Lesson-End Project

Implementing Job Chaining in Jenkins

Project agenda: To build and execute a Jenkins job chaining workflow with the build pipeline plugin and GitHub integration for efficient continuous integration and deployment processes

Description: Imagine a development team at a financial technology company working on a critical Java application that manages customer transactions. The team uses Maven for building their application and Jenkins for continuous integration. The project is hosted on GitHub, and the team wants to build a pipeline that includes tasks such as checking out code, compiling code, and packaging code. In this project, you will create a pipeline in Jenkins where jobs can be set up and chained together by integrating a project hosted on GitHub.

Tools required: Jenkins and GitHub

Prerequisites: You must have Jenkins and GitHub access in the lab to proceed.

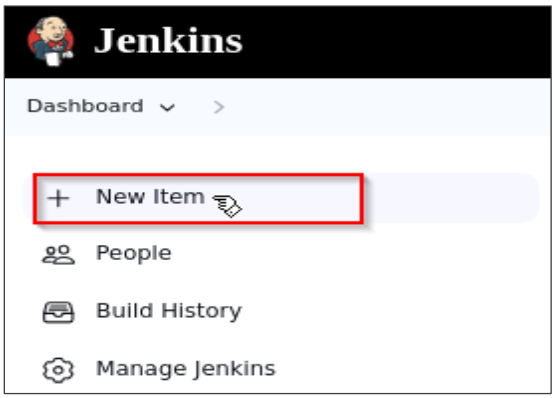
Expected deliverables: A Jenkins pipeline demonstrating job chaining for continuous integration

Steps to be followed:

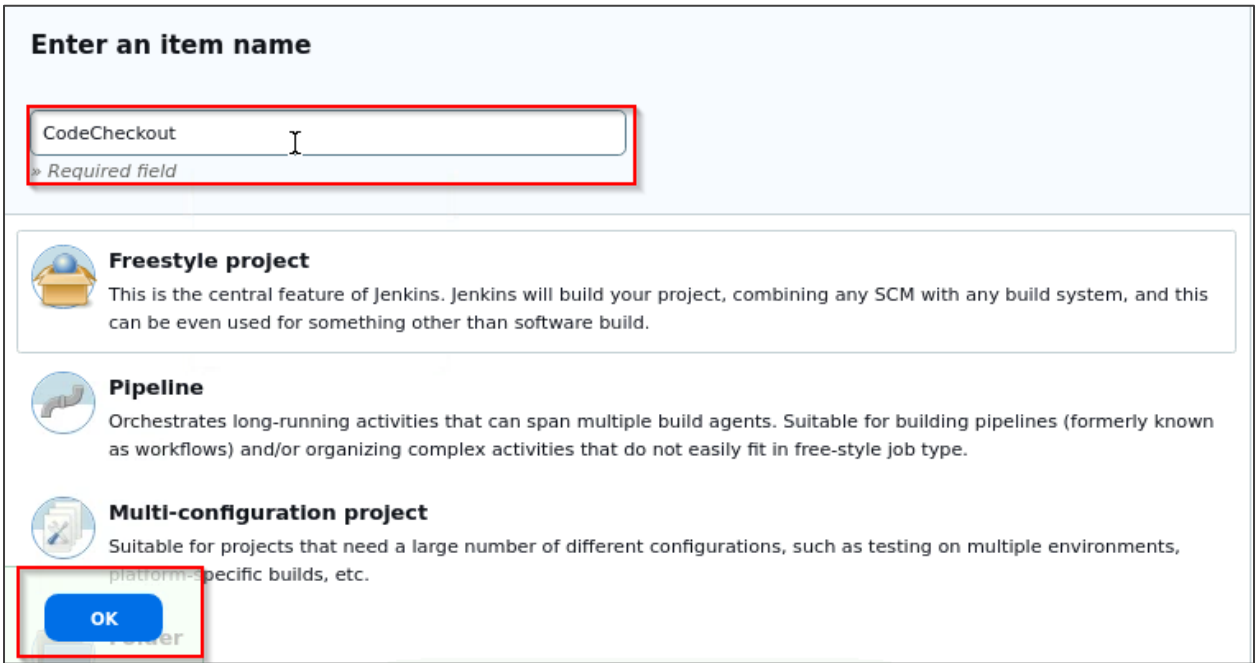
1. Create jobs in a Jenkins pipeline
2. Install the build pipeline plugin in Jenkins
3. Trigger the execution of jobs

Step 1: Create jobs in a Jenkins pipeline

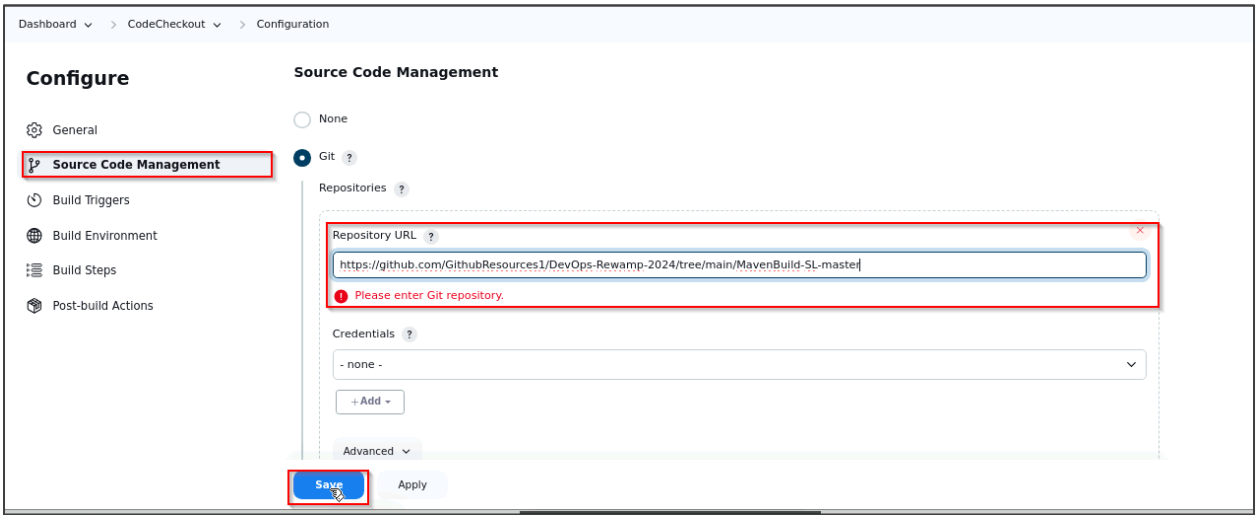
1.1 Click on the **New item** box to create the first job



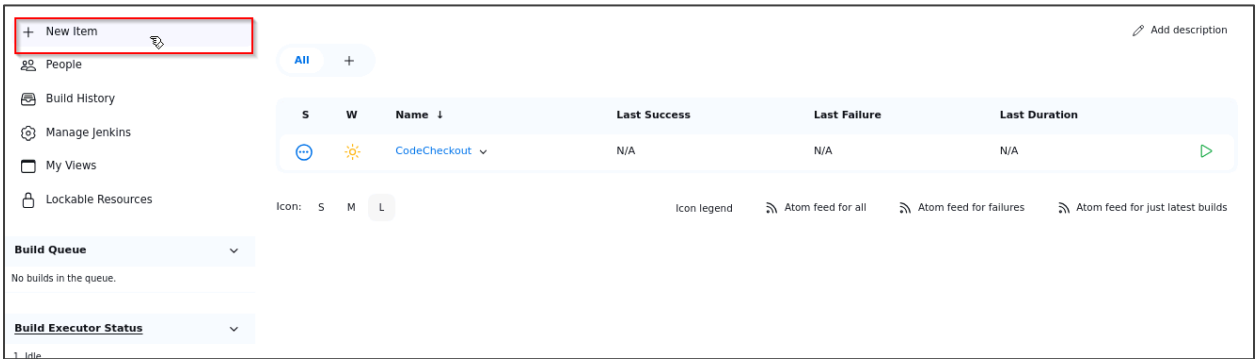
1.2 Enter the item name as **CodeCheckout,** select the type as **Freestyle project**, and click on **OK**



