

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

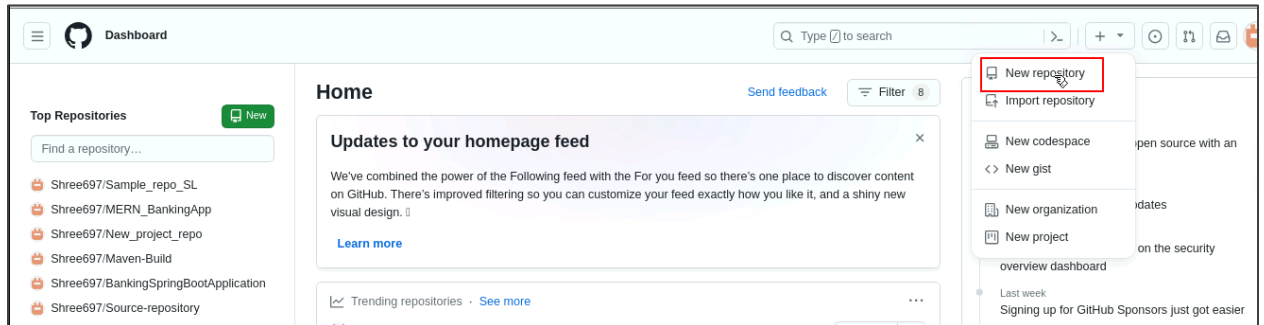
Expected deliverables: A Jenkins pipeline job set up to perform tasks such as code checkout, Maven-based building, and testing whenever updates are pushed to GitHub

Steps to be followed:

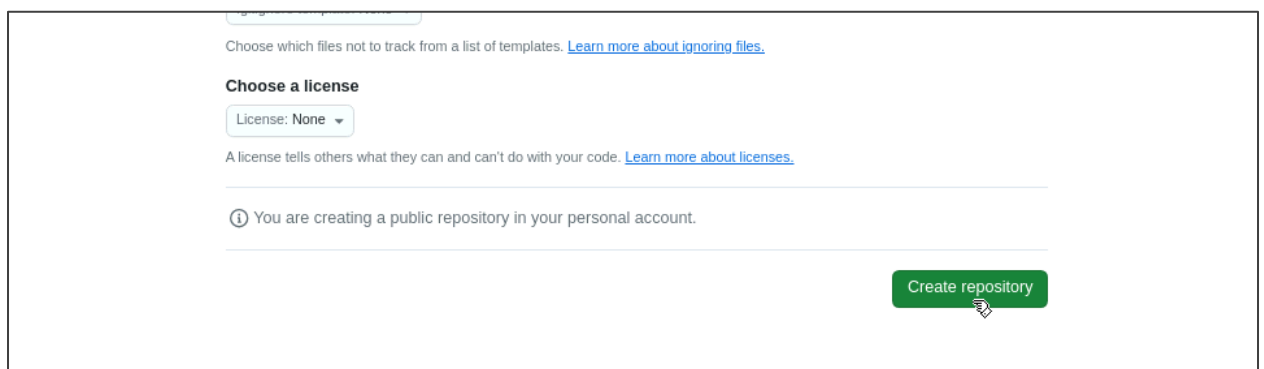
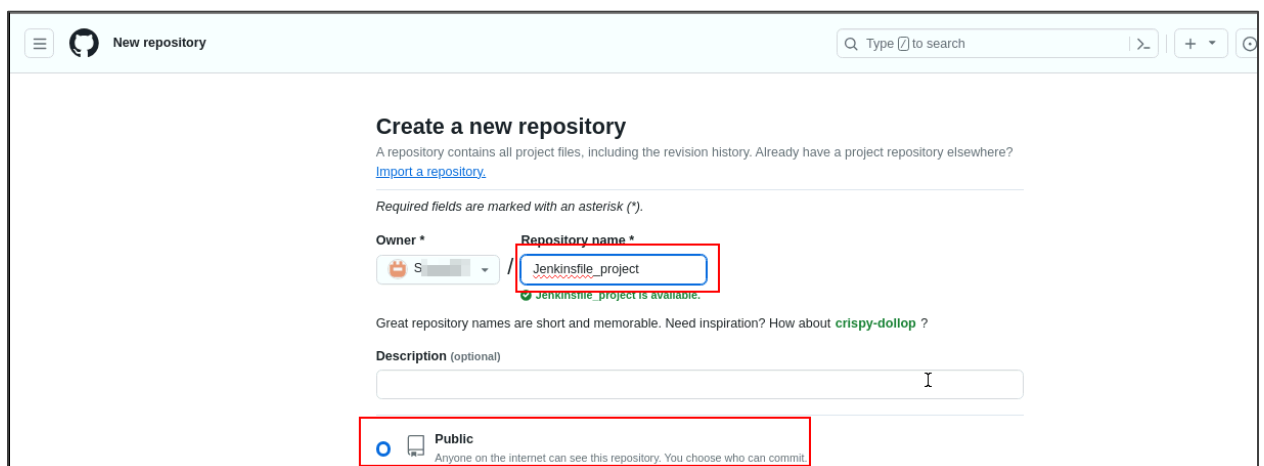
1. Create a Jenkinsfile pipeline script file in a GitHub repository
