

Lesson 08 Demo 03

Configuring Multibranch Jenkins Pipeline Job

Objective: To configure a Multibranch Jenkins pipeline job to facilitate managing multiple projects within a single job setup

Tools required: Jenkins, Git, and GitHub

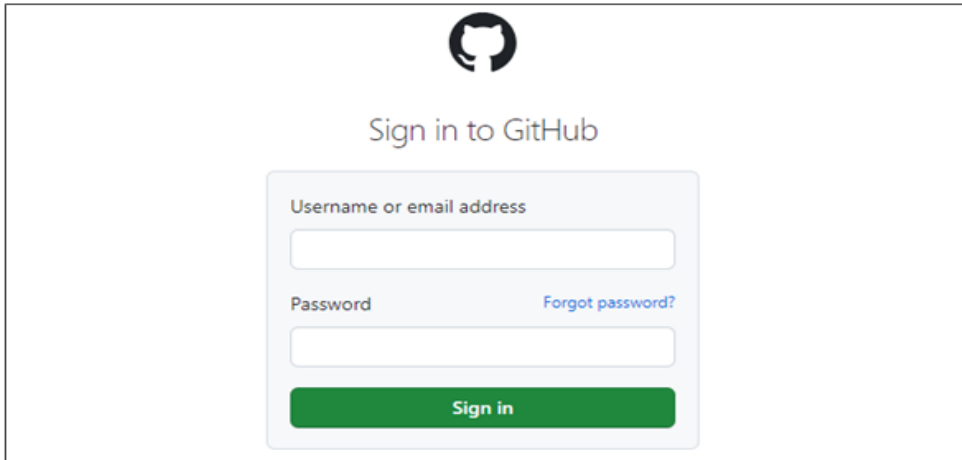
Prerequisites: None

Steps to be followed:

1. Create a Git repository
2. Push the code file into the Git repository
3. Create a Multibranch Pipeline

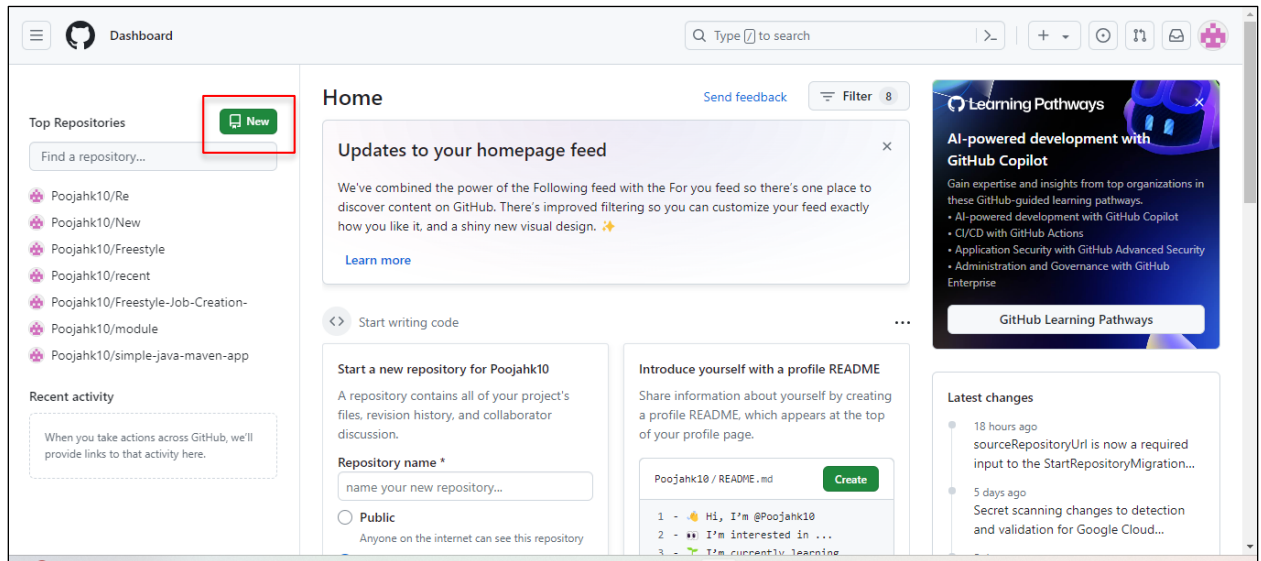
Step 1: Create a Git repository

1.1 Open the browser in your lab, go to **github.com**, and sign in to your account

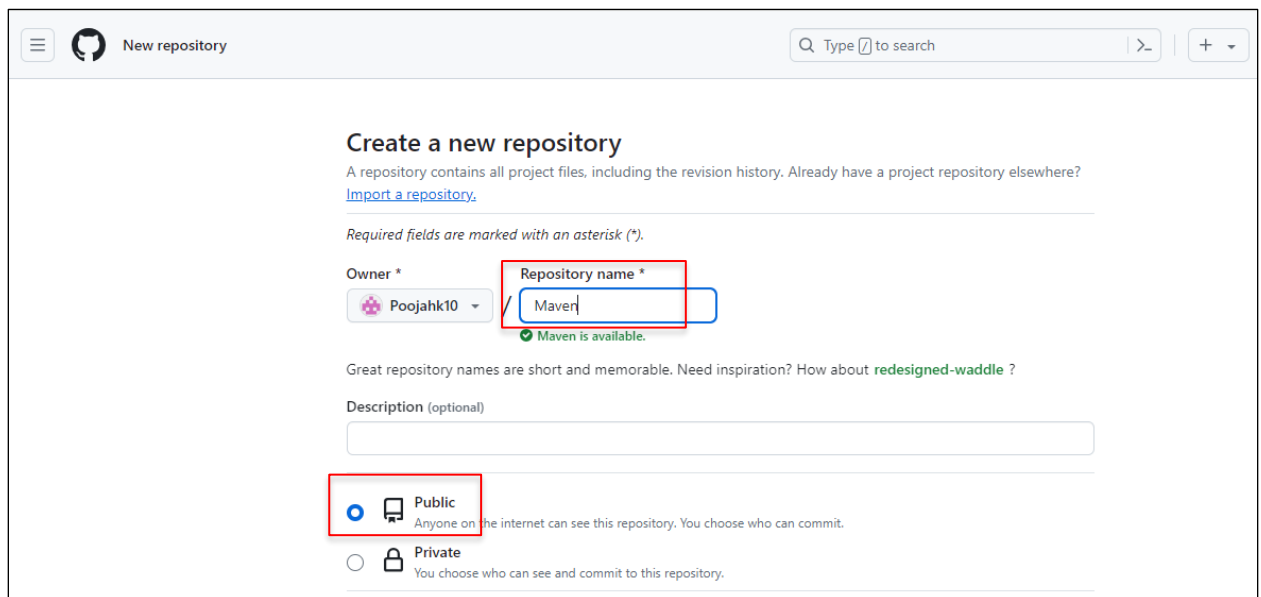
A screenshot of the GitHub sign-in page. At the top center is the GitHub logo (an octocat). Below it, the text "Sign in to GitHub" is displayed. Underneath is a light gray rounded rectangle containing the sign-in fields. The first field is labeled "Username or email address" and has a white input box. The second field is labeled "Password" and has a white input box. To the right of the password field is a blue link that says "Forgot password?". At the bottom of the form is a green button with the text "Sign in" in white.