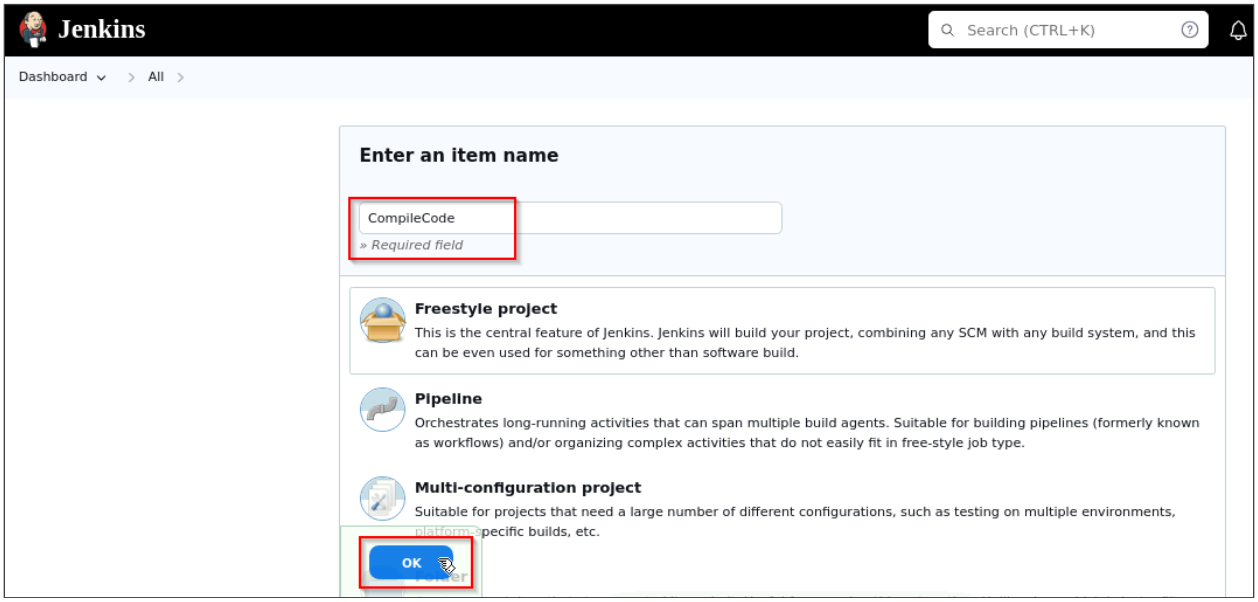
1.3 Click on **Source Code Management** on the job configuration page, select Git, and under Repository URL, give the URL as **https://github.com/GithubResources1/DevOps-Rewamp-2024/tree/main/MavenBuild-SL-master**



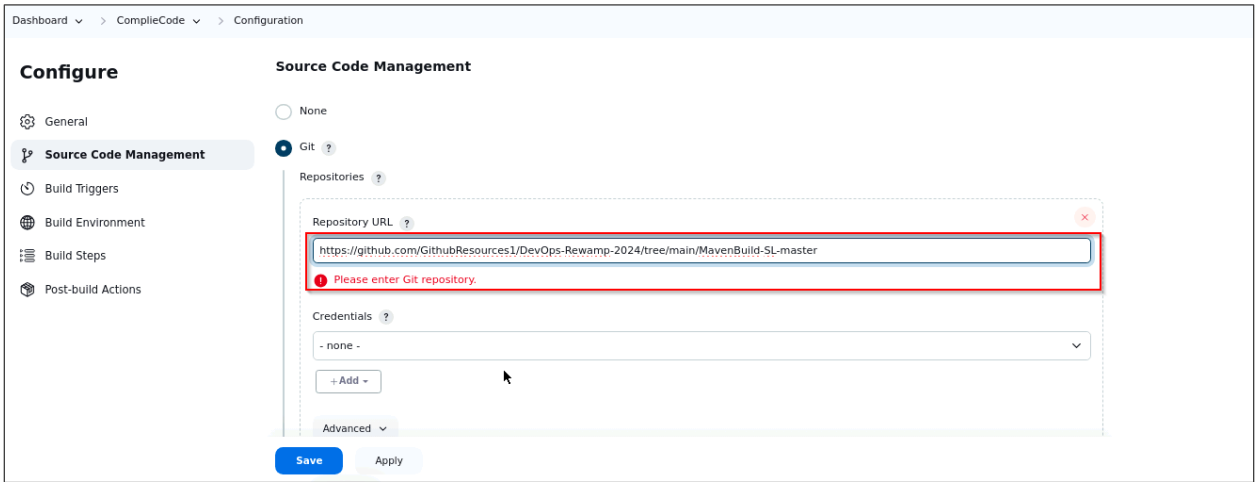
1.4 Go to the Jenkins dashboard and click on the **New item** tab to create the second job



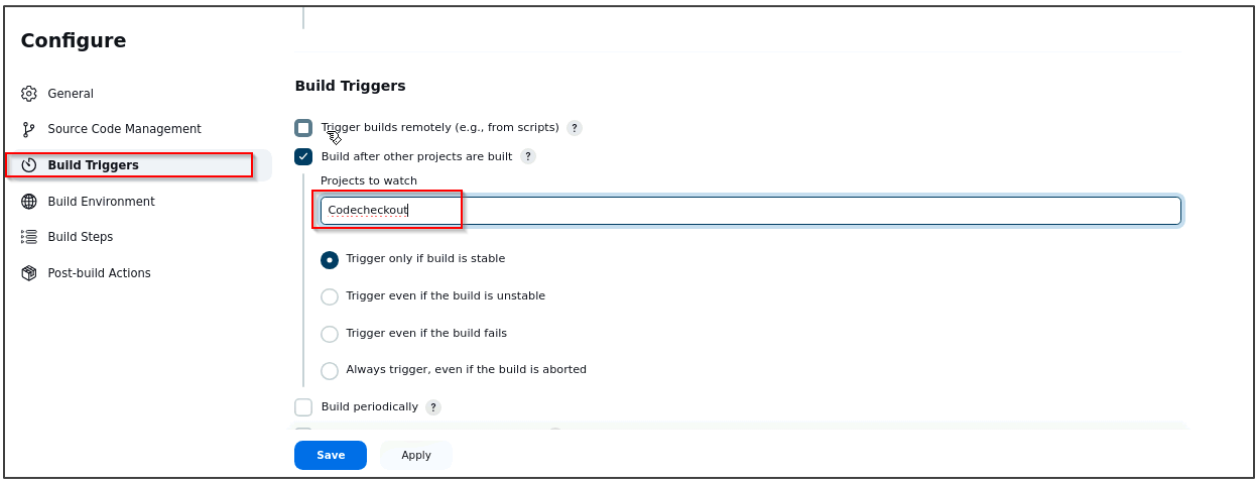
1.5 Enter the item name as **CompileCode,** select the type as **Freestyle project**, and click on **OK**



