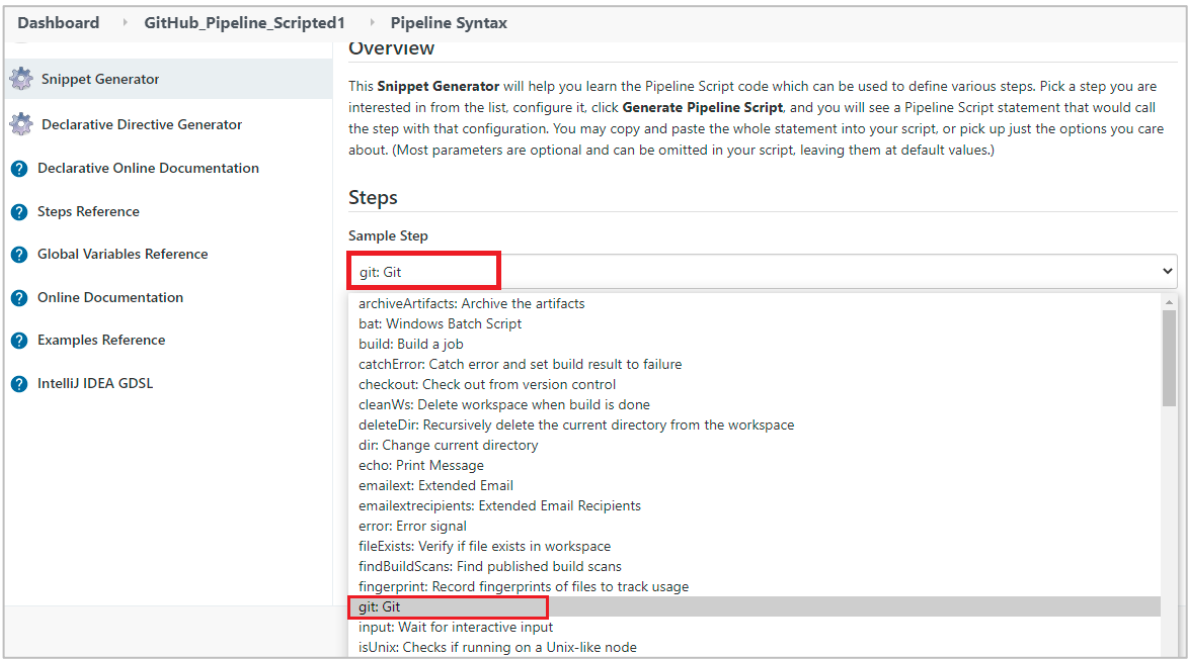
Link: [oralabuser/PipelineScript (github.com)]



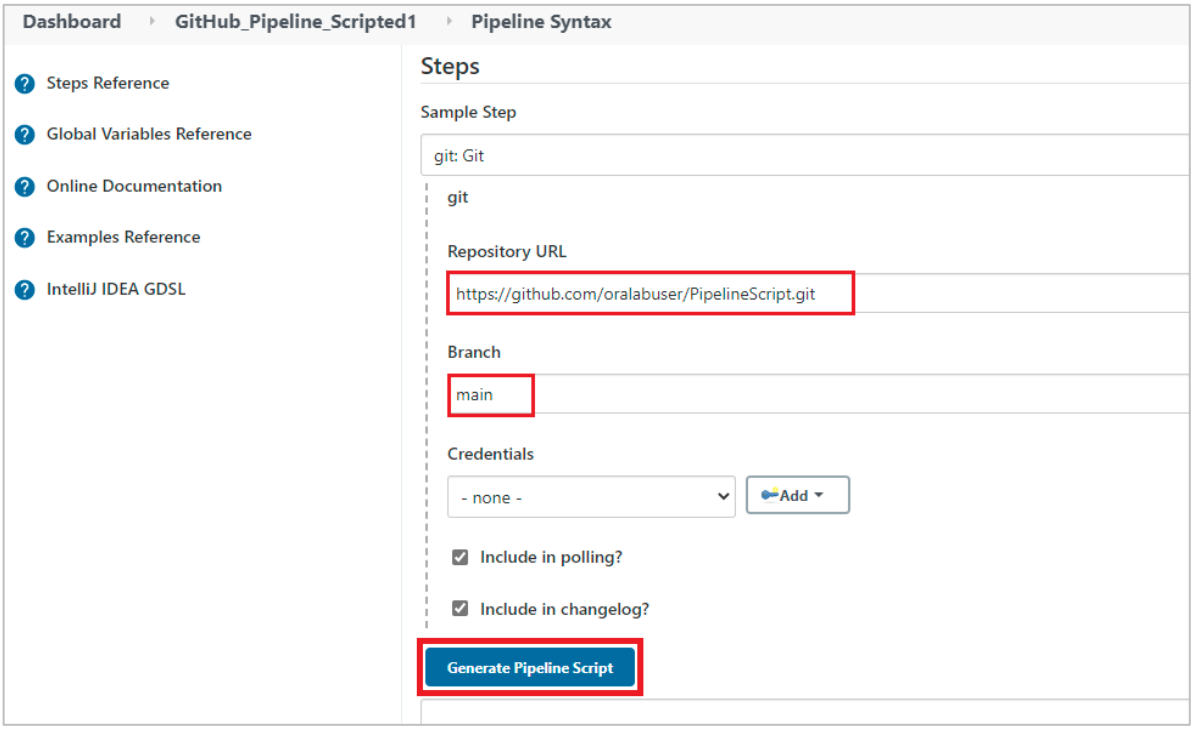
g. Click on **Code** and copy the **HTTPS** link as shown below to access the shell script files.



