

## Lesson 08 Demo 02

### Setting up Jenkins Pipeline Job from Git

**Objective:** To set up a Jenkins pipeline job from Git version control system to enable automated CI for building, testing, and potentially deploying software upon code changes

**Tools required:** Jenkins, Git and Linux

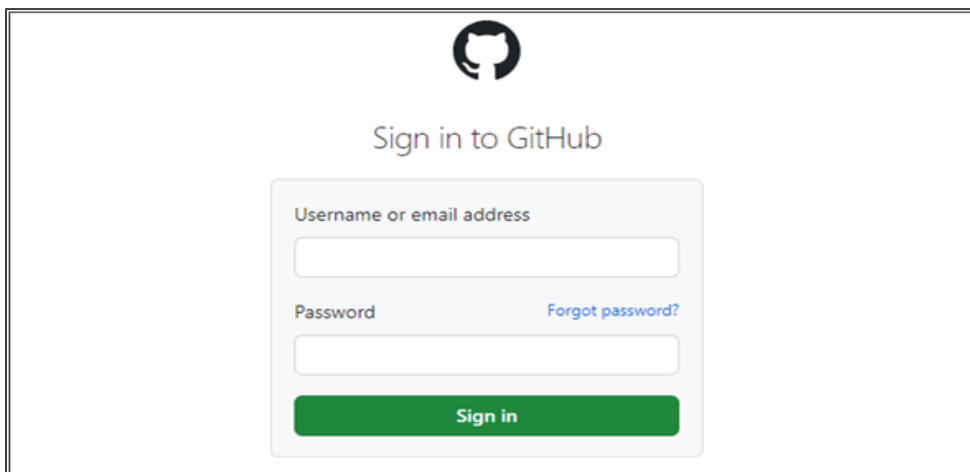
**Prerequisites:** None

Steps to be followed:

1. Create a Git repository
2. Push the pipeline script into the Git repository
3. Create a pipeline script-based freestyle job

