Lesson-End Project

Integrating GitHub with Jenkins

Project agenda: To create a Jenkinsfile pipeline script in GitHub and use it to set up a Jenkins pipeline job for cloning, compiling, testing, and packaging the codebase

Description: You are a software developer managing a web application on GitHub. To enhance the efficiency of the deployment process, you have taken the initiative to set up a Jenkins server. As part of this setup, a Jenkinsfile must be integrated into your project's GitHub repository. This Jenkinsfile is responsible for orchestrating essential tasks such as code checkout, Maven-based building, and testing. Whenever you push updates to GitHub, Jenkins automatically triggers the pipeline, ensuring that your changes are seamlessly integrated and deployed.

Tools required: GitHub and Jenkins

Prerequisites: None

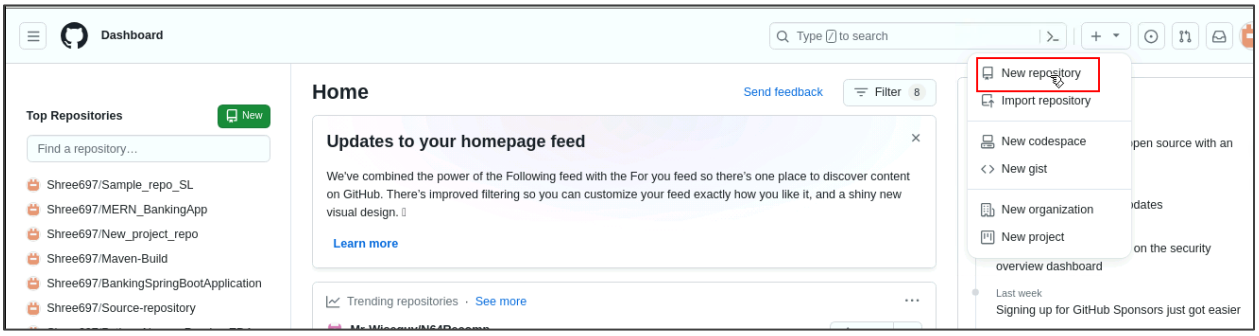
Expected deliverables: A Jenkins pipeline job set up to perform tasks such as code checkout, Maven-based building, and testing whenever updates are pushed to GitHub

Steps to be followed:

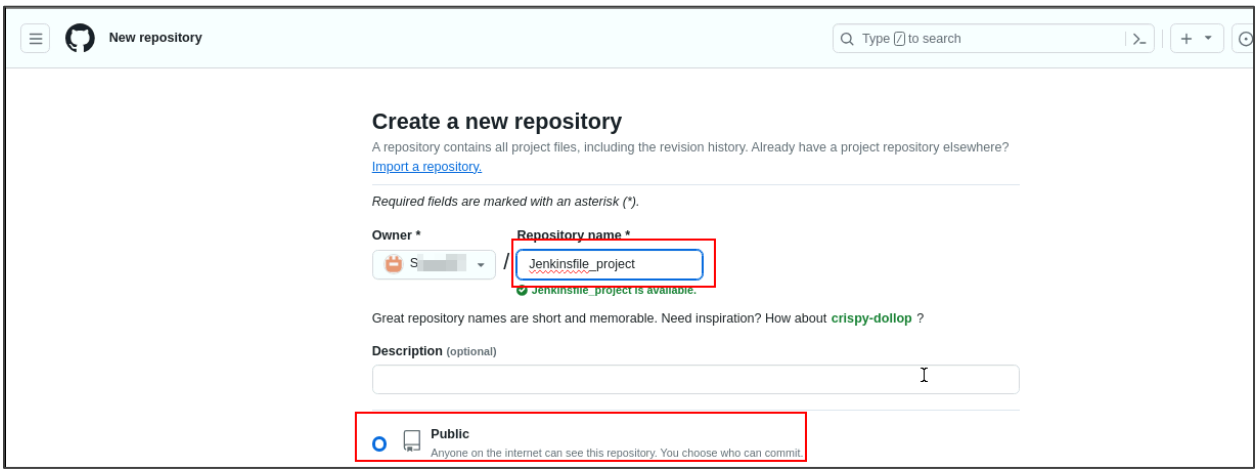
1. Create a Jenkinsfile pipeline script file in a GitHub repository
2. Create the Jenkins pipeline job
3. Execute the Jenkins job