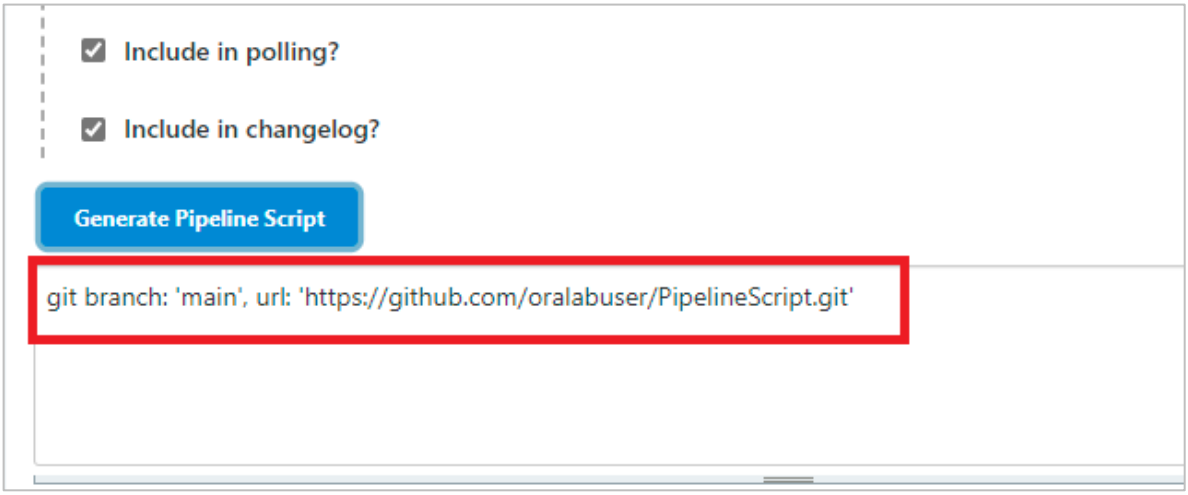
h. In the **Pipeline Syntax Generator** page, navigate to **Sample Step** and select **git: Git** as shown below.