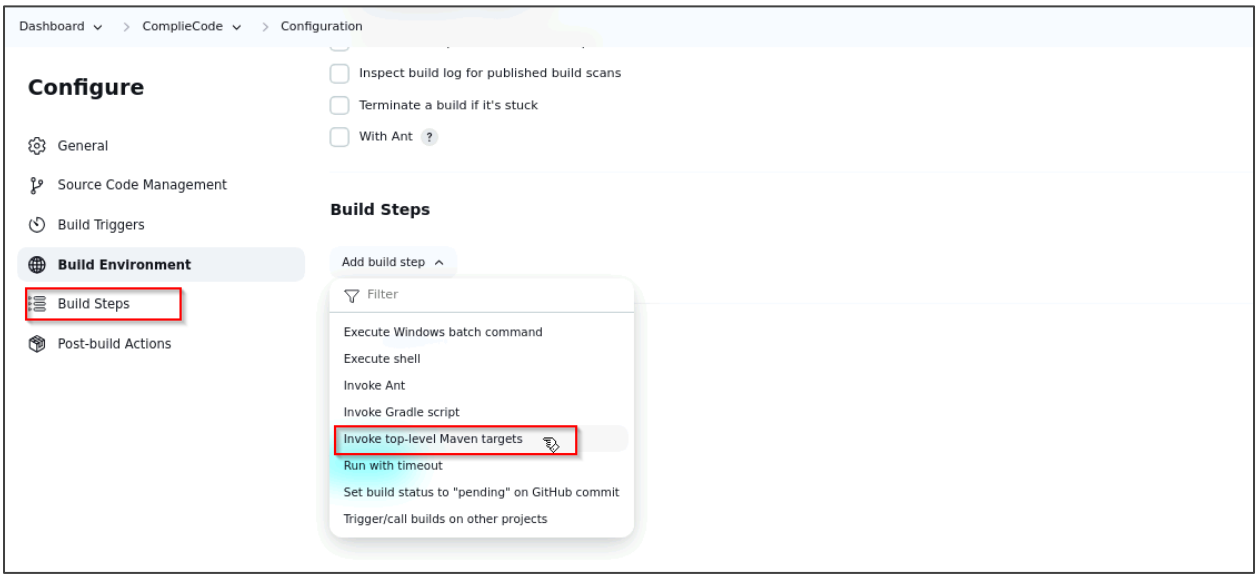
1.6 Click on **Source Code Management** on the job configuration page**,** select Git, and under Repository URL give the URL as **https://github.com/GithubResources1/DevOps-Rewamp-2024/tree/main/MavenBuild-SL-master**



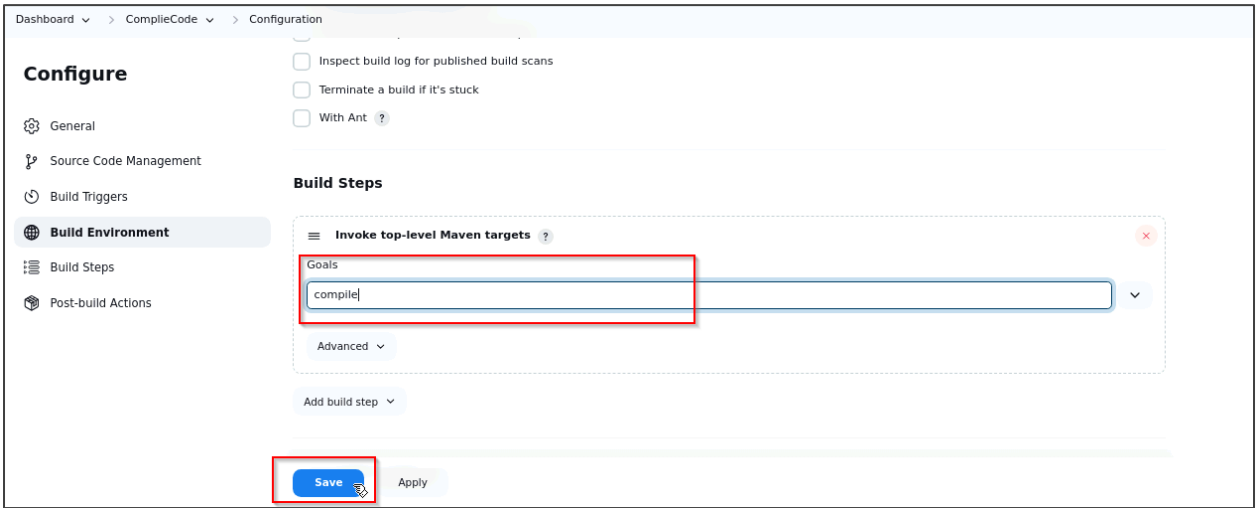
1.7 Click on **Build Triggers**, select the option **Build after other projects are built,** and enter the name as **Codecheckout** in the **Projects to watch** field



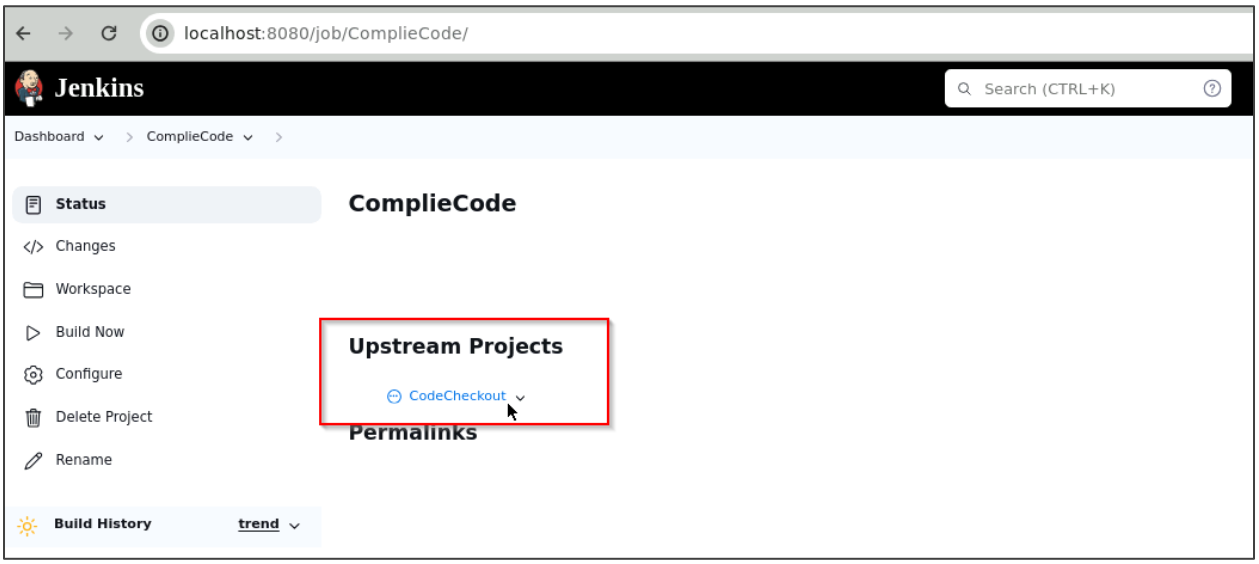
1.8 Click on **Build Steps** and select the option **Invoke top-level Maven targets**



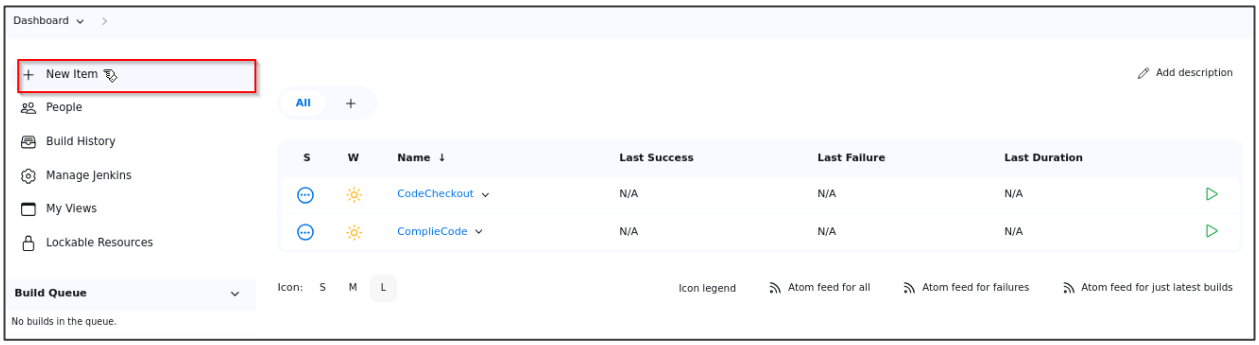
1.9 Enter **compile** as the goal in the **Goals** field and click on **Save**



