AIR UNIVERSITY, ISLAMABAD

Department of Cyber Security

Secure Software Design & Development Lab (CY- 256L)

REG ID: 231312

ASSIGNMENT III

Kiran Hashmi

BS-CYS 4 A

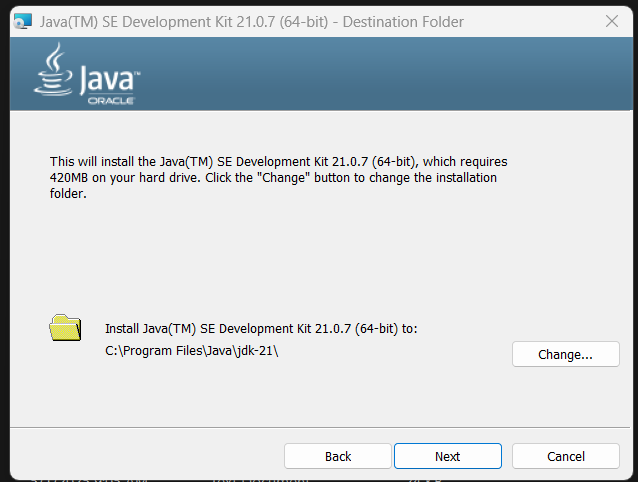
# What is Jenkins?

Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment

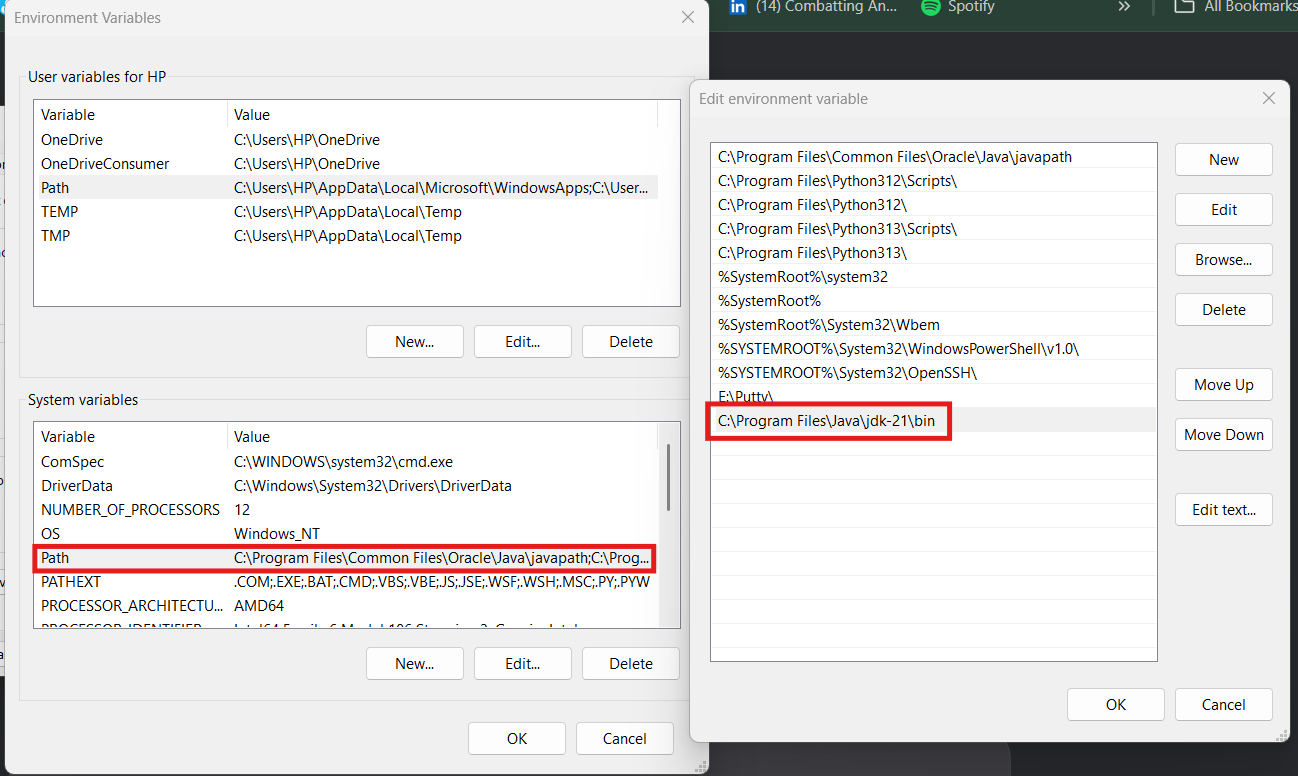
Pre requisites:

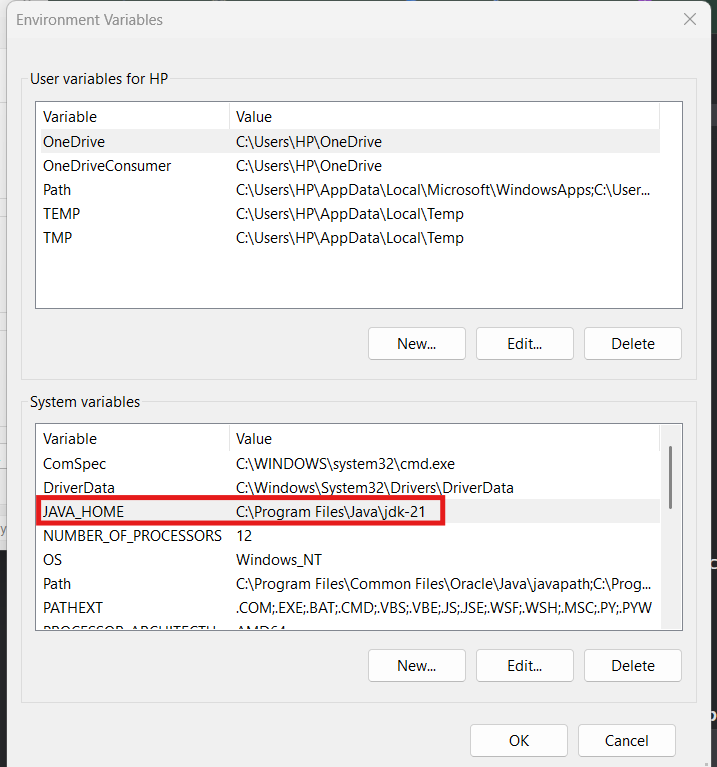
Java should be installed on your machine.

# Installation of JAVA:

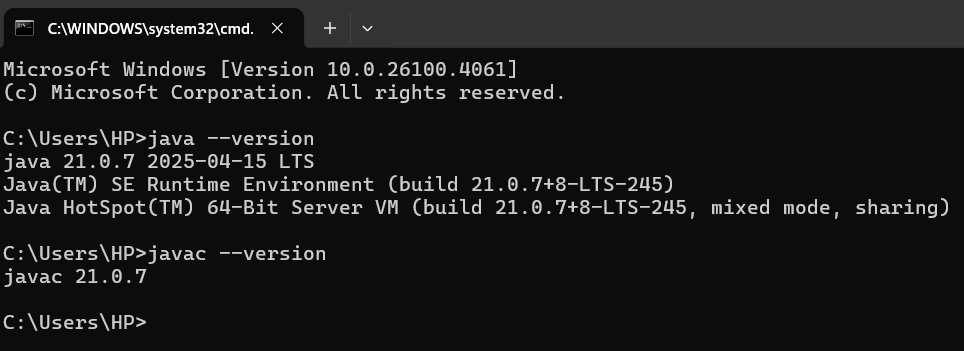


## Configuring java:

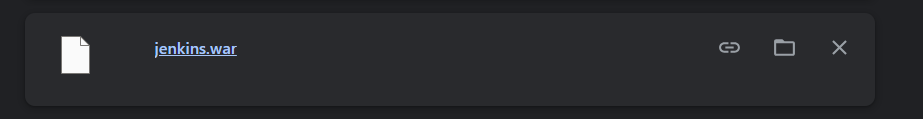




Installed Java:



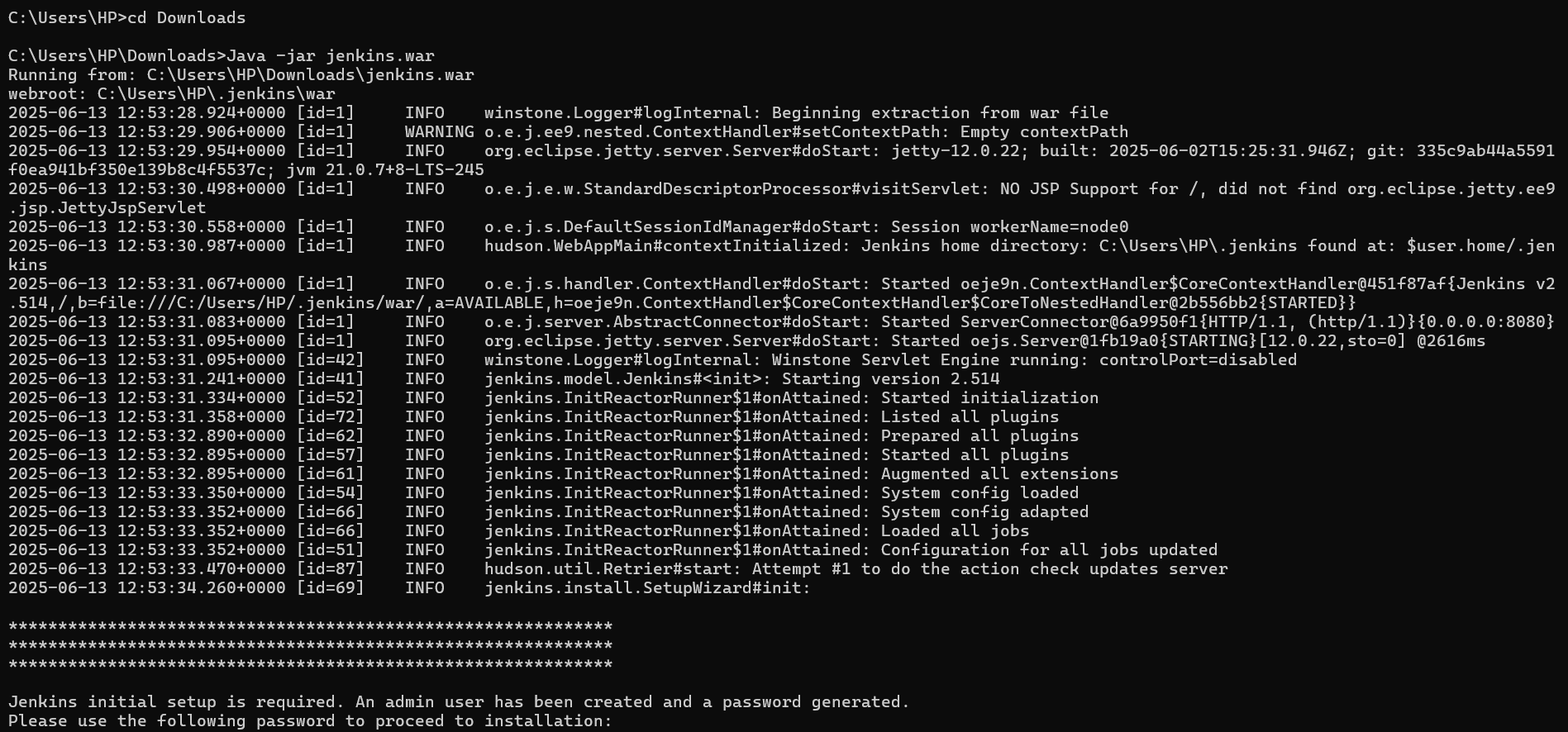
# Download Jenkins:



## Starting Jenkins

Open the command prompt. From the command prompt, browse to the directory where the jenkins.war file is present. Run the following command

Java -jar Jenkins.war



After the command is run, various tasks will run, one of which is the extraction of the war file which is done by an embedded webserver called winstone.

