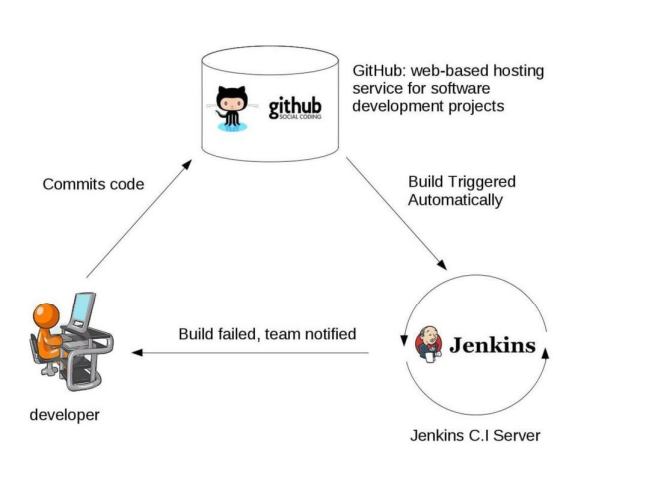
**JENKINS**

* Jenkins is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.
* Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.
* In Continuous Integration after a code commit, the software is built and tested immediately.
* In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for deployment.
* If deployment is a success, the code is pushed to production. This commit, build, test, and deploy is a continuous process and hence the name continuous integration/deployment.



How Jenkins works:

* Jenkins is distributed as a WAR archive and as installer packages for the major operating systems, as a Homebrew package, as a Docker image, and as [source code](https://github.com/jenkinsci/jenkins). Jenkins also supports [installation and scaling on Kubernetes](https://www.jenkins.io/doc/book/installing/kubernetes/). The source code is mostly Java, with a few Groovy, Ruby.
* You can run the Jenkins WAR standalone or as a servlet in a Java application server such as Tomcat. In either case, it produces a web user interface and accepts calls to its REST API.
* When you run Jenkins for the first time, it creates an administrative user with a long random password, which you can paste into its initial web page to unlock the installation.

## Jenkins for CI/CD

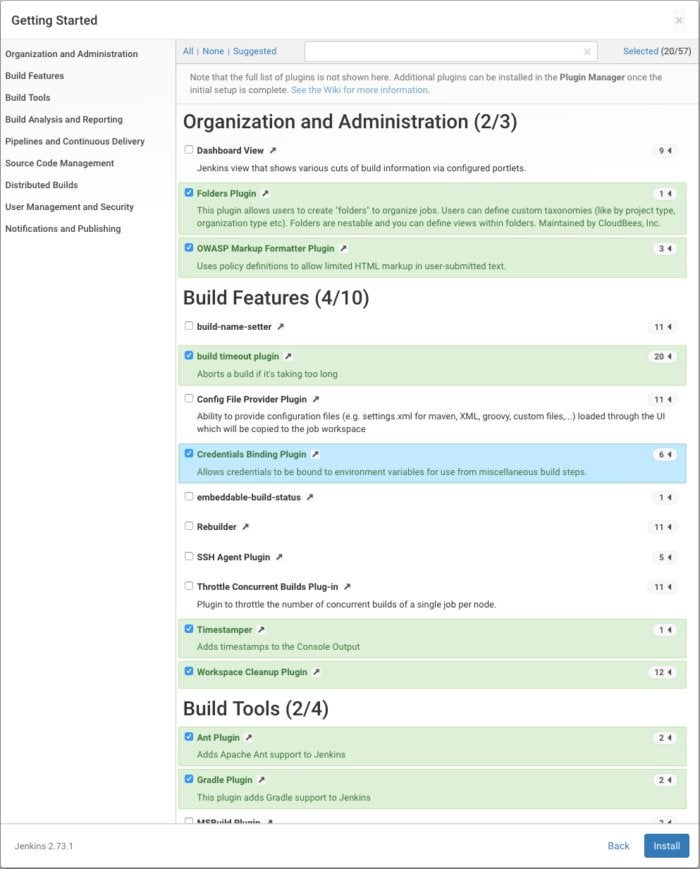
* Overall, Jenkins offers a simple way to set up a CI/CD environment for pretty much any combination of languages and source code repositories using pipelines, as well as automating several other routine development tasks.
* While Jenkins does not eliminate the need to create scripts for individual steps, it does give you a quicker and more robust way to integrate your entire chain of build, test, and deployment tools than you could easily build yourself.

**Why uses Jenkins?**

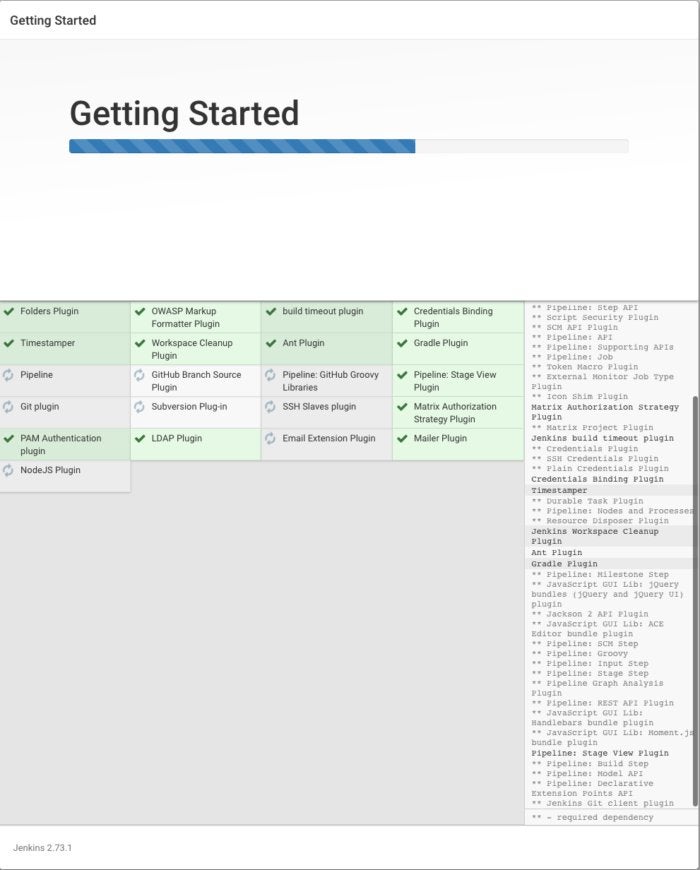
* The Jenkins Pipeline plugin we have been using supports a general continuous integration/continuous delivery (CICD) use case, which is probably the most common use for Jenkins. There are specialized considerations for some other use cases.
* Android runs a kind of Java, but introduces the issue of how to test on the wide range of Android devices. The Android emulator plugin allows you to build and test on as many emulated devices as you care to define. The Google Play Publisher plugin lets you send builds to an alpha channel in Google Play for release or further testing on actual devices.
* I have shown examples where we specified a Docker container as the agent for a pipeline and where we ran Jenkins and Blue Ocean in a Docker container. [Docker containers](https://jenkins.io/solutions/docker/) are very useful in a Jenkins environment for improving speed, scalability, and consistency.
* There are two major use cases for Jenkins and GitHub. One is building integration, which can include a service hook to trigger Jenkins on every commit to your GitHub repository. The second is the use of GitHub authentication to control access to Jenkins via OAuth.
* Jenkins supports many other languages besides Java. For C/C++, there are plugins to capture errors and warnings from the console, generate build scripts with Make, run unit tests, and perform static code analysis. Jenkins has a few integrations with PHP tools.