Note: If you do not have a GitHub account, visit the official website at <https://github.com/signup> and create a new account

1.2 Click on **New** as shown in the screenshot below:



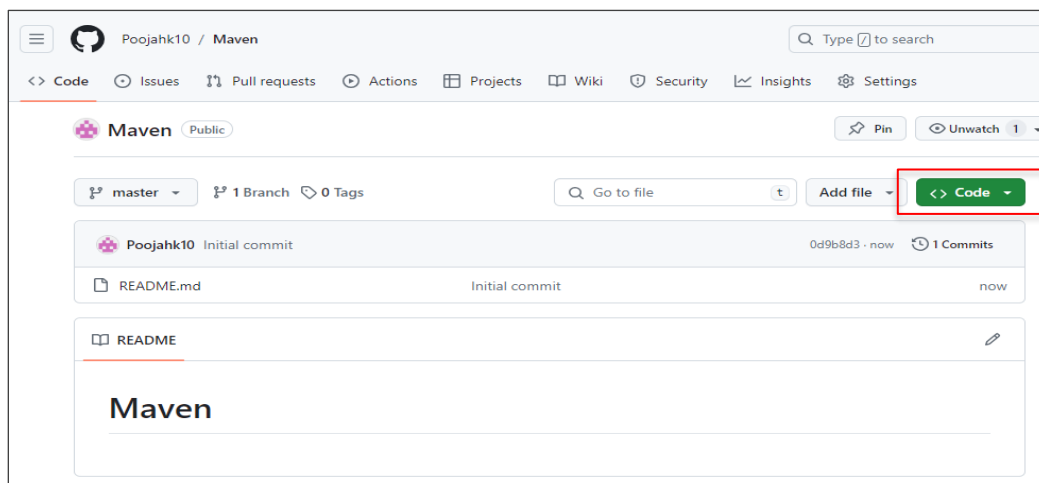
1.3 Enter a desired name for your repository and choose **Public** as shown in the screenshot below:



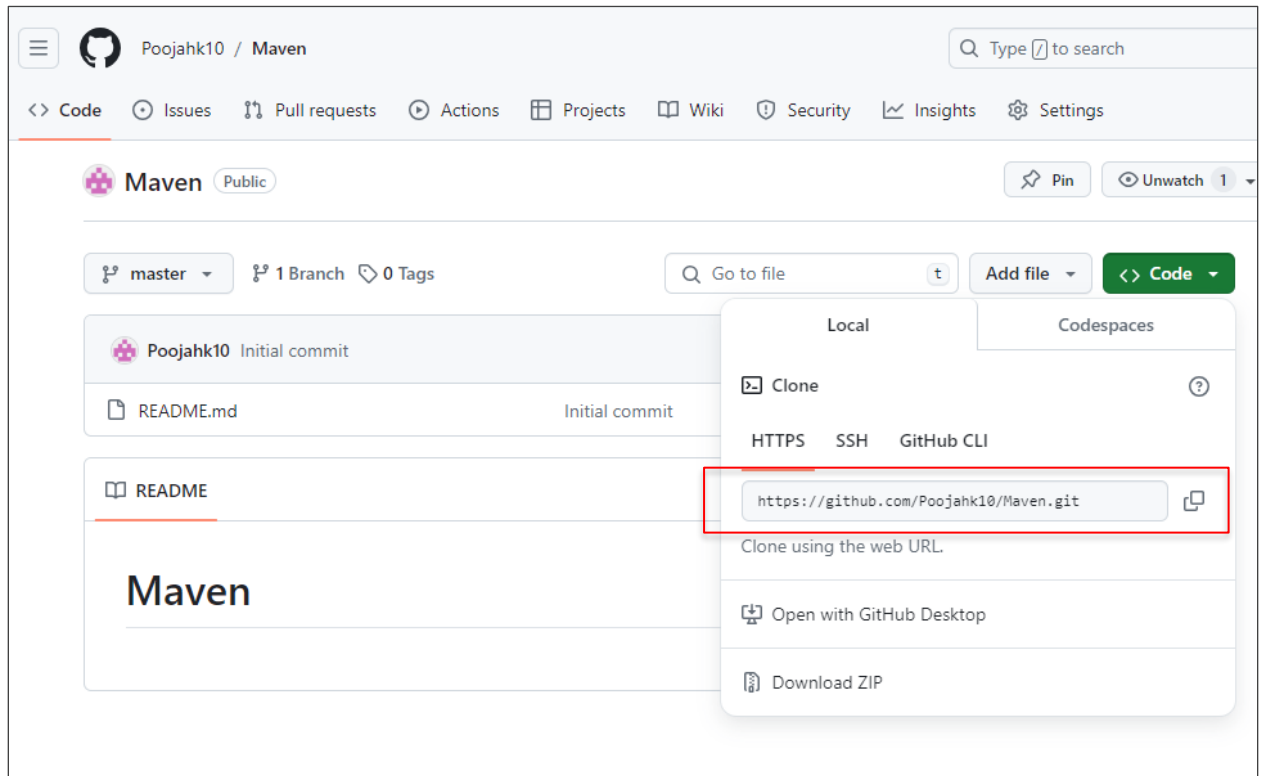
1.4 Click on **Add a README file** and then click on **Create repository** as shown in the screenshot below:

The screenshot shows the GitHub repository creation interface. At the top, there are two radio button options: 'Public' (selected) and 'Private'. Below this, a section titled 'Initialize this repository with:' contains a checkbox labeled 'Add a README file', which is checked and highlighted with a red box. Further down, there are sections for 'Add .gitignore' (with a dropdown set to 'None') and 'Choose a license' (with a dropdown set to 'None'). At the bottom right, a green button labeled 'Create repository' is highlighted with a red box. A note at the bottom states: 'You are creating a public repository in your personal account.'

1.5 Click on **Code** as shown in the screenshot below:



1.6 Copy the repository URL as shown in the screenshot below:



Step 2: Push the code file into the Git repository

2.1 Open the Linux terminal in your lab and clone the repository using the below command:
git clone RepositoryURL

```
syedsharozsimpl@ip-172-31-40-171:~$ git clone https://github.com/Poojahk10/Maven.git
Cloning into 'Maven'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
syedsharozsimpl@ip-172-31-40-171:~$
```

2.2 Navigate inside the repository that you had created using the below command:
cd RepositoryName

```
syedsharozsimpl@ip-172-31-40-171:~$ cd Maven
syedsharozsimpl@ip-172-31-40-171:~/Maven$
```

2.3 Initialize the Git using the below command:

git init

```
syedsharozsimpl@ip-172-31-40-171:~/Maven$ git init
Reinitialized existing Git repository in /home/syedsharozsimpl/Maven/.git/
syedsharozsimpl@ip-172-31-40-171:~/Maven$ █
```

2.4 Create a file using the below command:

nano demofile

```
syedsharozsimpl@ip-172-31-40-171:~/Maven$ nano demofile █
```

2.5 Paste the below pipeline script inside the file as shown in the screenshot below:

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        // Checkout your source code from version control
        git 'https://github.com/your/repository.git'
      }
    }
    stage('Build') {
      steps {
        // Use Maven to build your project
        sh 'mvn clean package'
      }
    }
    stage('Test') {
      steps {
        // Run tests if applicable
        sh 'mvn test'
      }
    }
    stage('Deploy') {
      steps {
        // Deploy your artifact, if necessary
        // Example: sh 'mvn deploy'
      }
    }
  }
}
```

```

post {
  success {
    // This block will be executed if the pipeline runs successfully
    echo 'Pipeline executed successfully!'
  }
  failure {
    // This block will be executed if the pipeline fails
    echo 'Pipeline failed!'
  }
}
}

```

Note: Ensure you provide your Git repository URL on line7, save, and exit the page by clicking on **ctrl+S** to save and **ctrl+X** to exit

The screenshot shows a terminal window with the title 'syedsharozsimpl@ip-172-31-40-171: ~/Maven'. The window contains the GNU nano 6.2 editor with a file named 'demofile *'. The editor displays a Jenkins pipeline configuration. The configuration starts with 'agent any', followed by a 'stages' block containing four stages: 'Checkout' (using 'git' to checkout from a repository), 'Build' (using 'sh' to run 'mvn clean package'), 'Test' (using 'sh' to run 'mvn test'), and 'Deploy'. The bottom of the window shows the nano editor's command palette with various shortcuts like ^G Help, ^X Exit, ^O Write Out, etc.

```

syedsharozsimpl@ip-172-31-40-171: ~/Maven
GNU nano 6.2 demofile *
pipeline {
  agent any

  stages {
    stage('Checkout') {
      steps {
        // Checkout your source code from version control
        git 'https://github.com/your/repository.git'
      }
    }
    stage('Build') {
      steps {
        // Use Maven to build your project
        sh 'mvn clean package'
      }
    }
    stage('Test') {
      steps {
        // Run tests if applicable
        sh 'mvn test'
      }
    }
    stage('Deploy') {
      steps {

```

2.6 Stage and commit the changes using the below commands:

git add .

git commit -m "initial commit"

```

syedsharozsimpl@ip-172-31-40-171:~/Maven$ git add .
syedsharozsimpl@ip-172-31-40-171:~/Maven$ git commit -m "initial commit"
[master a0e2b3a] initial commit
1 file changed, 41 insertions(+)
create mode 100644 demofile
syedsharozsimpl@ip-172-31-40-171:~/Maven$ █

