

Lesson 01 Demo 01

Implementing the DevOps Model

Objective: To implement DevOps using GitHub to store a Java program and Jenkins to build consistent code packages, enabling continuous integration

Tools required: Git, GitHub, and Jenkins

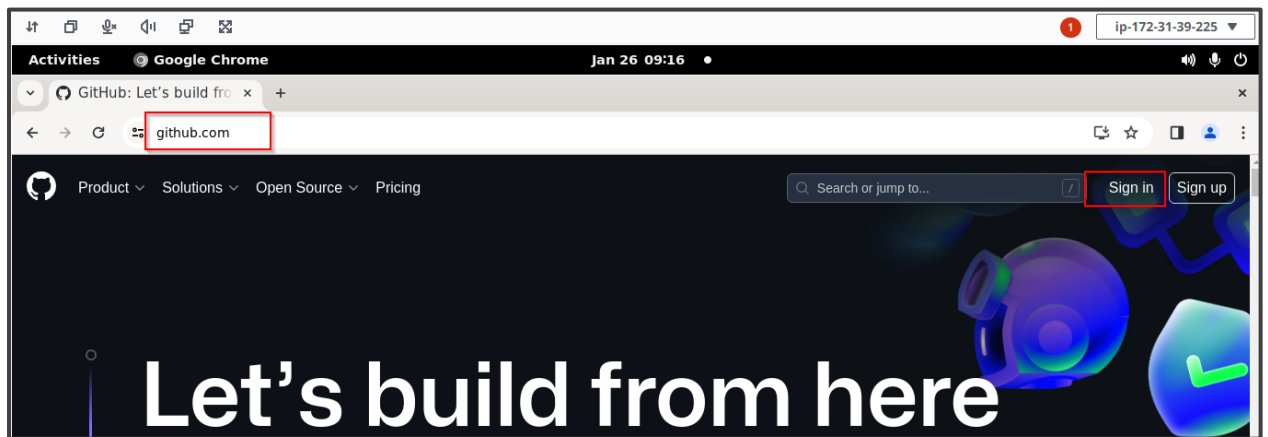
Prerequisites: None

Steps to be followed:

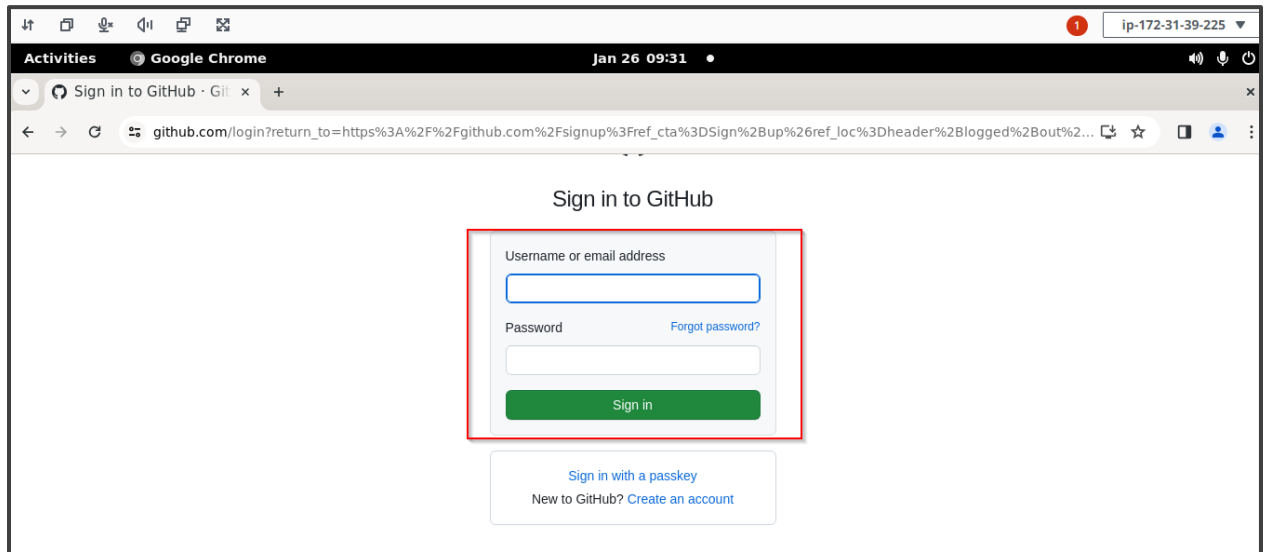
1. Create a GitHub repository
2. Add a Java program to the repository
3. Create a freestyle build job in Jenkins
4. Build the Java program with Jenkins