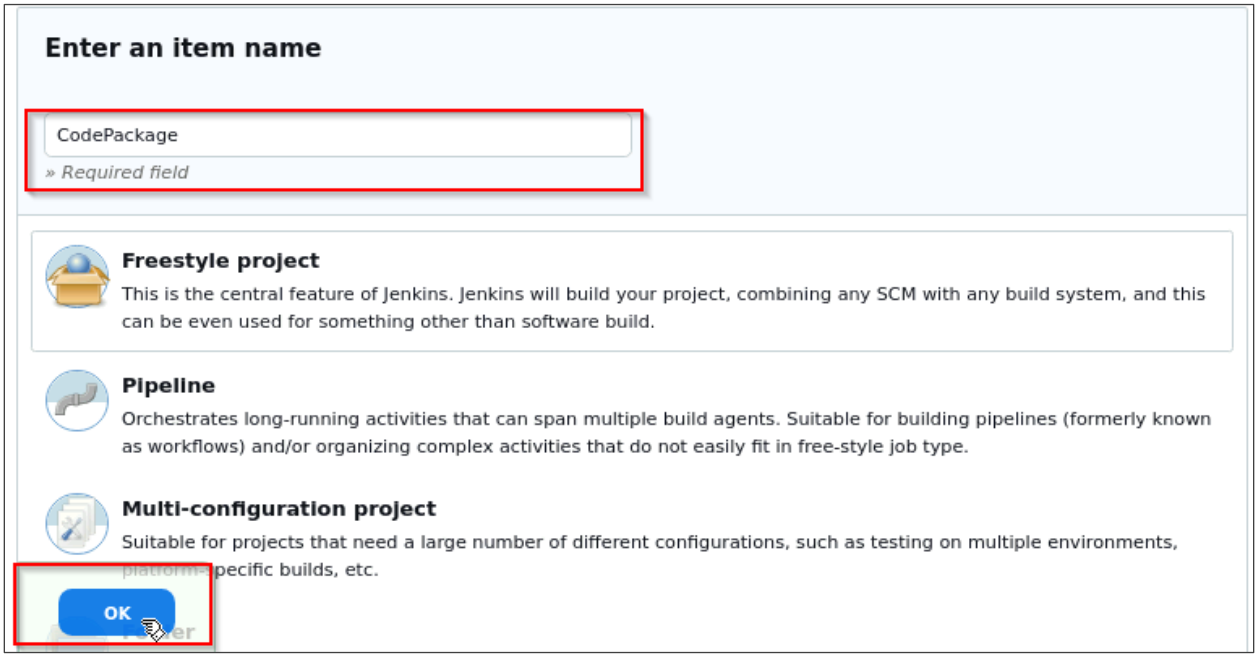
1.10 Refresh the page, and you will see the upstream job added



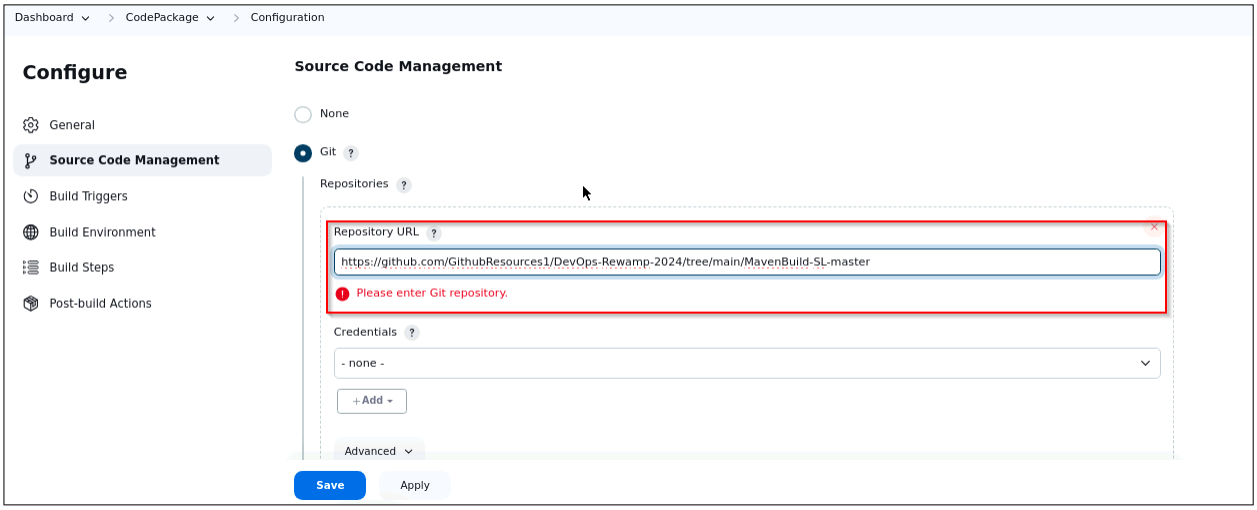
1.11 Go to the Jenkins dashboard and click on the **New item** box to create the third job



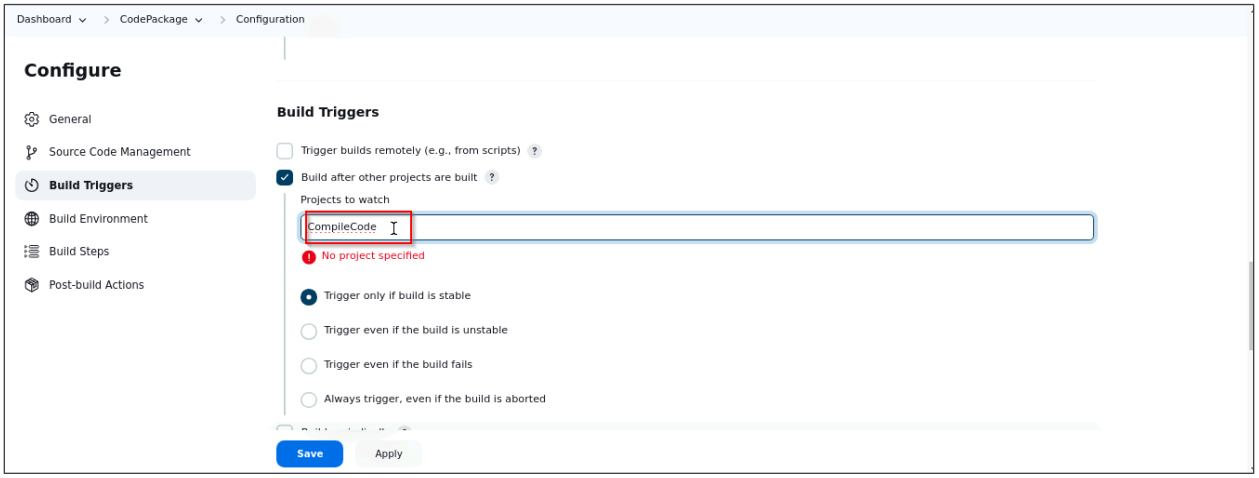
1.12 Enter the item name as **CodePackage,** select the type as **Freestyle project,** and click on **ok**