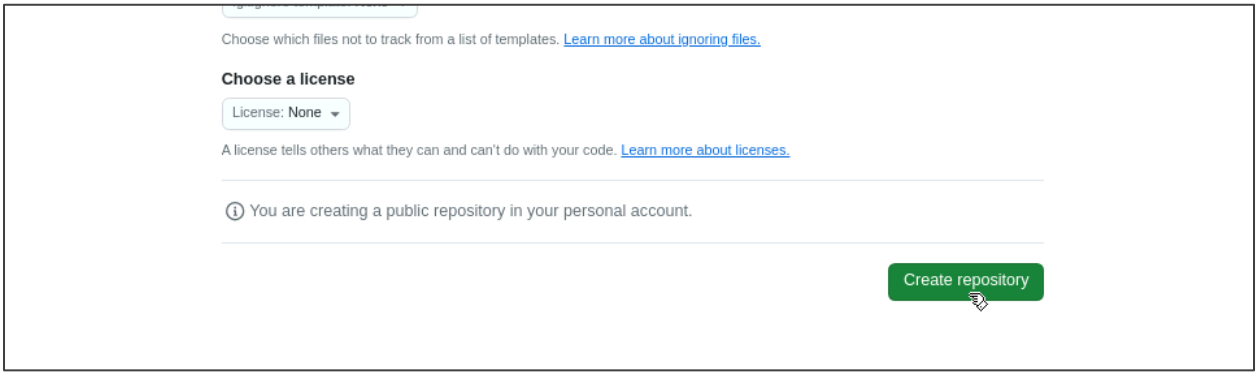
Step 1: Create a Jenkinsfile pipeline script file in a GitHub repository

