**Lab Experiment 1: Setting Up a Jenkins Job for Maven Build**

**Objective: Create a Jenkins job that builds a Maven project using Jenkins and triggers the build on changes in the version control repository.**

**Prerequisites:**

* Jenkins server up and running.
* Maven installed on the Jenkins server.
* A Maven project hosted in a version control repository (e.g., Git).

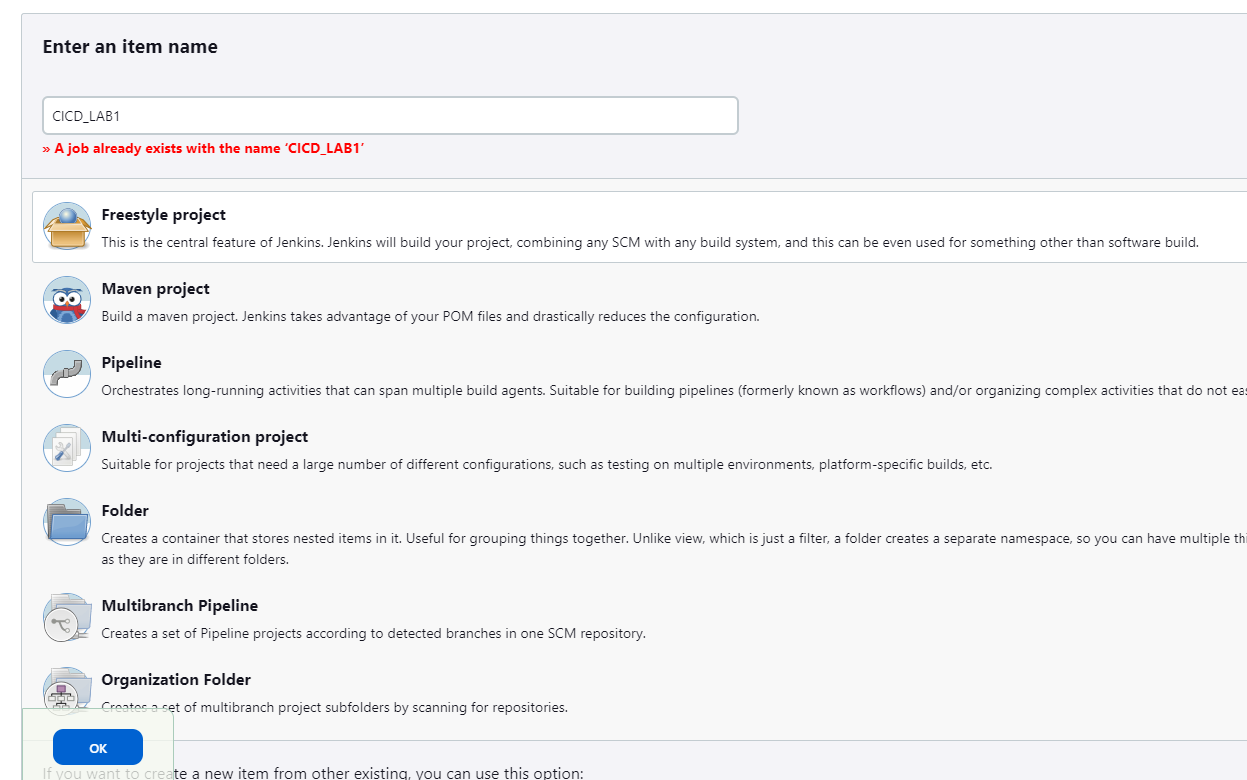
**Steps:**

**Jenkins Configuration:**

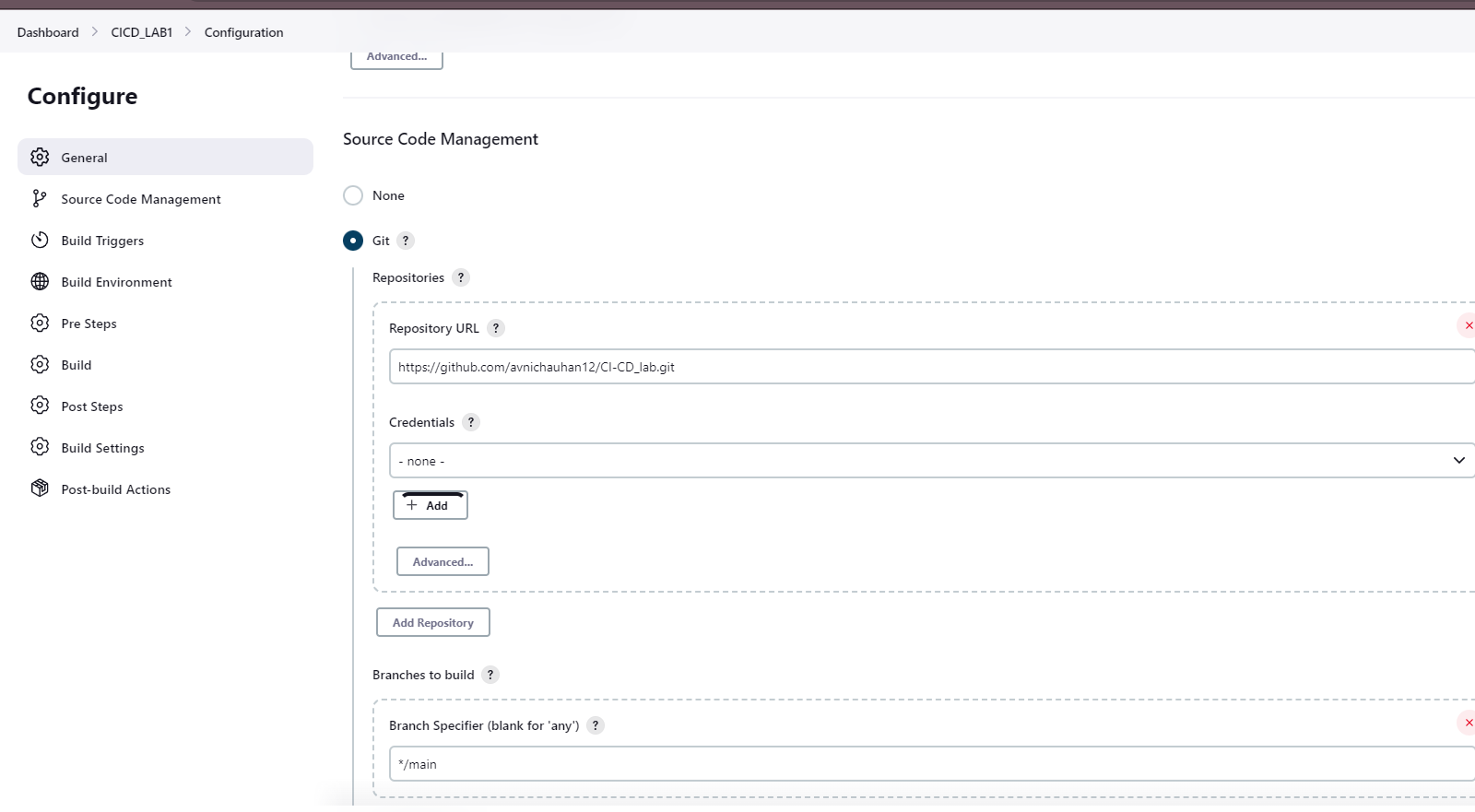
* Ensure that Jenkins is installed and accessible.
* Install necessary plugins: Maven Integration Plugin.

