

Lab Exercise 6.2

Creating a New Jenkins Job to Checkout Source Code

Objective: To set up a Jenkins job to manage source code, specifically by configuring the Source Code Management section to check out code from a Git repository

Tools required: Jenkins

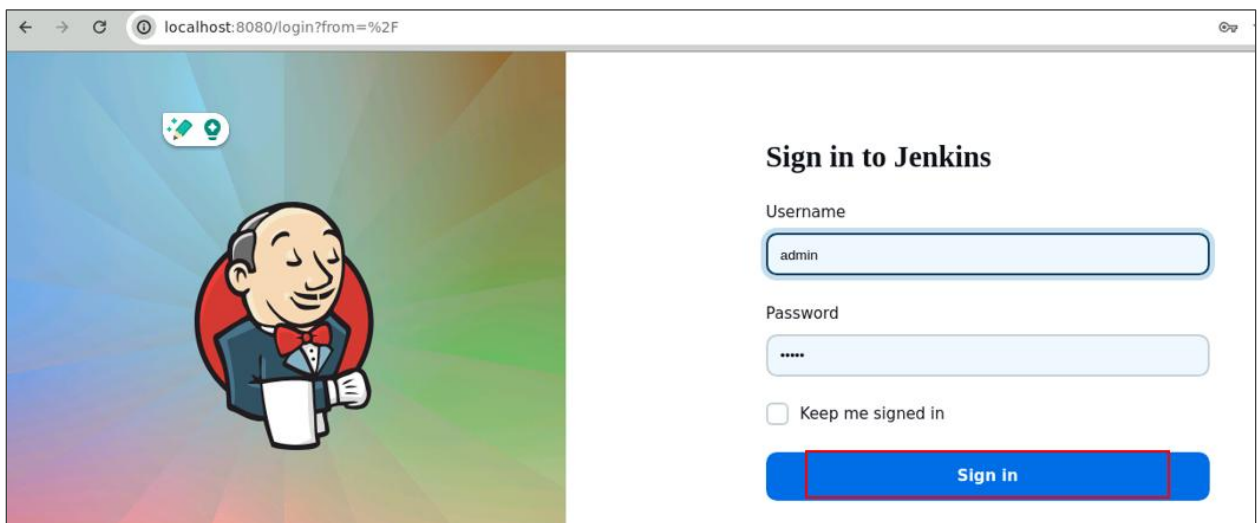
Prerequisites: Jenkins must be operational.

Steps to be followed:

1. Log in and create a Jenkins job
2. Configure source code management


Step 1: Log in and create a Jenkins job

- 1.1 Navigate to **localhost:8080** in your web browser, enter your credentials, and click on **Sign In**



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


1.2 Create a new Jenkins job by clicking on **New Item**


 **Jenkins** / All / New Item


New Item

Enter an item name

Select an item type

 **Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.


OK

1.3 Provide custom job name inside the field **Enter an item name**, select the **Freestyle project** option, and click on the **OK** button to save the job