Once the processing is complete without major errors, the following line will come in the output of the command prompt.

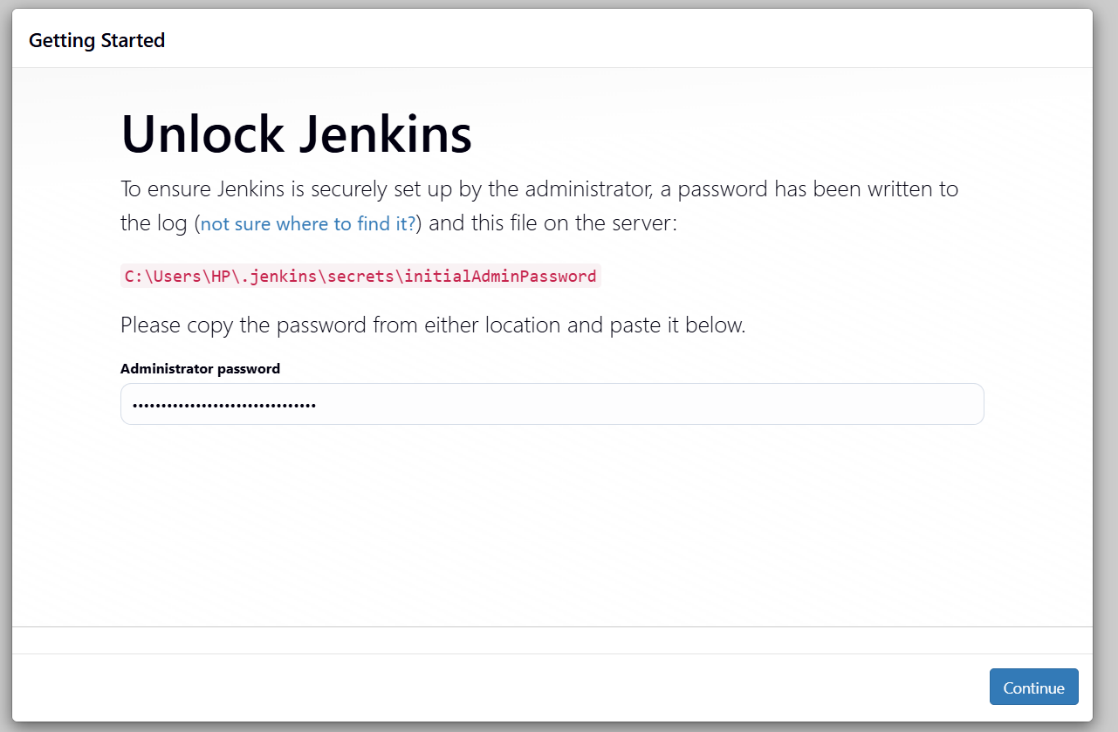
Copy the password and save it somewhere for later use.