Jenkins plugins:

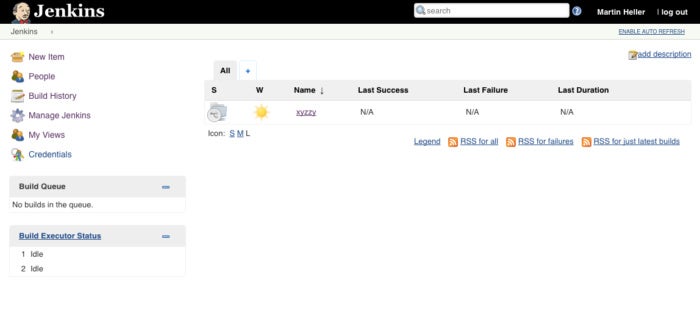
Once installed, Jenkins allows you to either accept the default plugin list or choose your own plugin.



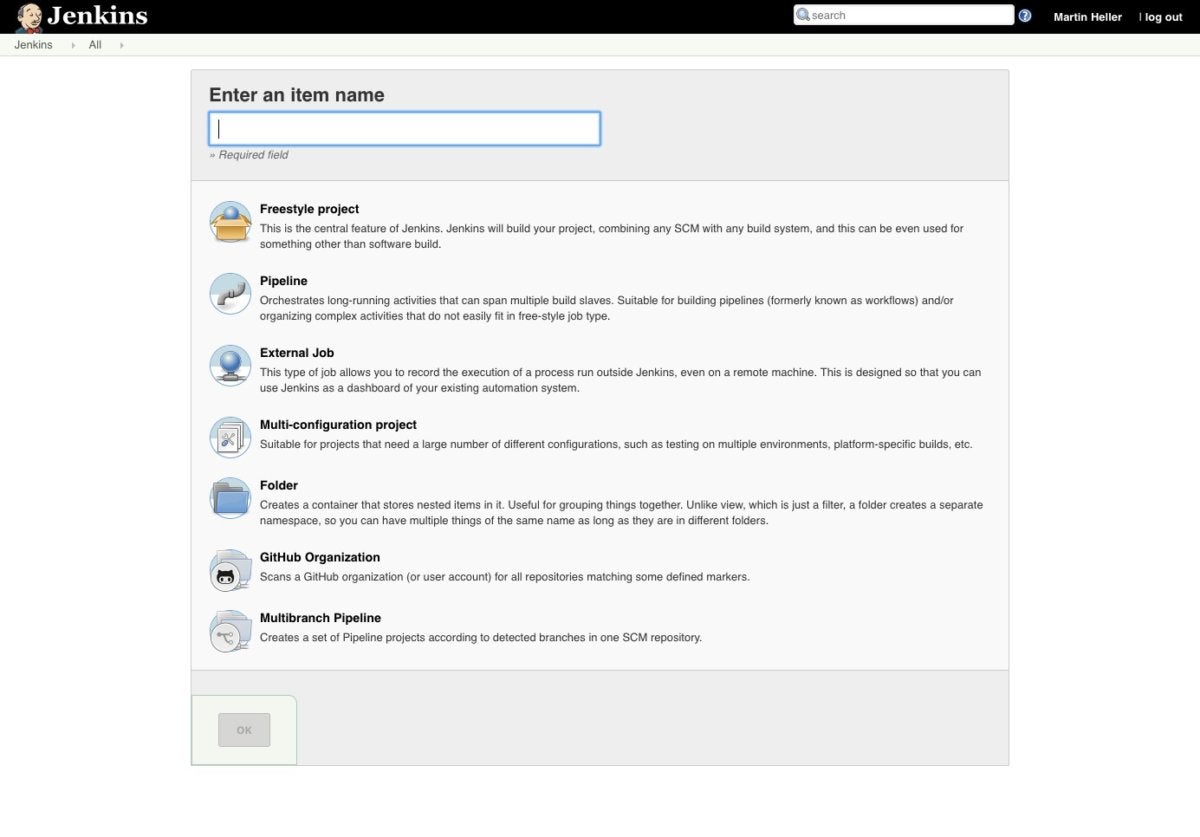
Once you have picked your initial set of plugins, click the Install button and Jenkins will add them.



The Jenkins main screen displays the current build queue and Executor status, and offers links to create new items (jobs), manage users, view build histories, manage Jenkins, look at your custom views, and manage your credentials.



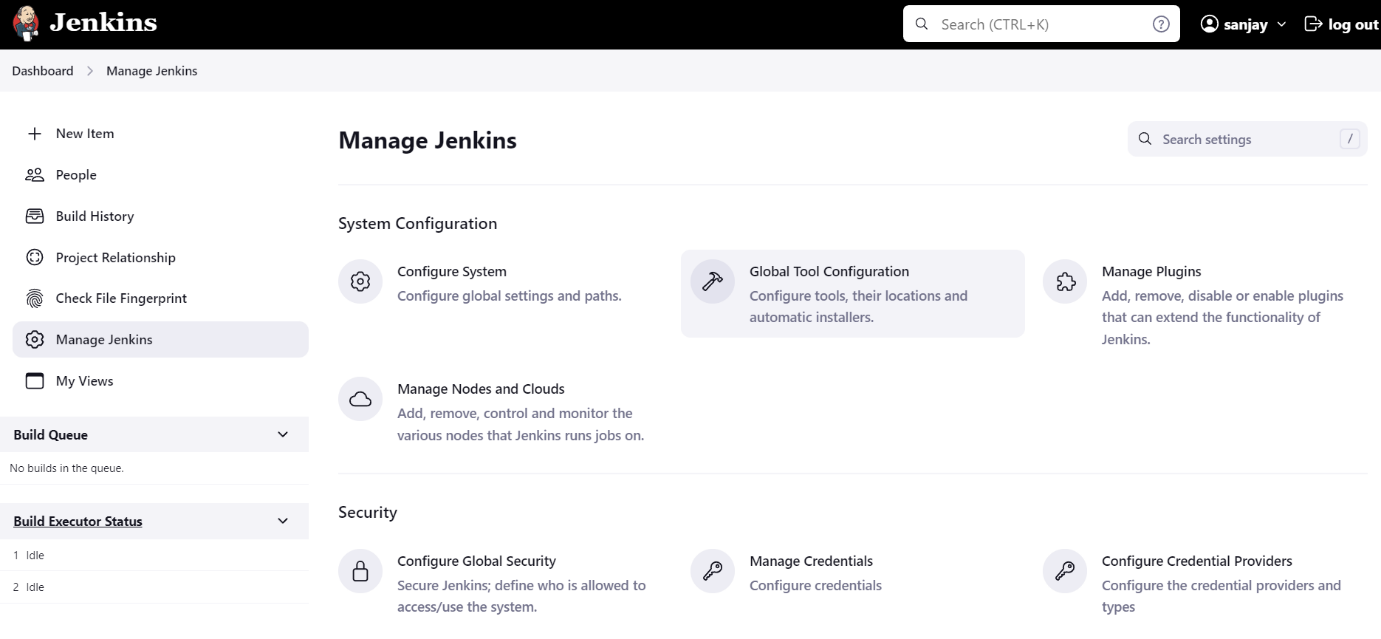
A new Jenkins item can be any of six types of job plus a folder for organizing items.



**JENKINS PLUGINS:**

* By default, Jenkins comes with a limited set of features. If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git.