2. Create the Jenkins pipeline job
3. Execute the Jenkins job

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

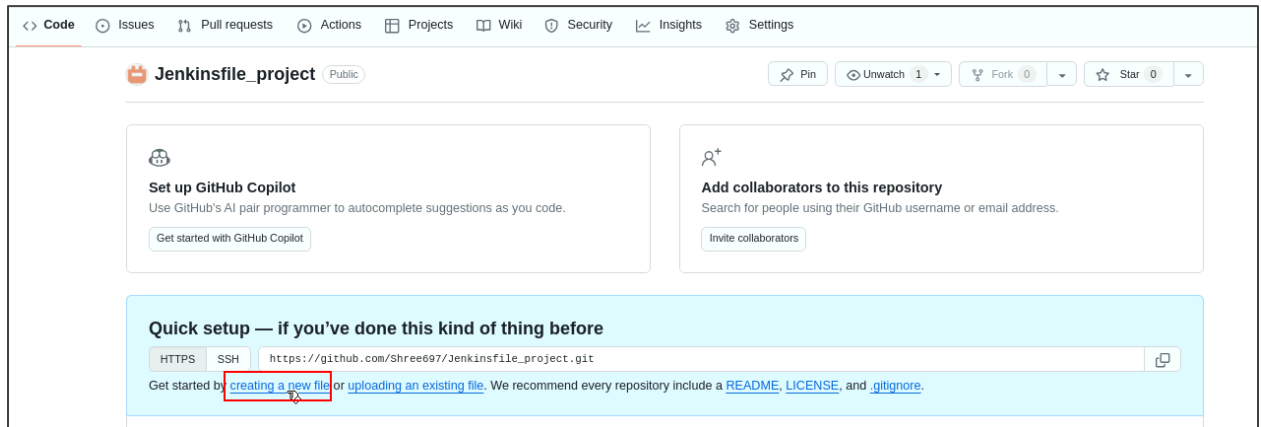
1.1 Log in to your GitHub account and click on **New Repository** to create a new repository



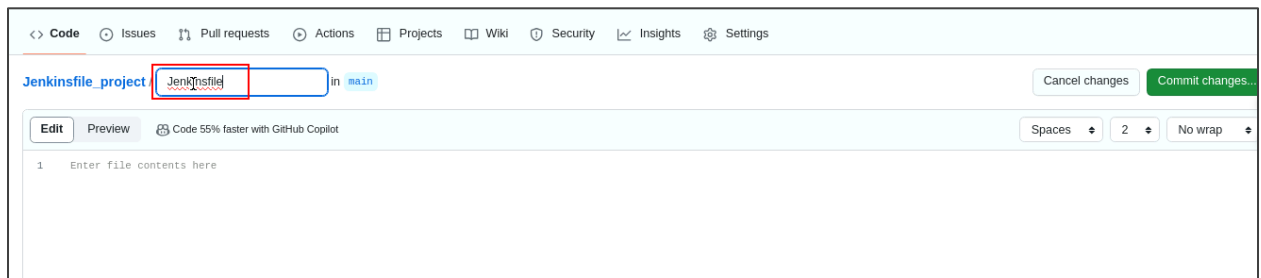
1.2 Enter a name for the repository, select **Public** as the repository type, and click on **Create repository**