Step 1: Create a GitHub repository

1.1 Open the browser in your lab, go to <https://github.com>, and click on the **Sign in** button

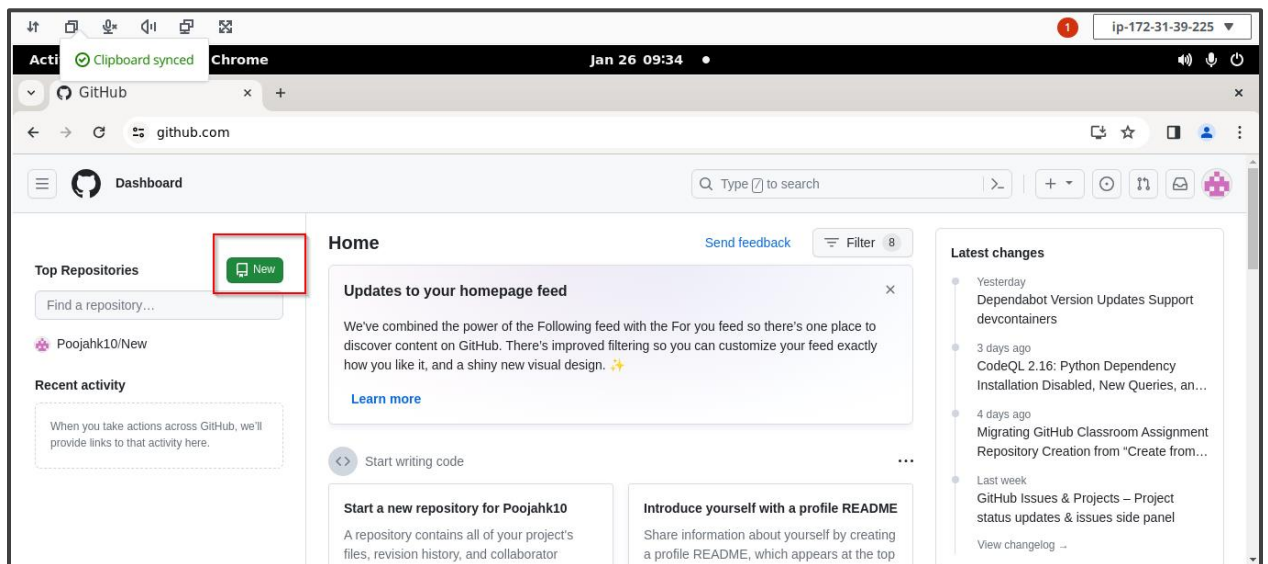


1.2 Enter the credentials of your GitHub account and click on **Sign in**



A screenshot of a web browser showing the GitHub sign-in page. The browser's address bar displays the URL `github.com/login?return_to=https%3A%2F%2Fgithub.com%2Fsignup%3Fref_cta%3DSign%2Bup%26ref_loc%3Dheader%2Blogged%2Bout%2...`. The page title is "Sign in to GitHub". The main content area features a sign-in form with two input fields: "Username or email address" and "Password". A green "Sign in" button is positioned below these fields. To the right of the password field is a link for "Forgot password?". Below the sign-in form is a section for "Sign in with a passkey" and a link for "New to GitHub? Create an account". A red rectangular box highlights the sign-in form and the "Sign in" button.

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



1.4 Add the **Repository** name as shown in the screenshot below:

Activities Google Chrome Jan 26 09:37

New repository x +

github.com/new

New repository

Q Type to search

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

Repository name *

Freestyle Job Creation

✓ Your new repository will be created as Freestyle-Job-Creation-.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. Need inspiration? How about **didactic-adventure** ?

Description (optional)