**FREESTYLE PROJECT:**



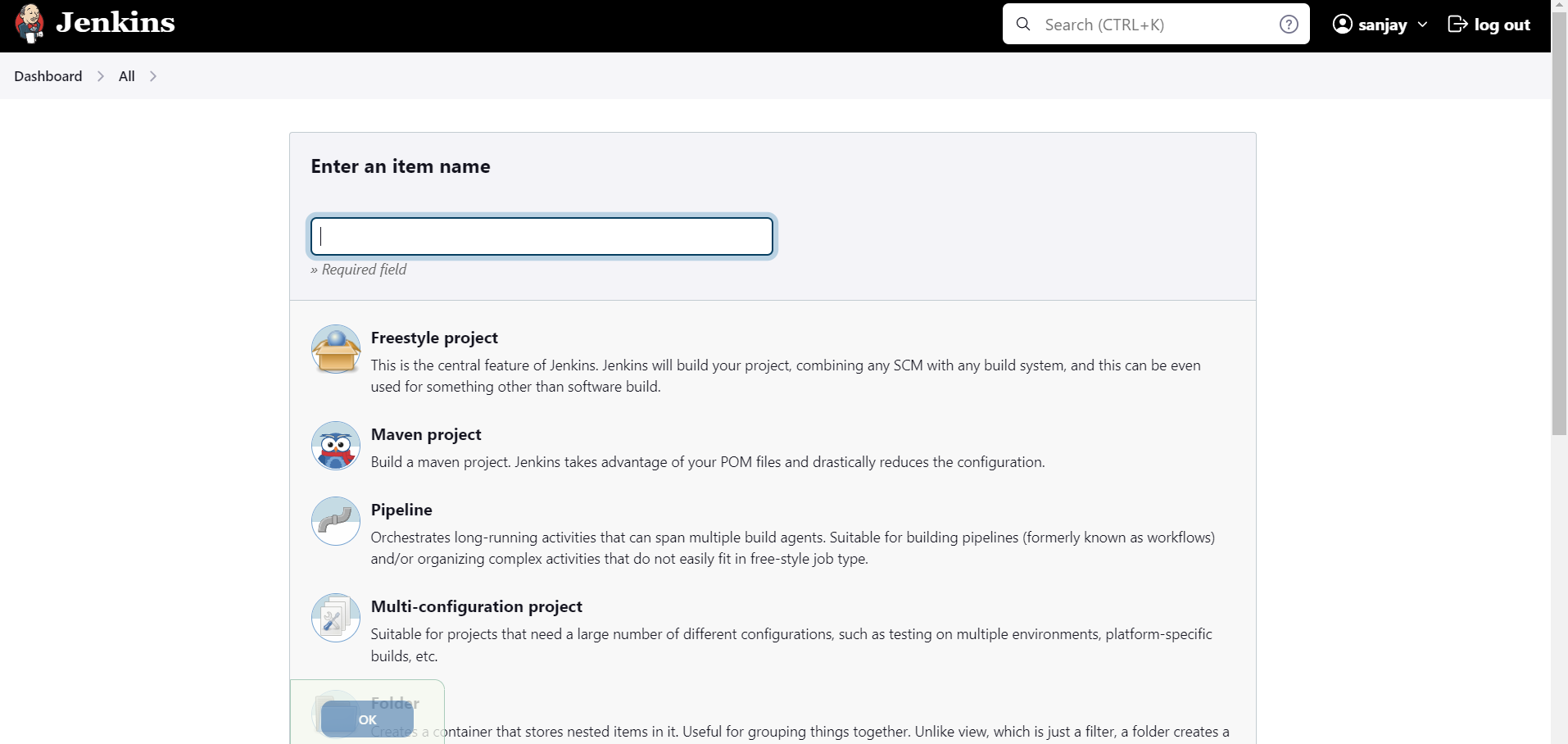
**Jenkins Plugin management: -**

>> Manage Jenkins Manage Plugins Available >> type your required package name >> install without restart.

Create a Jenkins job:

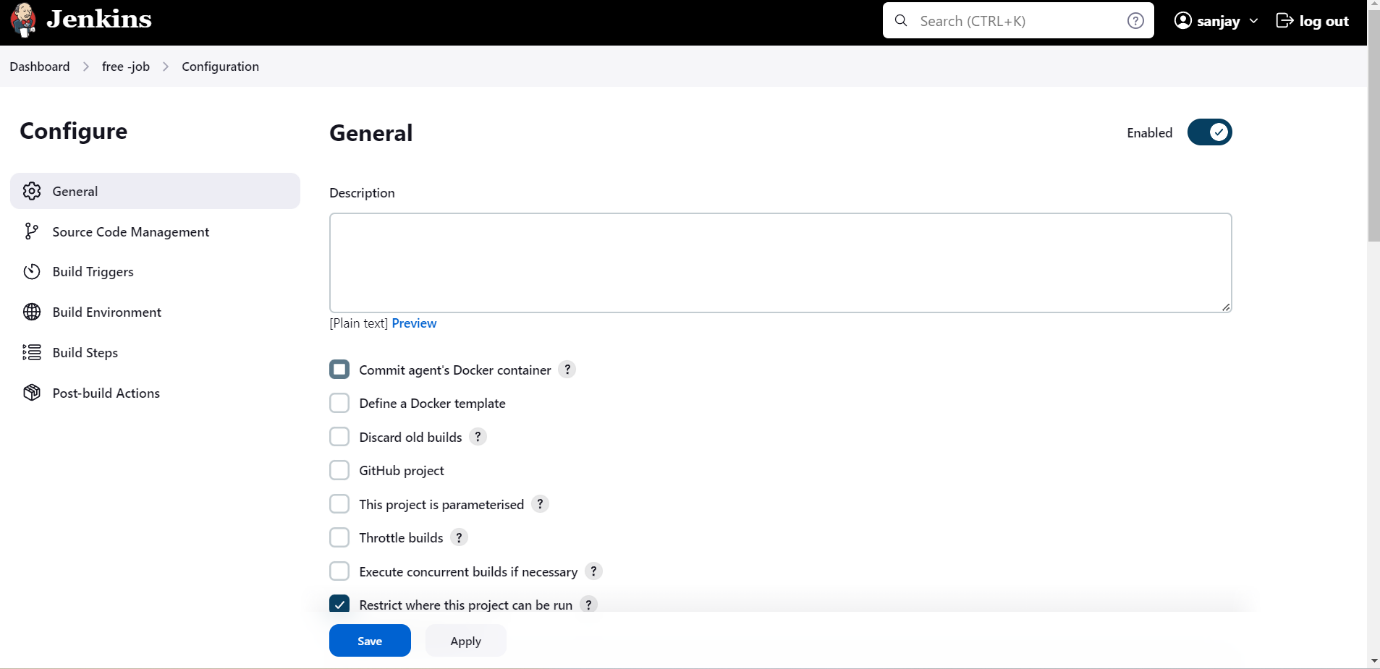
In Dashboard, Click the item button

Enter the item name and add a freestyle project.