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'

        }
    }
}

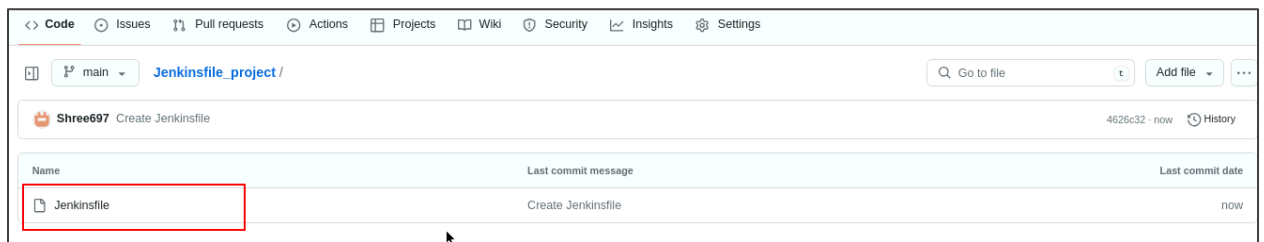
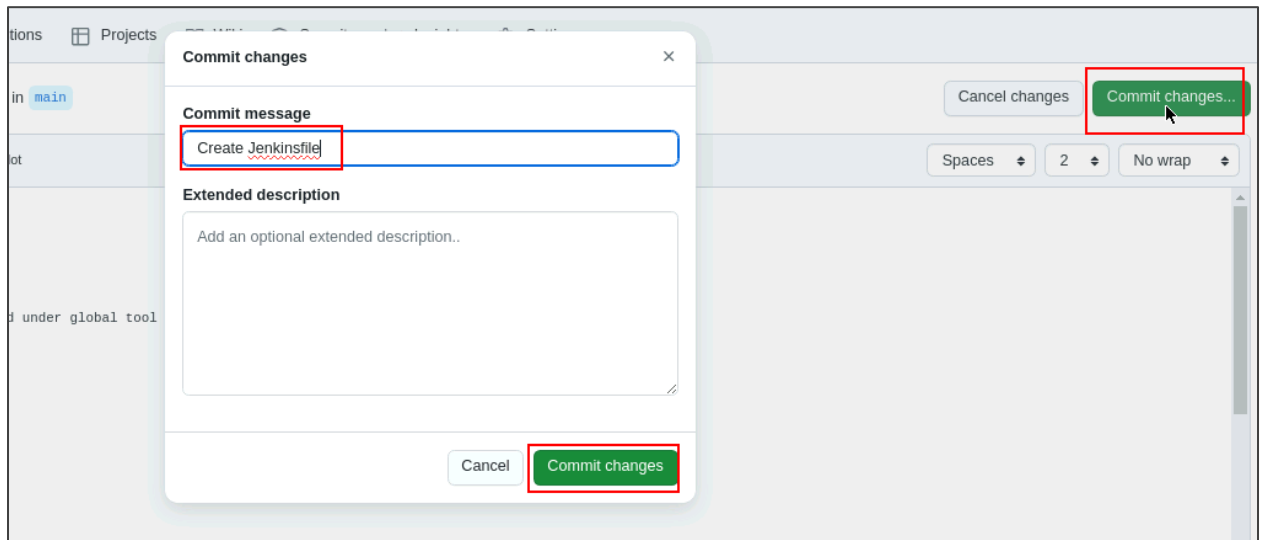
```

```

1 pipeline{
2   // need to add agents
3   agent any
4
5   tools{
6     // here mymaven is tool configured under global tool configuration
7     // new tools added
8     maven 'mymaven'
9   }
10
11  stages{
12    stage('clone repo')