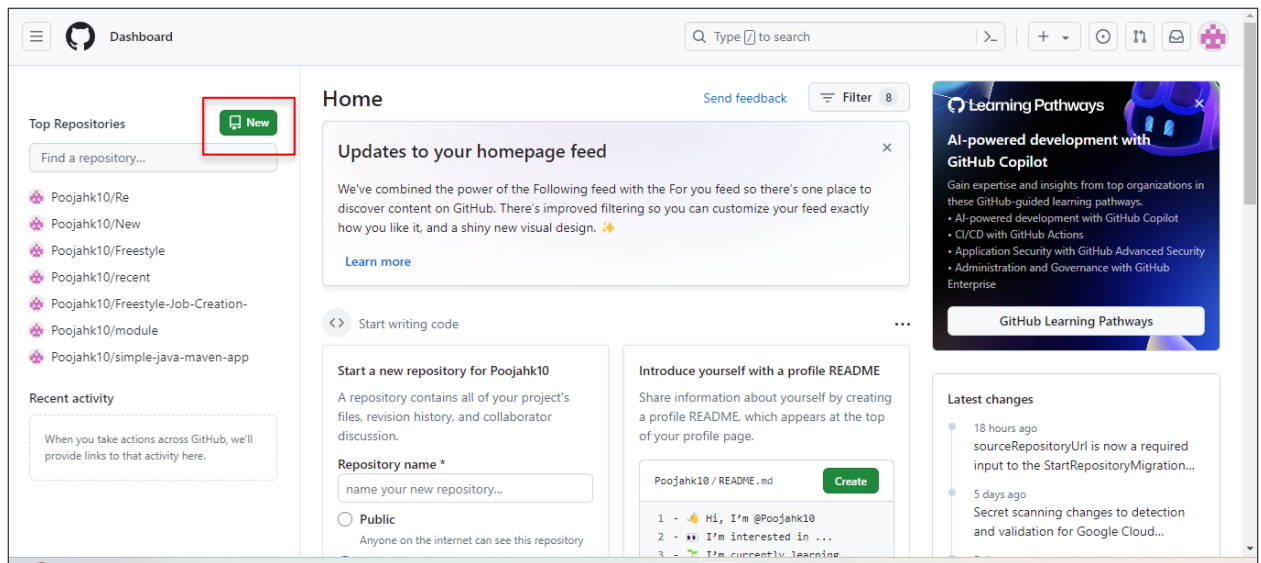
#### Step 1: Create a Git repository

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

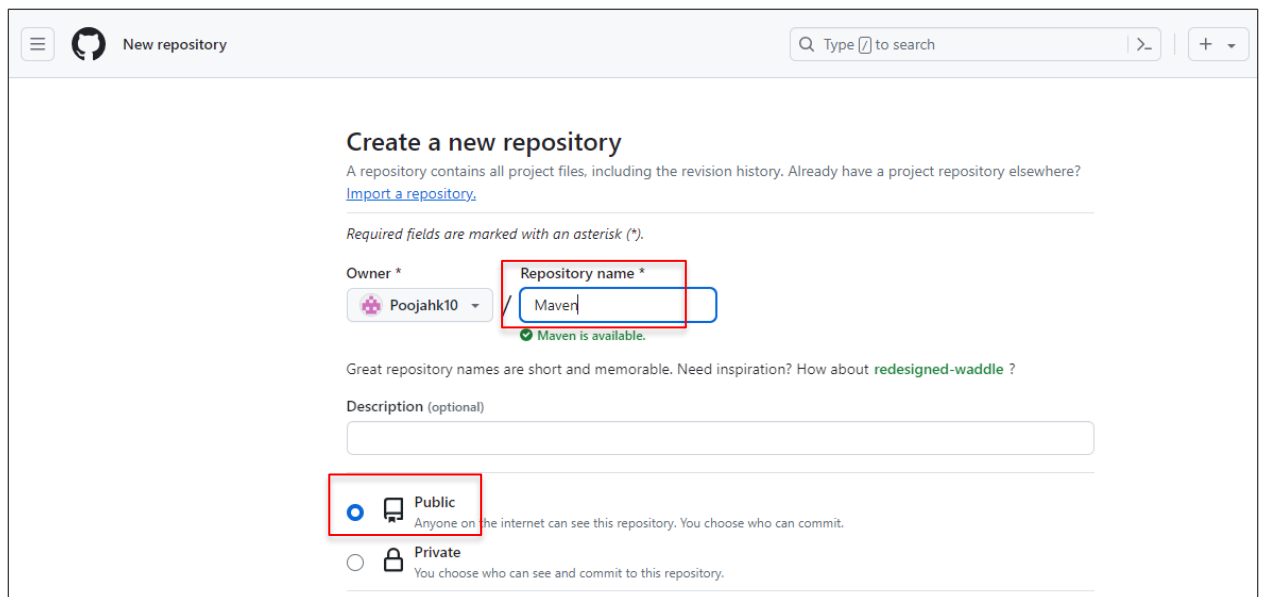


**Note:** If you do not have a GitHub account, visit the official website at <https://github.com/signup> and create a new account

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



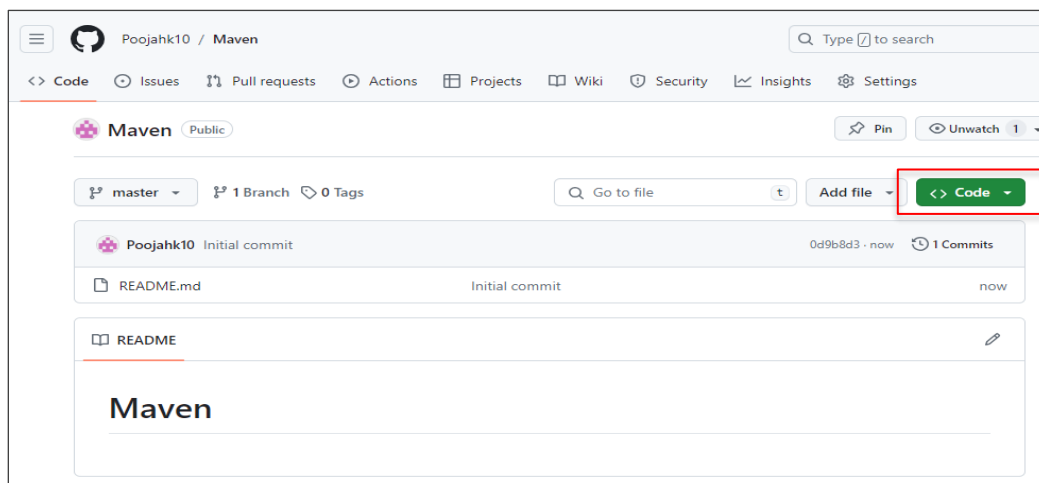
1.3 Enter a desired name for your repository and choose **Public** as shown in the screenshot below:



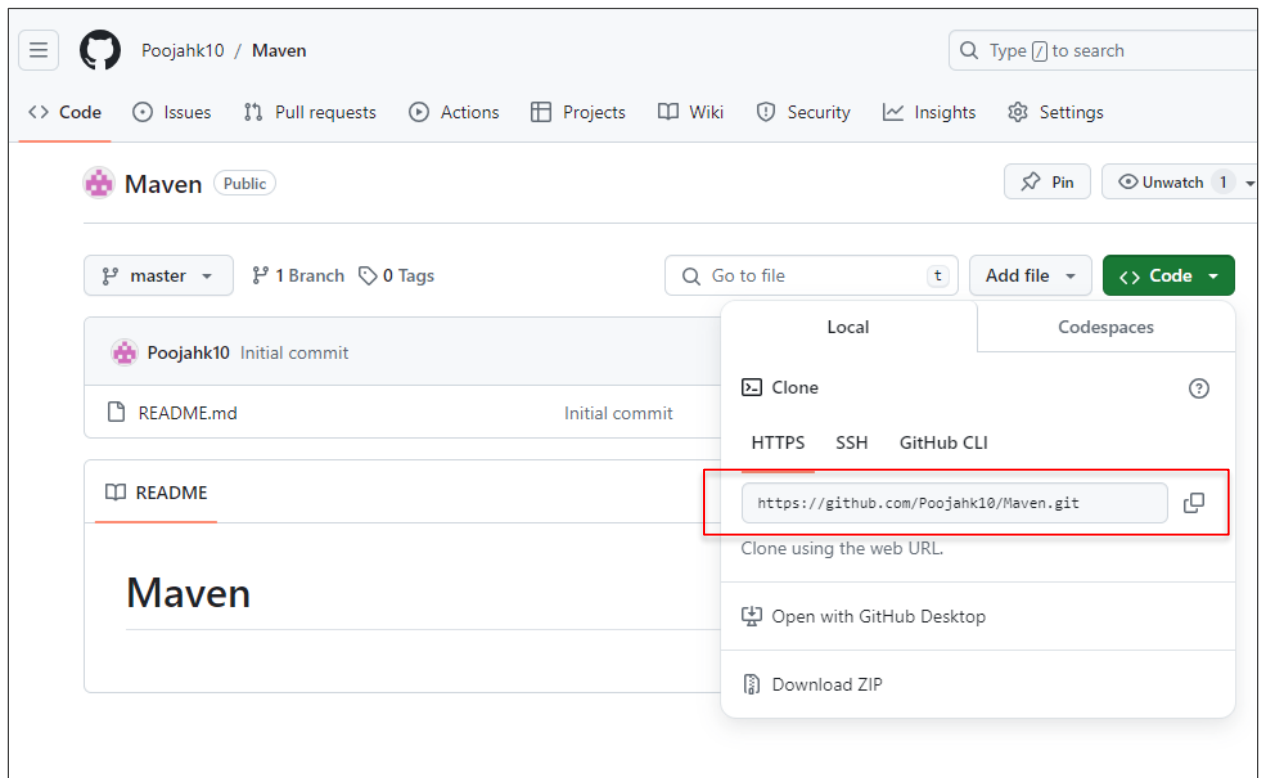
1.4 Click on **Add a README file** and then click on **Create repository** as shown in the screenshot below:

The screenshot shows the GitHub repository creation interface. At the top, there are two radio buttons for repository visibility: '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 pipeline script 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 editor is GNU nano 6.2, editing a file named 'demofile \*'. The content of the file is a Jenkins pipeline configuration:

```

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 {

```

The bottom of the terminal shows the nano editor's command palette with various shortcuts like ^G Help, ^X Exit, ^O Write Out, etc.