**Creating a Jenkins Job:**

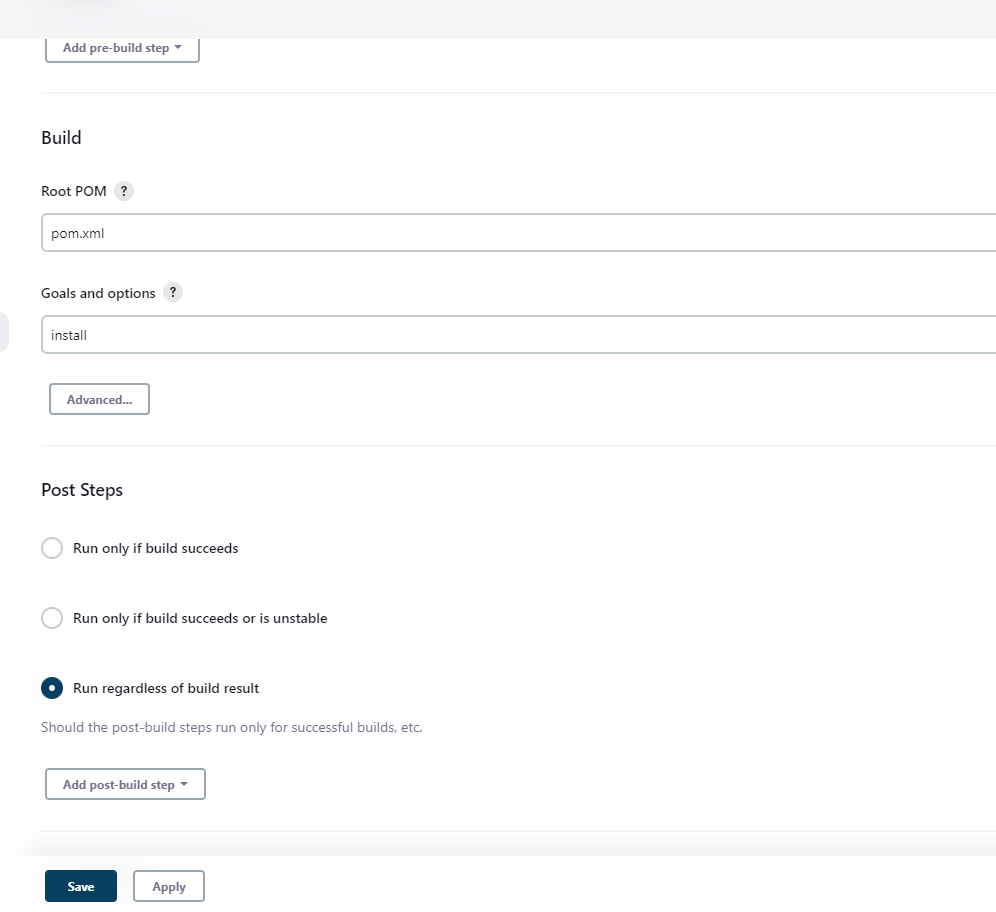
* Log in to your Jenkins instance.
* Click on "New Item" to create a new Jenkins job.
* Enter a name for the job (e.g., "Maven\_Build") and select "Freestyle project."



**Configuring Source Code Management:**

* Under the "Source Code Management" section, choose your version control system (e.g., Git).
* Provide the repository URL and credentials if needed.
* 