Enter an item name

FreeStyle


» Required field

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

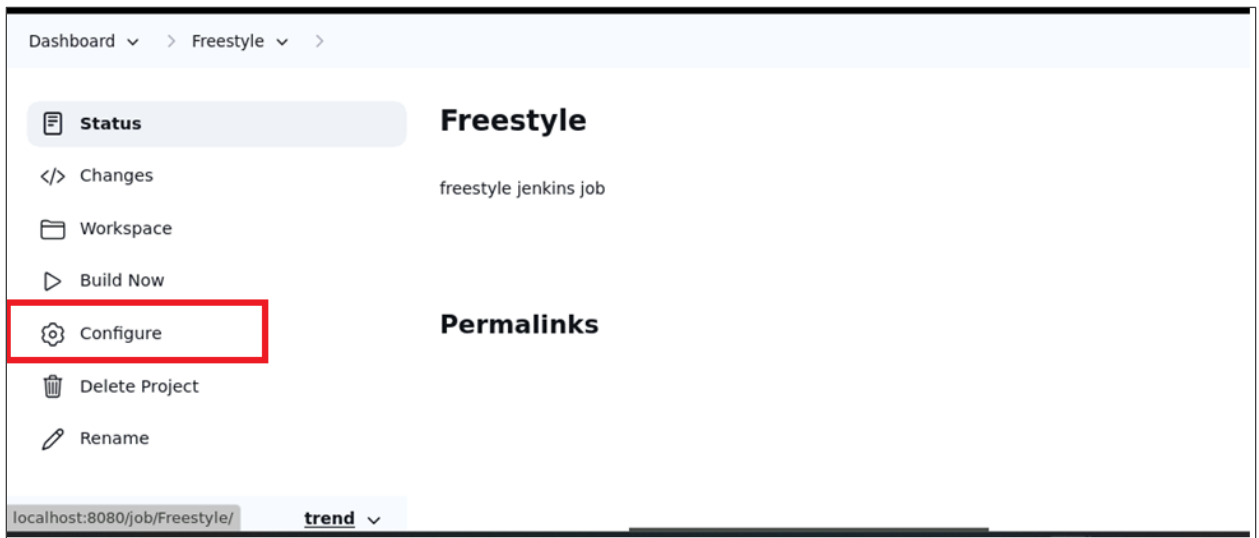
**GitHub Organization**

GitHub organization (or user account) for all repositories matching some defined markers.