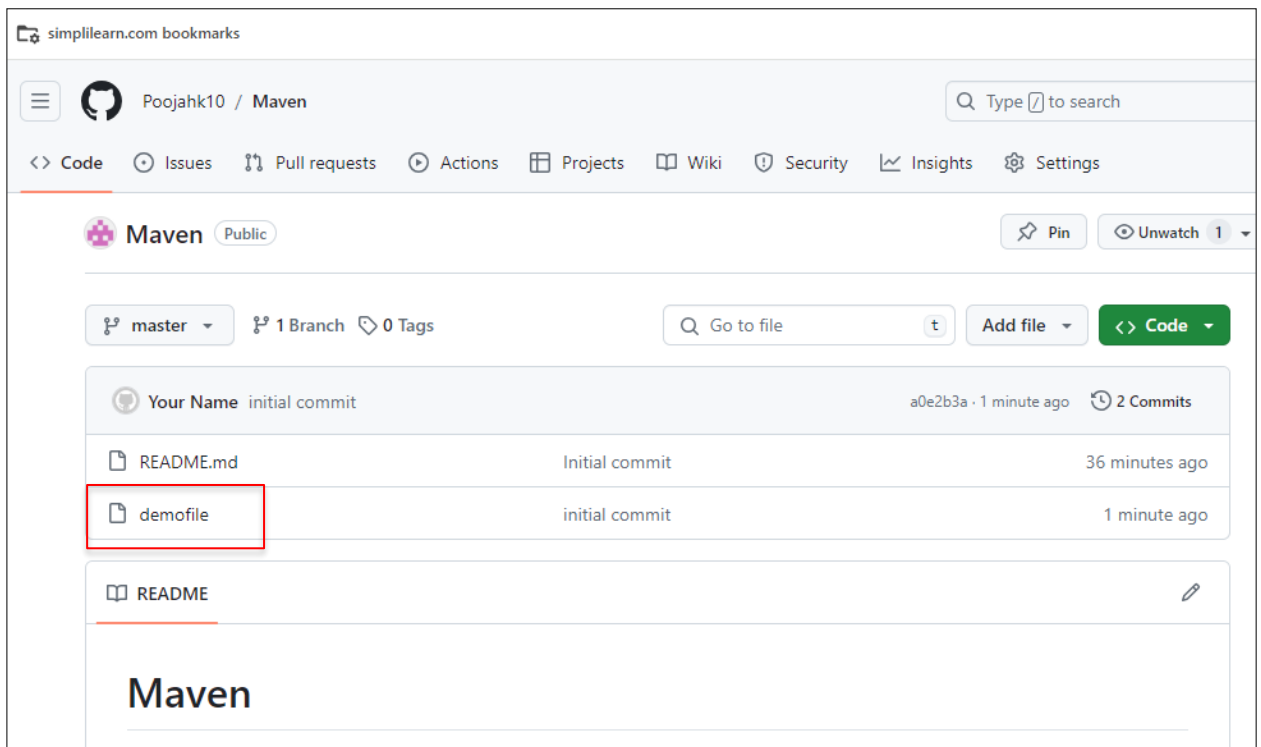
```

2.7 Push the file to the Git repository using the below command:

git push