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$ █

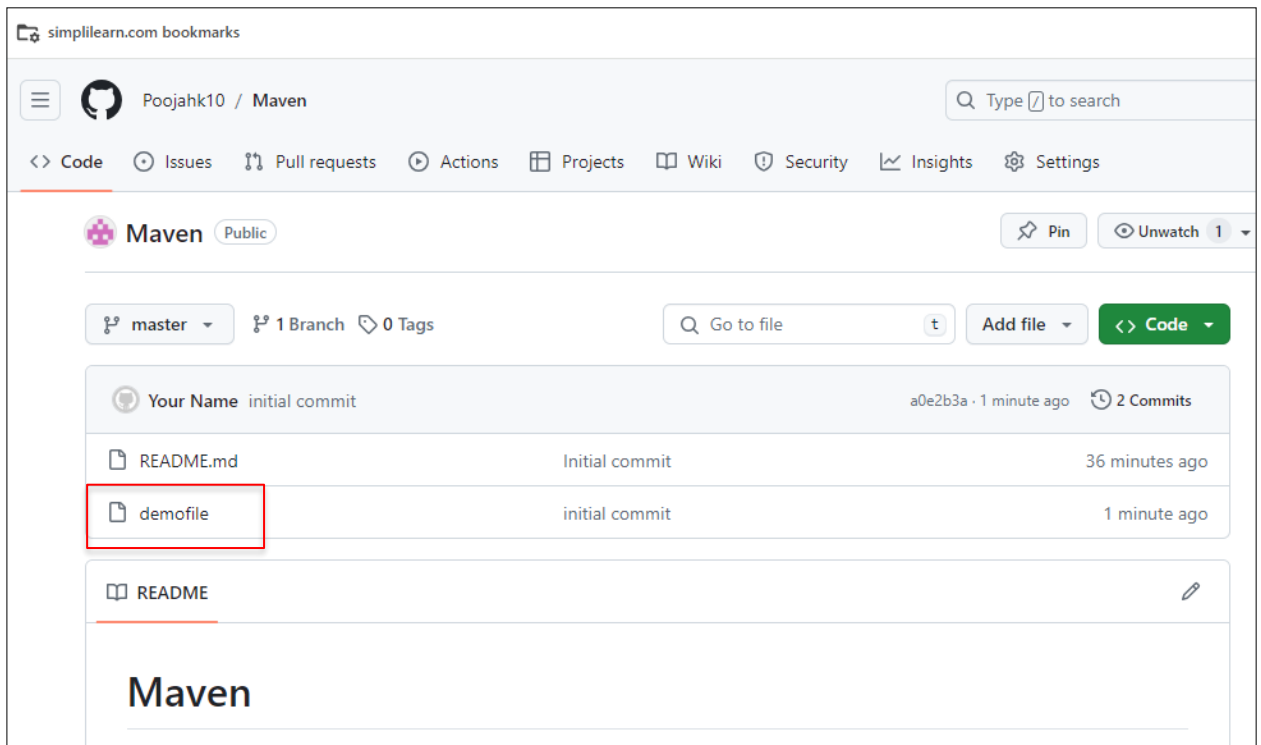
```

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

**git push**

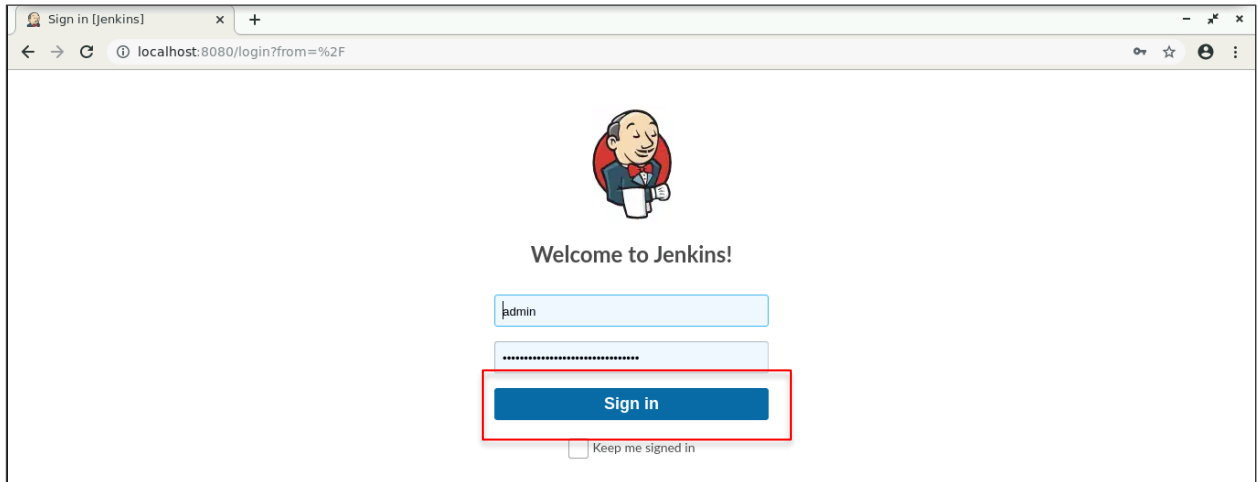
```
syedsharozsimpl@ip-172-31-40-171:~/Maven$ git push
Username for 'https://github.com': Poojahk10
Password for 'https://Poojahk10@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 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 as shown in the screenshot below:



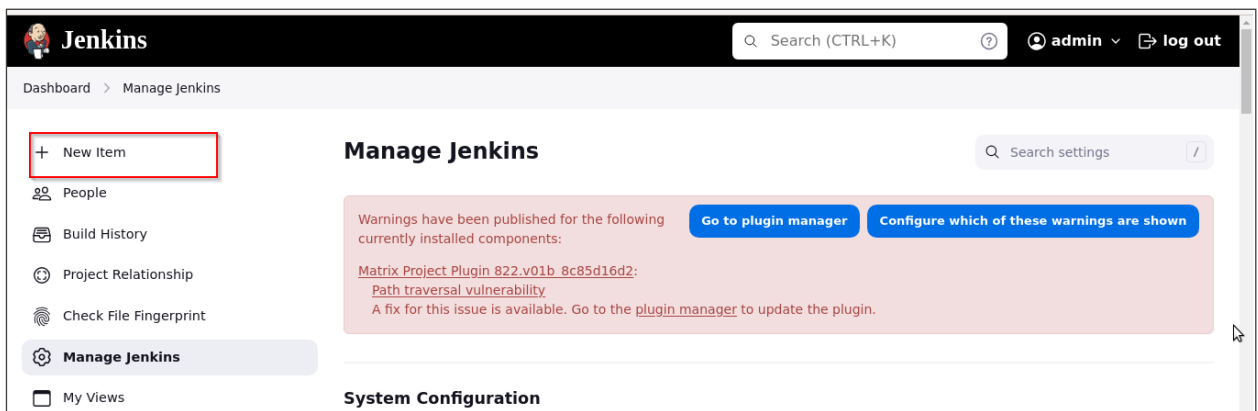
### Step 3: Create a pipeline script-based freestyle job