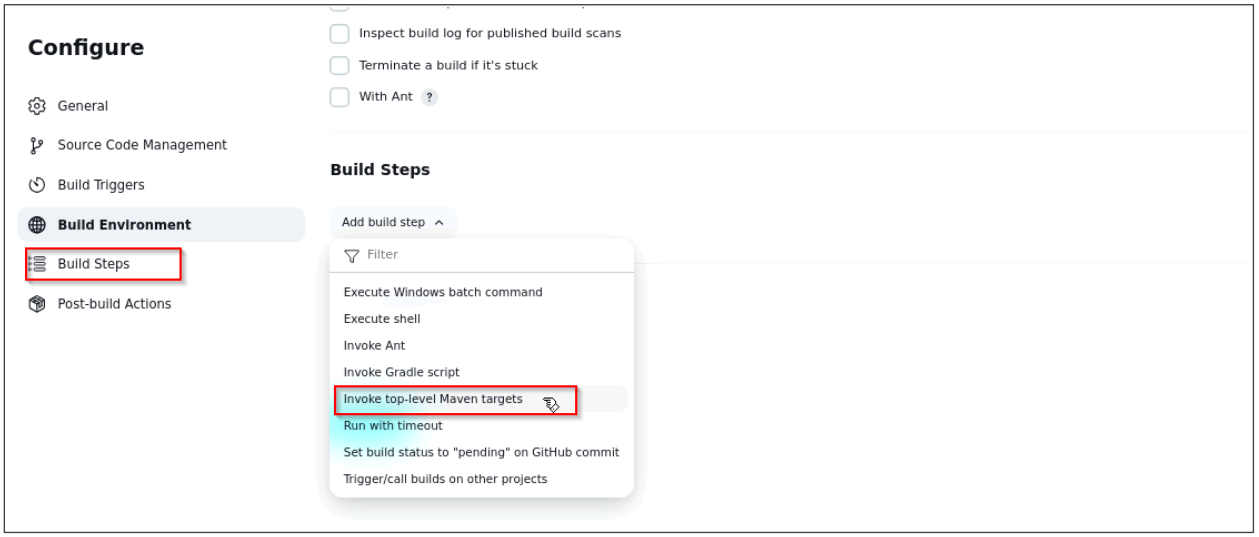
1.13 Click on **Source Code Management** on the configuration screen**,** select Git, and under Repository URL, give the URL as **https://github.com/GithubResources1/DevOps-Rewamp-2024/tree/main/MavenBuild-SL-master**



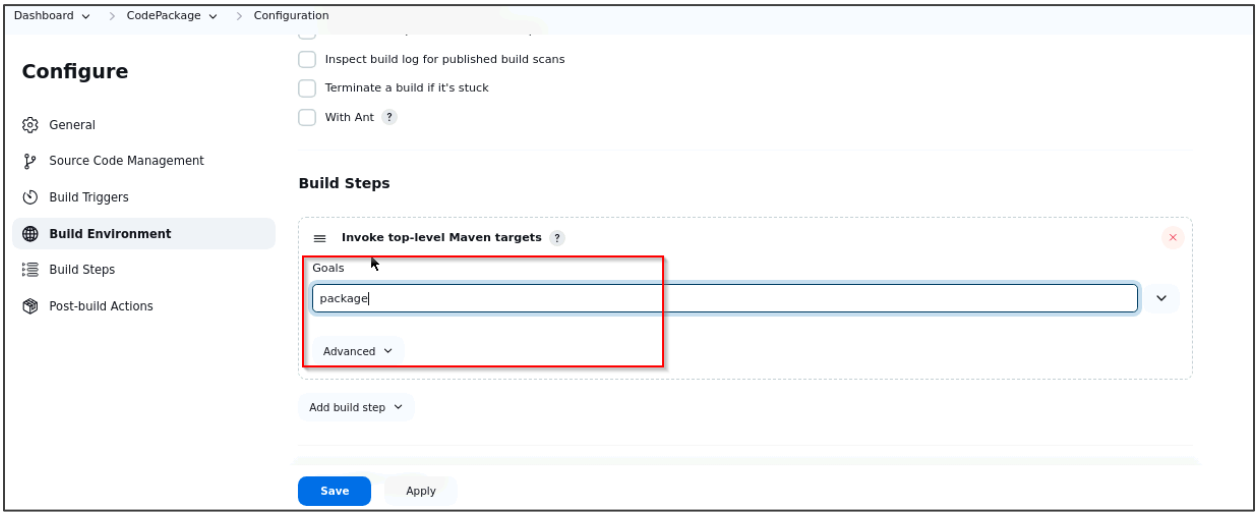
1.14 Click on **Build Triggers**, select the option **Build after other projects are built**, and enter the name as **CompileCode** in the **Projects to watch** field



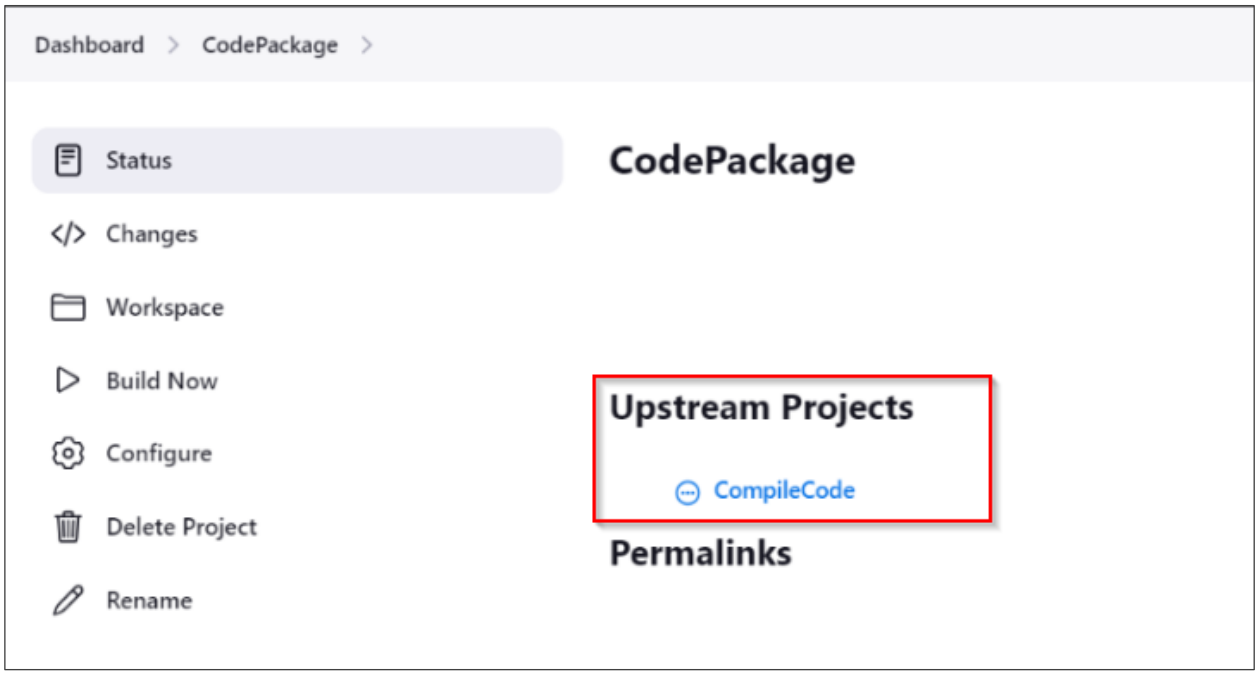
1.15 Click on **Build Steps** and select the option **Invoke top-level Maven targets**