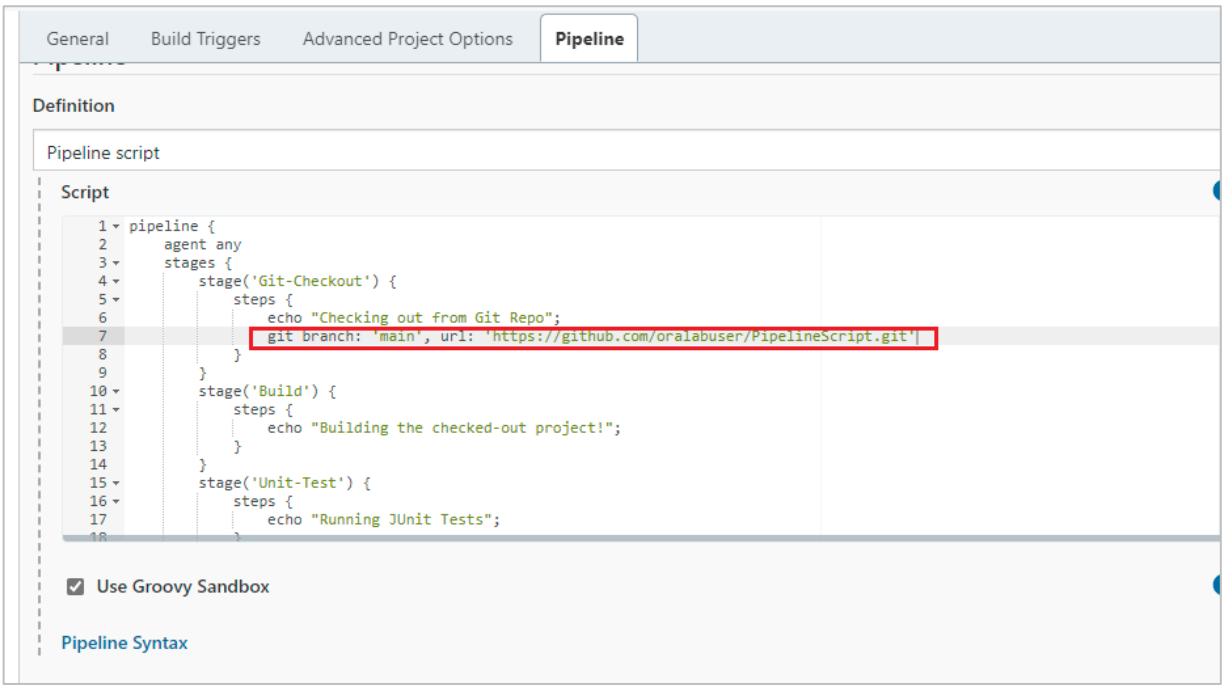
i. Paste the HTTPS GitHub link copied in the **Repository URL**, provide the **Branch** as **main** and click **Generate Pipeline Script** to generate the GitHub code snippet.



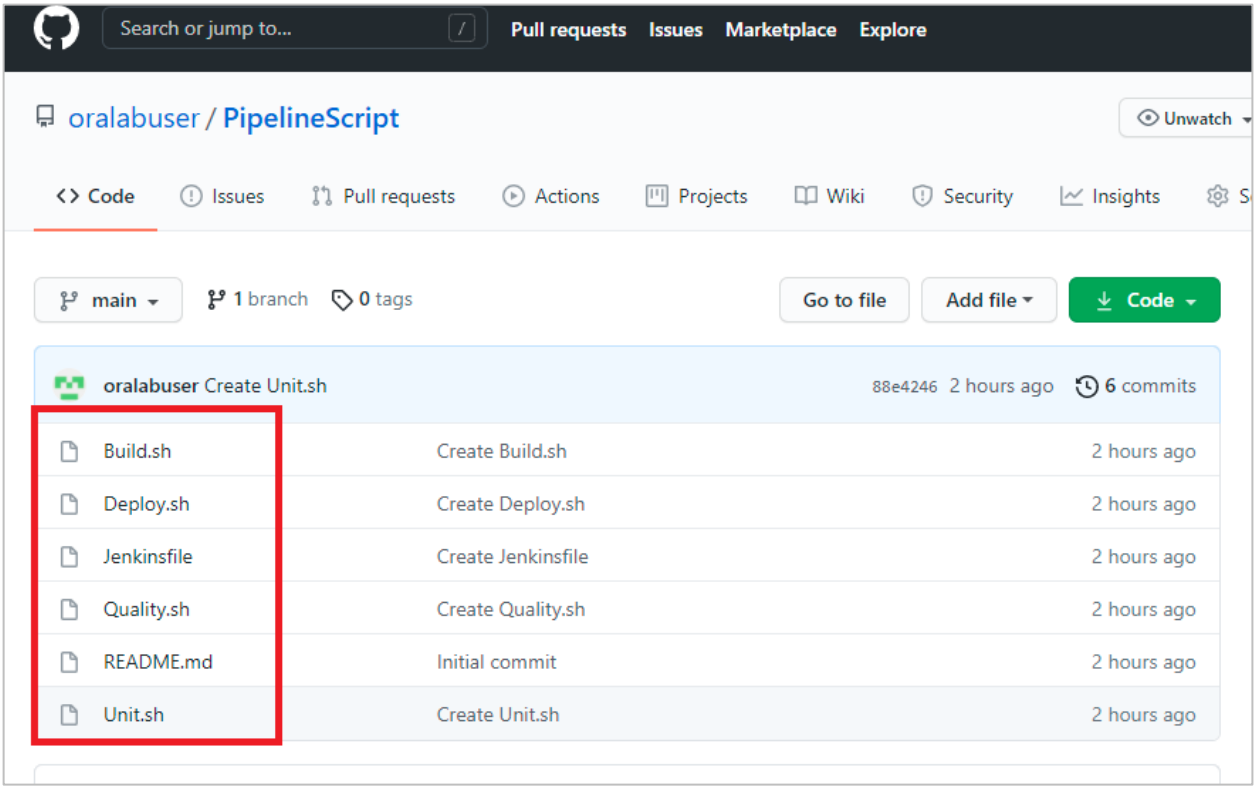
j. As shown below the GitHub Pipeline Script in Groovy is generated. Copy the script.