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



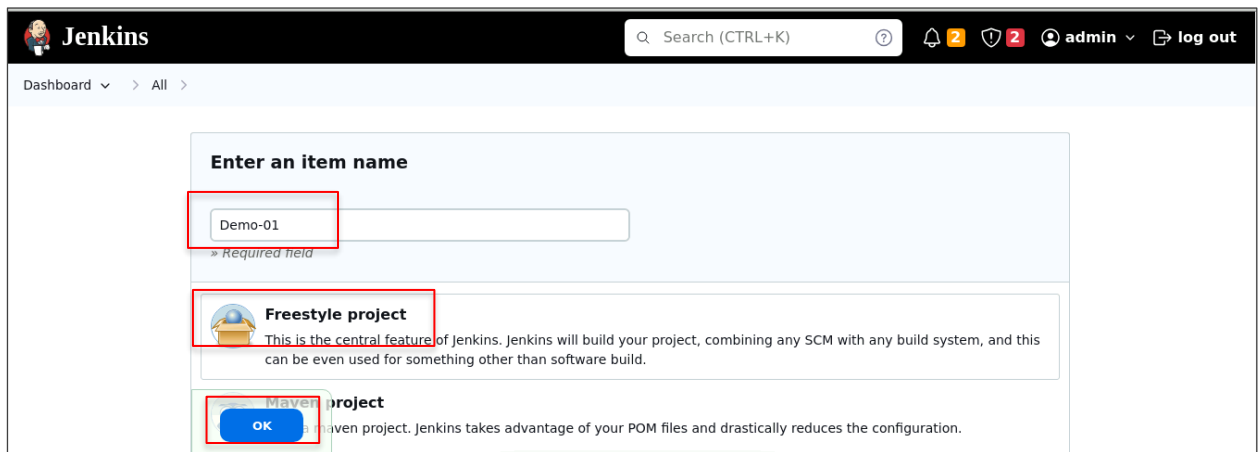
**Note:** Use the given credentials to access Jenkins in the lab: **Username** is admin and **Password** is Root123\$

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

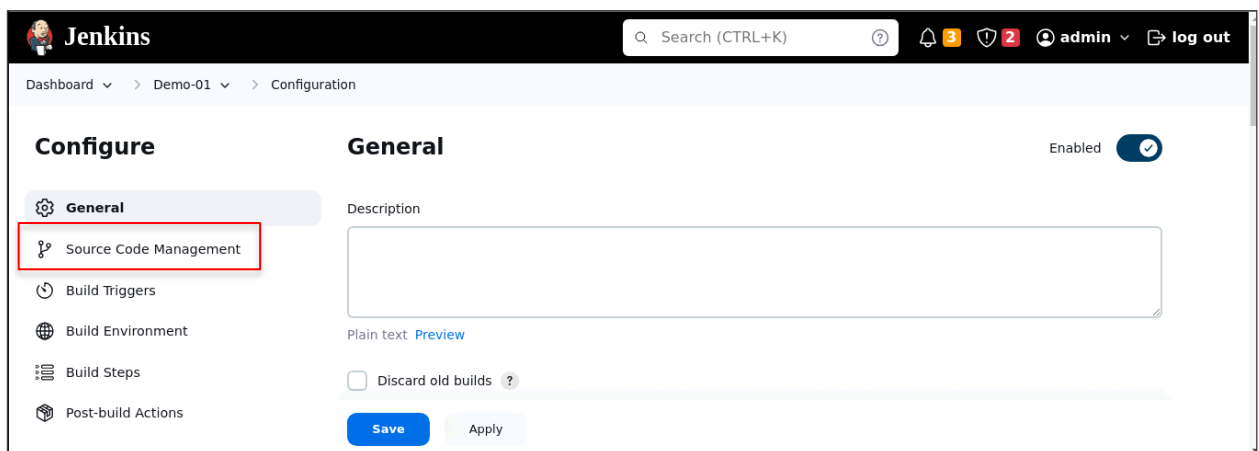




3.3 Enter a desired name for the project, select **Freestyle project**, and then click on **OK** as shown in the screenshot below:



3.4 Click on **Source Code Management** as shown in the screenshot below:



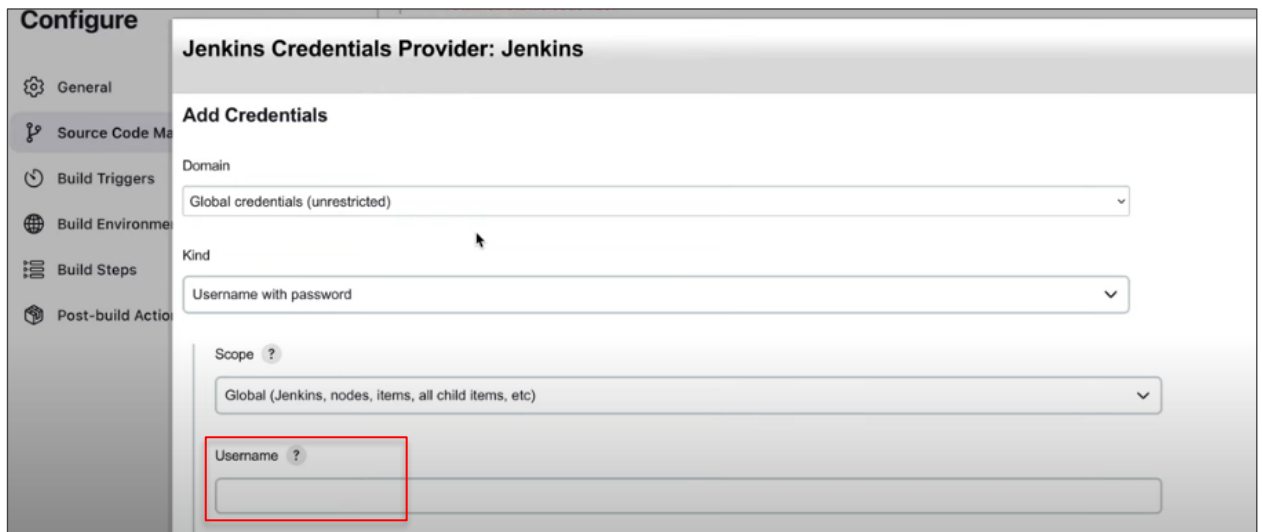
3.5 Click on **Git** as shown in the screenshot below:

The screenshot shows the Jenkins Configuration page for 'Demo-01'. The left sidebar has a 'Configure' section with options: General, Source Code Management (selected), Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main area is titled 'Source Code Management' and shows two radio buttons: 'None' and 'Git'. The 'Git' option is selected and highlighted with a red box. Below the radio buttons, there is a 'Repositories' section with a 'Repository URL' field and a 'Credentials' dropdown. A red error message 'Please enter Git repository.' is displayed. At the bottom, there are 'Save' and 'Apply' buttons.

3.6 Enter the repository URL and click on **Add** as shown in the screenshot below:

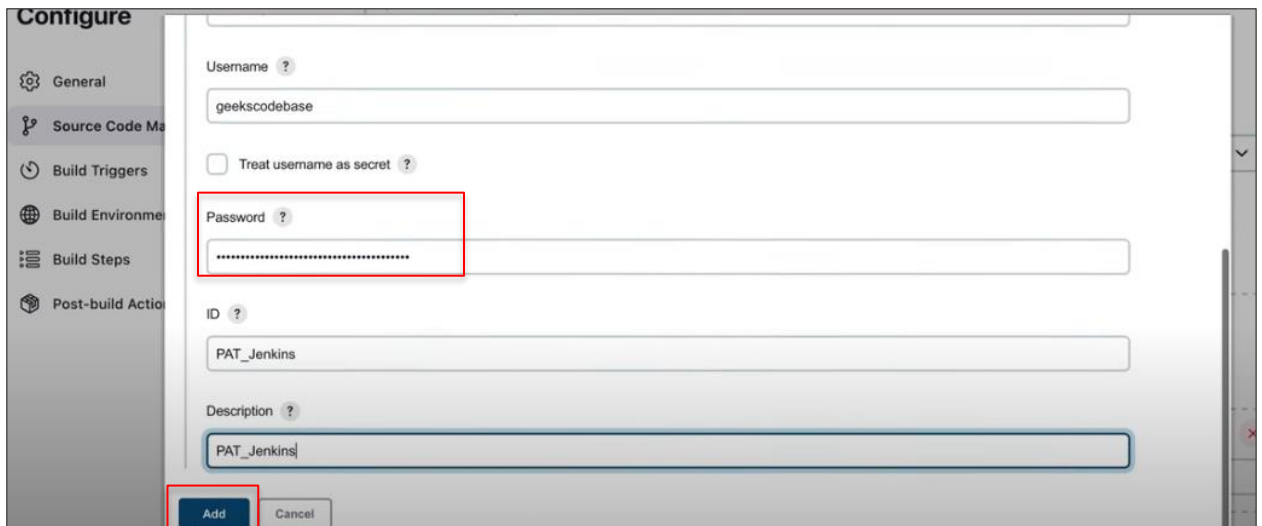
The screenshot shows the same Jenkins Configuration page, but now the 'Repository URL' field is filled with 'https://github.com/Poojahk10/Maven.git'. The field is highlighted with a red box. Below the field, the 'Credentials' dropdown is open, showing '- none -' and an 'Add' button. The 'Add' button is also highlighted with a red box. The red error message 'Please enter Git repository.' is still present. At the bottom, there are 'Save' and 'Apply' buttons.