1.16 Enter the goal as **package** in the **Goals** field and click on **Save**

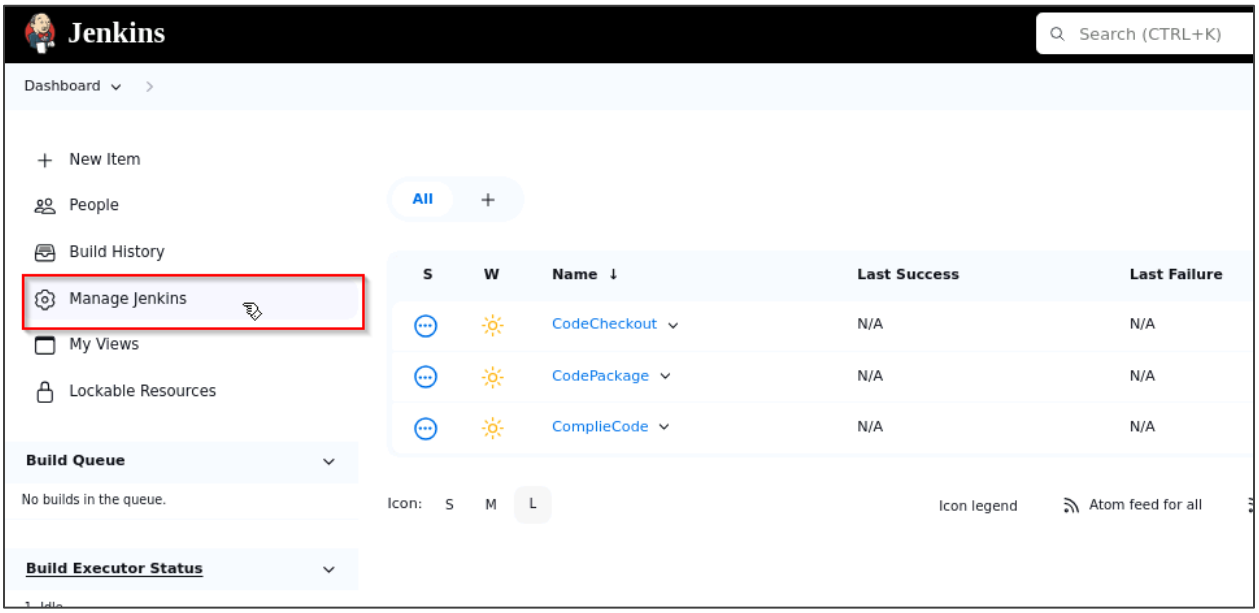


1.17 Refresh the page, and you will see the upstream job added

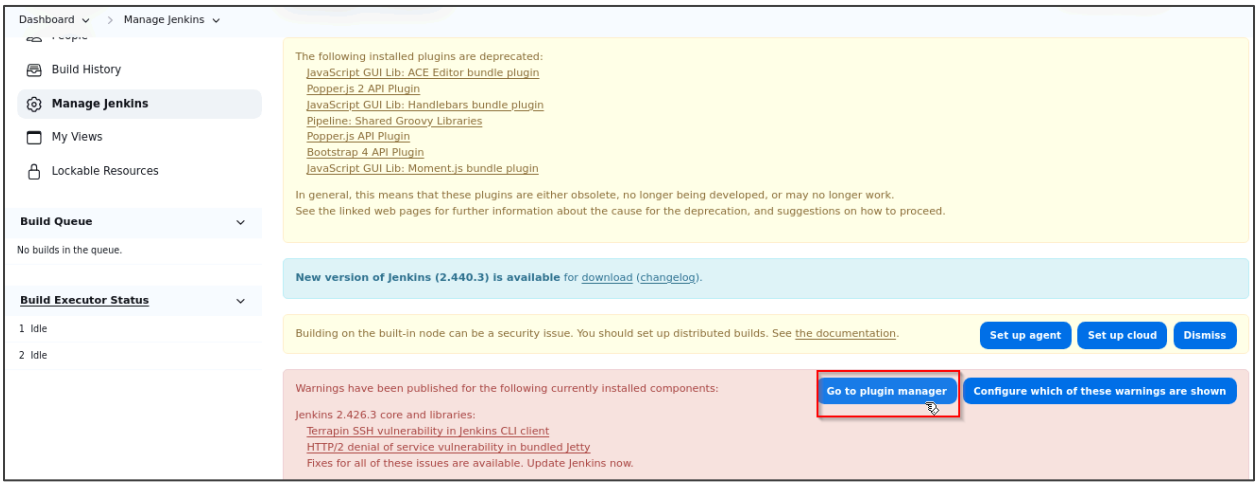


Step 2: Install the build pipeline plugin in Jenkins

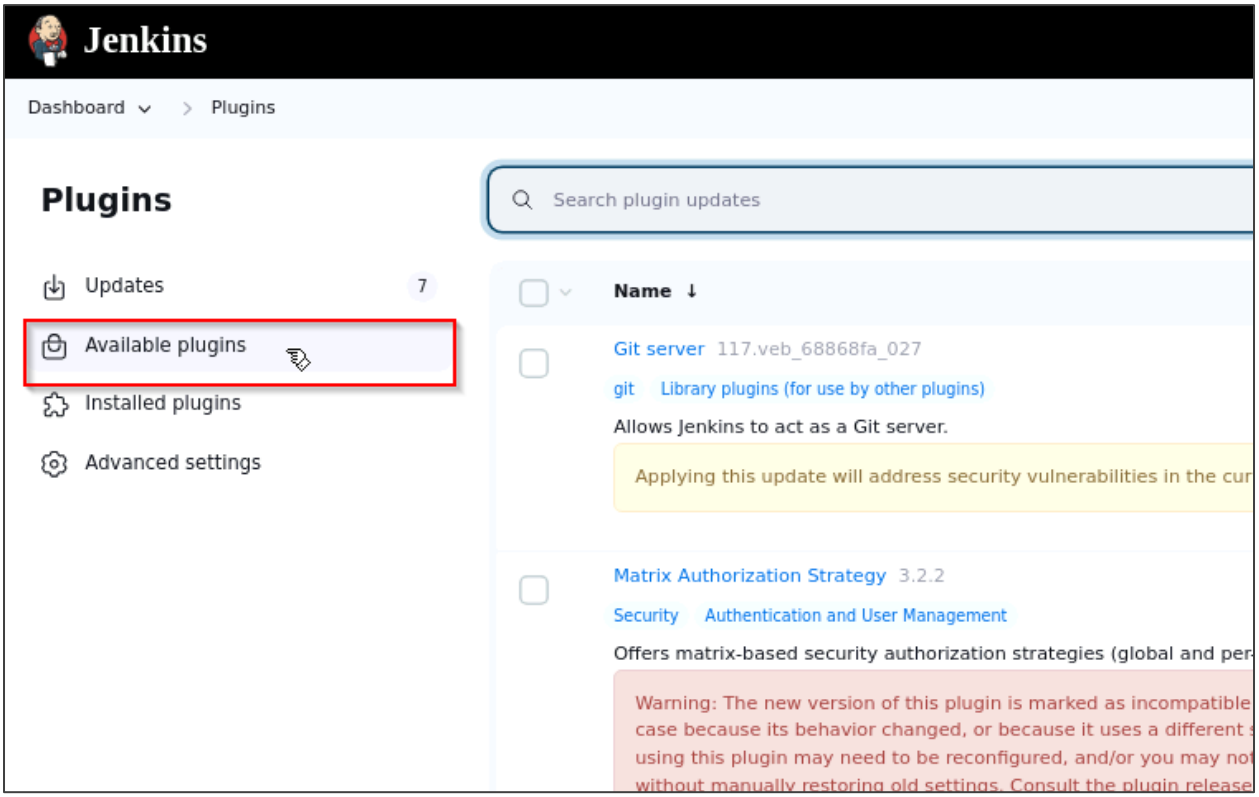
2.1 Go to the **Dashboard** and click on **Manage Jenkins**