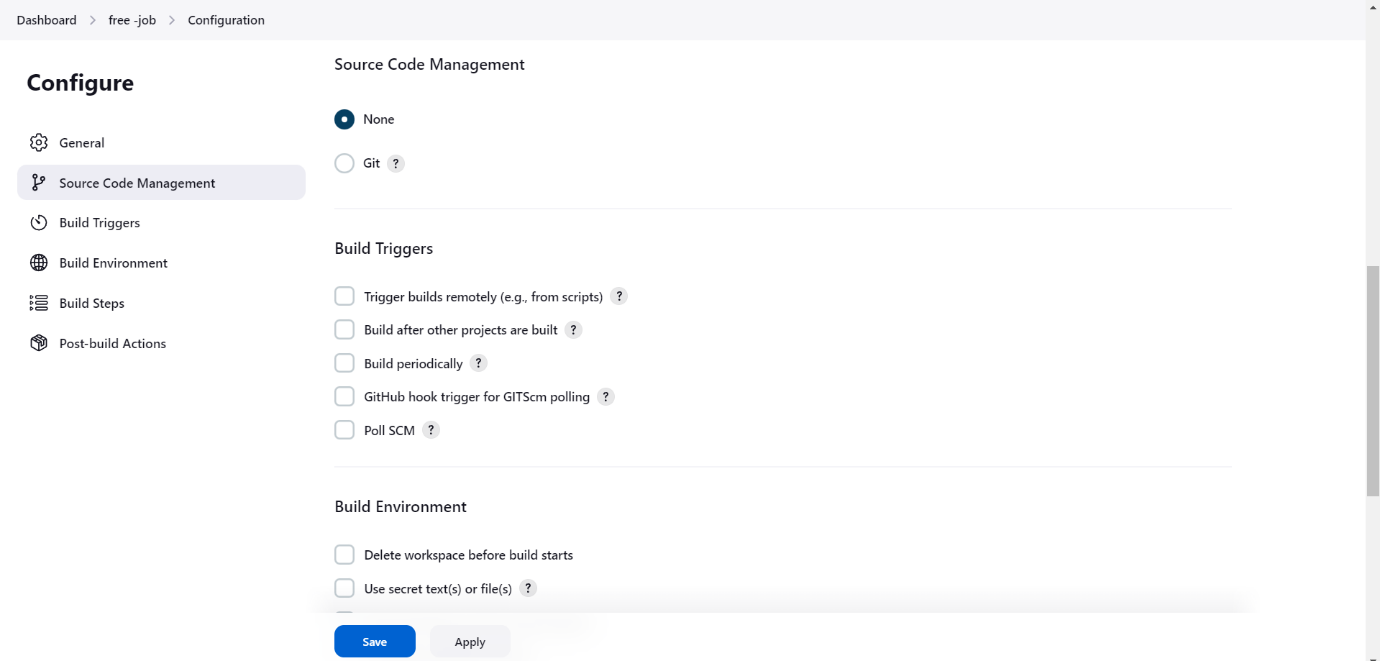


You need to install a plugin called GitHub plugin.

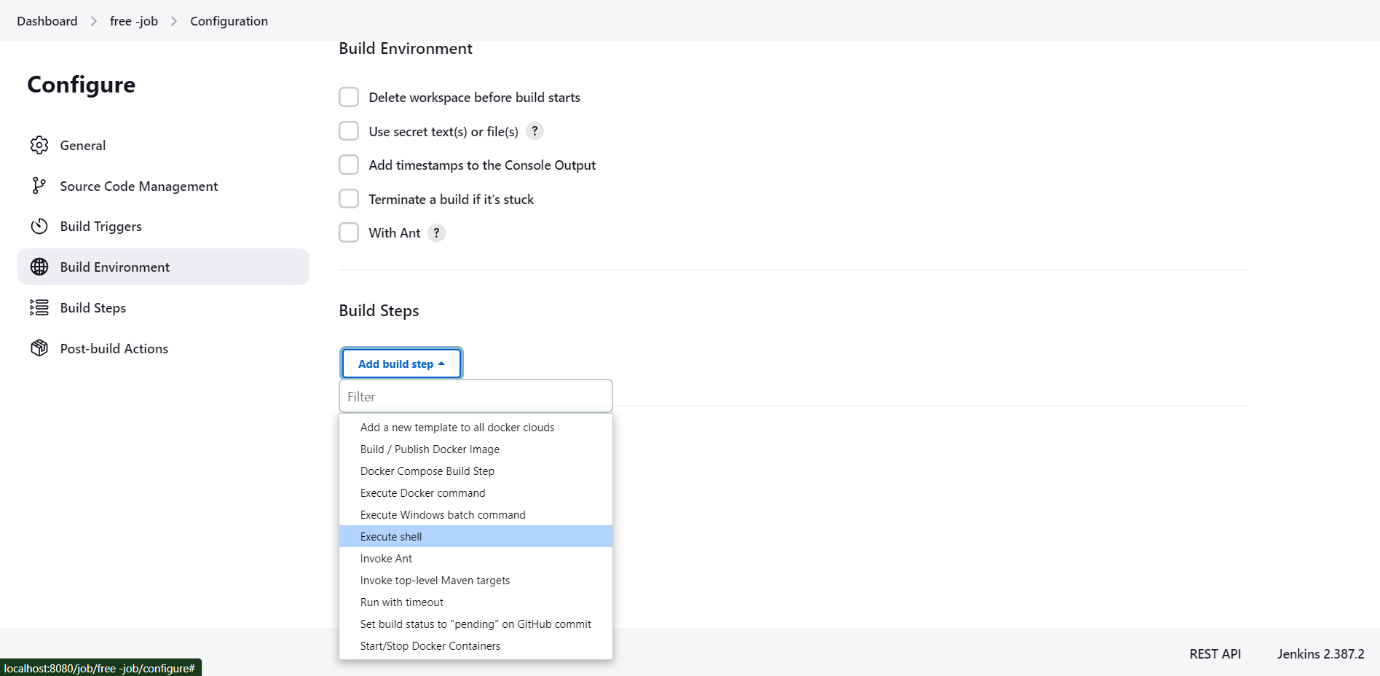
If you had code in GitHub, choose the GitHub project



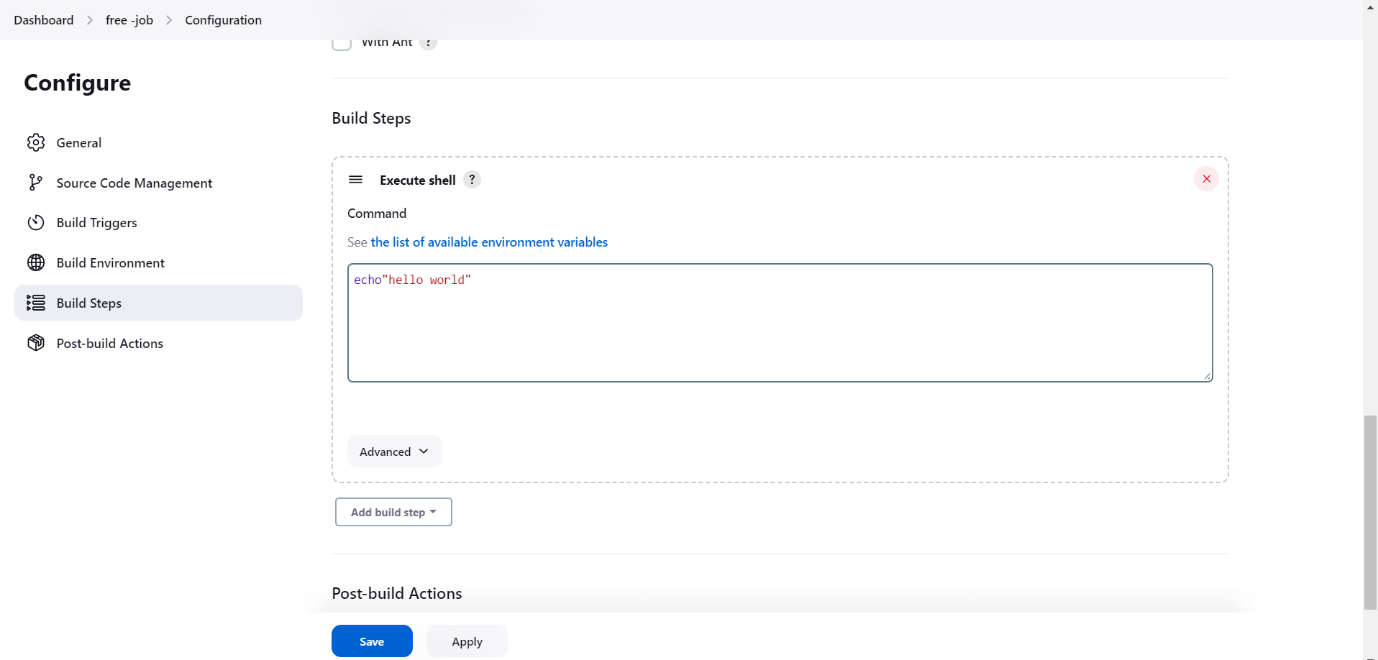
If you have code in the GitHub repository you need to choose git in source code management.