3.7 Enter a desired name for the **Username** as shown in the screenshot below:



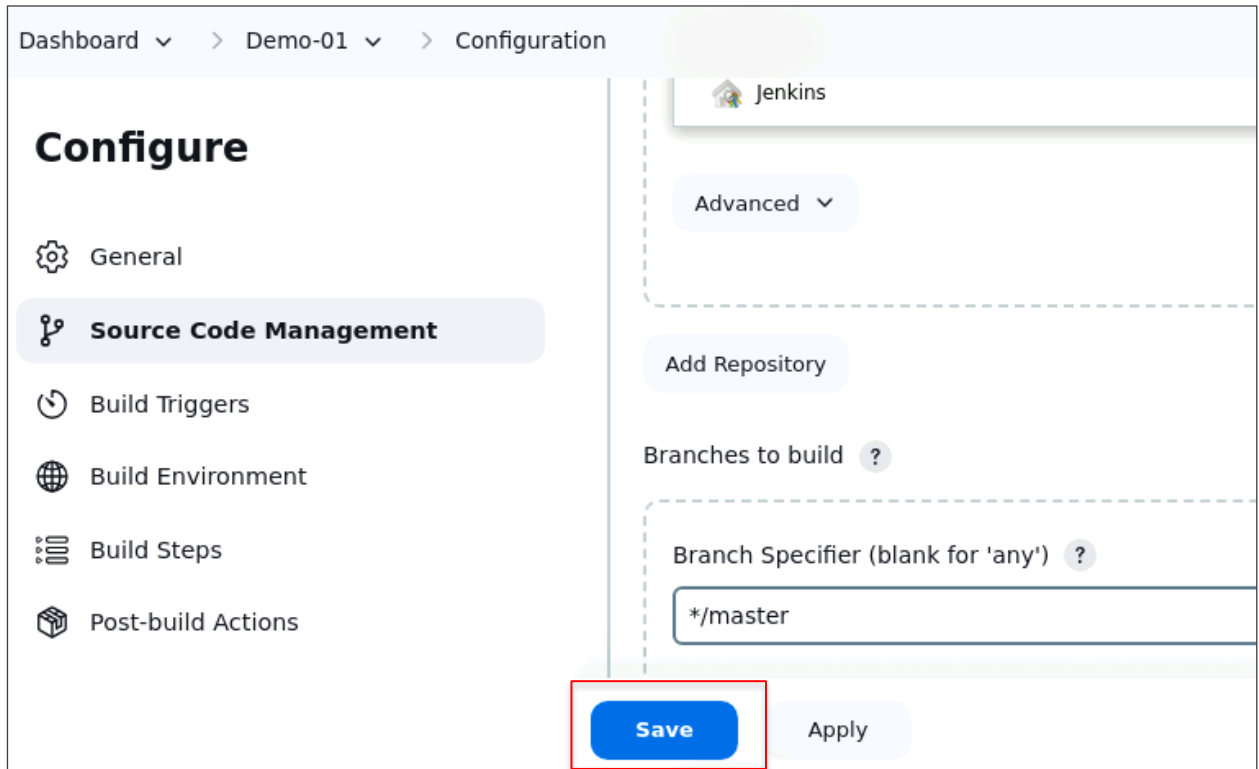
The screenshot shows the 'Configure' page for the 'Jenkins Credentials Provider: Jenkins'. The 'Add Credentials' section is active. The 'Domain' dropdown is set to 'Global credentials (unrestricted)'. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field is highlighted with a red box and is currently empty.

3.8 Paste the Git token under **Password** section and then click on **Add** as shown in the screenshot below:

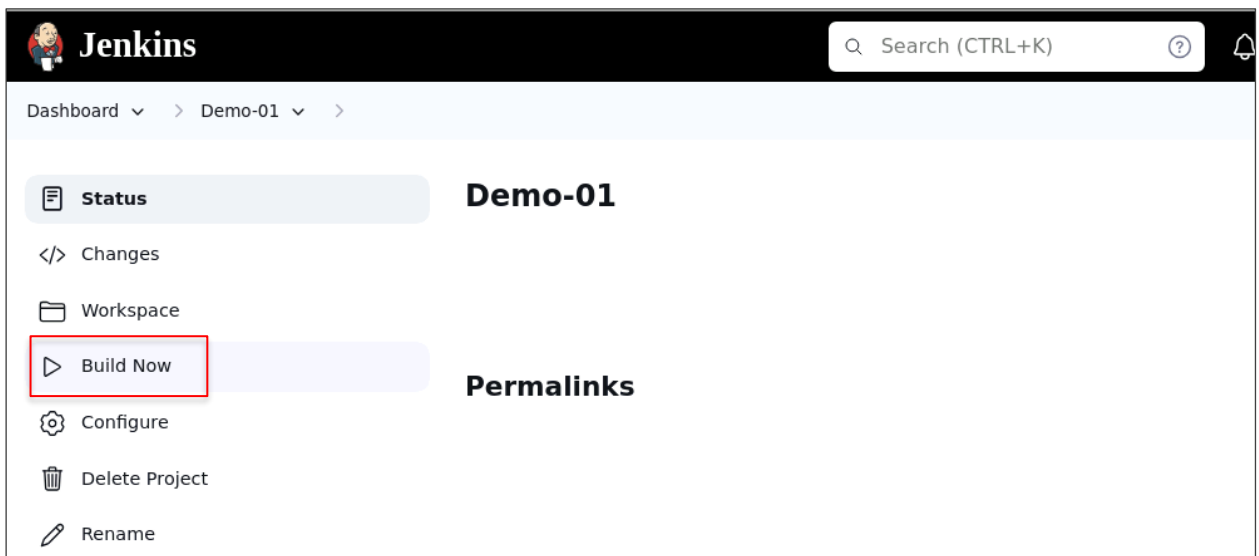


The screenshot shows the 'Configure' page for the 'Jenkins Credentials Provider: Jenkins'. The 'Add Credentials' section is active. The 'Username' field is filled with 'geekscodbase'. The 'Treat username as secret' checkbox is unchecked. The 'Password' field is highlighted with a red box and contains a masked password (dots). The 'ID' field is filled with 'PAT\_Jenkins'. The 'Description' field is filled with 'PAT\_Jenkins'. The 'Add' button is highlighted with a red box.