2.2 Click on **Go to plugin manager**

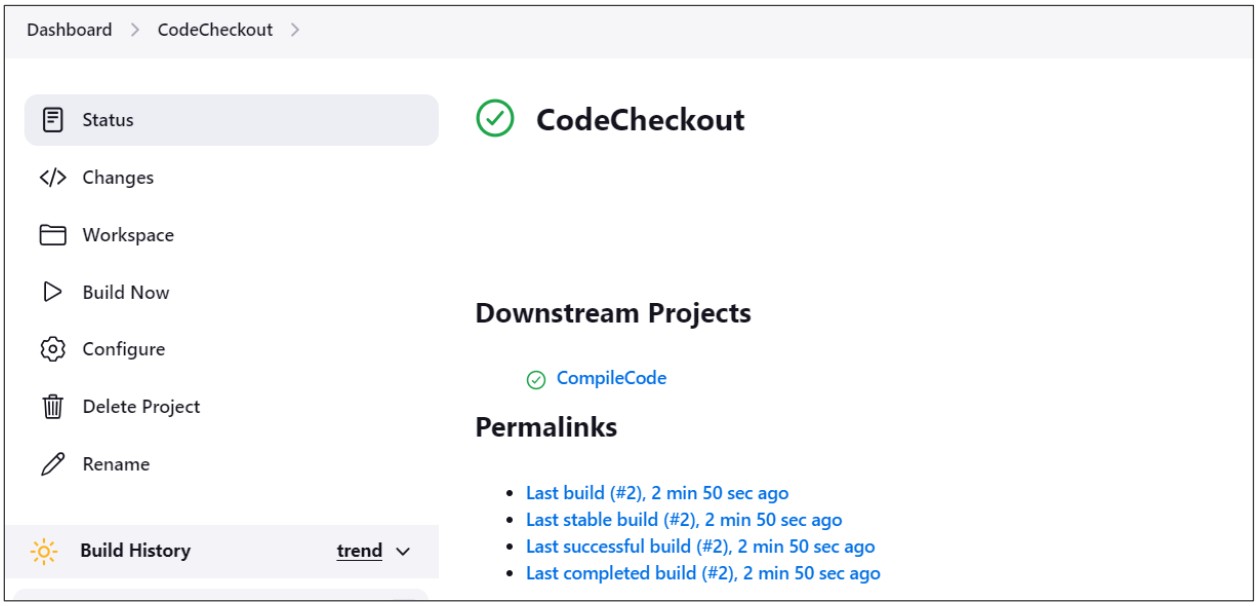


2.3 Select the **Available plugins** option

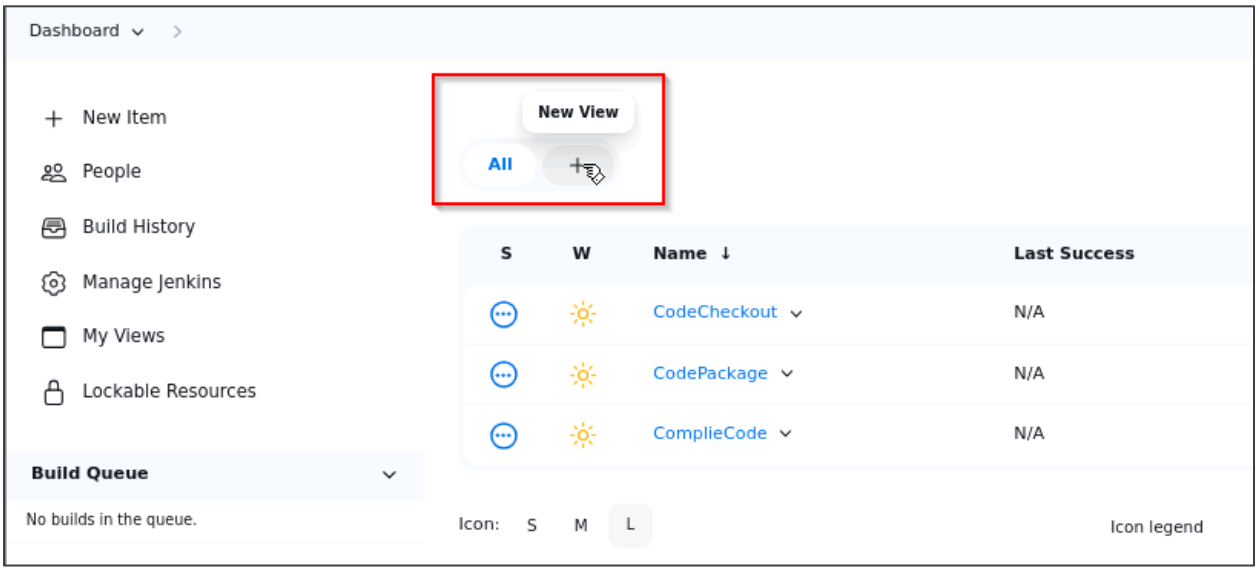


2.4 Search for the **Build Pipeline** plugin, select it, and click the **Install** button

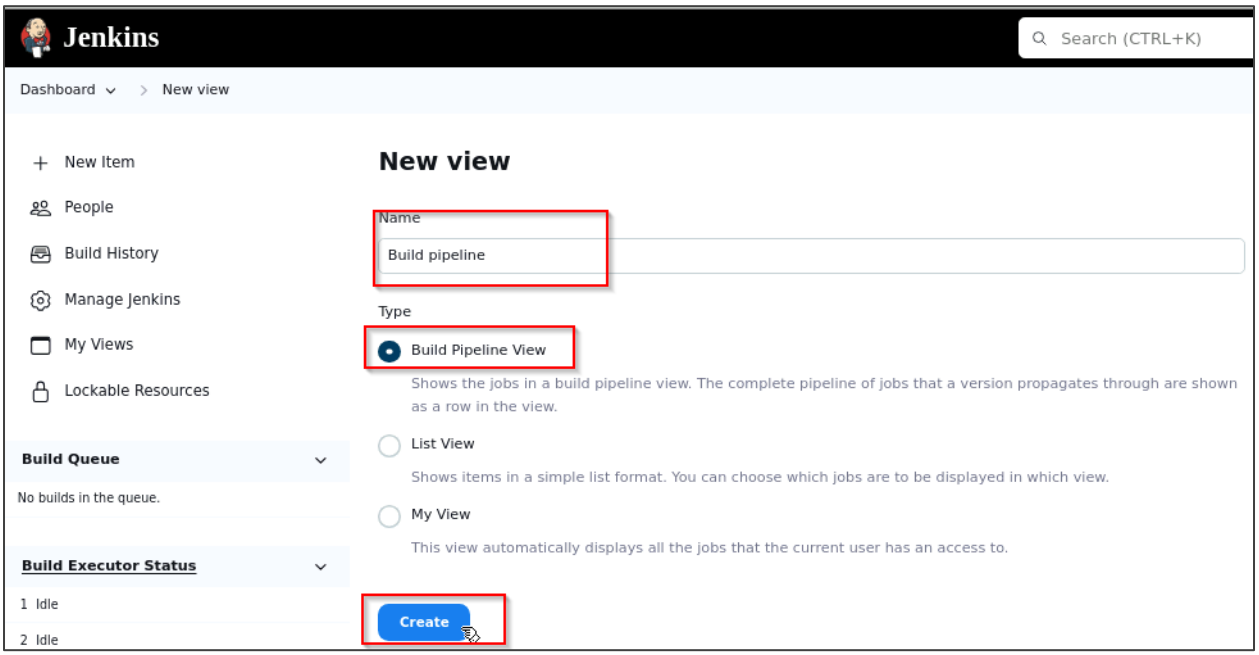
The build pipeline plugin will be installed.