1.1 Log in to your GitHub account and click on **New Repository** to create a new repository

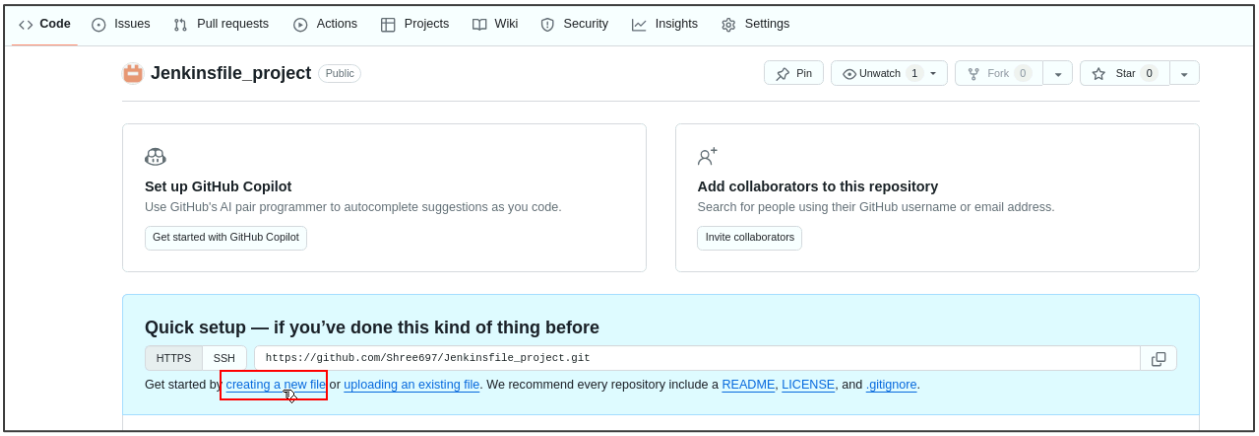


1.2 Enter a name for the repository, select **Public** as the repository type, and click on **Create repository**

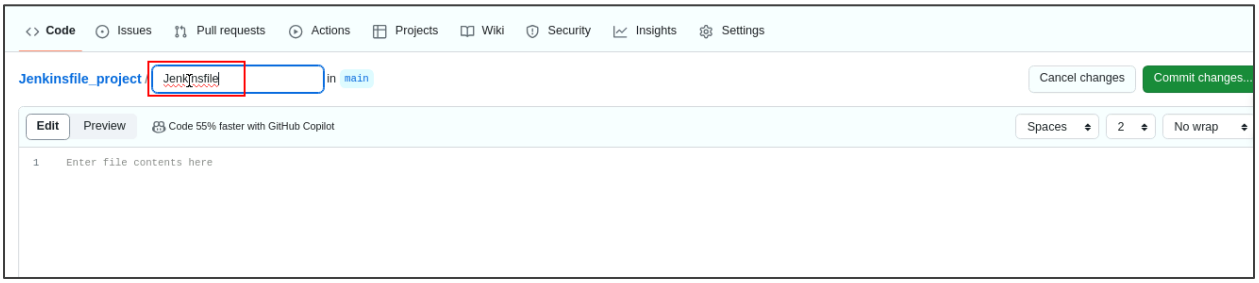




1.3 Click on **creating a new file** to create a new file within the repository



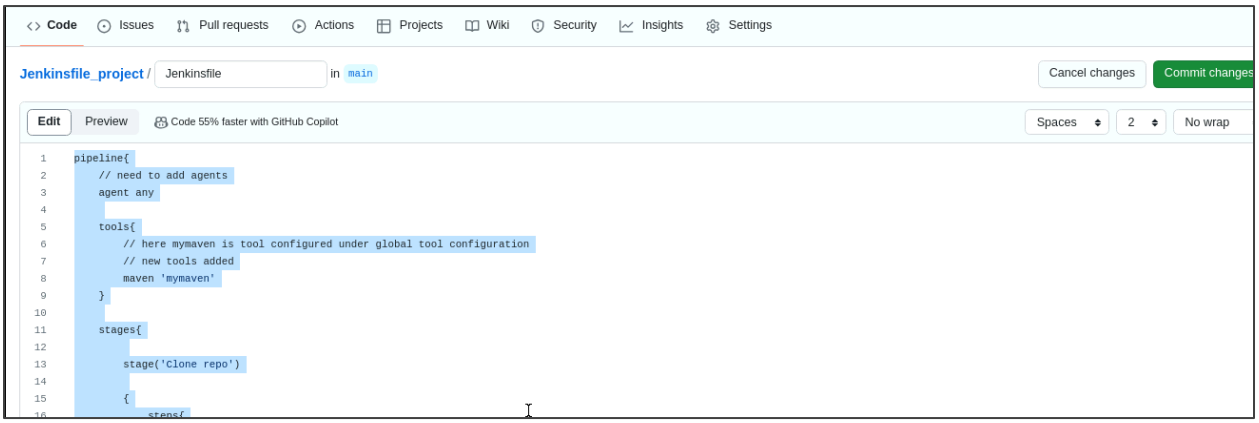
1.4 Enter the file name as **Jenkinsfile**



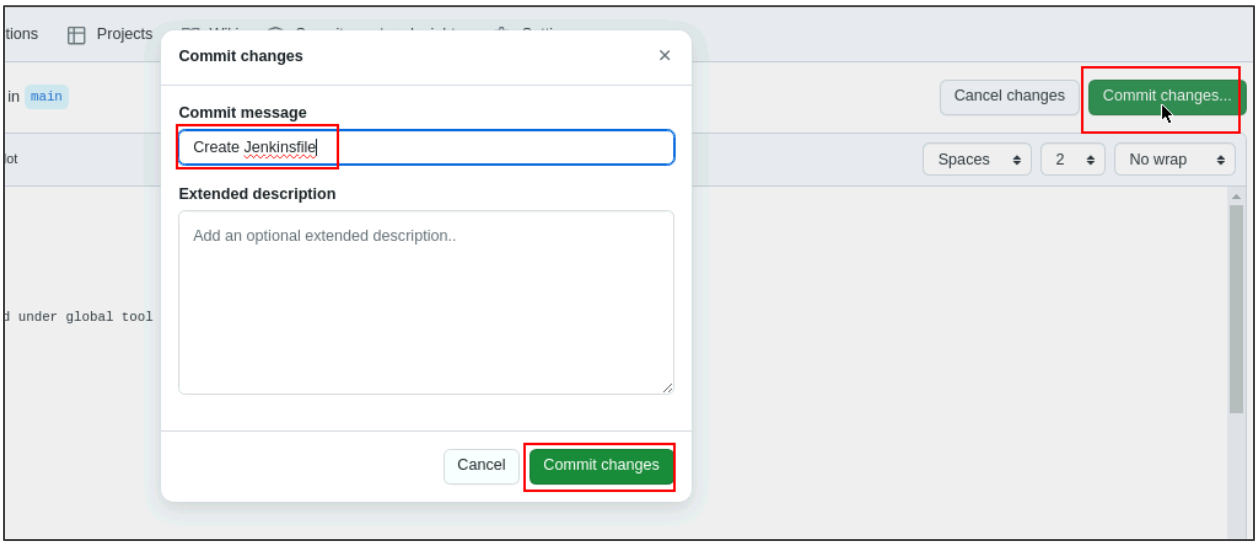
1.5 Add the following script to the file you created:

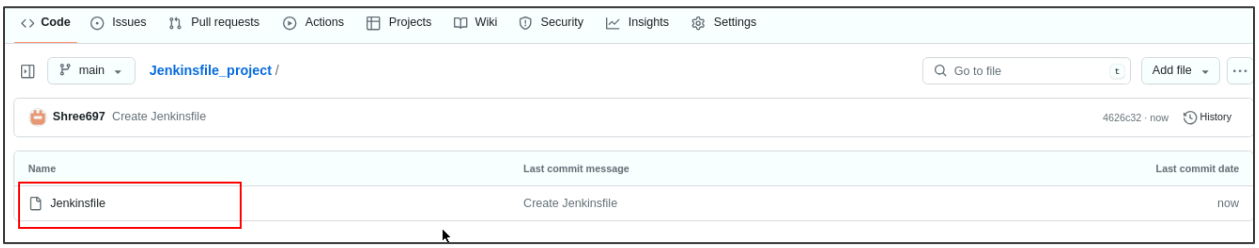
```
pipeline{
  // need to add agents
  agent any

  tools{
    // here mymaven is tool configured under global tool configuration
    // new tools added
    maven 'mymaven'
  }

  stages{

    stage('Clone repo')

    {
      steps{
        git 'https://github.com/github-simplilearn-net/MavenBuild.git'
      }
```

```
        }

        stage('Compile Code')
        {
          steps{

            sh 'mvn compile'
          }
        }

            stage('Test Code')
        {

          steps{

            sh 'mvn test'
          }
          post{
            success{
              junit 'target/surefire-reports/*.xml'
            }
          }
        }
         stage('Package Code')
        {
          steps{

            sh 'mvn package'
          }
        }
      }
}
```



1.6 Click on **Commit changes**, provide a commit message, and then click **Commit changes** again to finalize them

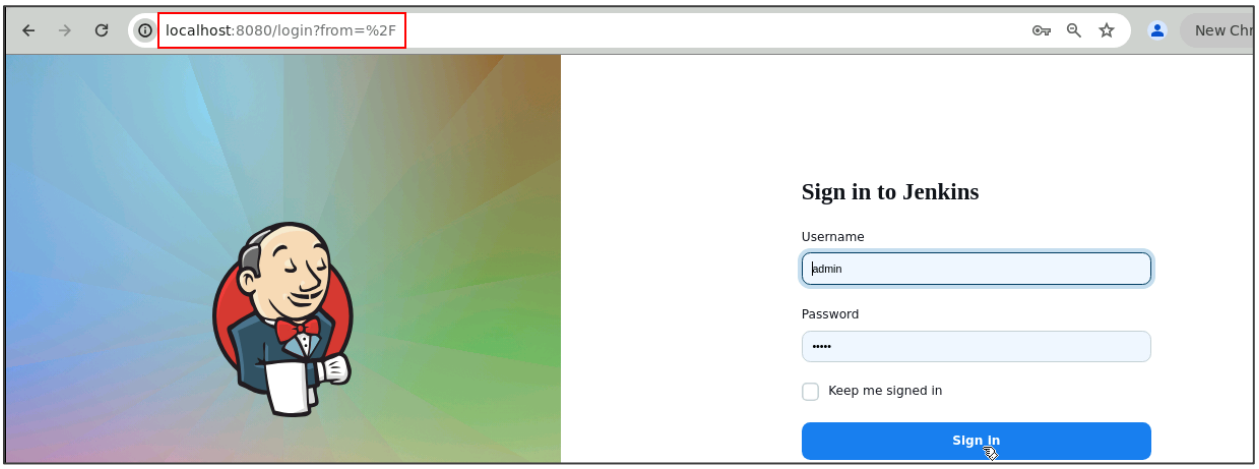




A Jenkinsfile pipeline script is created within the project repository.