```
syedsharozsimpl@ip-172-31-40-171:~/Maven$ git push
Username for 'https://github.com': Poojahk10
Password for 'https://Poojahk10@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 614 bytes | 614.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Poojahk10/Maven.git
    0d9b8d3..a0e2b3a  master -> master
syedsharozsimpl@ip-172-31-40-171:~/Maven$
```

2.8 Navigate to your Git repository to check for the file that is pushed:

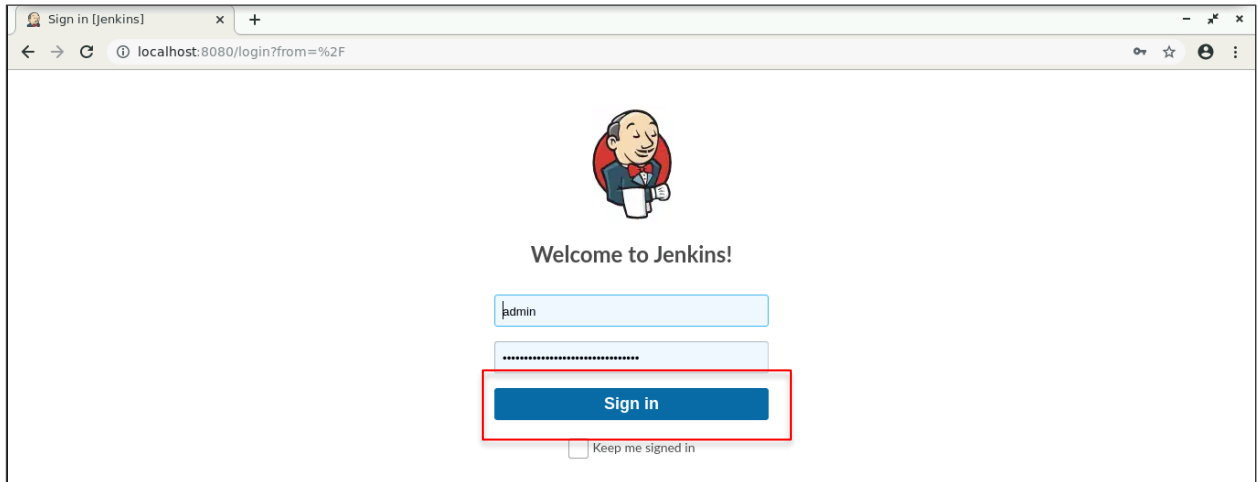


The screenshot shows the GitHub interface for a repository named 'Maven' by user 'Poojahk10'. The repository is public. The commit history shows two commits: 'Your Name initial commit' (a0e2b3a, 1 minute ago) and 'Your Name initial commit' (a0e2b3a, 1 minute ago). The commit history table lists the files changed in each commit: 'README.md' and 'demofile'. The 'demofile' is highlighted with a red box. Below the commit history, there is a 'README' section with the title 'Maven'.

File	Commit	Time
README.md	Initial commit	36 minutes ago
demofile	initial commit	1 minute ago

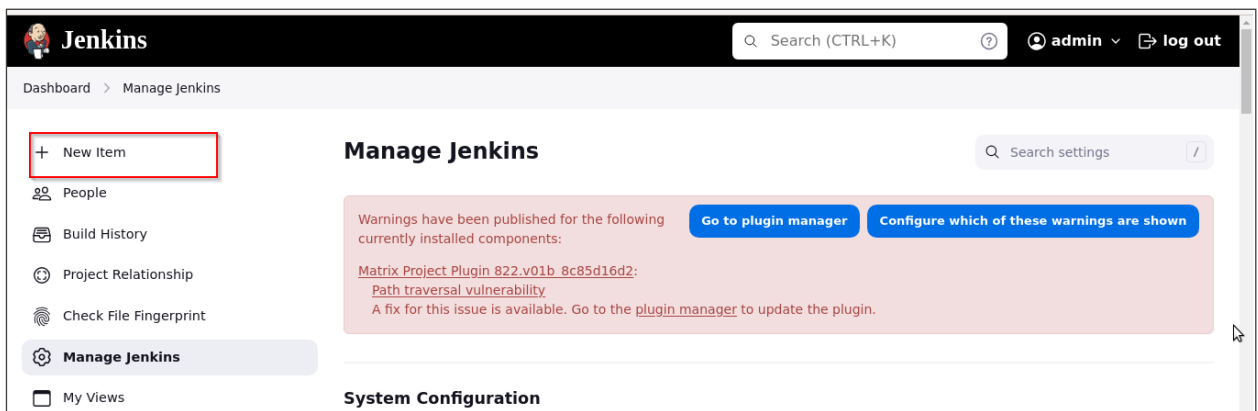
Step 3: Create a Multibranch Pipeline

3.1 Open the browser, go to the Jenkins **Dashboard** by typing **localhost:8080** in your browser, provide the credentials, and click the **Sign in** button