1.5 Select the check box of **ADD a README file** and click on **Create repository**

Activities Google Chrome Jan 26 09:41

New repository x +

github.com/new

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

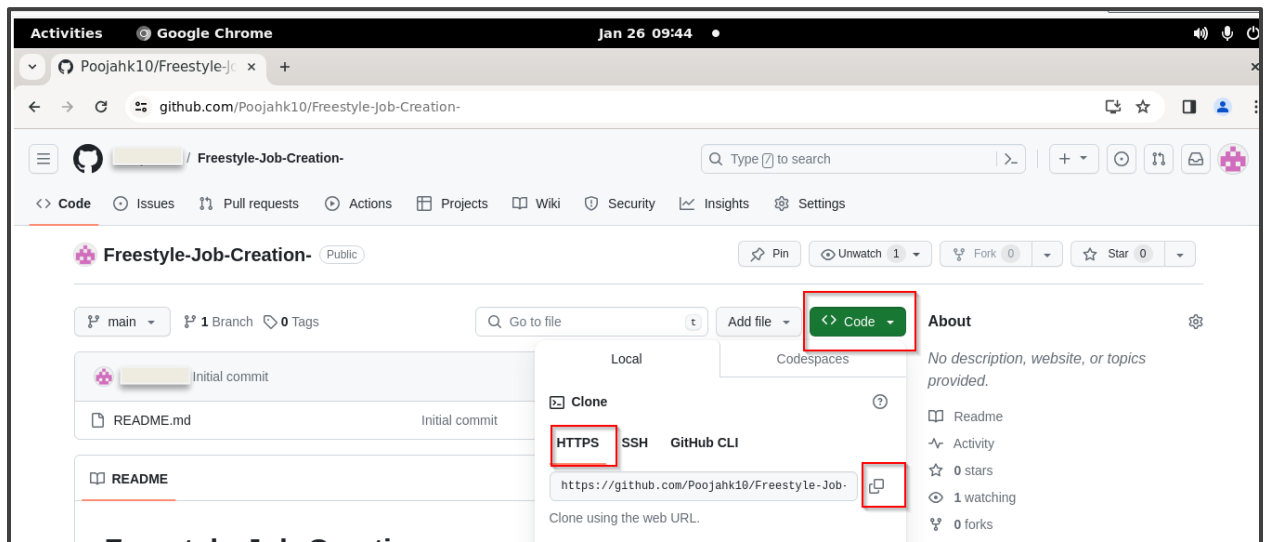
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set **main** as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

Create repository

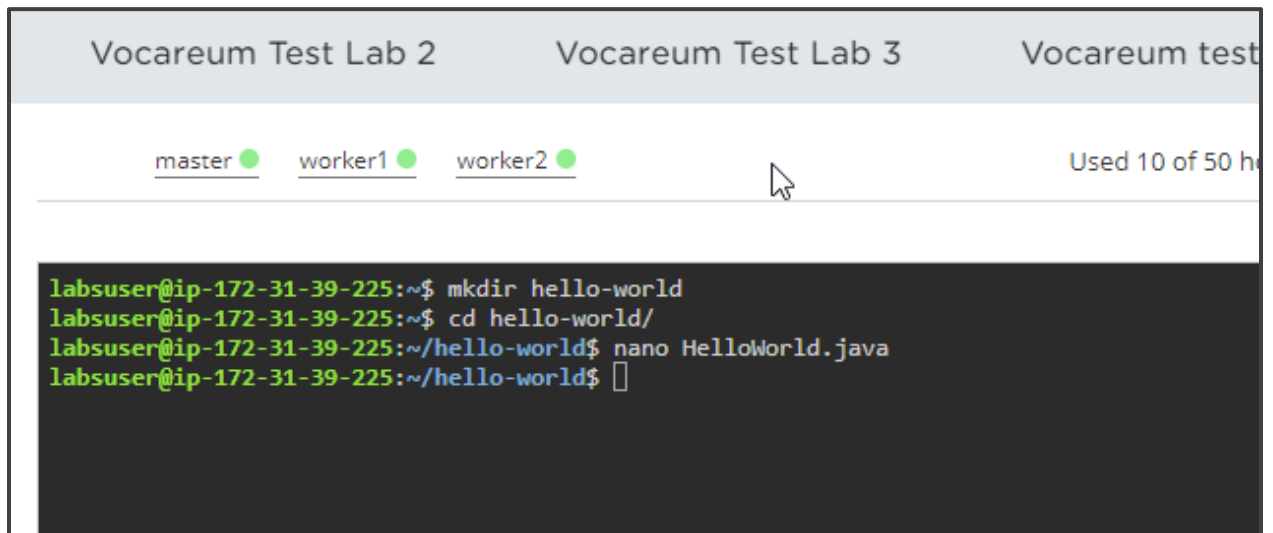
1.6 Click on <> **Code**, then **HTTPS**, and finally copy the repository URL