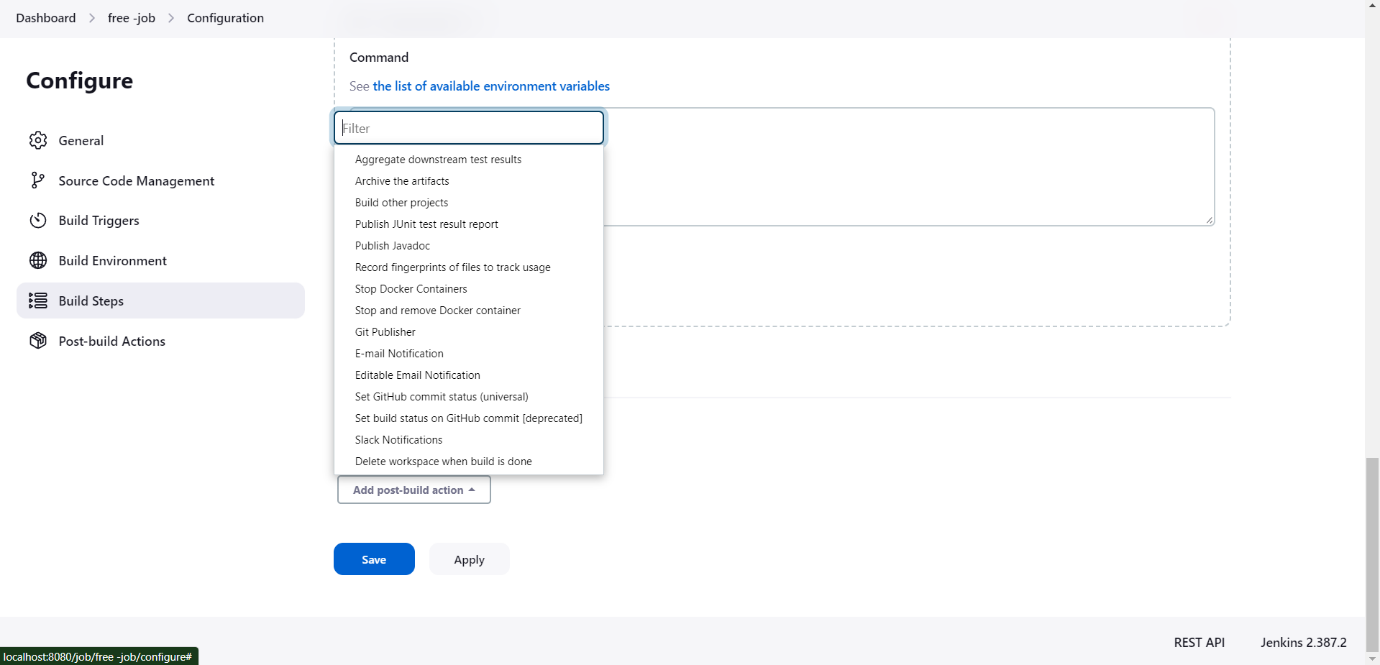


Then move on to the build steps you need to choose the execute shell for run the commands in Jenkins.

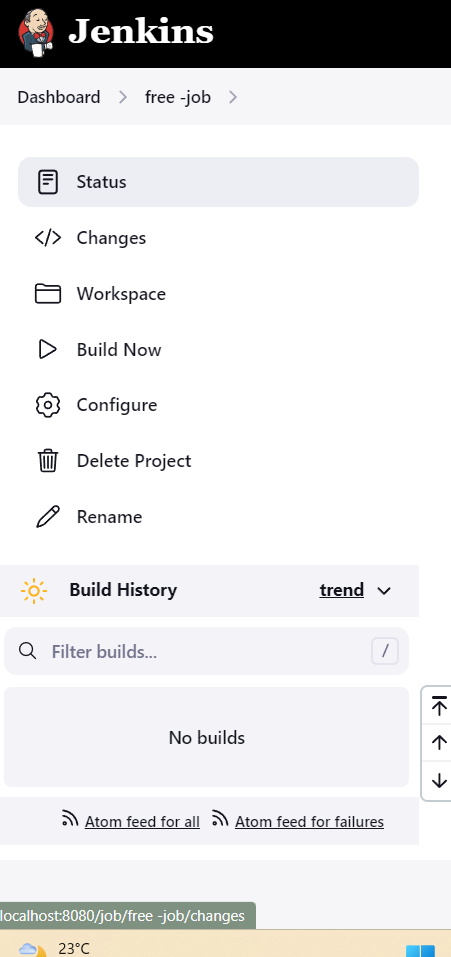


In that shell I had run a simple echo command to print the content.