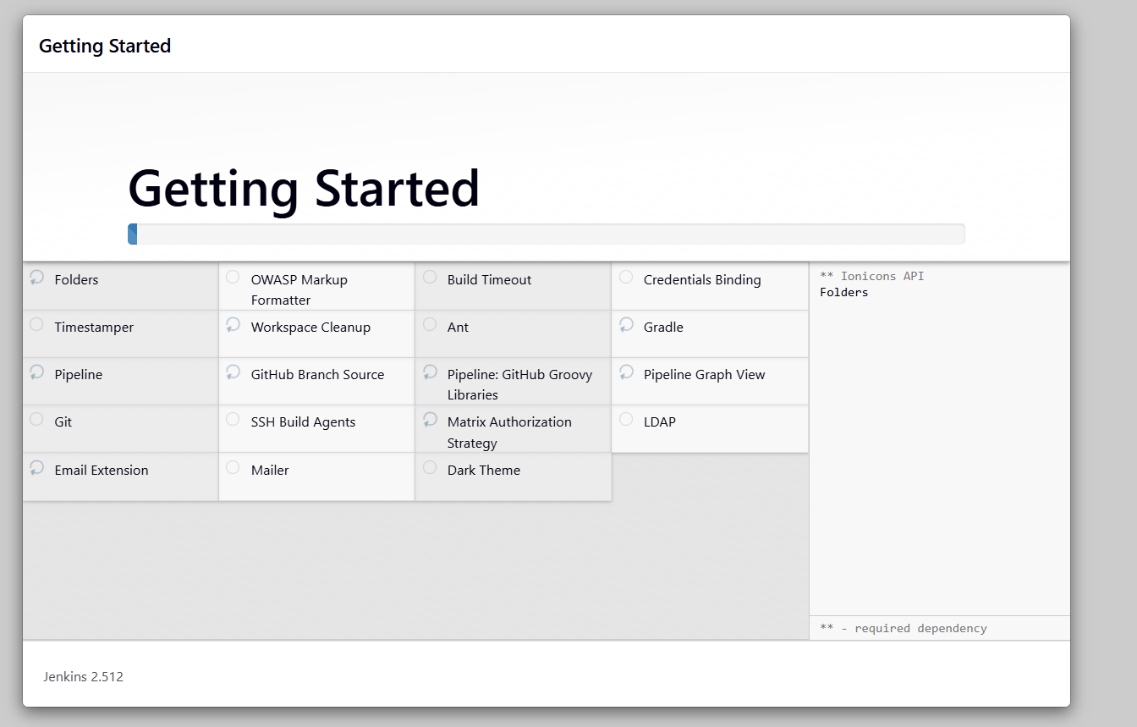
## Accessing Jenkins

Once Jenkins is up and running, one can access Jenkins from the

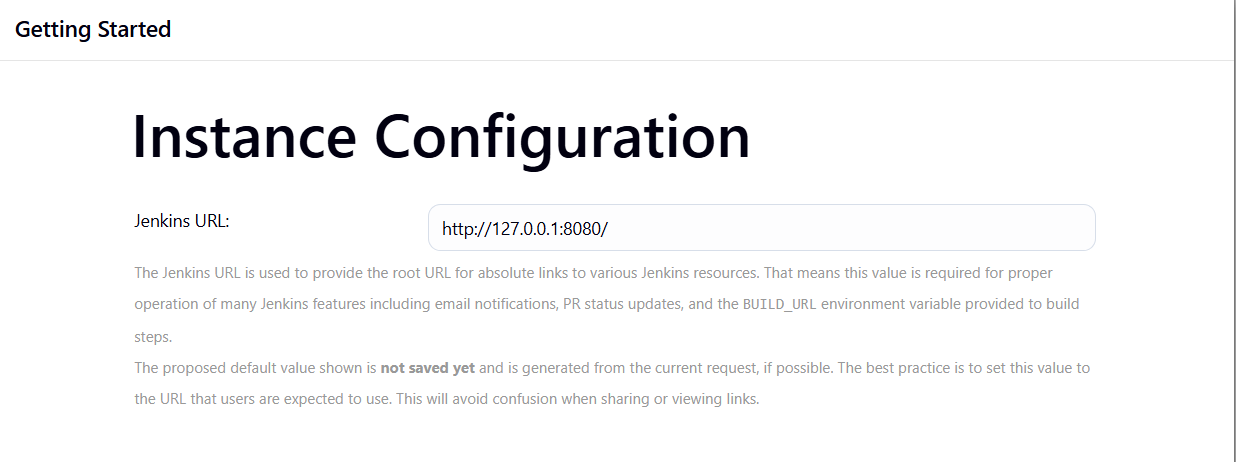