Step 2: Add a Java program to the repository

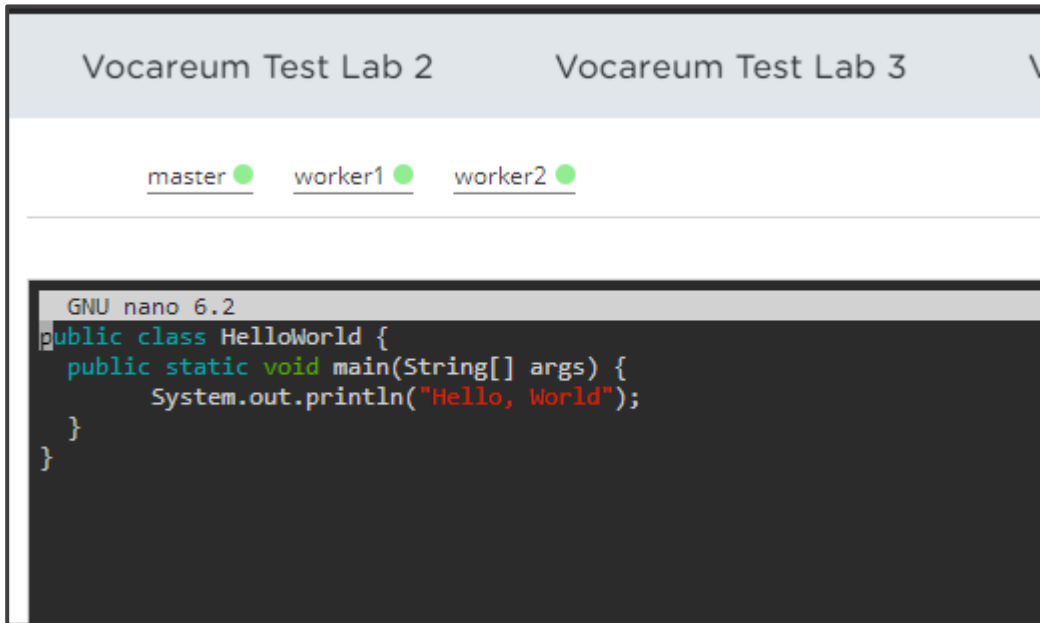
2.1 Open the terminal, run the following commands to create a directory, navigate to the **hello-world** directory, and open the Java file in a text editor as shown in the screenshot below:

```
mkdir hello-world
cd hello-world
nano HelloWorld.java
```



2.2 Copy and paste the below code into the file, save the file, and exit from the text editor:

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World");  
    }  
}
```



2.3 Run the following commands:

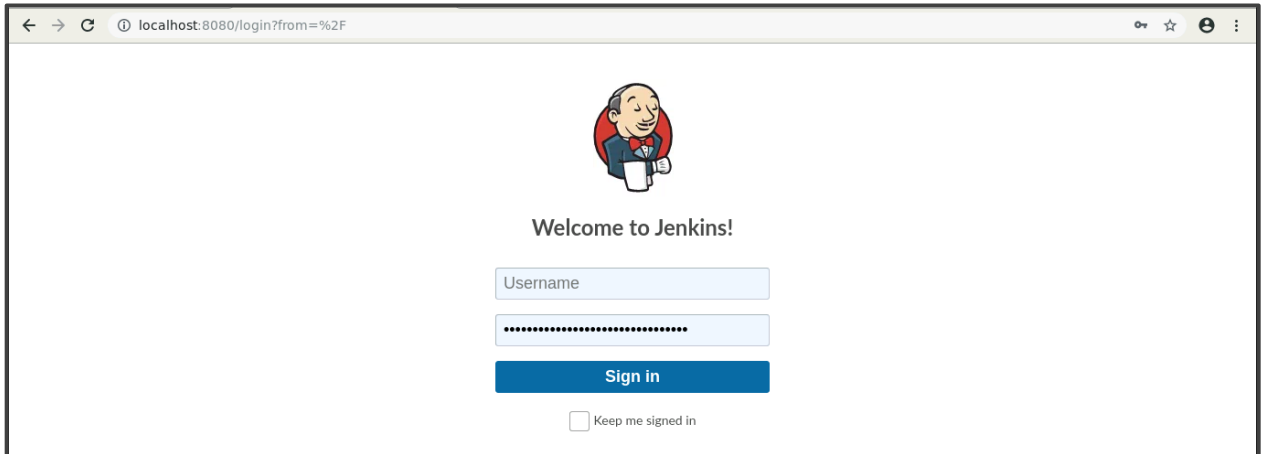
```
git init  
git add .  
git commit -m "Add new files"  
git remote add origin <Repository_URL>  
git push -u origin master
```