13
14    {
15      steps{
16

```

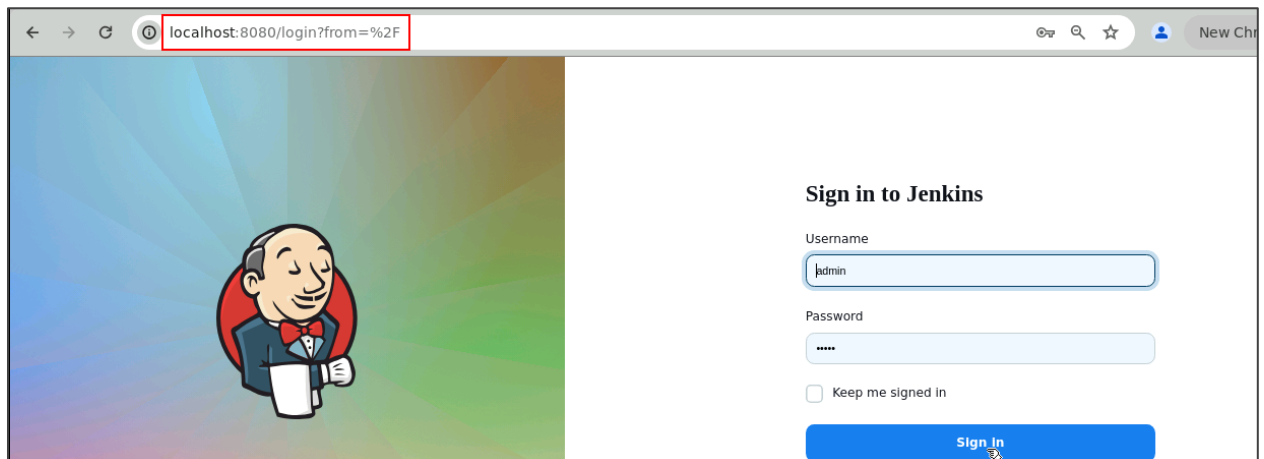
1.6 Click on **Commit changes**, provide a commit message, and then click **Commit changes** again to finalize them



A Jenkinsfile pipeline script is created within the project repository.

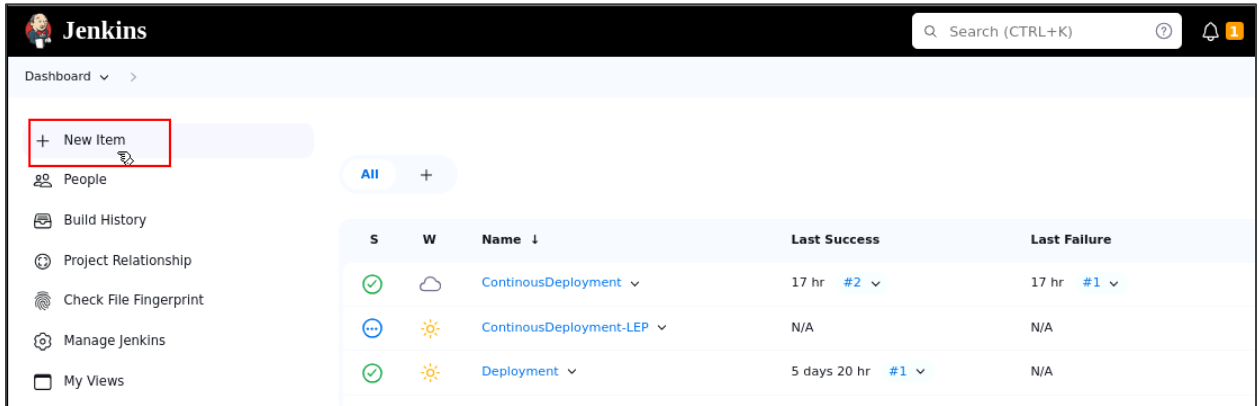
Step 2: Create the Jenkins pipeline job

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**.