link − http://localhost:8080

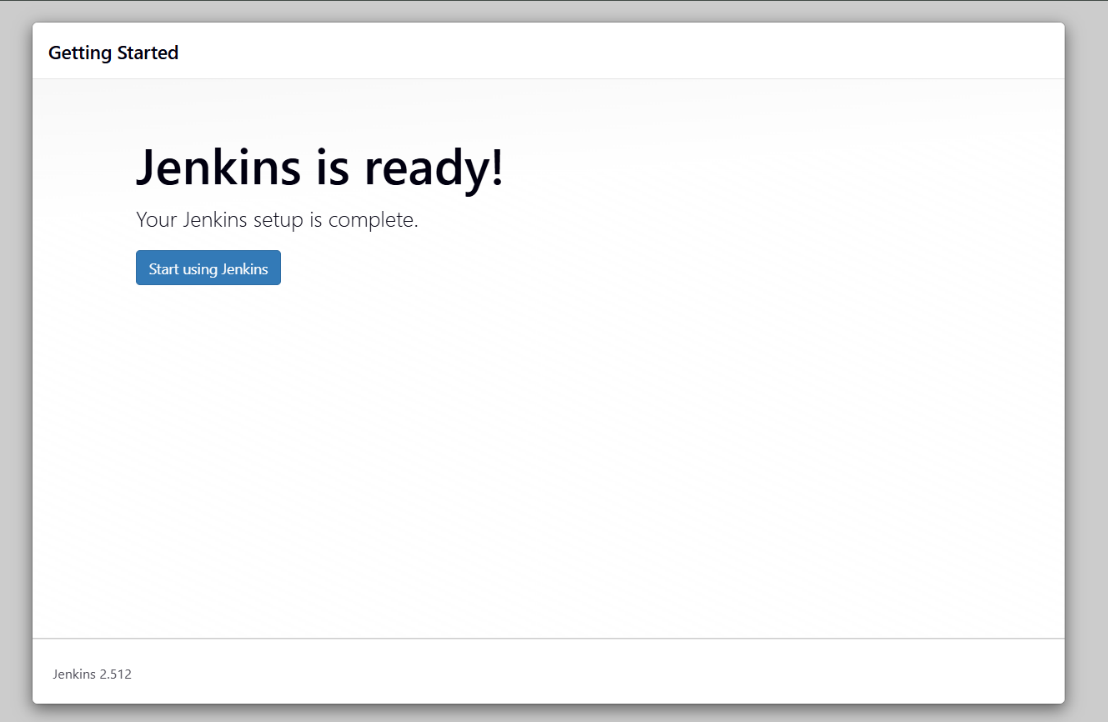
This link will bring up the Jenkins dashboard. Write password that you saved previously.