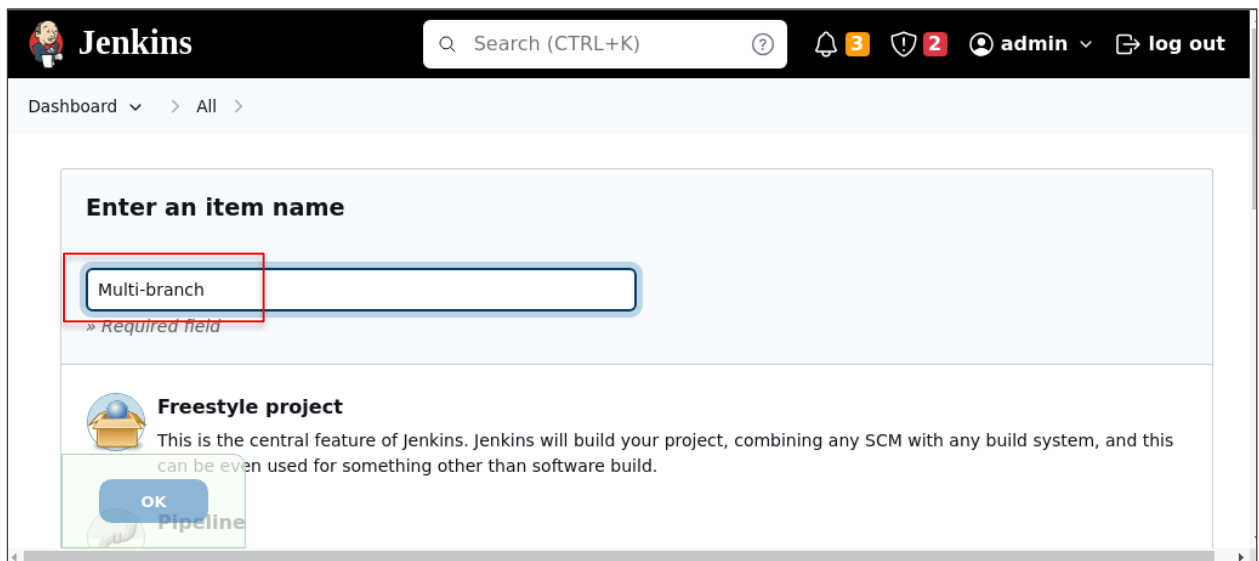


Note: Use the given credentials to access Jenkins in the lab: **Username** is admin and **Password** is Root123\$

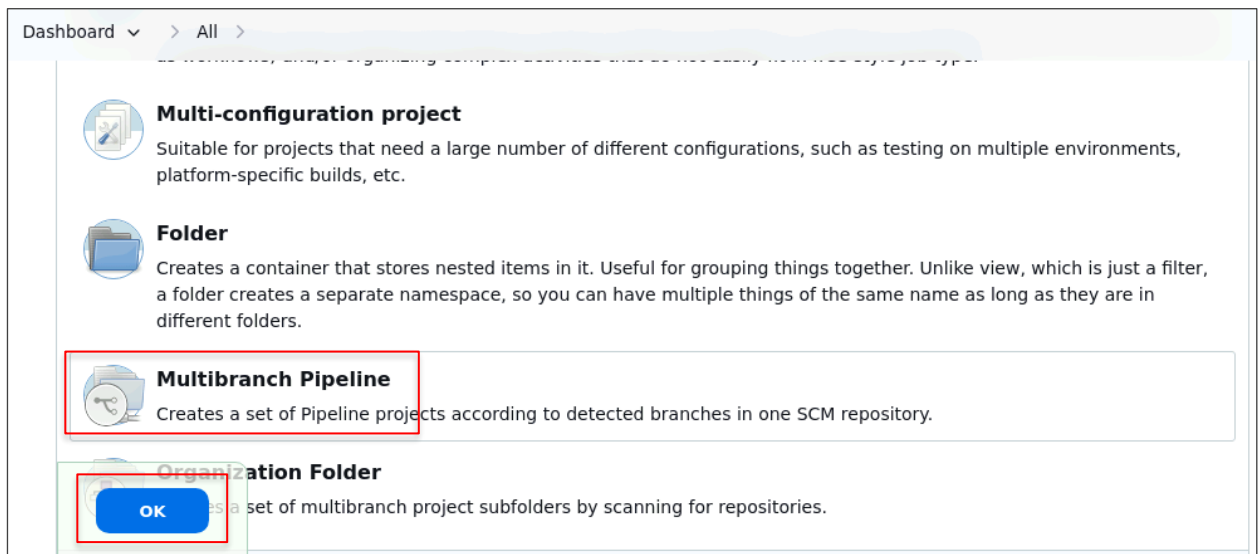
3.2 Click on **New Item** as show in the screenshot below:



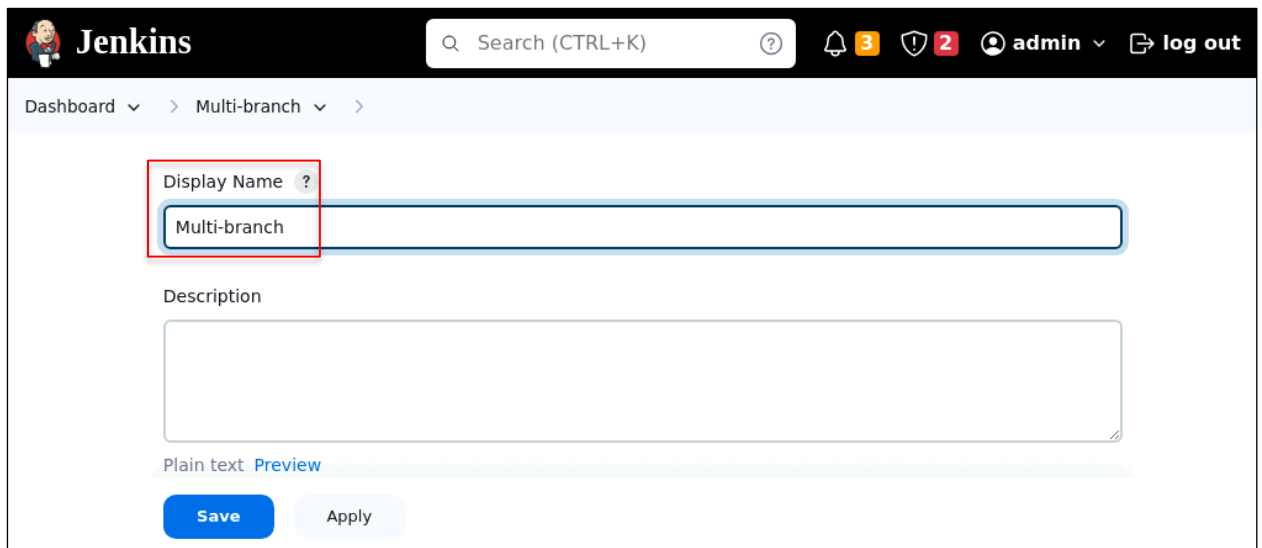
3.3 Enter a desired name for the project as shown in the screenshot below:



3.4 Scroll down to **Multibranch Pipeline**, click on it, and then click on the **OK** button as shown in the screenshot below:

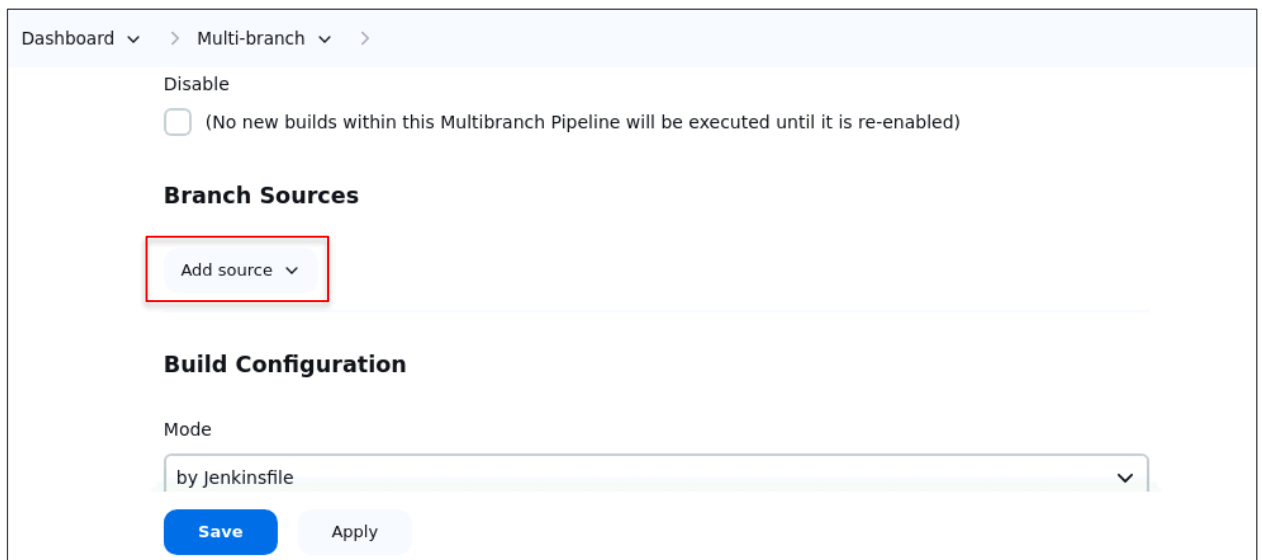


3.5 Enter a desired name for **Display Name** as shown in the screenshot below:



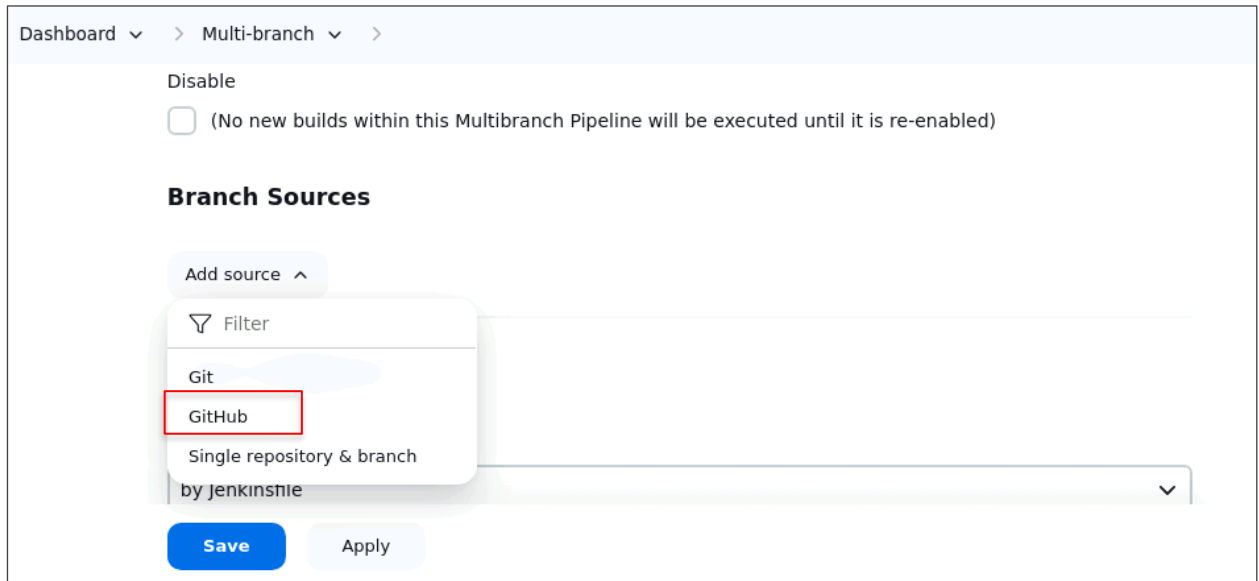
The screenshot shows the Jenkins web interface. At the top, there's a navigation bar with the Jenkins logo, a search bar, and user information. Below the navigation bar, the breadcrumb trail reads "Dashboard > Multi-branch >". The main content area is titled "Display Name" with a help icon. A text input field contains the value "Multi-branch". Below this is a "Description" section with a large text area. At the bottom, there are "Save" and "Apply" buttons.

3.6 Scroll down to **Branch Sources** and then click on **Add source** as shown in the screenshot below:

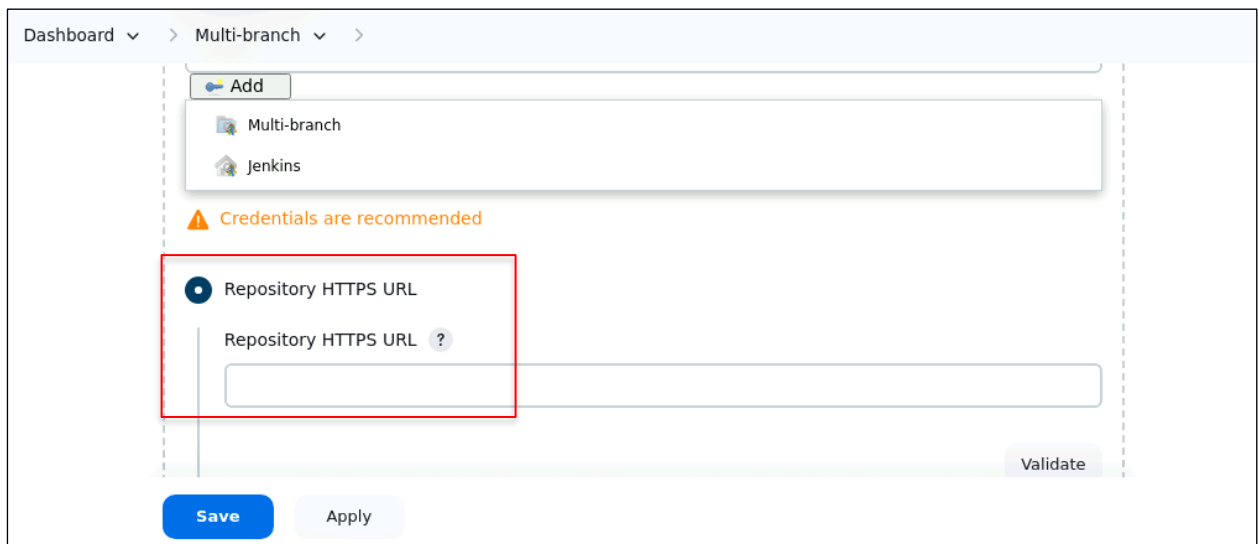


