

## 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

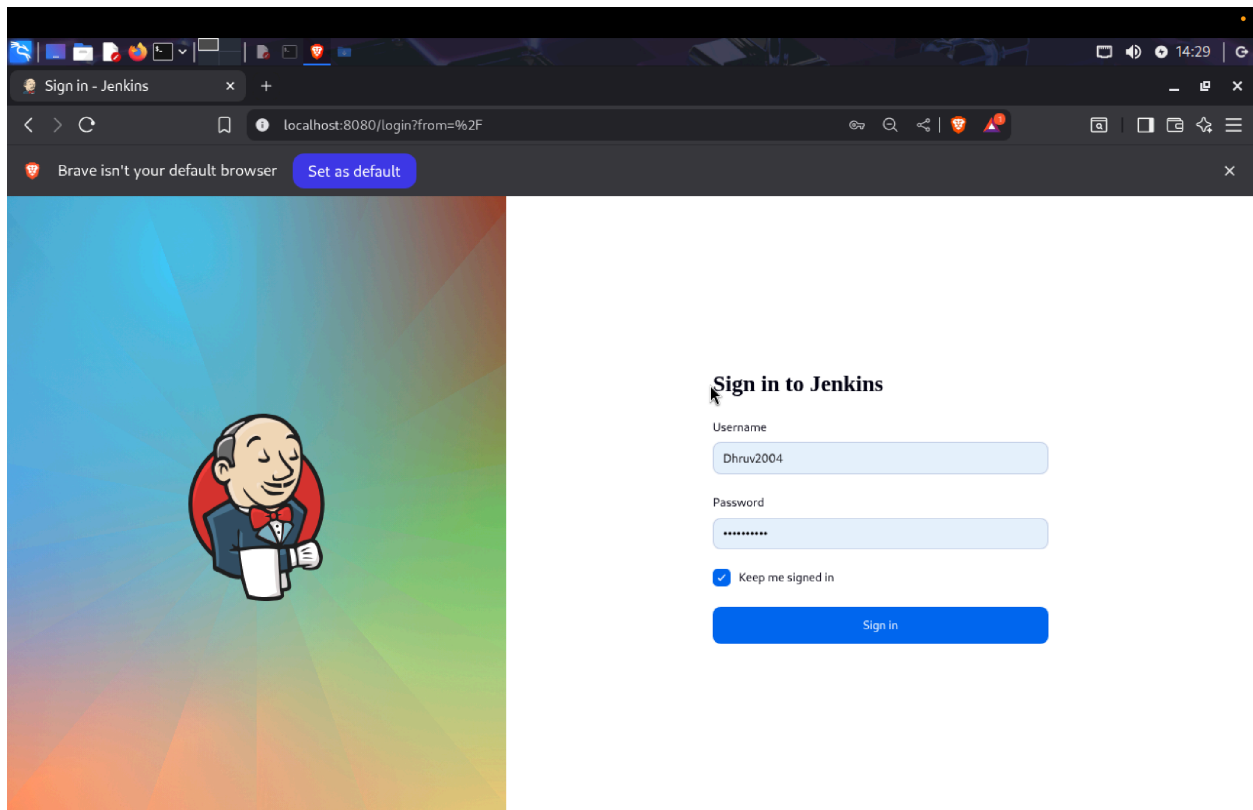
**Prerequisites:** Jenkins must be operational.

Steps to be followed:

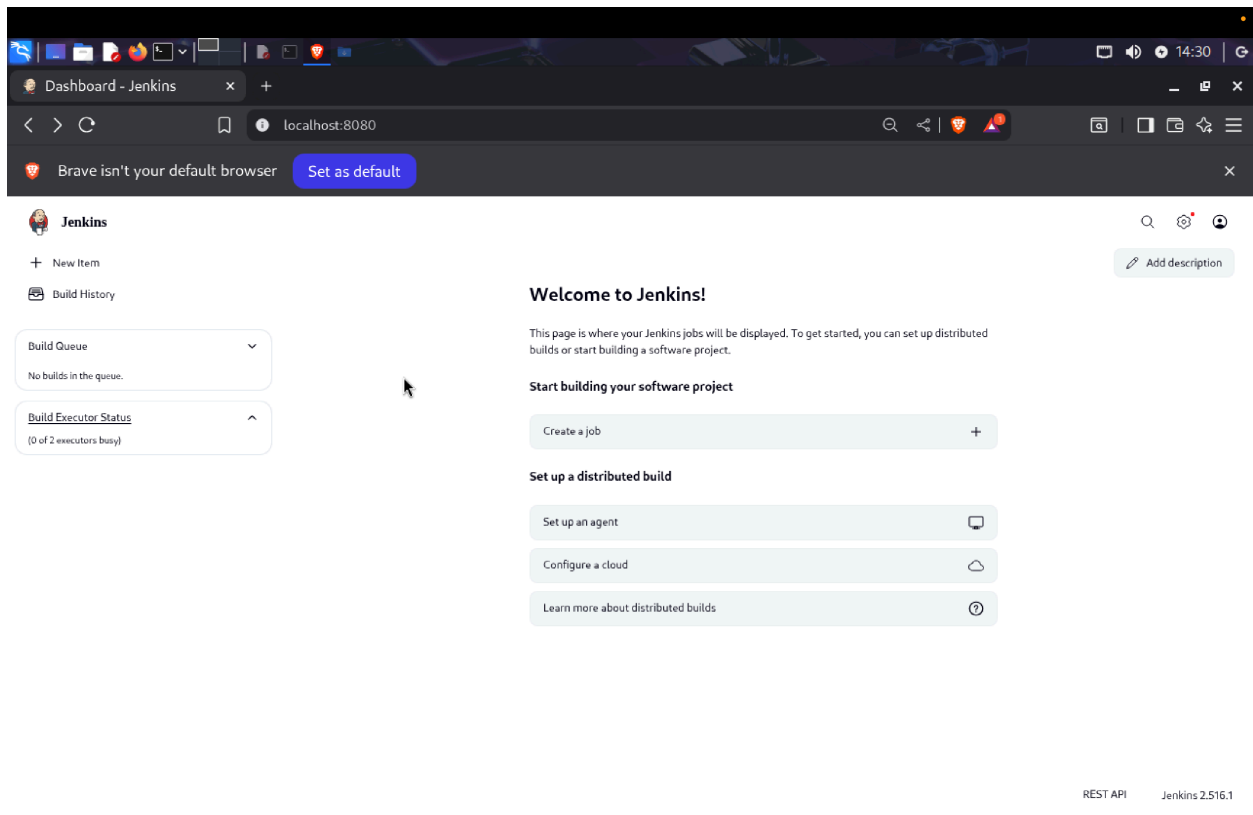
1. Log in and create a Jenkins job
2. Configure source code management

#### Step 1: Log in and create a Jenkins job

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



## 2. Create a new Jenkins job by clicking on **New Item**




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.