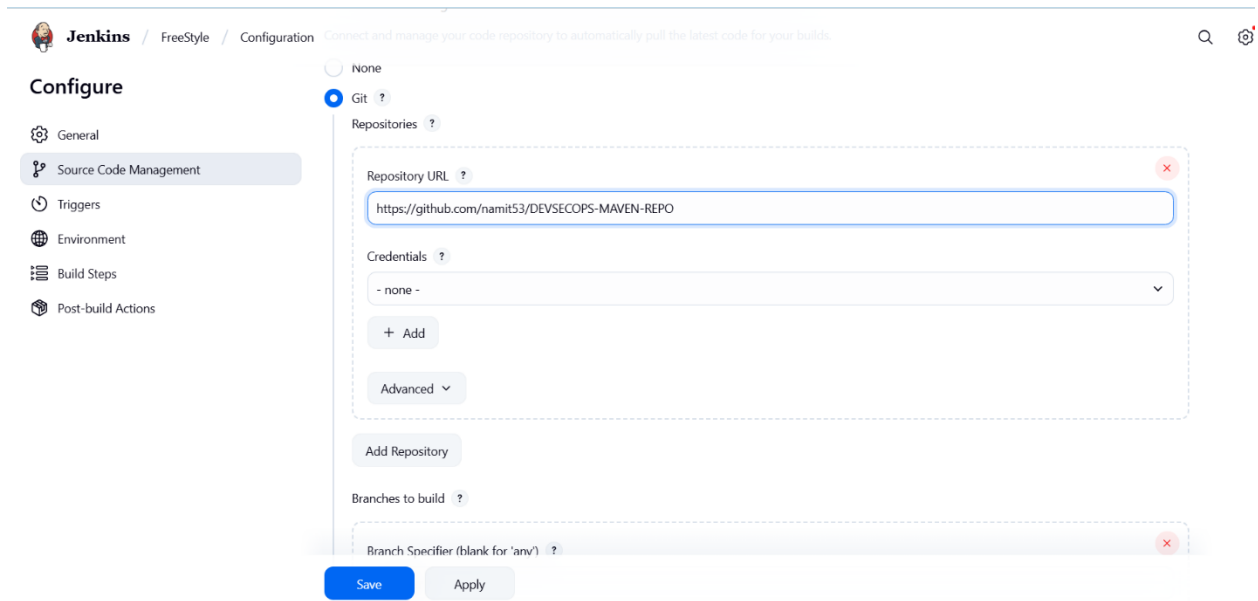
OK

Step 2: Configure source code management

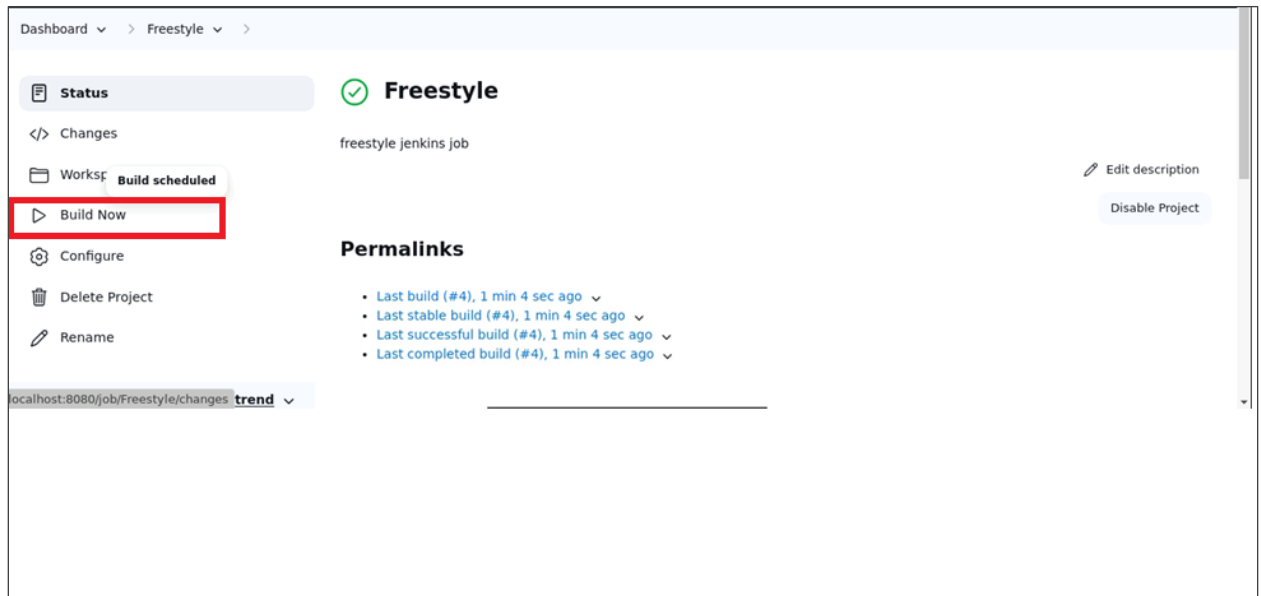
2.1 Access the newly created job's configuration screen by clicking on **Configure**



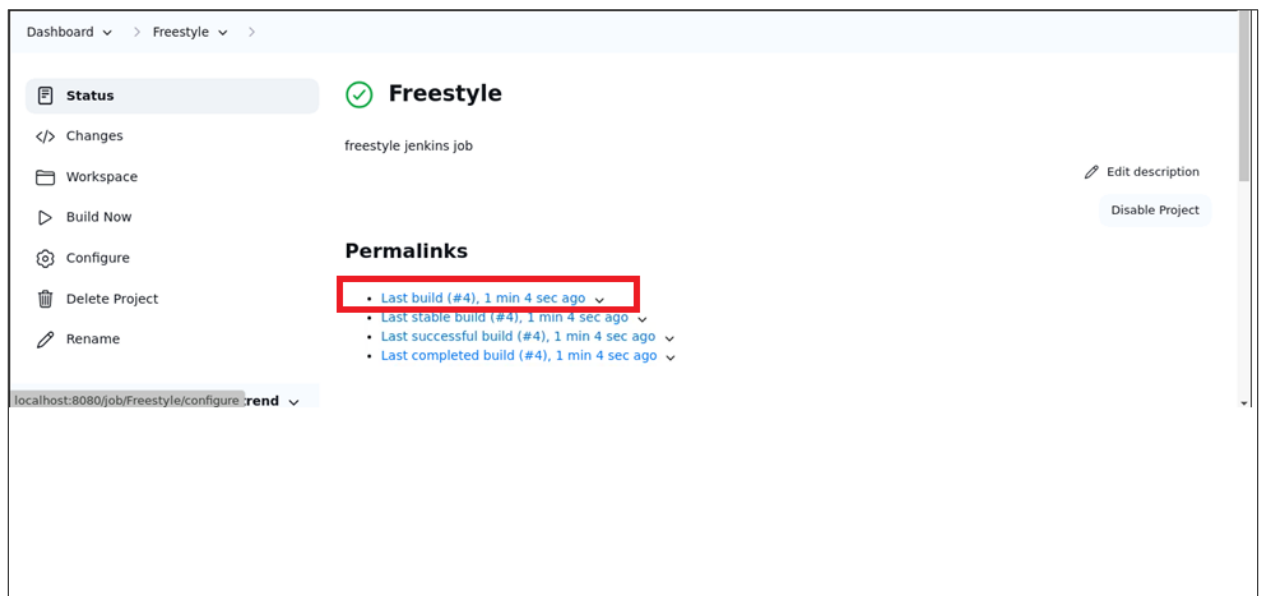
2.2 Navigate to the **Source Code Management** tab, provide Git repository configuration inside the **Repository URL** field, and click on the **Save** button