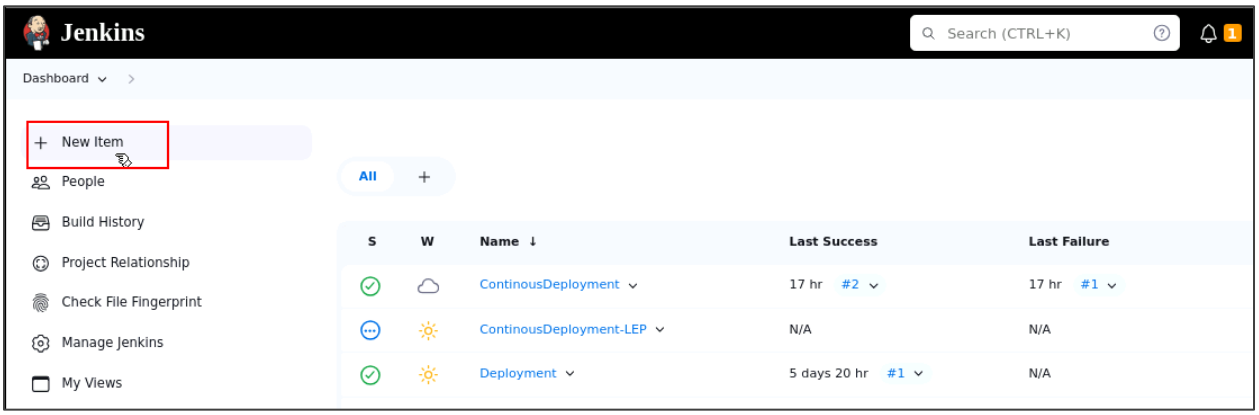
Step 2: Create the Jenkins pipeline job

2.1 Visit **localhost:8080** and sign in to the Jenkins CI tool

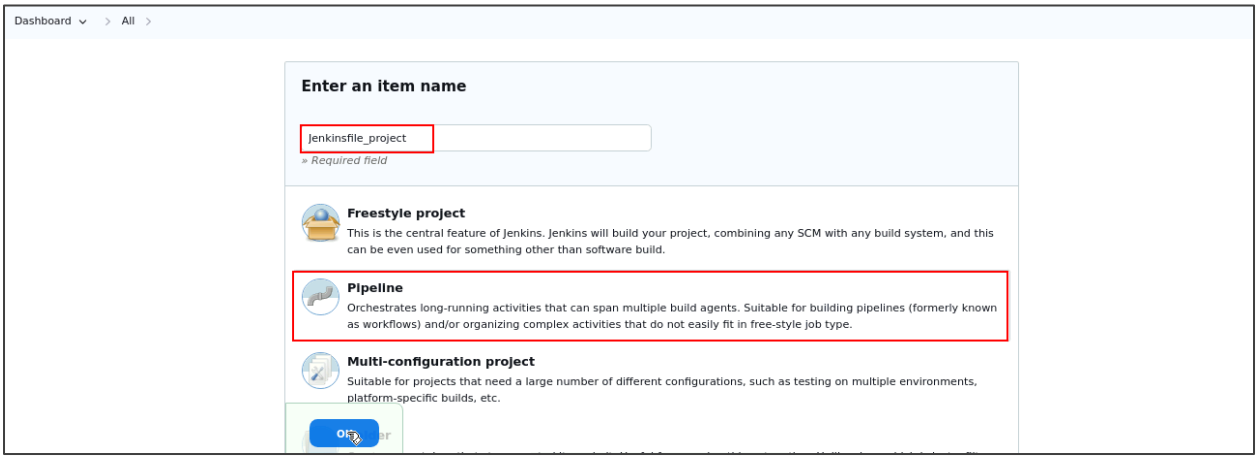


Note: The credentials for accessing Jenkins in the lab are Username: **admin** and Password: **admin**.

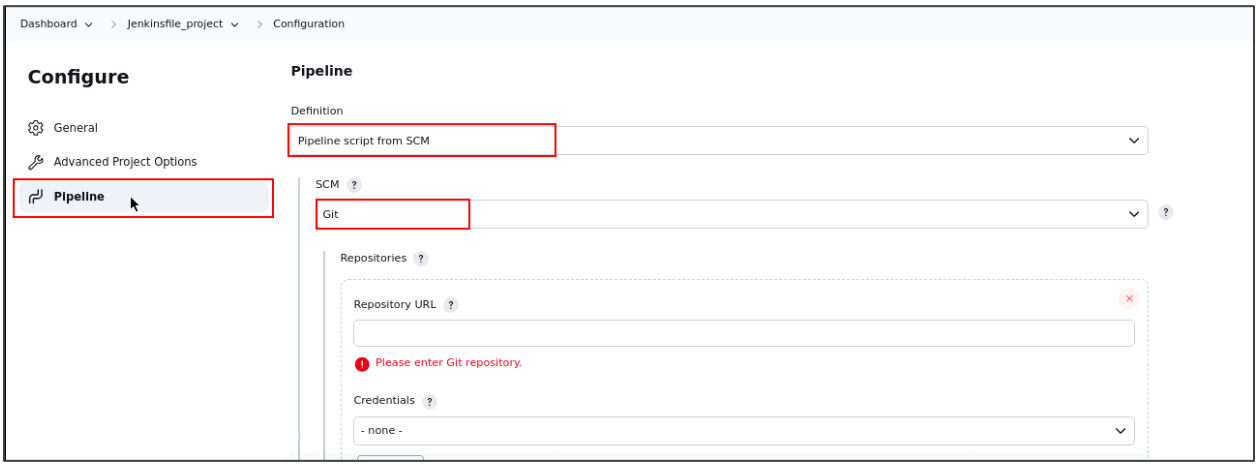
2.2 In the Jenkins dashboard, click on **New Item** to create a new Jenkins job