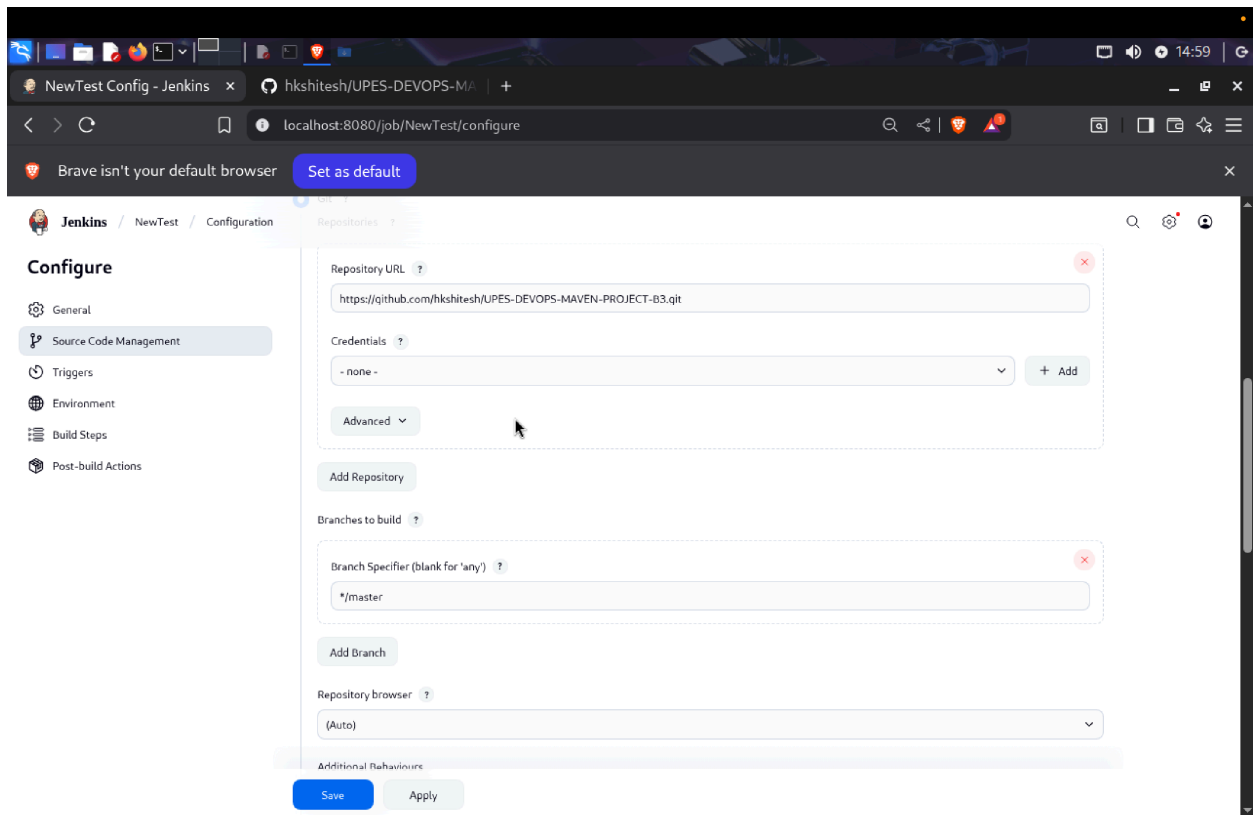
**GitHub Organization**

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

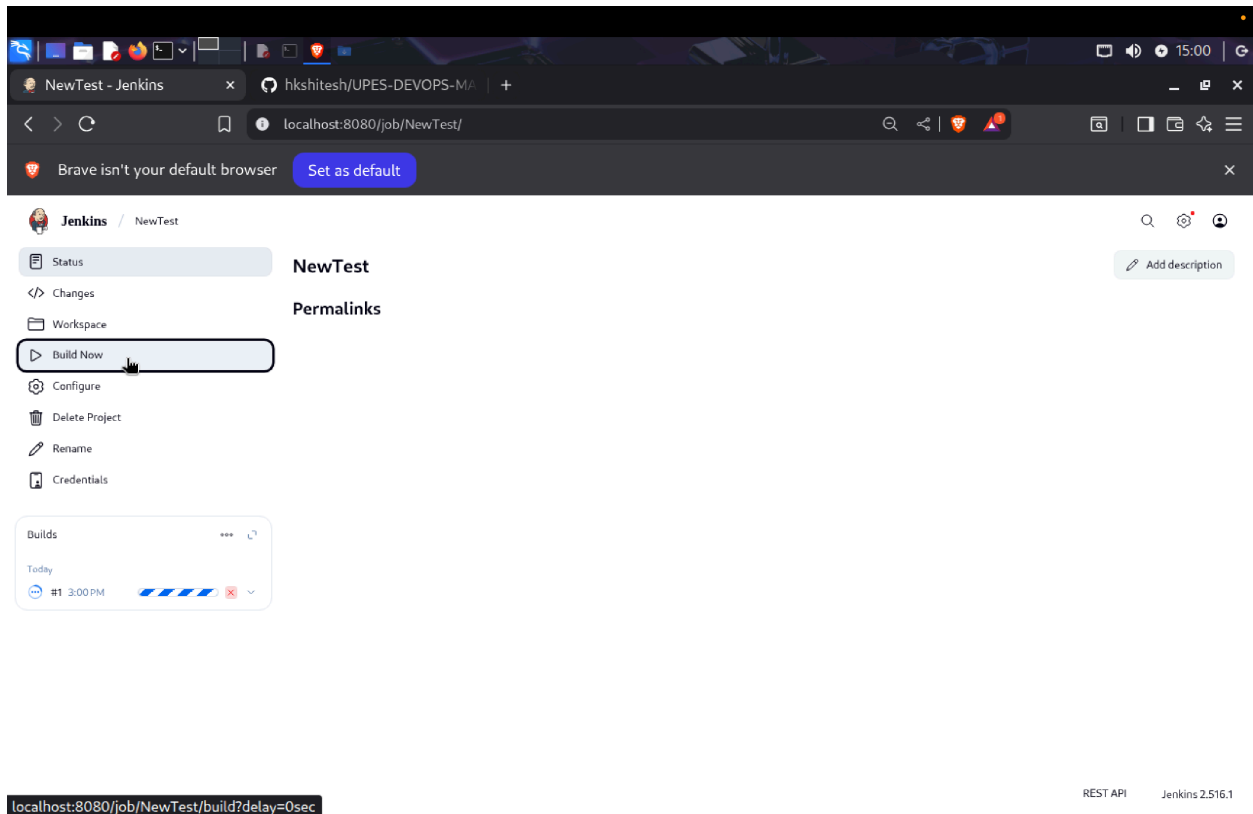
OK

## Step 2: Configure source code management

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

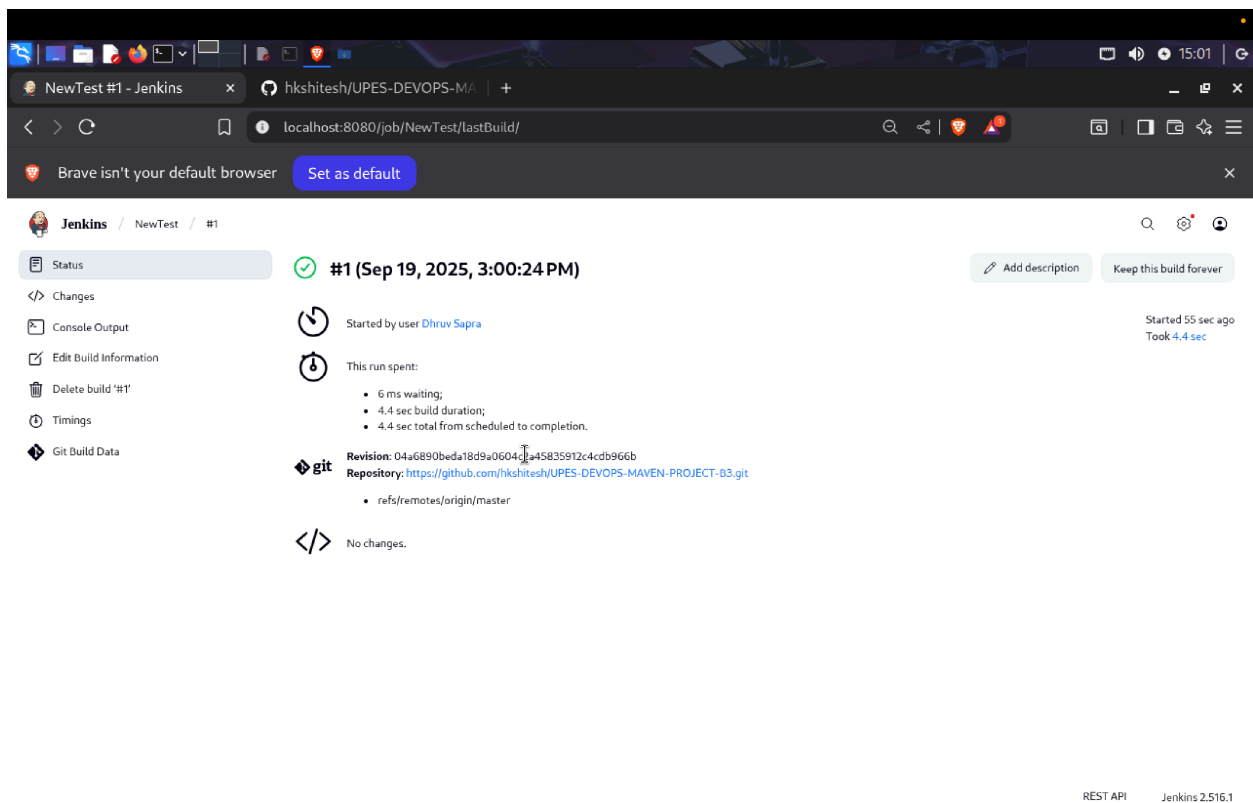


2. Then, click on the **Build Now** option to schedule a build



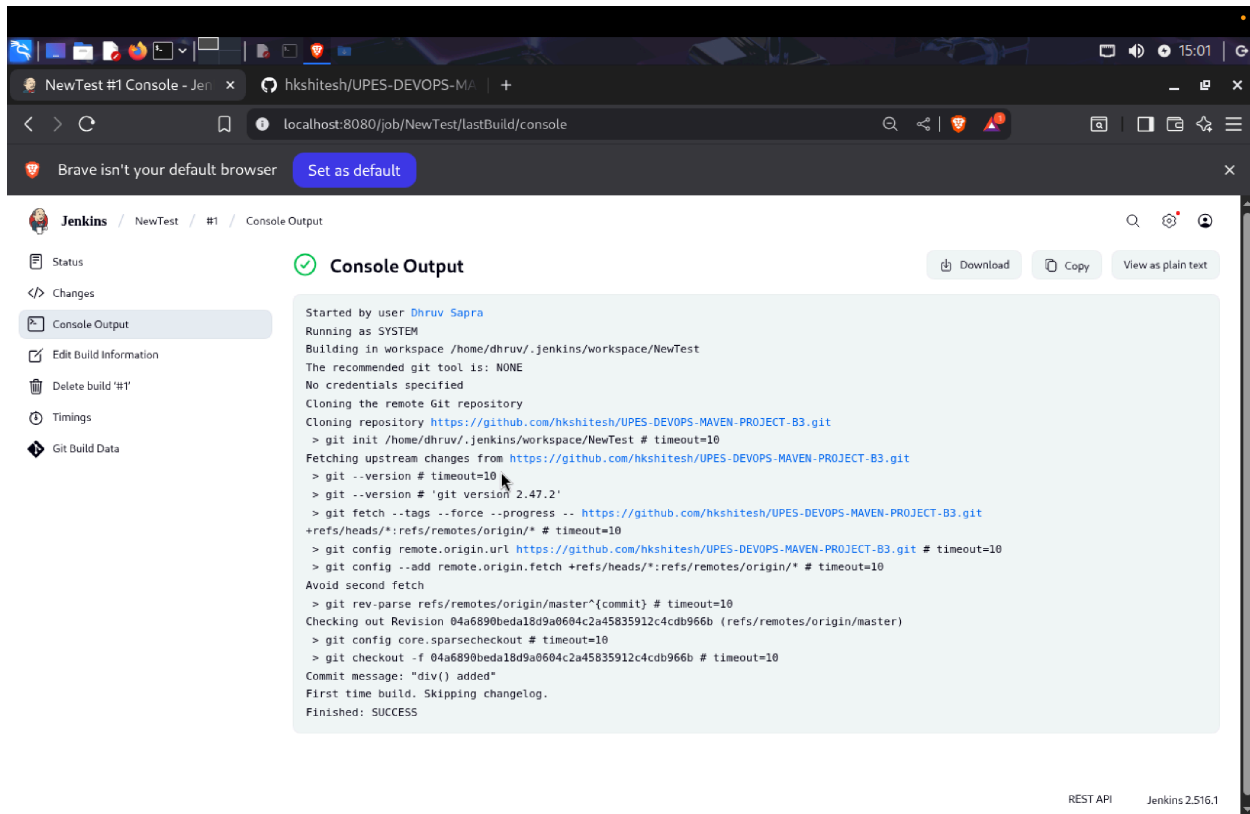
The screenshot shows the Jenkins web interface for a job named 'NewTest'. On the left sidebar, the 'Build Now' button is highlighted with a red rectangle. The main content area shows the job name 'NewTest' and a 'Permalinks' section. Below