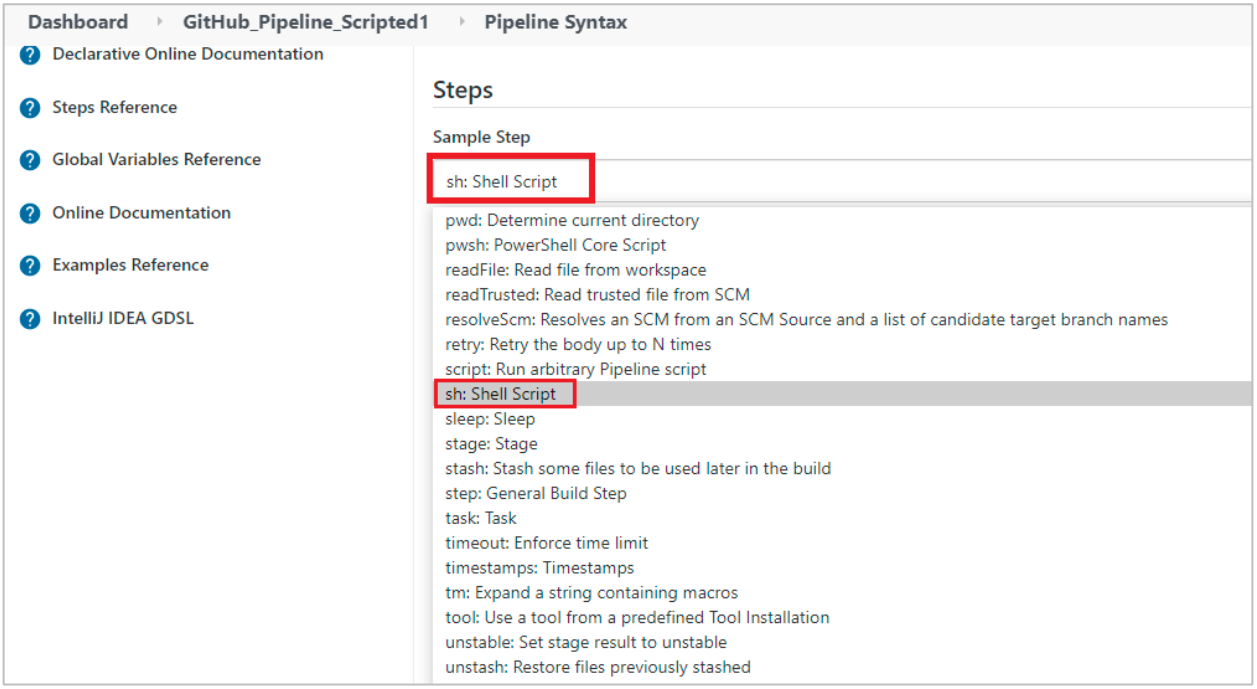
k. Paste the GitHub script in the code as shown below to integrate with the GitHub account and get all the files in the GitHub repository to the Jenkins workspace.