```
labuser@ip-172-31-39-225:~/hello-world$ git init  
hint: Using 'master' as the name for the initial branch. This default branch name  
hint: is subject to change. To configure the initial branch name to use in all  
hint: of your new repositories, which will suppress this warning, call:  
hint:  
hint:   git config --global init.defaultBranch <name>  
hint:  
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and  
hint: 'development'. The just-created branch can be renamed via this command:  
hint:  
hint:   git branch -m <name>  
Initialized empty Git repository in /home/labuser/hello-world/.git/  
labuser@ip-172-31-39-225:~/hello-world$ git add .  
labuser@ip-172-31-39-225:~/hello-world$ git commit -m HelloWorld.java  
[master (root-commit) 2bbb537] HelloWorld.java  
1 file changed, 6 insertions(+)  
create mode 100644 HelloWorld.java  
labuser@ip-172-31-39-225:~/hello-world$ git remote add origin https://github.com/Poojahki0/freestyle.git  
labuser@ip-172-31-39-225:~/hello-world$ git push -u origin master  
Username for 'https://github.com': Poojahki0  
Password for 'https://github.com': Poojahki0@github.com:  
remote: Support for password authentication was removed on August 13, 2021.  
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
```

Note: Ensure that the password to be added is your **GitHub** account **Token**

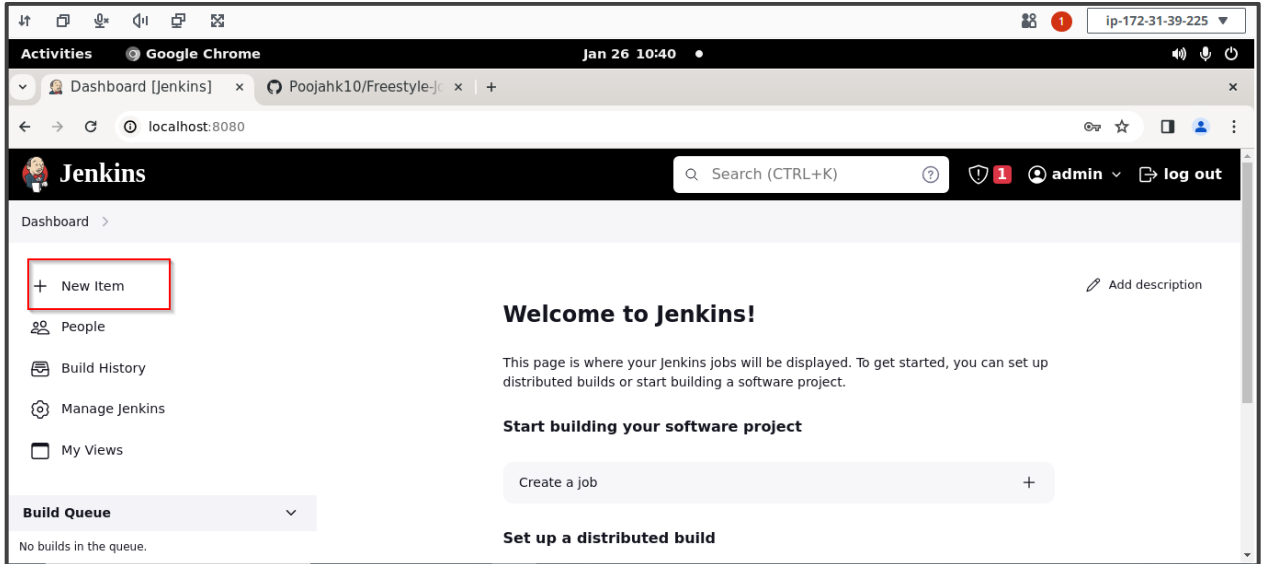
Step 3: Create a freestyle build job in Jenkins

3.1 Open the browser, type **localhost:8080**; this will open Jenkins. Provide the credentials and then click on **Sign in**

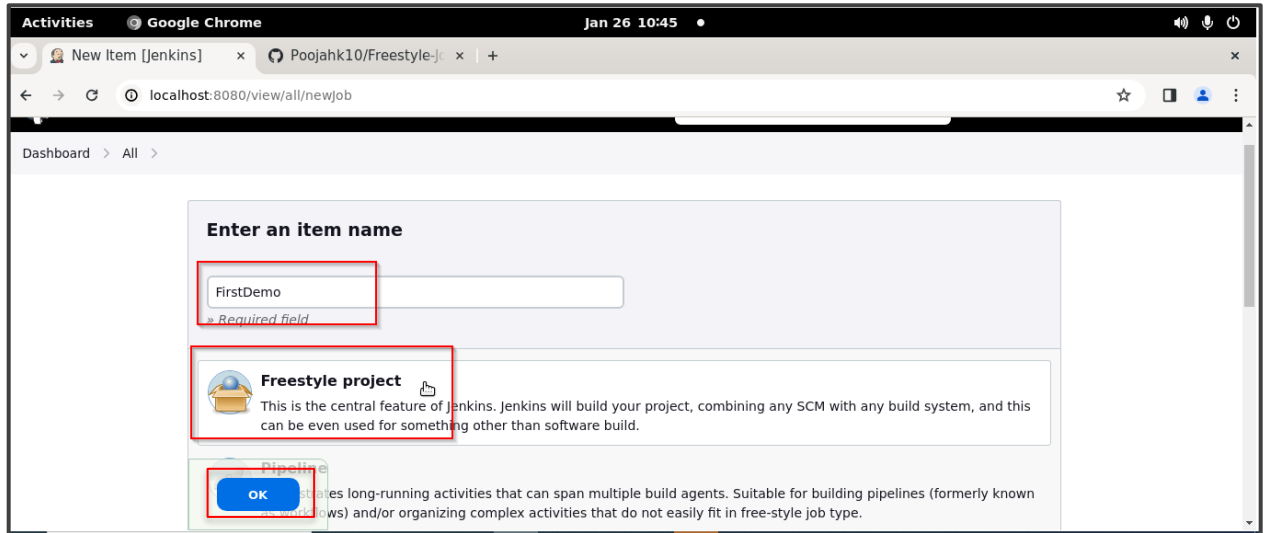
A screenshot of the Jenkins login page in a web browser. The browser's address bar shows 'localhost:8080/login?from=%2F'. The page features the Jenkins logo (a cartoon man with a red bow tie) and the text 'Welcome to Jenkins!'