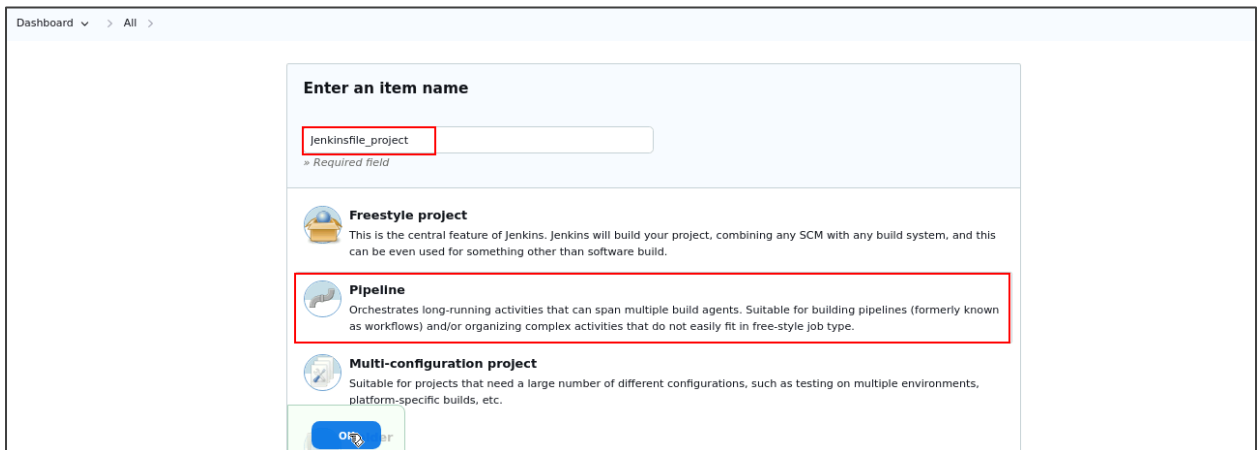
2.2 In the Jenkins dashboard, click on **New Item** to create a new Jenkins job



The screenshot shows the Jenkins dashboard. On the left sidebar, the 'New Item' button is highlighted with a red box. The main area displays a table of existing jobs. The table has columns for status (S), icon (W), Name, Last Success, and Last Failure.

S	W	Name ↓	Last Success	Last Failure
✓	☁	ContinuousDeployment ↓	17 hr #2 ↓	17 hr #1 ↓
...	☀	ContinuousDeployment-LEP ↓	N/A	N/A
✓	☀	Deployment ↓	5 days 20 hr #1 ↓	N/A

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

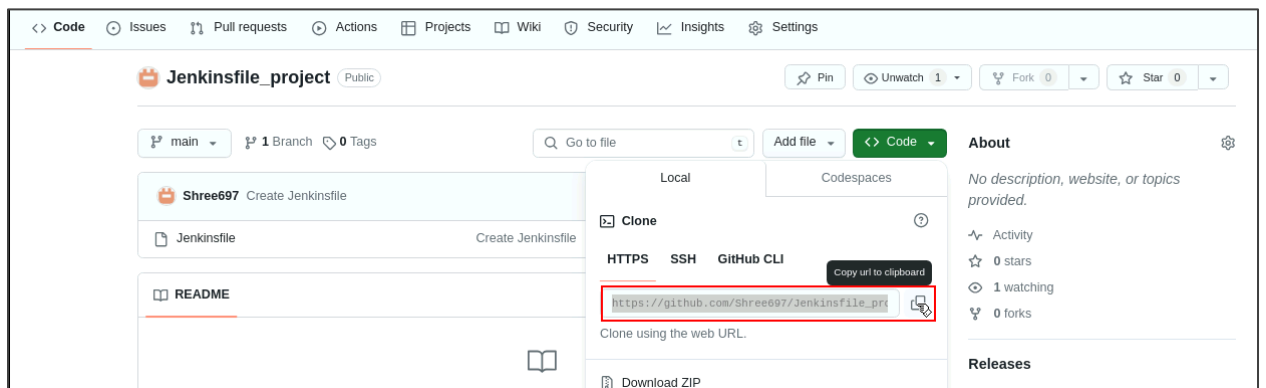


The screenshot shows the 'Enter an Item name' dialog box. The input field contains 'jenkinsfile_project' and is highlighted with a red box. Below the input field, three project types are listed: Freestyle project, Pipeline, and Multi-configuration project. The 'Pipeline' option is highlighted with a red box. At the bottom, there is an 'OK' button.