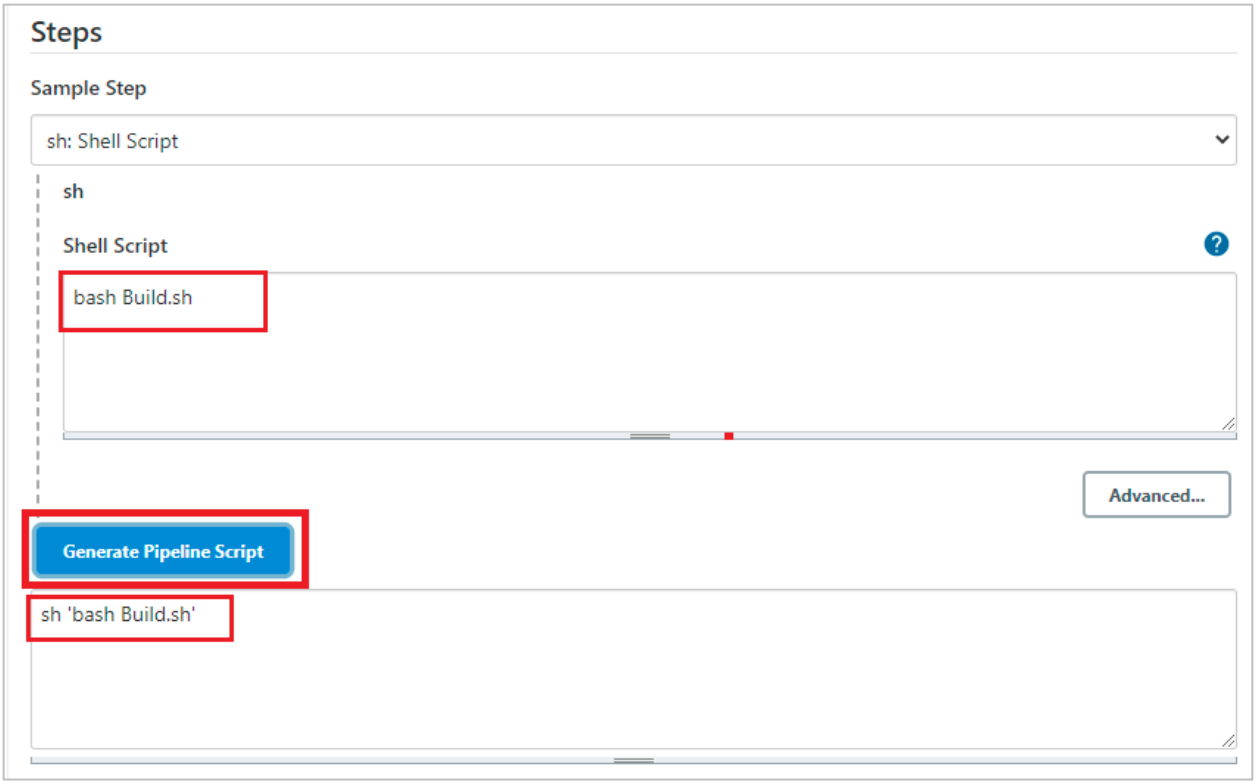
l. As shown below these are the list of files that will be copied to the Jenkins workspace.