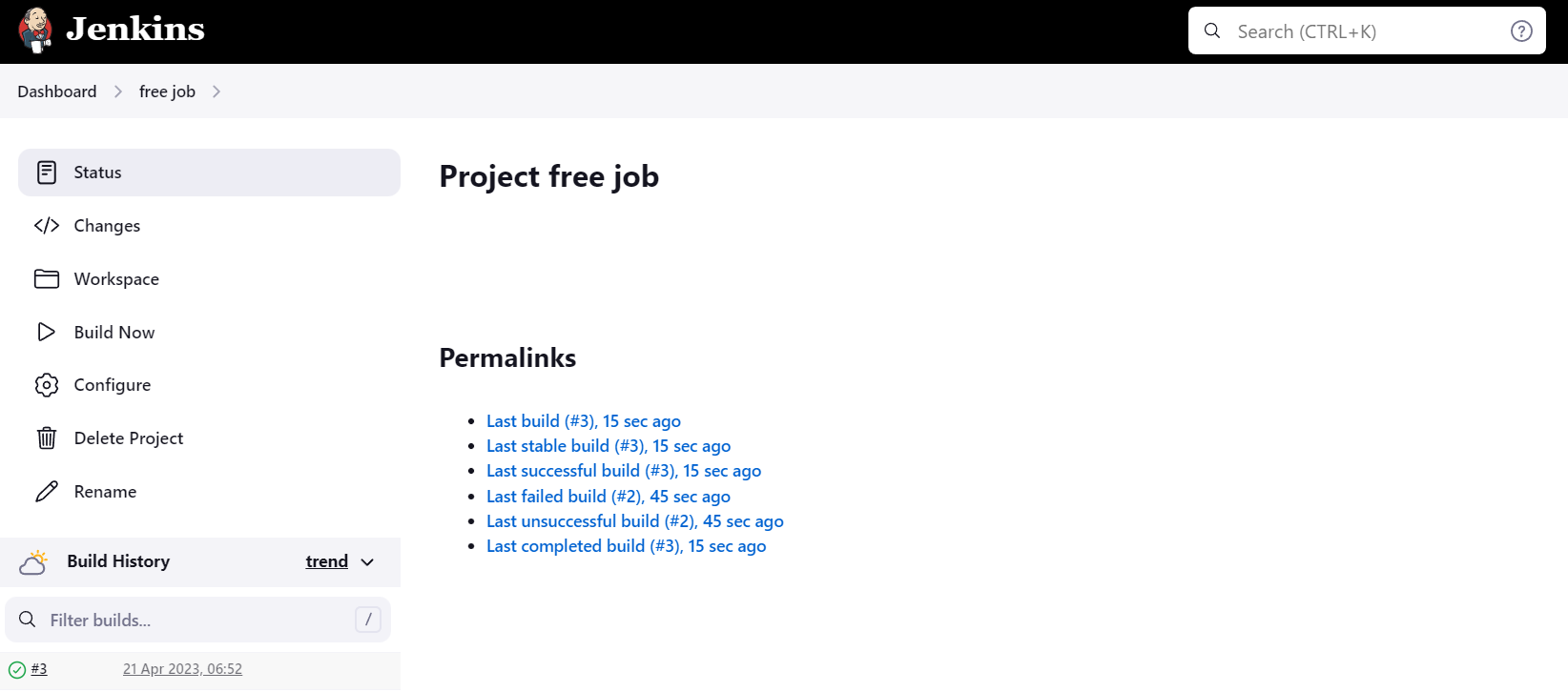


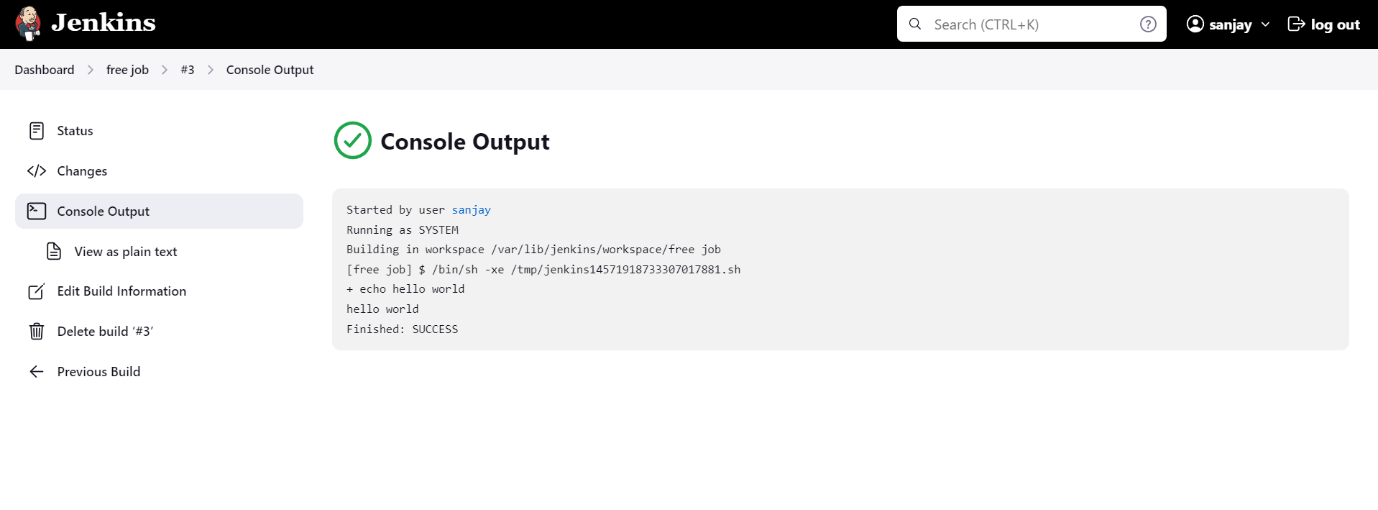
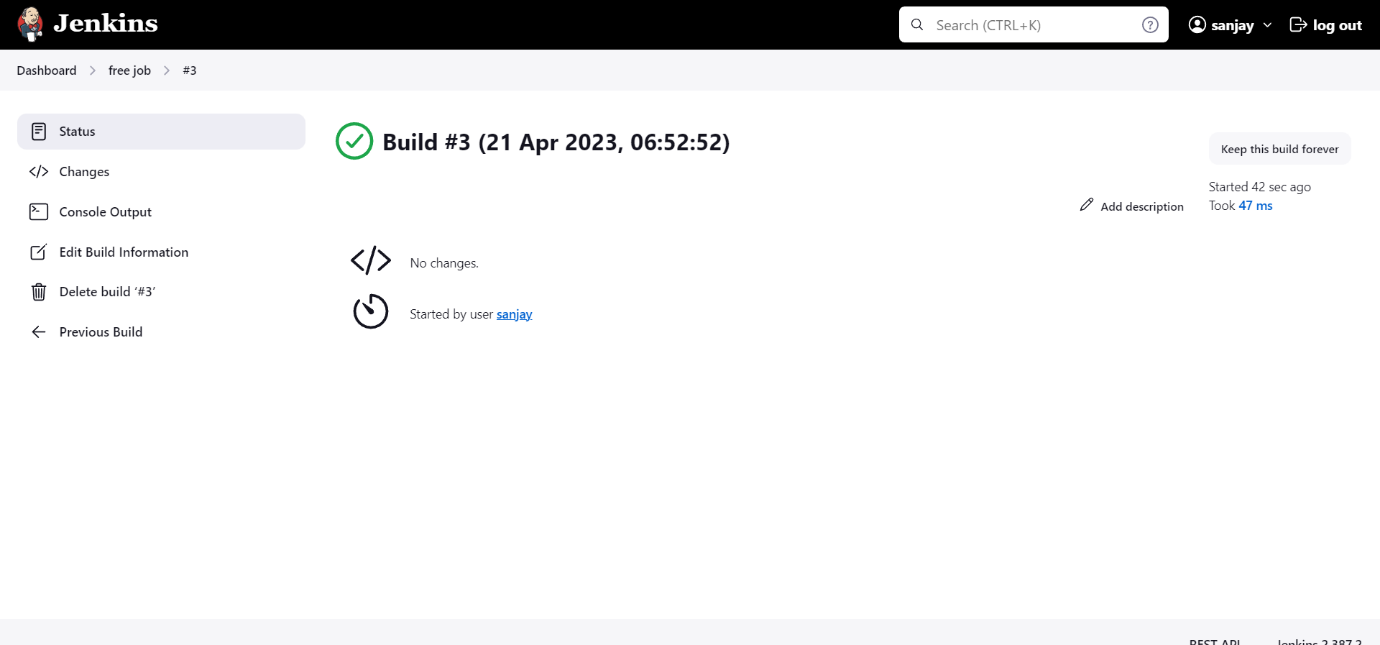
In Post build actions if you want to get notify from the Jenkins, you need to integrate with the Jenkins. 

Click the save button and build the project.



Check the build history, click the console output, you can get the desired output.