. Below this, there are two input fields: 'Username' and a password field represented by dots. A blue 'Sign in' button is positioned below the password field. At the bottom, there is a checkbox labeled 'Keep me signed in'.

Note: The credentials for accessing Jenkins in the lab are Username: **admin** and Password: **Root123\$**.

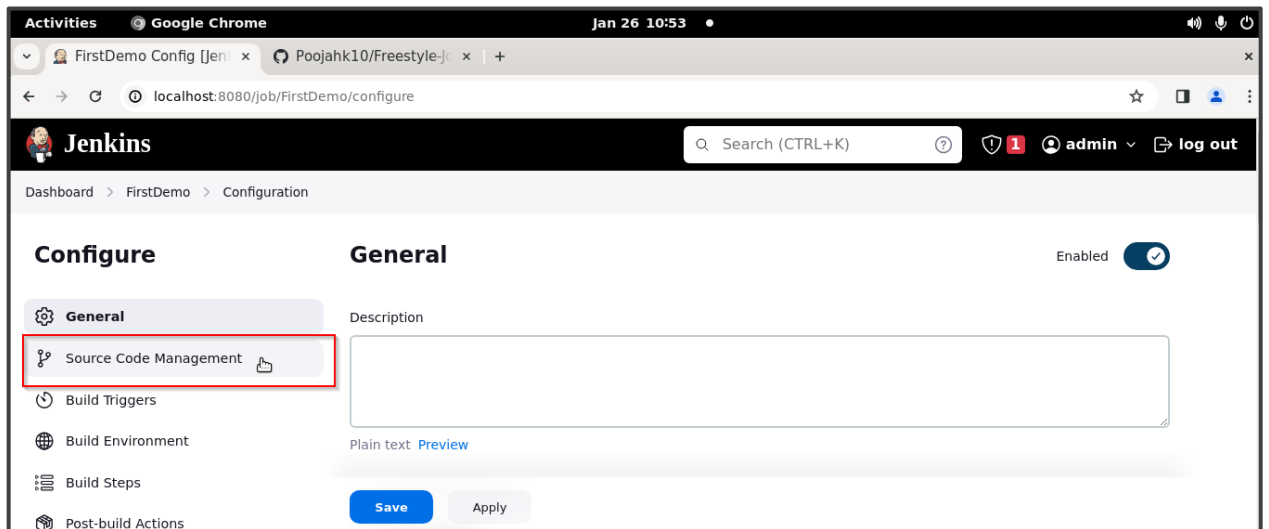
3.2 Click on **New Item** in the Jenkins Dashboard

A screenshot of the Jenkins Dashboard in a web browser. The browser's address bar shows 'localhost:8080'. The dashboard has a dark header with the Jenkins logo, a search bar, and user information ('admin'). On the left sidebar, the 'New Item' button is highlighted with a red rectangle. Other sidebar options include 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area displays 'Welcome to Jenkins!' and instructions on how to start building software projects. There is a 'Create a job' button and a 'Set up a distributed build' section. The 'Build Queue' section at the bottom indicates 'No builds in the queue.'