the sidebar, there is a 'Builds' section showing a single build labeled '#1' at '3:00 PM' with a status icon. The browser's address bar shows 'localhost:8080/job/NewTest/'. At the bottom of the page, the URL 'localhost:8080/job/NewTest/build?delay=0sec' is displayed, along with 'REST API' and 'Jenkins 2.516.1'.

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



The screenshot shows the Jenkins web interface for the 'NewTest' job, specifically the build details page for build '#1 (Sep 19, 2025, 3:00:24 PM)'. The left sidebar shows various options like 'Status', 'Changes', 'Console Output', 'Edit Build Information', 'Delete build '#1'', 'Timings', and 'Git Build Data'. The main content area displays the build status as a green checkmark, the user 'Dhruv Sapra', and the run duration '4.4 sec'. It also shows the revision '04a6890beds18d9a06041245835912c4cd966b' and the repository 'https://github.com/hkshitesh/UPES-DEVOPS-MAVEN-PROJECT-B3.git'. The browser's address bar shows 'localhost:8080/job/NewTest/lastBuild/'. At the bottom of the page, the URL 'localhost:8080/job/NewTest/build?delay=0sec' is displayed, along with 'REST API' and 'Jenkins 2.516.1'.

### 3. Click on **Console Output** to check out the process during the build process



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.