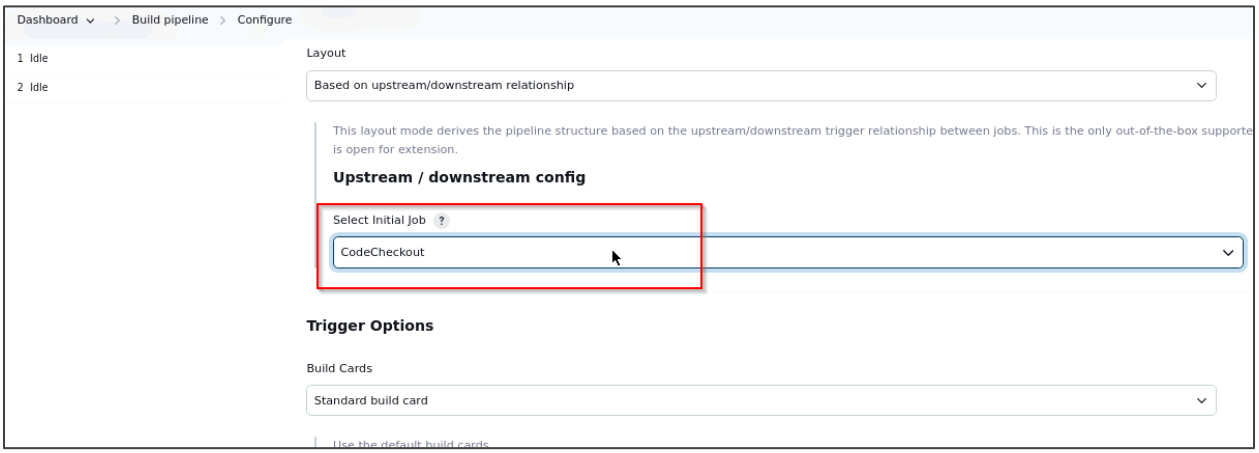
2.5 Go to the Jenkins dashboard and click on the **New View** tab



2.6 Enter the name as **Build pipeline** in the **New View** tab, select the type as **Build Pipeline View,** and click on **Create**



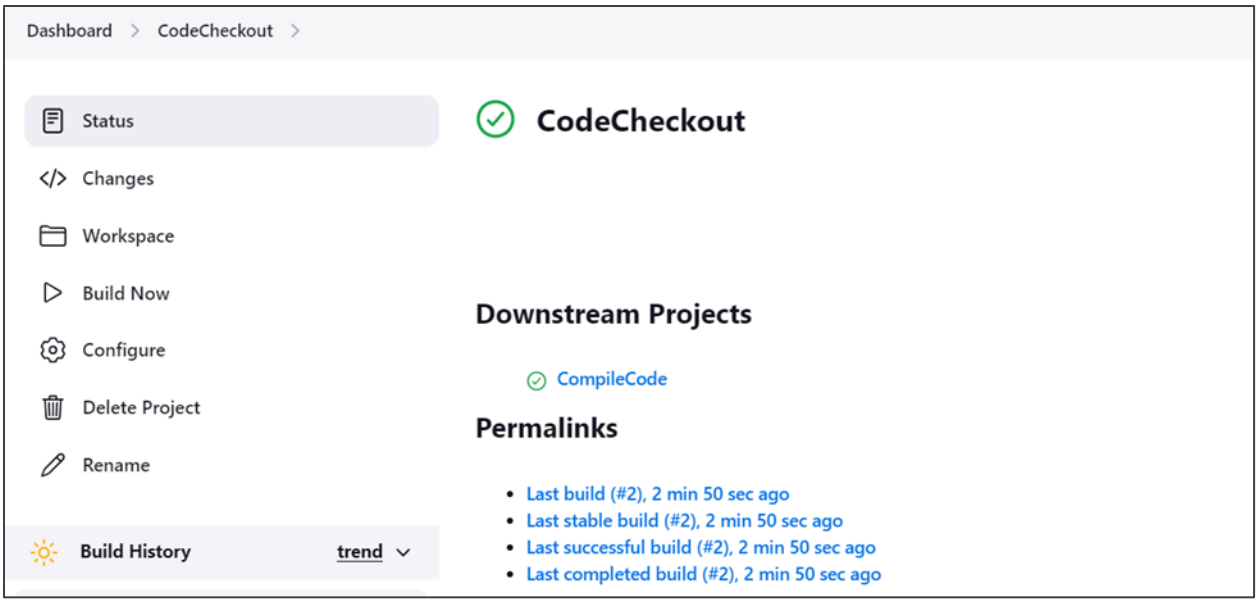
2.7 Scroll down on the configure page until you see the **Select Initial Job** drop-down. In this, **select Initial Job** as **CodeCheckout**



2.8 Click on the **OK** button

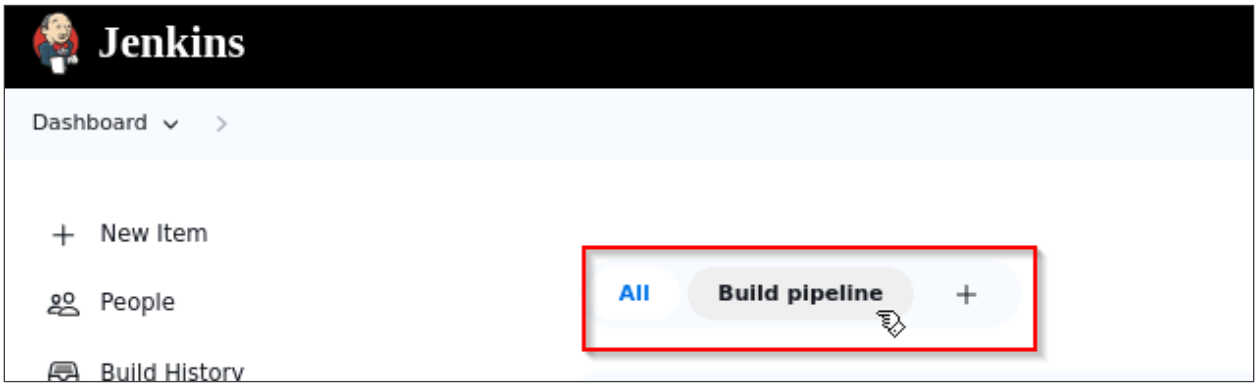
Step 3: Trigger the execution of jobs

3.1 Click on the dashboard and trigger the execution of the first job, which is **CodeCheckout**

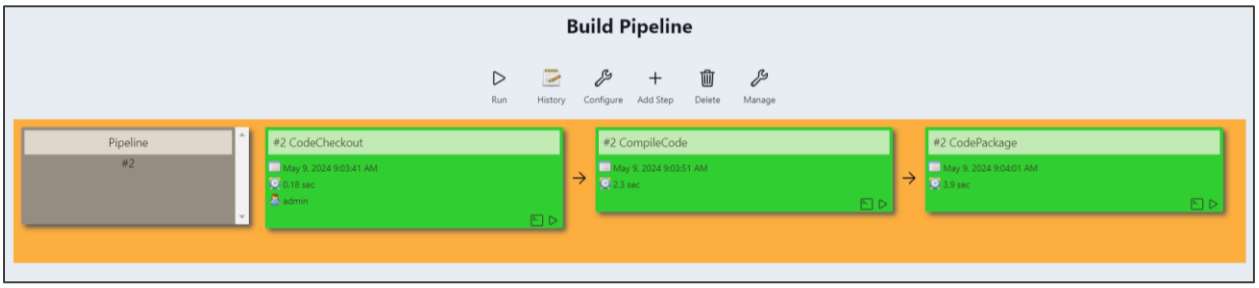


The job will run successfully, and it will automatically trigger the execution of the second job which in turn will trigger the execution of the third job sequentially.

3.2 Click on the Jenkins dashboard, select the **build pipeline**, and refresh the Jenkins page



You will be able to see the plugin-based build pipeline



By following these steps, you have successfully built and executed a Jenkins job chaining workflow with the build pipeline plugin and GitHub integration for efficient continuous integration and deployment processes.