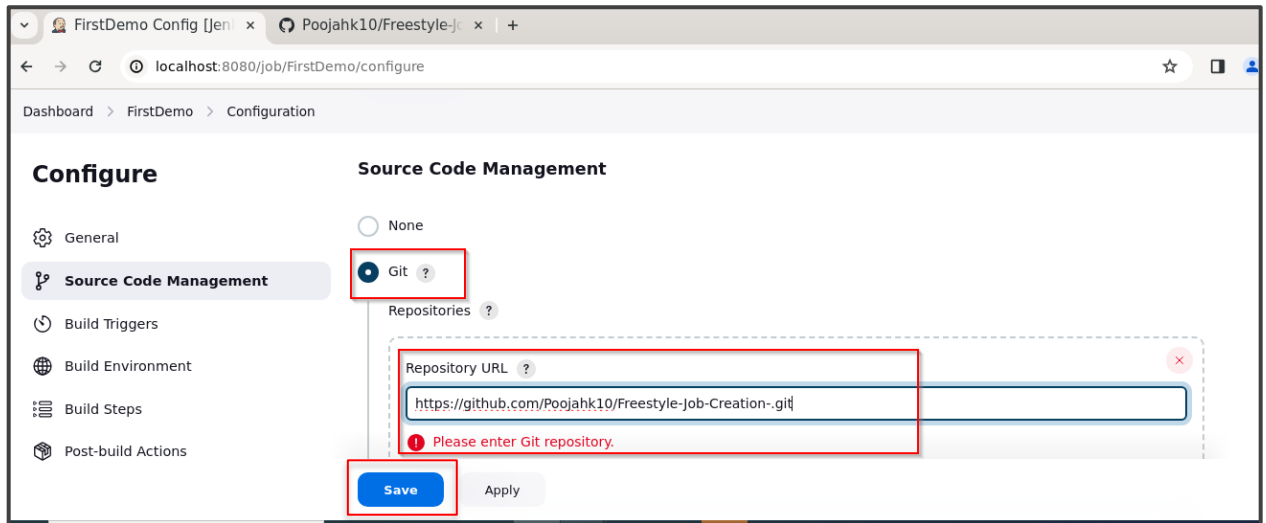
3.3 Enter a name for your project, select **Freestyle project** as the build job type, and click on **OK**



3.4 Click on **Source Code Management**

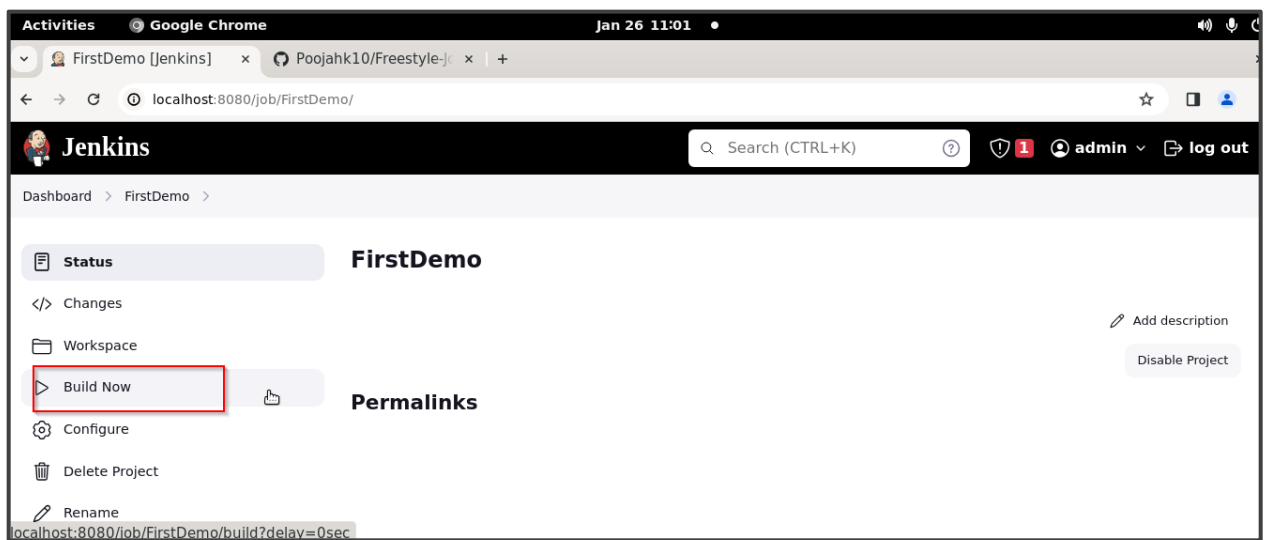


3.5 Select **Git**, enter the **Repository URL**, and then click on **Save**

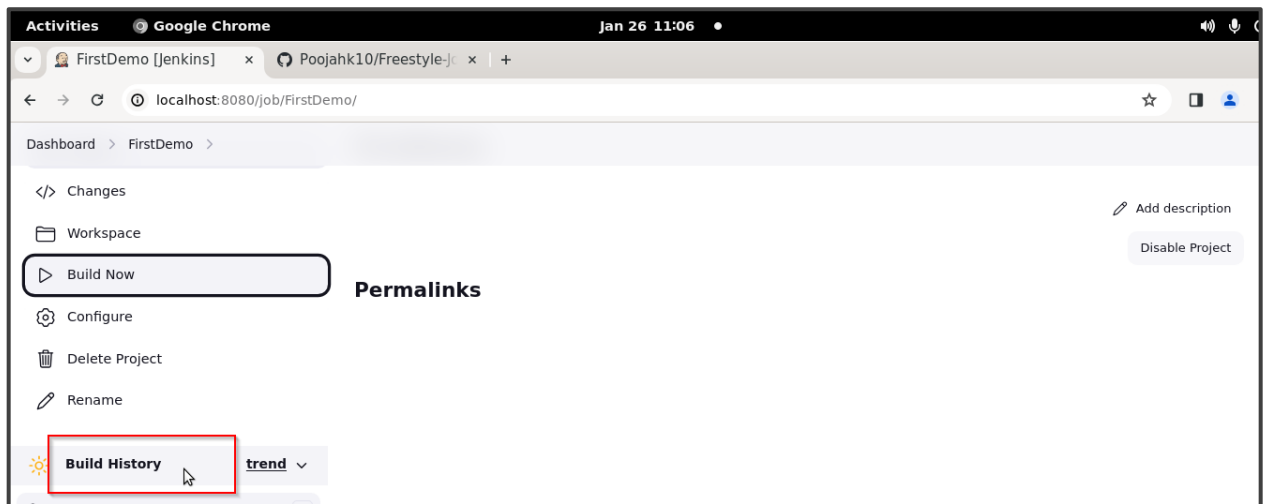


Step 4: Build the Java program with Jenkins

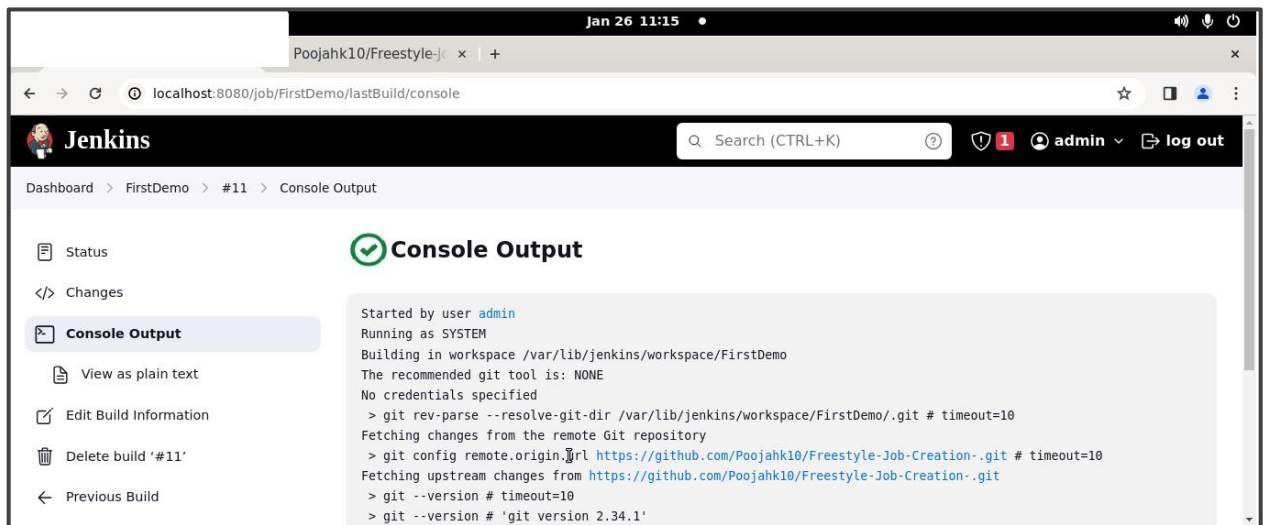
4.1 Click on **Build Now** to build your project



4.2 Click on **Build History** to view the build results



4.3 Click on the **Console Output** to view the build logs



By following these steps, you have successfully implemented DevOps using GitHub to store a Java program and Jenkins to build consistent code packages, enabling continuous integration.