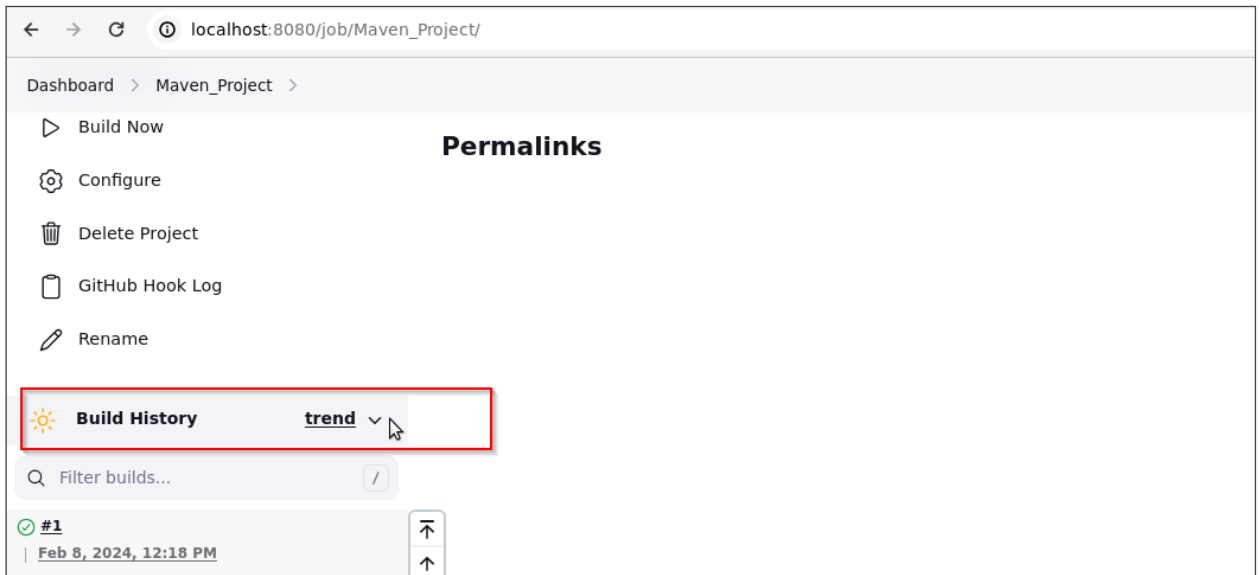
3.9 Click on **Save** as shown in the screenshot below:



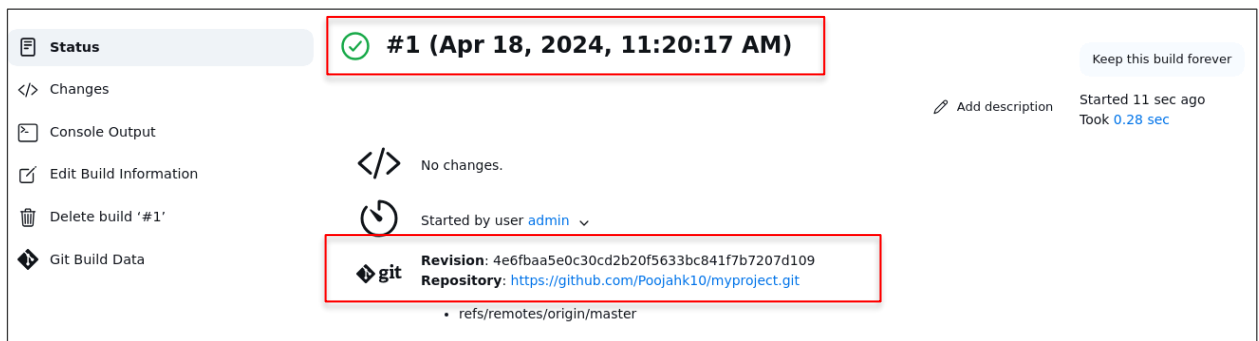
3.10 Click on **Build Now** in the left section after the creation of the job as shown in the below screenshot:



3.11 Click on **Build History** as shown in the screenshot below:



3.12 Verify that the output indicates the status (successful) as shown in the screenshot below:



By following these steps, you have successfully set up the Jenkins freestyle job from the Git version control system to enable automated CI for building, testing, and potentially deploying software upon code changes.