The screenshot shows the Jenkins web interface, scrolled down to the "Branch Sources" section. Above this section, there's a "Disable" checkbox and a note: "(No new builds within this Multibranch Pipeline will be executed until it is re-enabled)". The "Branch Sources" section has a red box around the "Add source" button. Below this is the "Build Configuration" section, which includes a "Mode" dropdown menu currently set to "by Jenkinsfile". At the bottom, there are "Save" and "Apply" buttons.

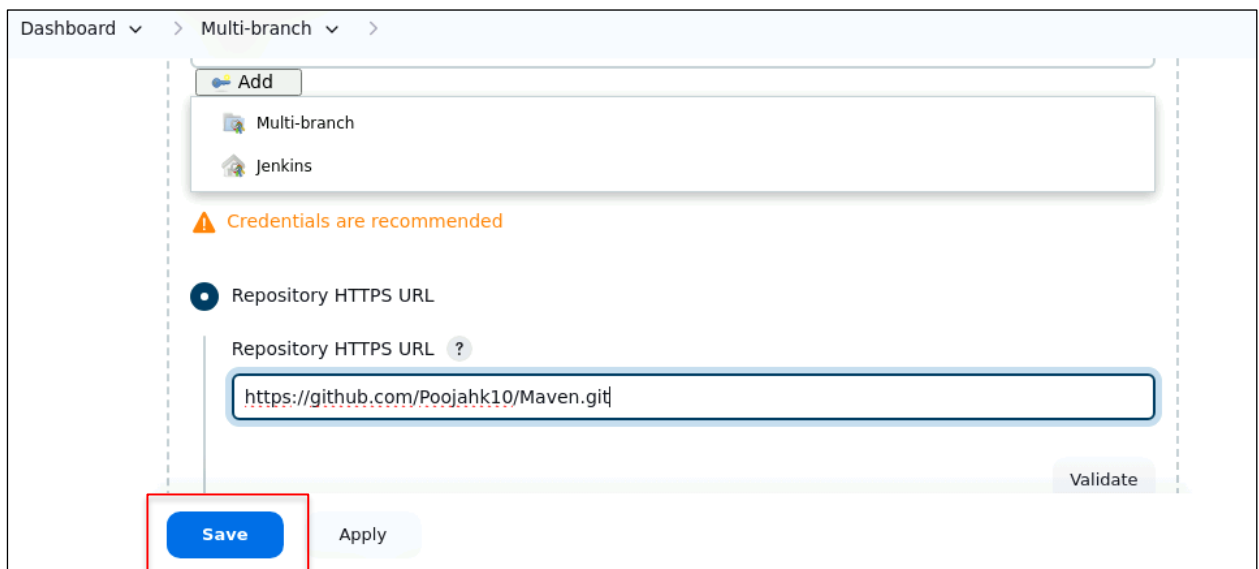
3.7 Click on **GitHub** as shown in the screenshot below:



3.8 Scroll down to **Repository HTTPS URL** as shown in the screenshot below:

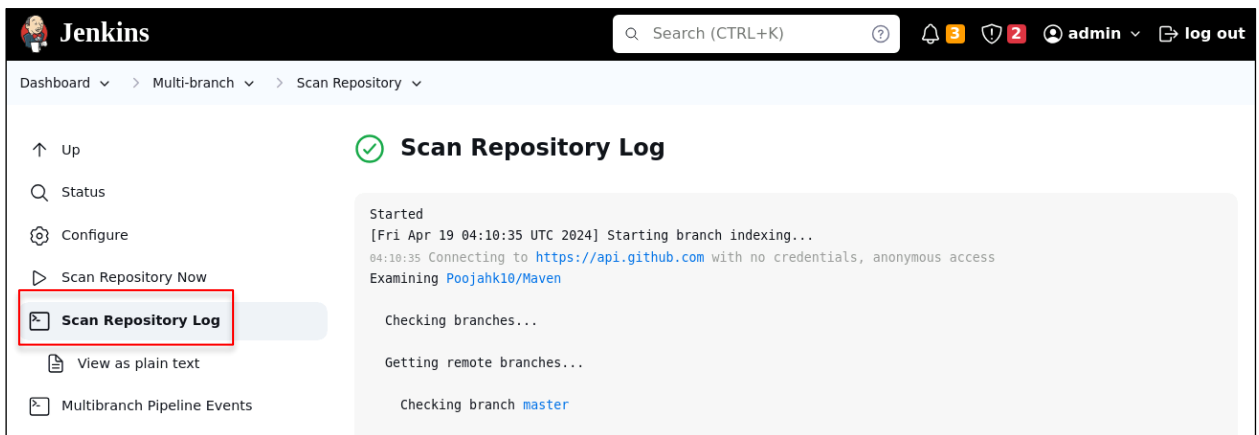


3.9 Enter the repository URL and then click on the **Save** button as shown in the screenshot below:



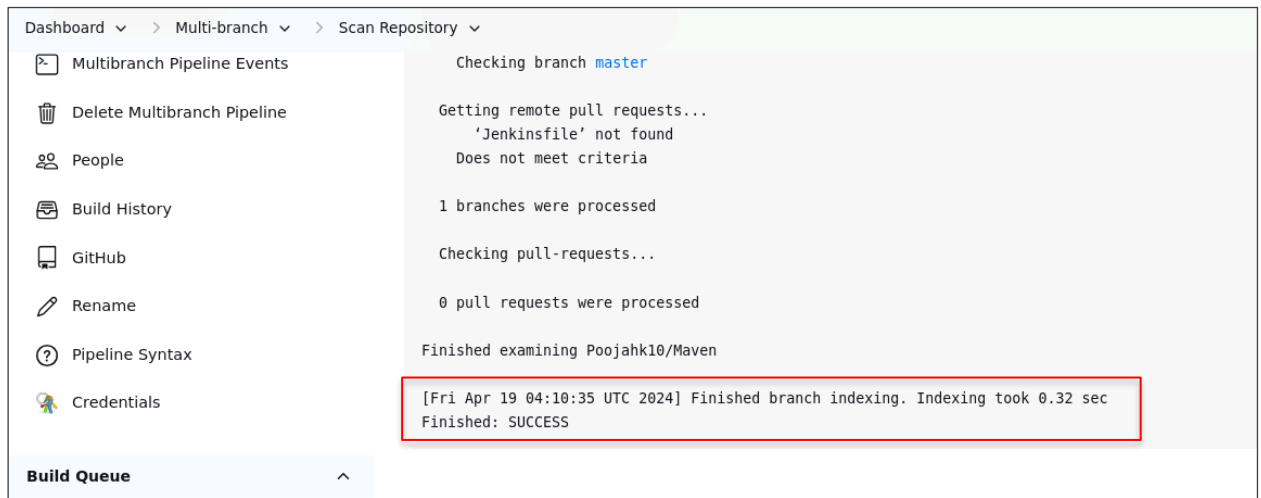
The screenshot shows the Jenkins 'Multi-branch' configuration page. At the top, there's a breadcrumb trail: 'Dashboard > Multi-branch >'. Below this, there's an 'Add' button and a list of repository types: 'Multi-branch' and 'Jenkins'. A warning message states 'Credentials are recommended'. The 'Repository HTTPS URL' section is active, showing a text input field with the URL 'https://github.com/Poojahk10/Maven.git'. To the right of the input field is a 'Validate' button. At the bottom, there are two buttons: 'Save' (highlighted with a red box) and 'Apply'.

3.10 Click on **Scan Repository Log** as shown in the screenshot below:



The screenshot shows the Jenkins 'Scan Repository Log' page. The breadcrumb trail is 'Dashboard > Multi-branch > Scan Repository >'. On the left sidebar, there are several links: 'Up', 'Status', 'Configure', 'Scan Repository Now', 'Scan Repository Log' (highlighted with a red box), 'View as plain text', and 'Multibranch Pipeline Events'. The main content area is titled 'Scan Repository Log' with a green checkmark icon. It displays a log of the scanning process, starting with 'Started' and '[Fri Apr 19 04:10:35 UTC 2024] Starting branch indexing...'. The log continues with '04:10:35 Connecting to https://api.github.com with no credentials, anonymous access', 'Examining Poojahk10/Maven', 'Checking branches...', 'Getting remote branches...', and 'Checking branch master'.

3.11 Scroll down to verify that the status is displayed as **SUCCESS** as shown in the screenshot below:



By following these steps, you have successfully configured a Multibranch Jenkins pipeline job to facilitate managing multiple projects within a single job setup.