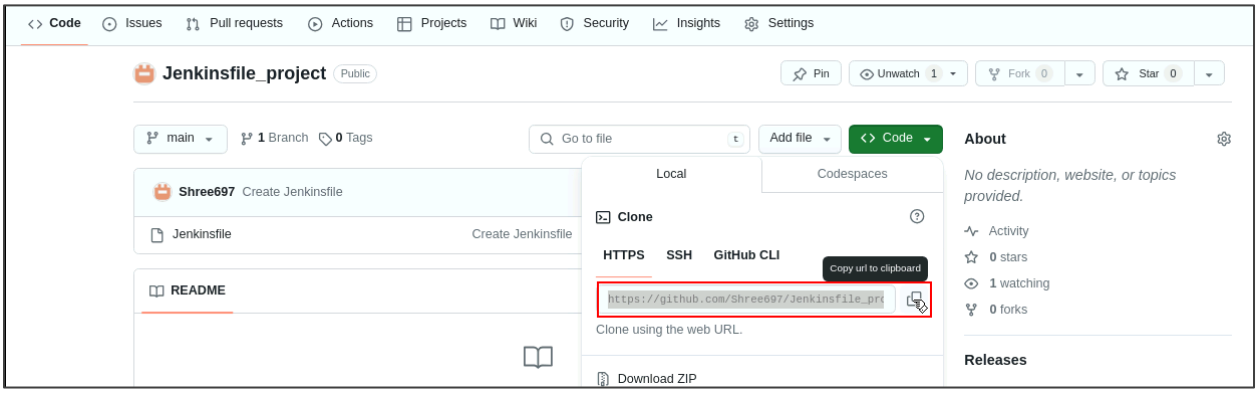
2.3 Enter a name for the Jenkins job, select **Pipeline,** and click on **OK**



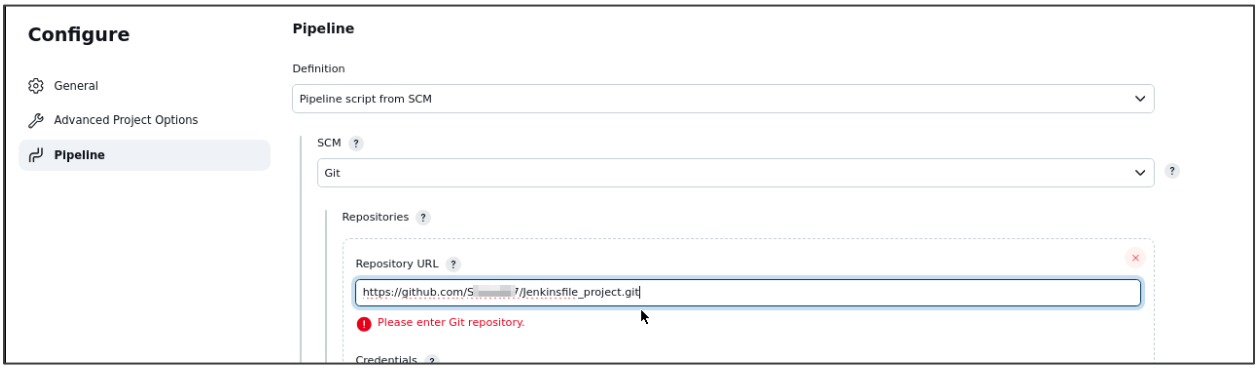
2.4 In the job configuration page, go to **Pipeline**, select **Pipeline script from SCM** as the pipeline definition, and choose **Git** as the SCM tool