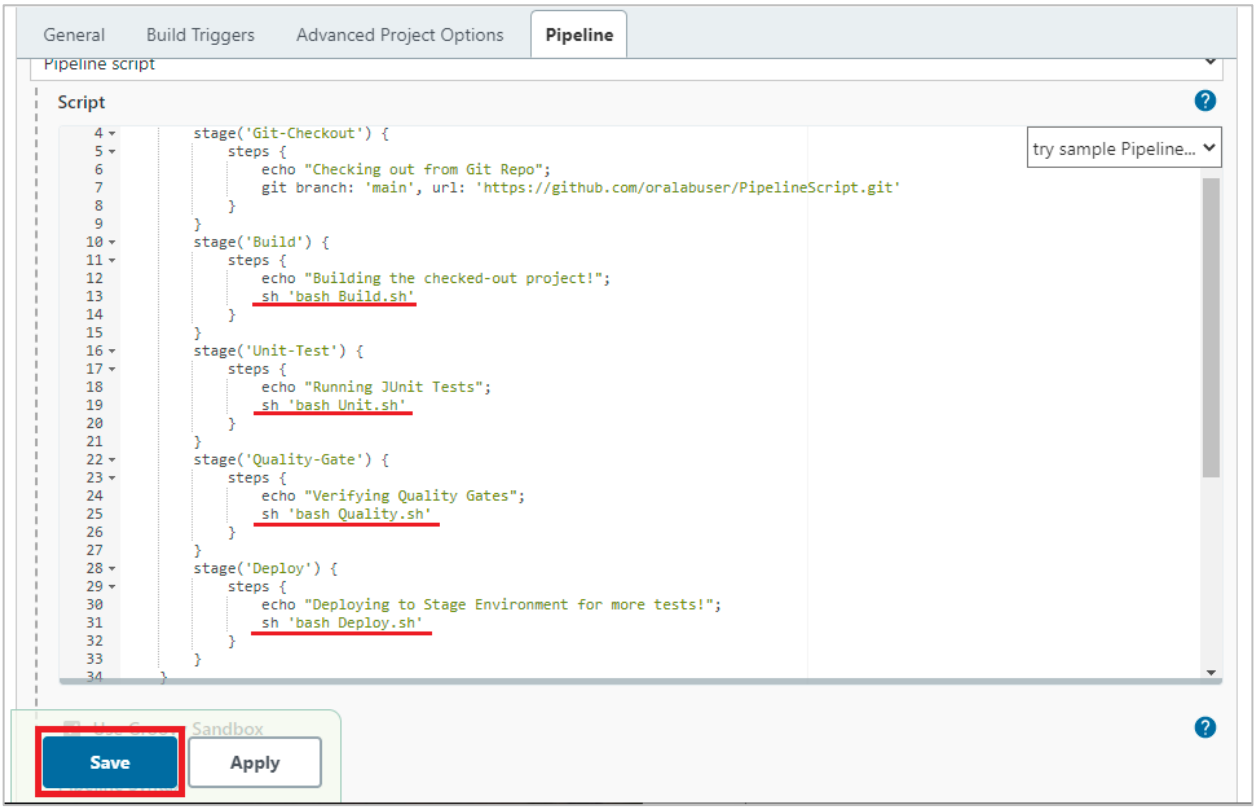
m. In the **Pipeline Syntax Generator** page, navigate to **Sample Step** and select **sh: Shell Script** as shown below.



n. In the **Shell Script** block, type the command as shown below to execute the **Build.sh** file in the Jenkins Pipeline. Click **Generate Pipeline Script** and copy the **sh** code.



o. Paste the code in the **Build** stage as shown below. Similarly, provide the code for **Unit.sh**, **Quality.sh** and **Deploy.sh**. Click **Save.**

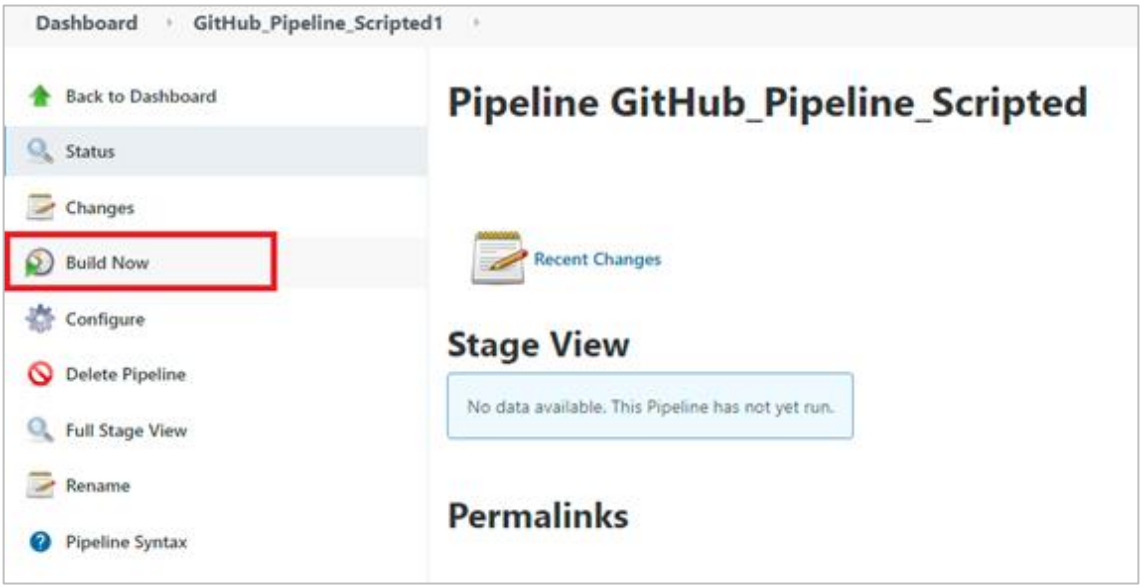


p. On updating the Groovy script, final code looks similar as shown below.