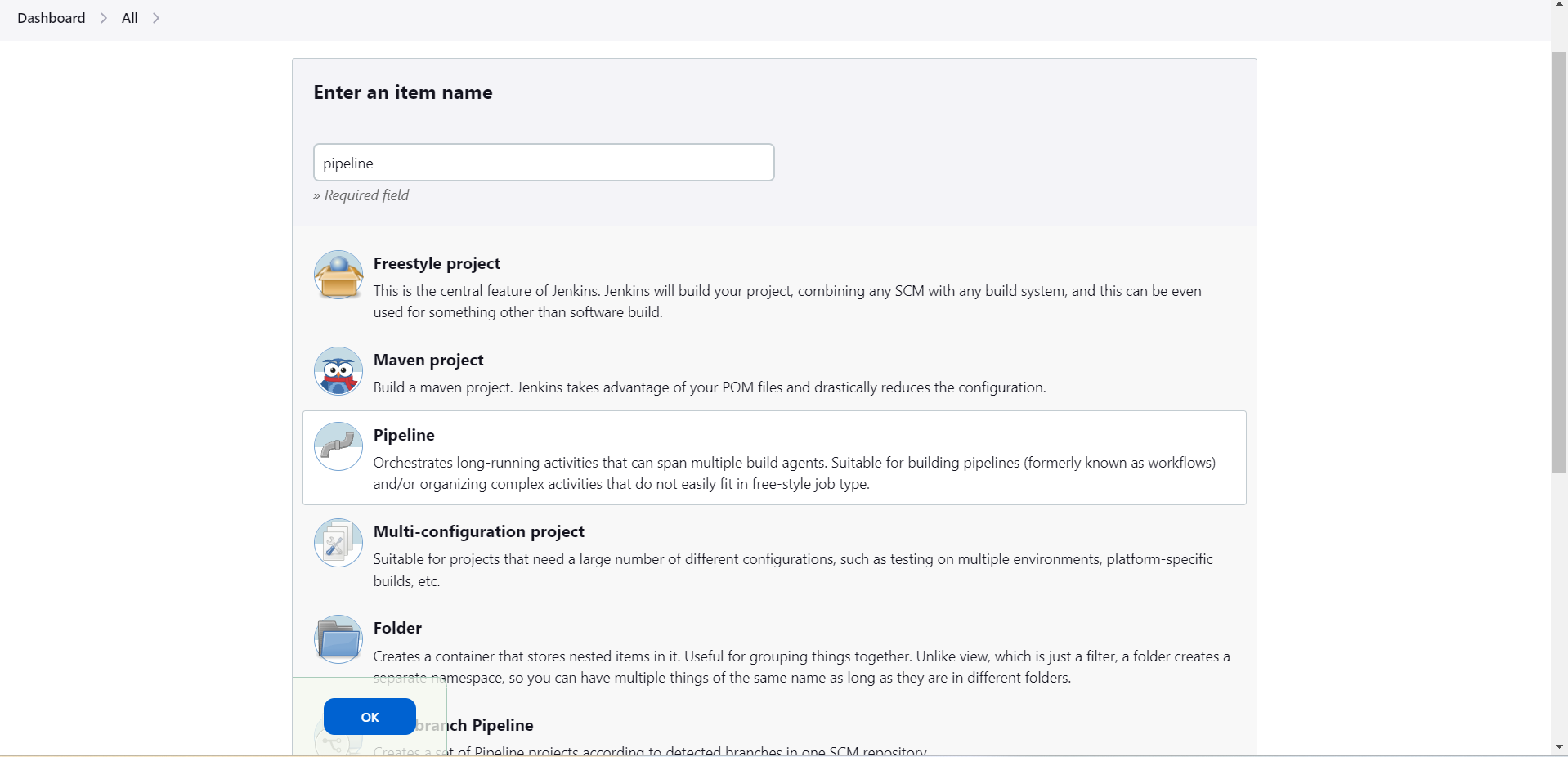




**PIPELINE PROJECT:**

Enter a new job and add name.

Choose the pipeline project.

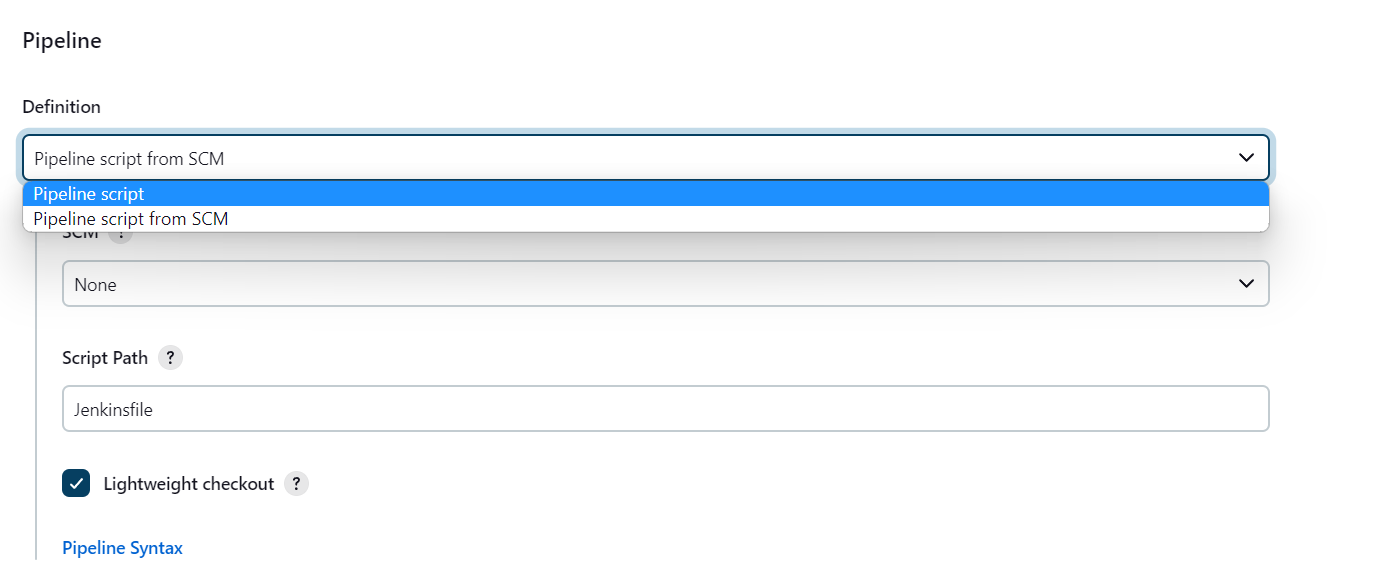


**Pipeline script:**

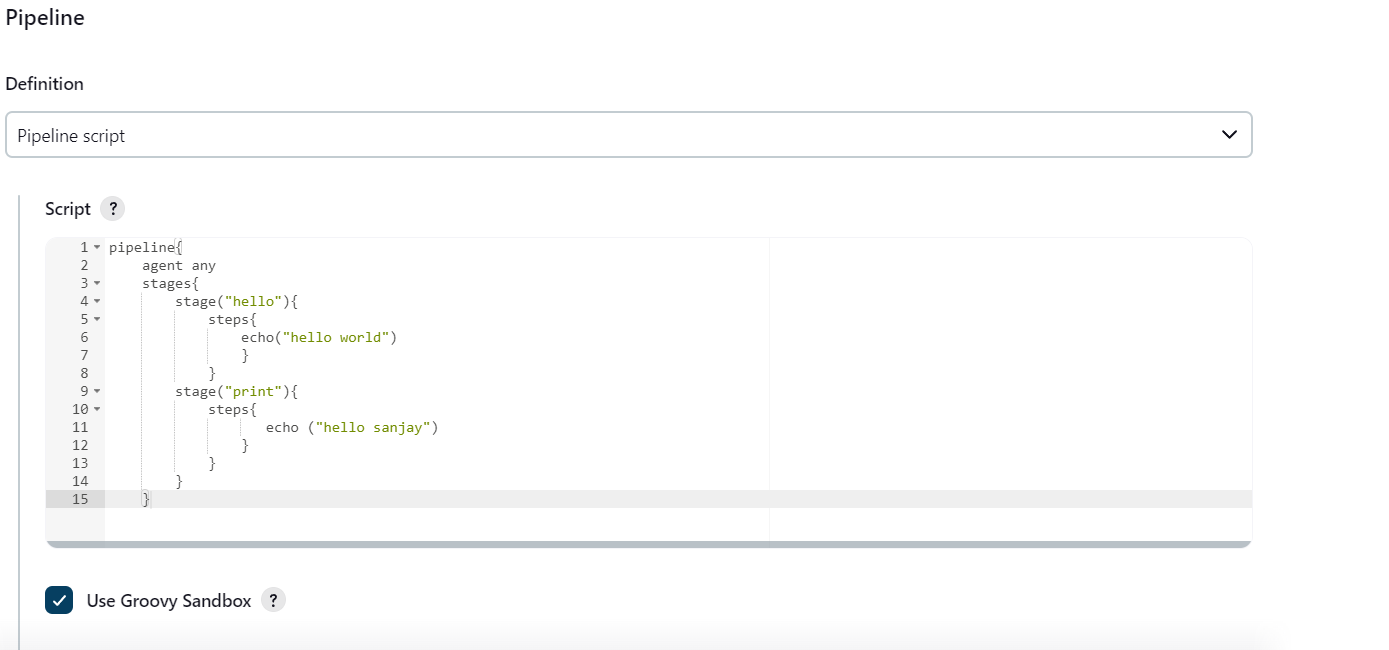
You have to write a Jenkins file for execute the pipeline in the Jenkins.