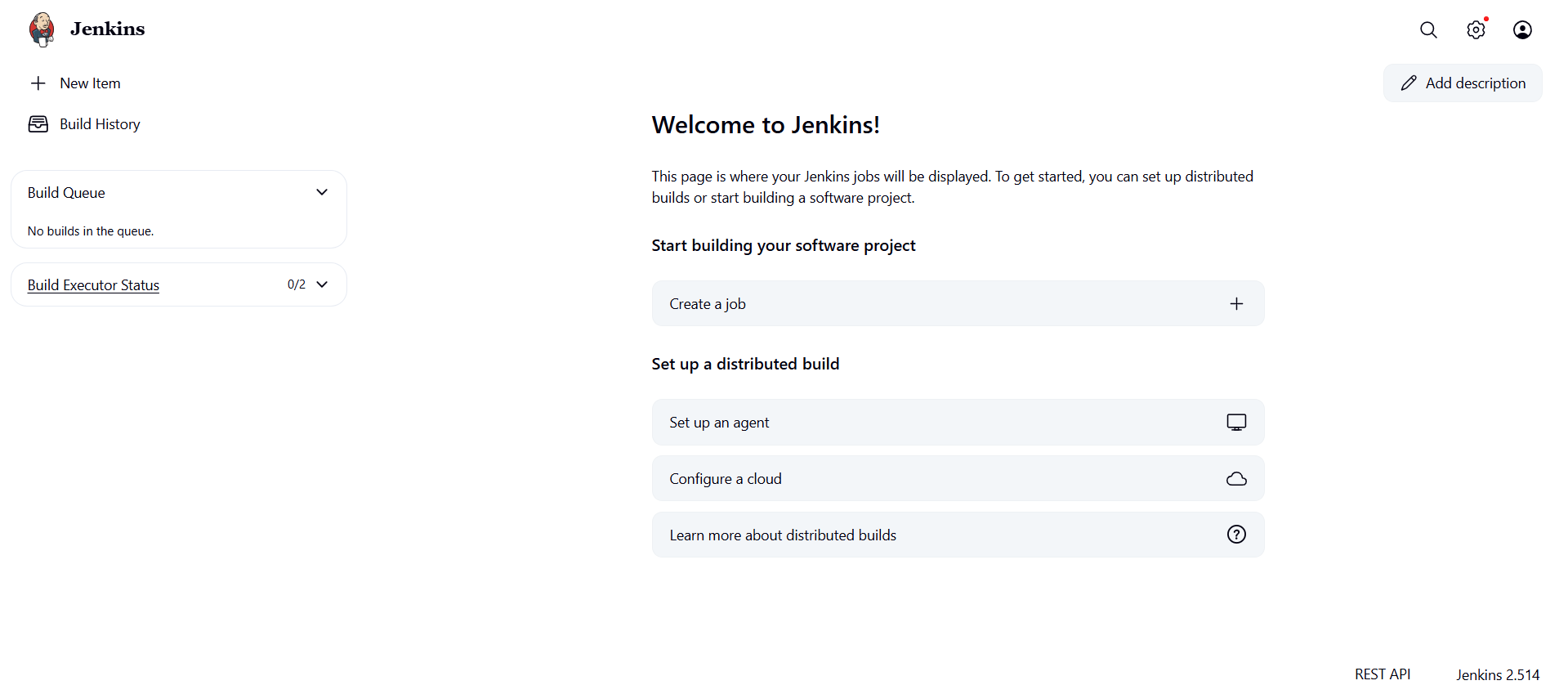












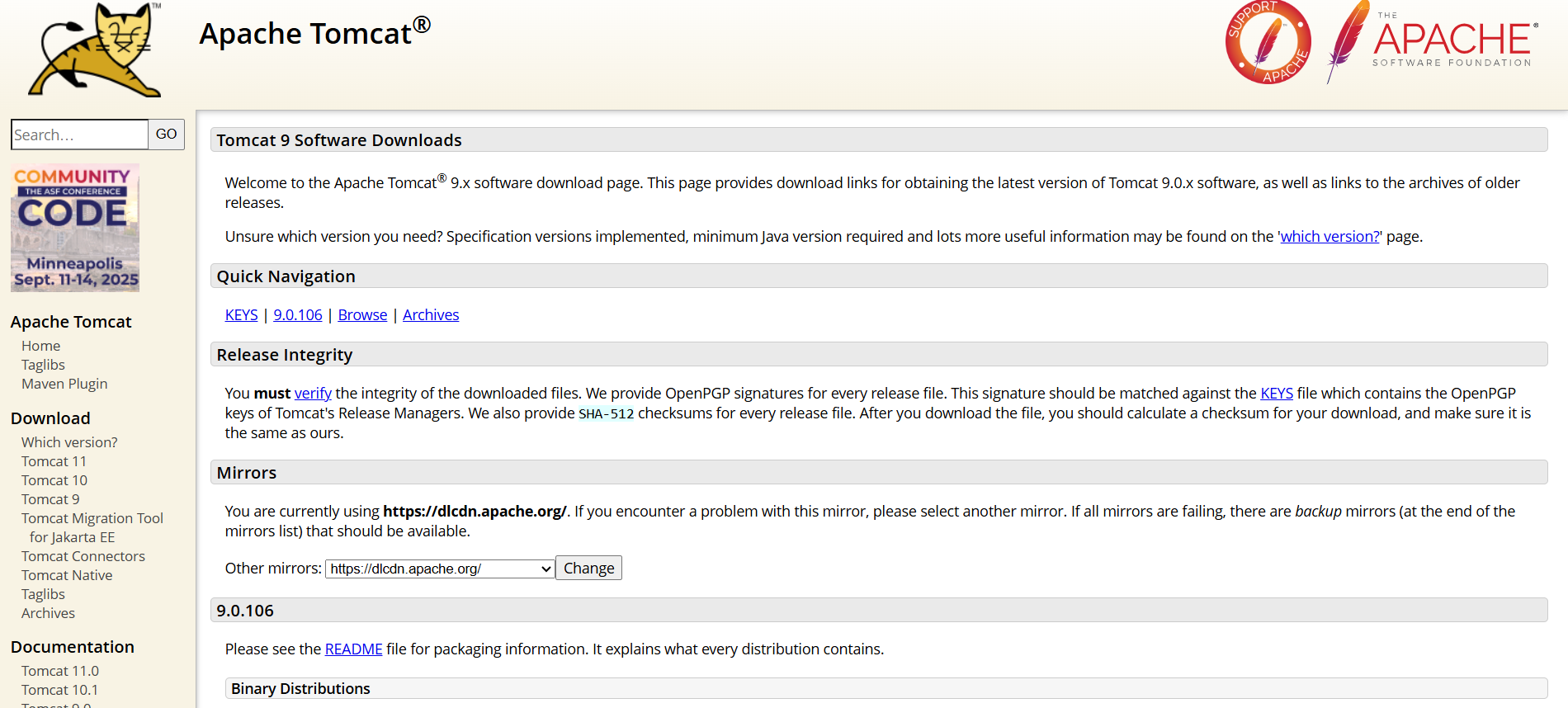
# What is Tomcat?