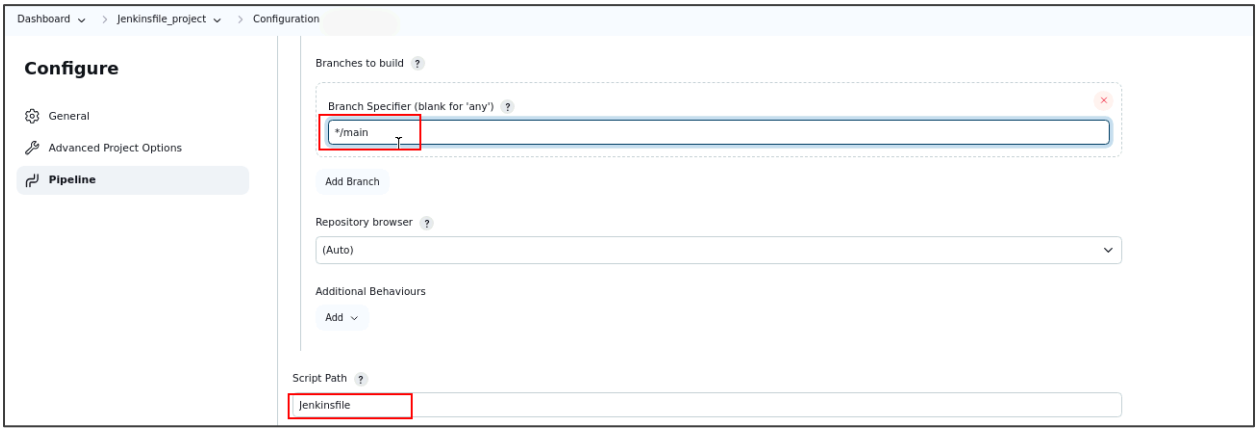
2.5 Go to the main project page in GitHub and copy the HTTPS URL



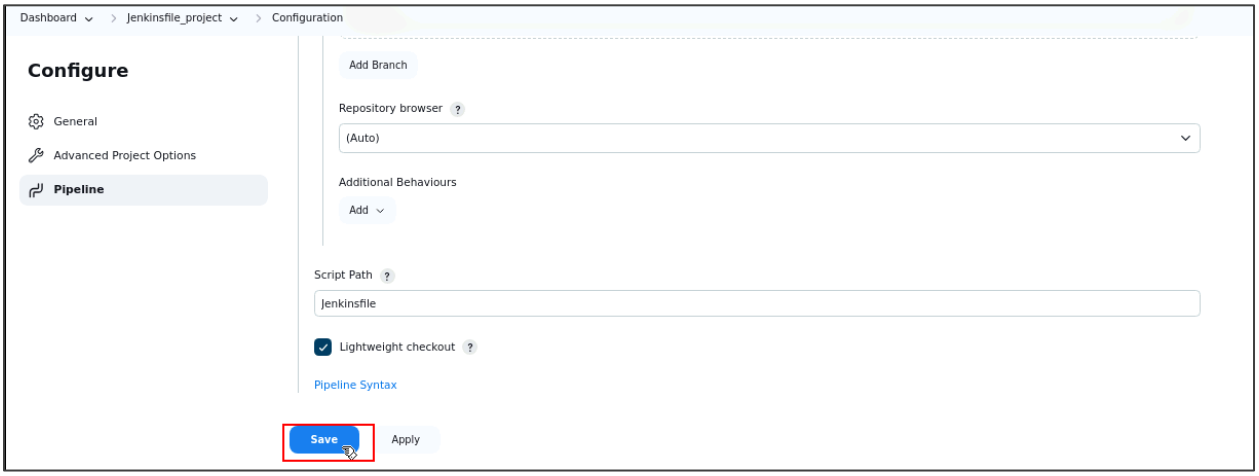
2.6 Paste the copied URL in the **Repository URL** field for the pipeline configuration