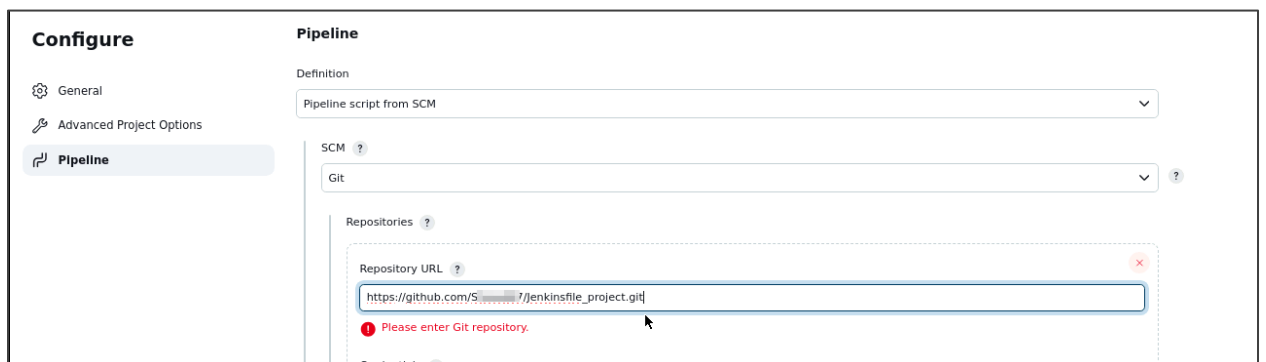
2.4 In the job configuration page, go to **Pipeline**, select **Pipeline script from SCM** as the pipeline definition, and choose **Git** as the SCM tool



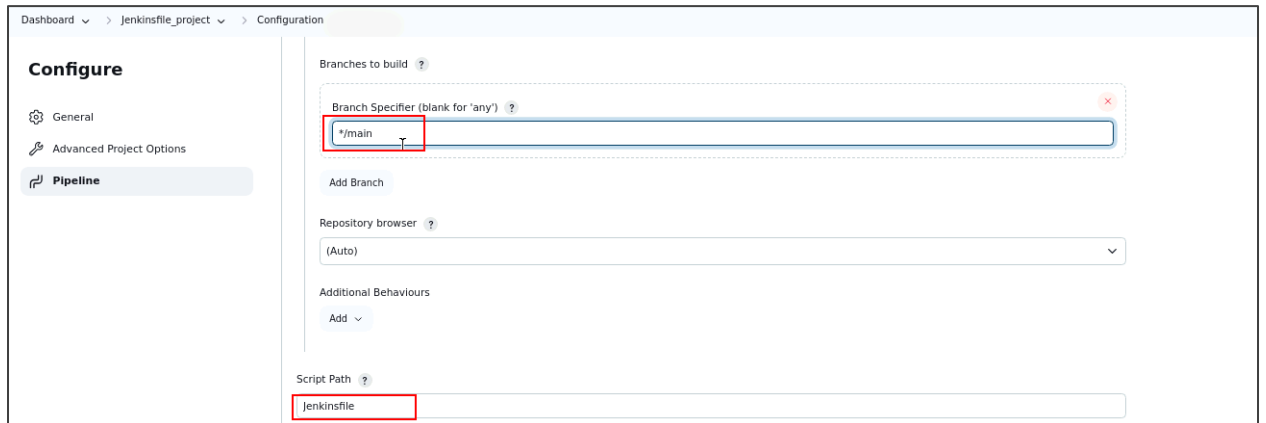
2.5 Go to the main project page in GitHub and copy the HTTPS URL



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

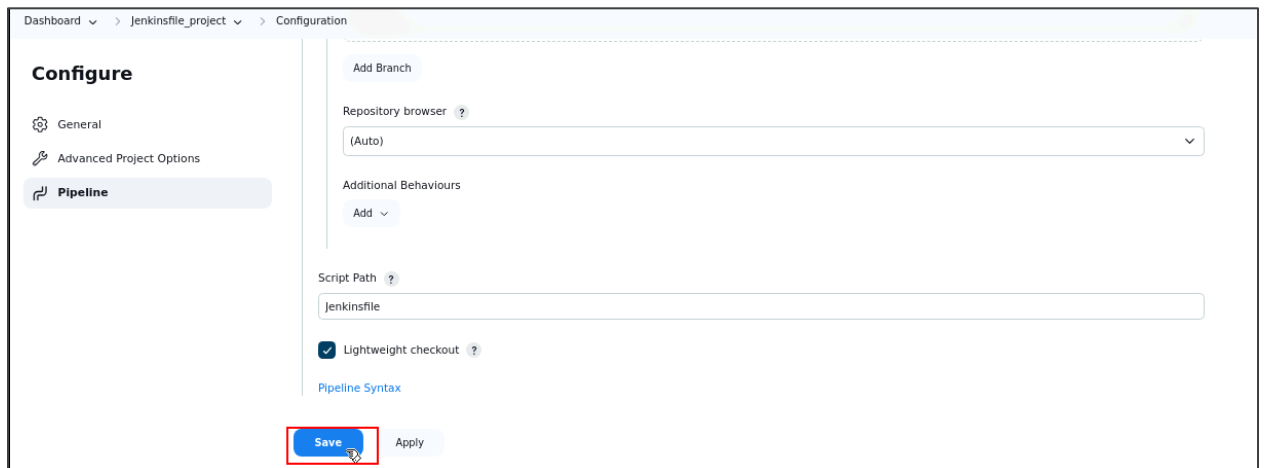


2.7 Scroll down, enter ***/main** in the **Branch Specifier**, and keep **Script Path** as **Jenkinsfile**



The screenshot shows the Jenkins Configuration page for a project named 'Jenkinsfile_project'. The left sidebar has a 'Configure' section with three tabs: 'General', 'Advanced Project Options', and 'Pipeline'. The 'Pipeline' tab is selected. The main content area is titled 'Configuration' and contains several sections. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' field with the value '*/main' entered, which is highlighted with a red box. Below this is an 'Add Branch' button. The 'Repository browser' section has a dropdown menu set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. The 'Script Path' section has a text field with the value 'Jenkinsfile', which is also highlighted with a red box.

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



The screenshot shows the same Jenkins Configuration page as before, but with additional options visible. The 'Script Path' field still contains 'Jenkinsfile'. Below it, the 'Lightweight checkout' checkbox is checked. There is a 'Pipeline Syntax' link. At the bottom of the configuration area, there are two buttons: 'Save' and 'Apply'. The 'Save' button is highlighted with a red box.

Step 3: Execute the Jenkins job

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

The screenshot shows the Jenkins web interface for a project named 'Jenkinsfile_project'. On the left sidebar, the 'Build Now' button is highlighted with a red rectangle. The main area shows the 'Stage View' with a message: 'No data available. This Pipeline has not yet run.' Below this is the 'Permalinks' section.