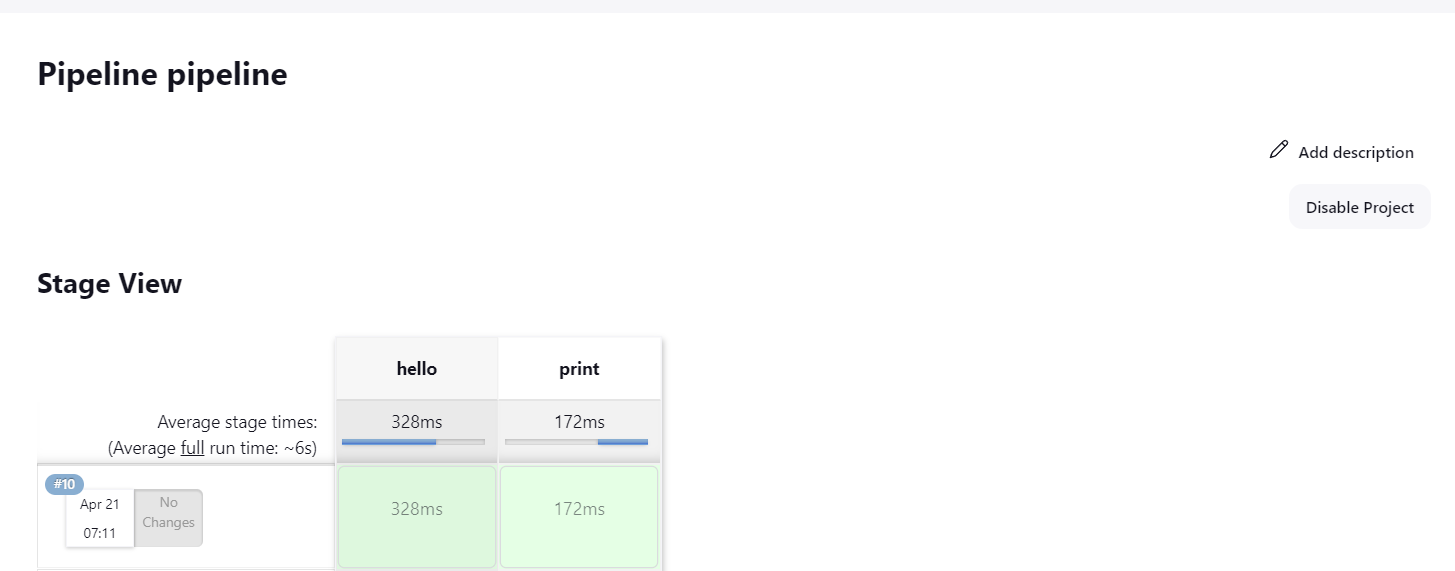
**Pipeline script from SCM:**

You have a Jenkins file in the GitHub repository, you have to mention that GitHub repository URL in that pipeline method and no need to add credentials if the repository is public.

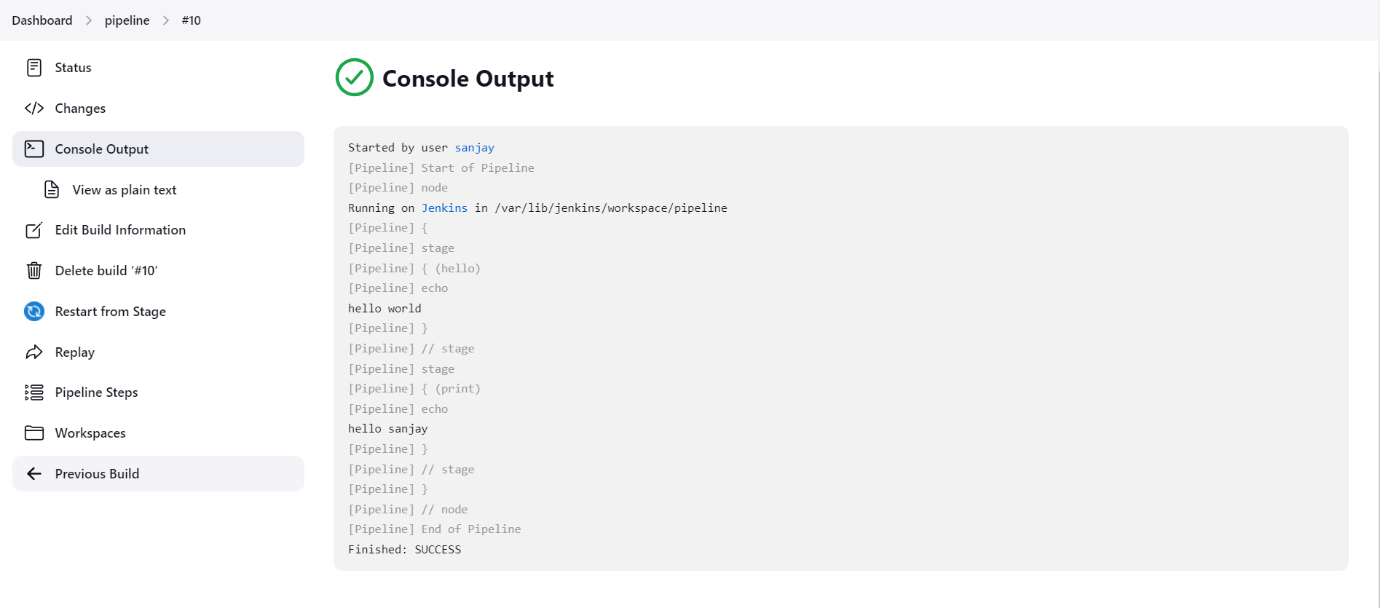


This is the pipeline syntax for Jenkins file



Click the save button and build the pipeline.

In pipeline it executes in stage wise that we mentioned in that stage:



Declarative pipeline:

pipeline {

agent {docker ‘node:6.3’ }

stages {

stage(‘build’) {

steps {

sh ‘npm —version’

}

}

}

}

### Scripted pipeline:

node(‘docker’) {

checkout scm

stage(‘**Build**’) {

docker.image(‘node:6.3’).inside {

sh ‘npm —version’

}

}

}