2.7 Scroll down, enter ***/main** in the **Branch Specifier**, and keep **Script Path** as **Jenkinsfile**

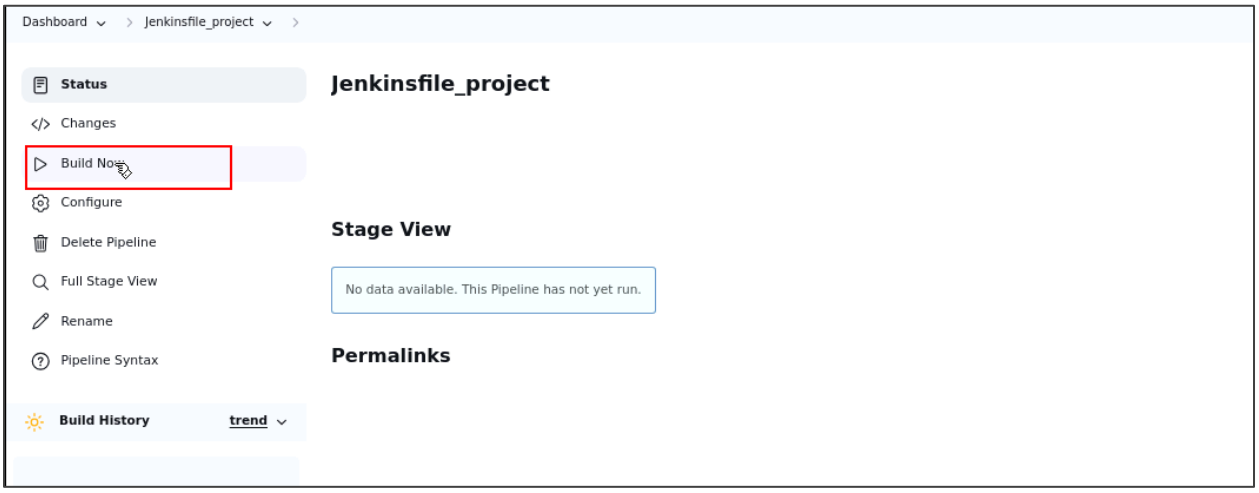


2.8 Click on **Save** to save the Jenkins pipeline job



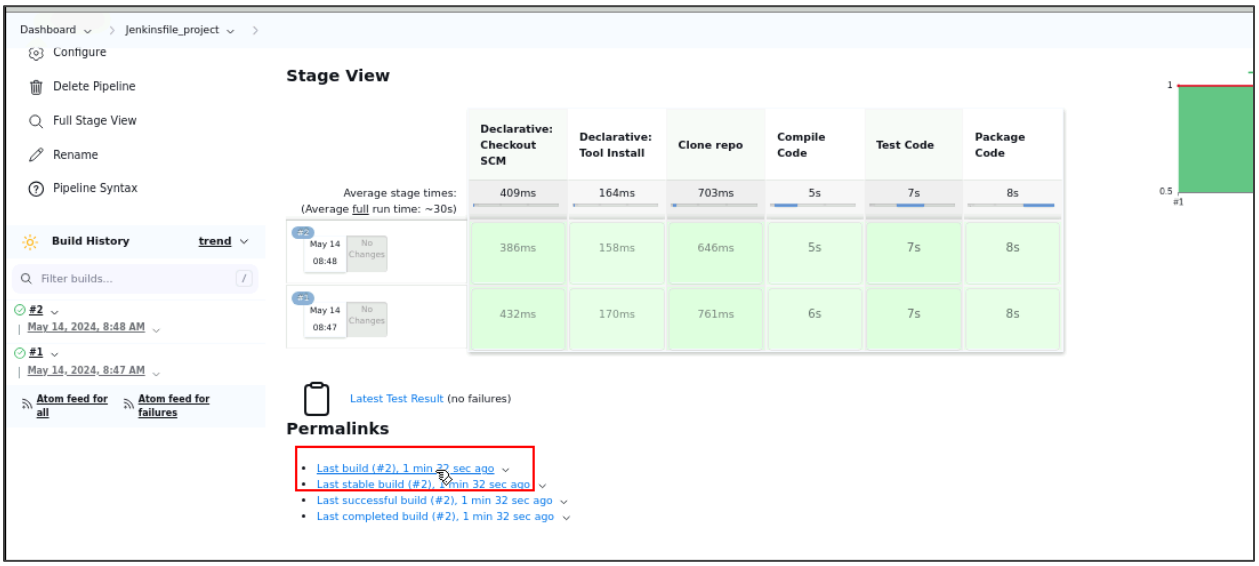
Step 3: Execute the Jenkins job

3.1 After saving the Jenkins pipeline job, click on **Build Now**



The build creation through the Jenkinsfile pipeline script is successful.

3.2 Click on any **Permalink** and select **Console Output** to view a detailed report of the build creation



By following these steps, you have successfully created a Jenkinsfile pipeline script in GitHub and used it to set up a Jenkins pipeline job for cloning, compiling, testing, and packaging the codebase.