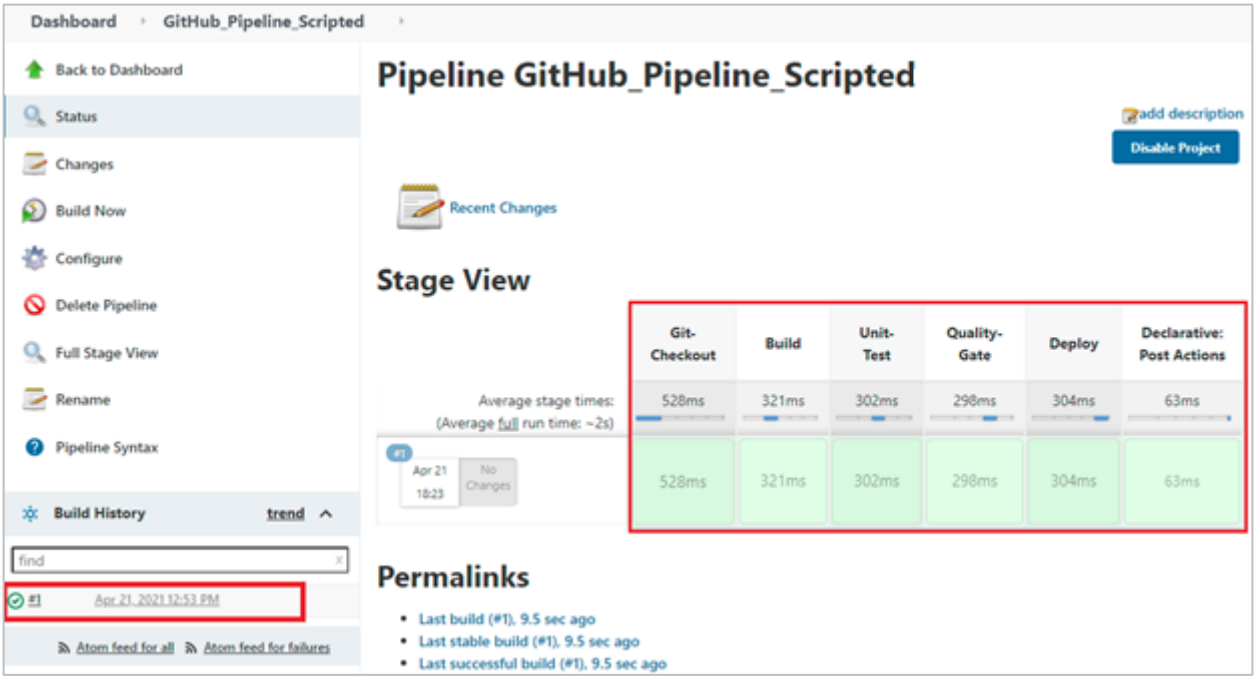
```
pipeline {
      agent any
      stages {
            stage('Git-Checkout') {
                  steps {
                        echo "Checking out from Git Repo";
                        git branch: 'main', url:
'https://github.com/oralabuser/PipelineScript.git'
                  }
            }
            stage('Build') {
                  steps {
                        echo "Building the checked-out project!";
                        sh 'bash Build.sh'
                  }
            }
            stage('Unit-Test') {
                  steps {
                        echo "Running JUnit Tests";
                        sh 'bash Unit.sh'
                  }
            }
            stage('Quality-Gate') {
                  steps {
                        echo "Verifying Quality Gates";
                        sh 'bash Quality.sh'
                  }
            }
            stage('Deploy') {
                  steps {
                        echo "Deploying to Stage Environment for more
tests!";
                        sh 'bash Deploy.sh'
                  }
            }
      }
      post {
            always {
                  echo 'This will always run'
            }
            success {
```

```
                echo 'This will run only if successful'
            }
            failure {
                echo 'This will run only if failed'
            }
            unstable {
                echo 'This will run only if the run was marked as
unstable'
            }
            changed {
                echo 'This will run only if the state of the pipeline
has changed'
                echo 'For example, if the Pipeline was previously
failing but is now successful'
            }
        }
}
```