The screenshot shows the Jenkins web interface for the 'Jenkinsfile_project' after a successful build. The 'Status' is green. The 'Build History' section shows two builds: #2 (May 14, 2024, 8:48 AM) and #1 (May 14, 2024, 8:47 AM). The 'Stage View' table shows the following data:

	Declarative: Checkout SCM	Declarative: Tool Install	Clone repo	Compile Code	Test Code	Package Code
Average stage times: (Average full run time: ~30s)	409ms	164ms	703ms	5s	7s	8s
May 14 08:48	386ms	158ms	646ms	5s	7s	8s
May 14 08:47	432ms	170ms	761ms	6s	7s	8s

On the right, the 'Test Result Trend' graph shows a green line indicating 'Passed' results.

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

Dashboard > Jenkinsfile_project > Stage View

Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax

Build History trend

Filter builds...

Atom feed for all
Atom feed for failures

Latest Test Result (no failures)

Permalinks

- Last build (#2), 1 min 25 sec ago
- Last stable build (#2), 1 min 32 sec ago
- Last successful build (#2), 1 min 32 sec ago
- Last completed build (#2), 1 min 32 sec ago

Dashboard > Jenkinsfile_project > #2

Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#2'
Git Build Data
Git Build Data
Test Result
Restart from Stage
Replay
Pipeline Steps
Workspaces

Console Output

Started by user admin
Obtained Jenkinsfile from git https://github.com/Shree697/Jenkinsfile_project.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Jenkinsfile_project2
[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
Cloning the remote Git repository
Cloning repository https://github.com/Shree697/Jenkinsfile_project.git
> git init /var/lib/jenkins/workspace/Jenkinsfile_project2 # timeout=10
Fetching upstream changes from https://github.com/Shree697/Jenkinsfile_project.git
> git --version # timeout=10
> git --version # 'git version 2.43.2'
> git fetch --tags --force --progress -- https://github.com/Shree697/Jenkinsfile_project.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/Shree697/Jenkinsfile_project.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch

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.