**Configuring the Build:**

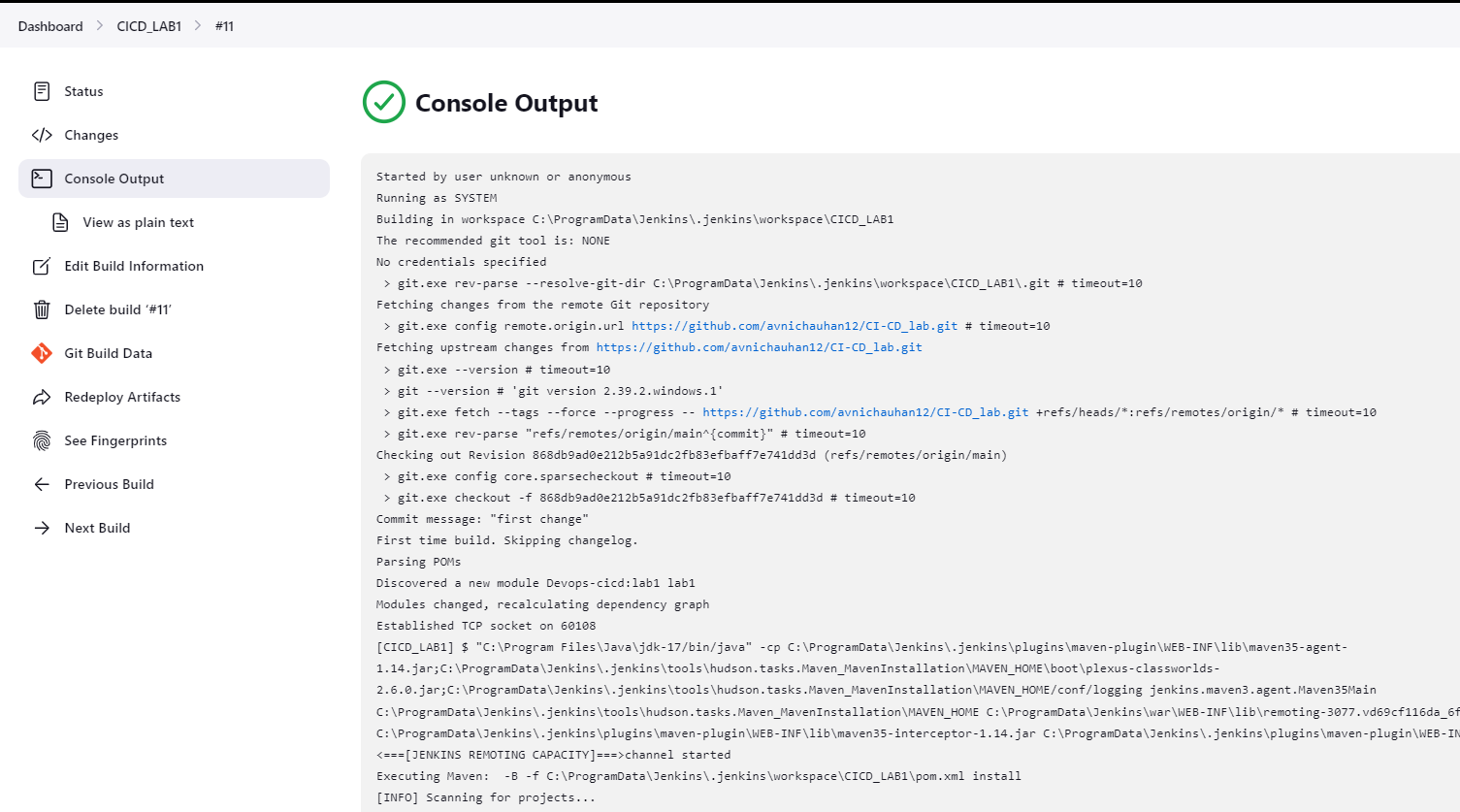
* In the "Build" section, click on "Add build step" and select "Invoke top-level Maven targets."
* In the "Goals" field, enter the Maven goals you want to execute (e.g., "clean install").
* Setting Up Polling for Changes:
* Scroll down to the "Build Triggers" section.
* Choose the option "Poll SCM" and specify the polling schedule (e.g., "\* \* \* \* \*" for polling every minute).
* 

**Save and Run the Job:**

* Click on "Save" to save the job configuration.
* Click on "Build Now" to manually trigger the job initially.

**Observing the Results:**

* Monitor the job's console output to see the Maven build process.
* Check the build status (success/failure) on the Jenkins dashboard.





**Automated Builds on Repository Changes:**

* Make changes to your Maven project and push them to the version control repository.
* Observe that Jenkins automatically triggers the job based on the polling schedule.