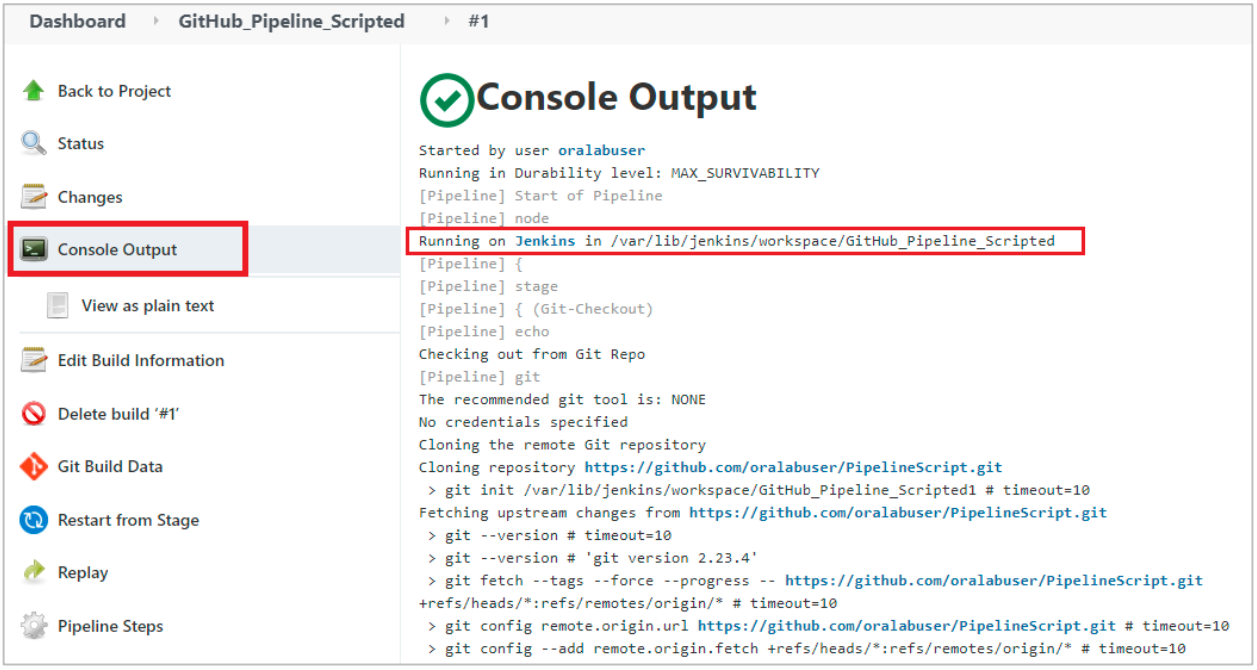
2. Build the Groovy Scripted Pipeline created in the Jenkins.
   a. Select **Build Now** from the menu to execute the Pipeline as shown below.



   b. As shown below, **view** the **stages** of **pipeline** getting executed and click on the link under **Build History**.



c. Click on **Console Output** to view the execution of the scripted pipeline integrating with the GitHub.



d. As shown below, the scripted pipeline integrating with GitHub is executed successfully.

```
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deploying to Stage Environment for more tests!
[Pipeline] sh
+ bash Deploy.sh
Deploying Build : 04/21/21 : 12:53:37
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
This will always run
[Pipeline] echo
This will run only if the state of the pipeline has changed
[Pipeline] echo
For example, if the Pipeline was previously failing but is now successful
[Pipeline] echo
This will run only if successful
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

e. Connect to Jenkins AWS instance from Putty to verify the path of the workspace consisting of the GitHub repository files as shown below.

```
[ec2-user@ip-172-31-33-131 ~]$ cd /var/lib/jenkins/workspace/GitHub_Pipeline_Scripted
[ec2-user@ip-172-31-33-131 GitHub_Pipeline_Scripted]$ ls
Build.sh  Deploy.sh  Jenkinsfile  Quality.sh  README.md  Unit.sh
[ec2-user@ip-172-31-33-131 GitHub_Pipeline_Scripted]$
```

3. Keep the Jenkins Dashboard and the AWS Management Console open for the next practice.