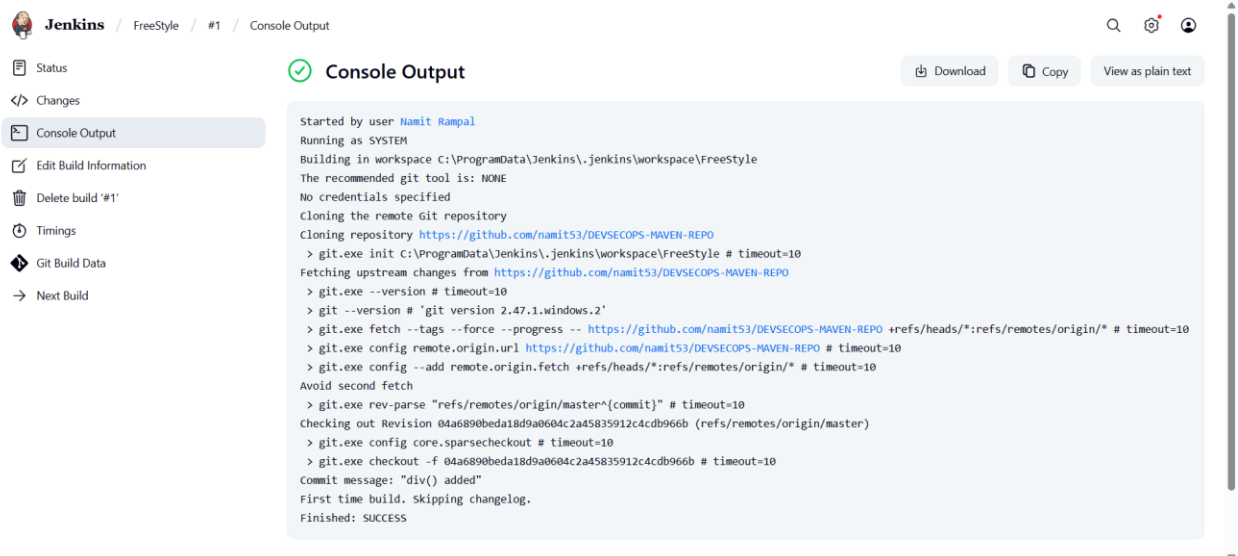
2.3 Then, click on the **Build Now** option to schedule a build



2.4 To schedule the build, click the required link under **Permalinks**



2.5 Click on **Console Output** to check out the process during the build process



The screenshot shows the Jenkins web interface for a job named 'FreeStyle' (build #1). The 'Console Output' tab is selected, displaying the build log. The log shows the build was started by user 'Namit Rampal' and is running as 'SYSTEM'. It details the process of cloning a Git repository from 'https://github.com/namit53/DEVSECOPS-MAVEN-REPO' and checking out a specific revision. The build concludes with the message 'Finished: SUCCESS'.

Console Output

```
Started by user Namit Rampal
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\FreeStyle
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/namit53/DEVSECOPS-MAVEN-REPO
> git.exe init C:\ProgramData\Jenkins\jenkins\workspace\FreeStyle # timeout=10
Fetching upstream changes from https://github.com/namit53/DEVSECOPS-MAVEN-REPO
> git.exe --version # timeout=10
> git --version # 'git version 2.47.1.windows.2'
> git.exe fetch --tags --force --progress -- https://github.com/namit53/DEVSECOPS-MAVEN-REPO +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config remote.origin.url https://github.com/namit53/DEVSECOPS-MAVEN-REPO # timeout=10
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 04a6890beda18d9a0604c2a45835912c4cdb966b (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 04a6890beda18d9a0604c2a45835912c4cdb966b # timeout=10
Commit message: "div() added"
First time build. skipping changelog.
Finished: SUCCESS
```

By following these steps, you have successfully set up a Jenkins job to automatically check out source code from a Git repository, enabling seamless integration and automation in your CI/CD pipeline.