Tomcat is an open-source web server and servlet. The Apache Software Foundation has developed it. It is used widely for hosting Java-based applications on the web. It is built on Java technologies and implements the Java Servlet and JavaServer Pages (JSP) specifications. Tomcat acts as a bridge between web servers and Java-based applications, facilitating the execution of dynamic content and processing client requests.

## Download Tomcat

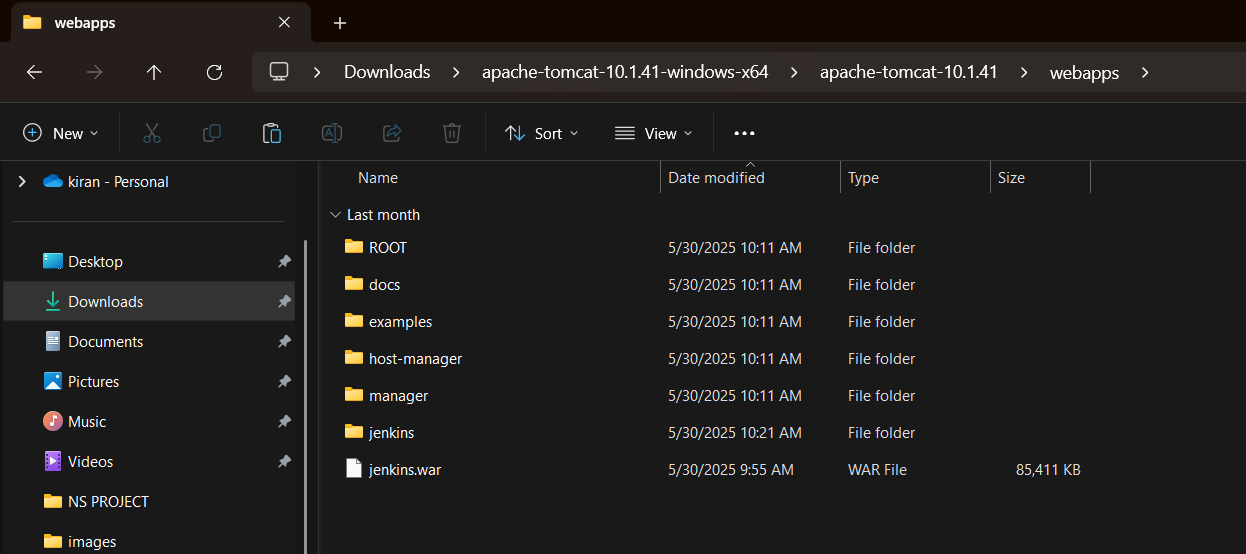
The official website for tomcat is Tomcat. If you click the given link, you can get the home page of the tomcat official website as shown below.

<https://tomcat.apache.org/download-90.cgi>



Go to the ‘Binary Distributions’ section. Download the 32-bit Windows zip file.

Then unzip the contents of the downloaded zip file.

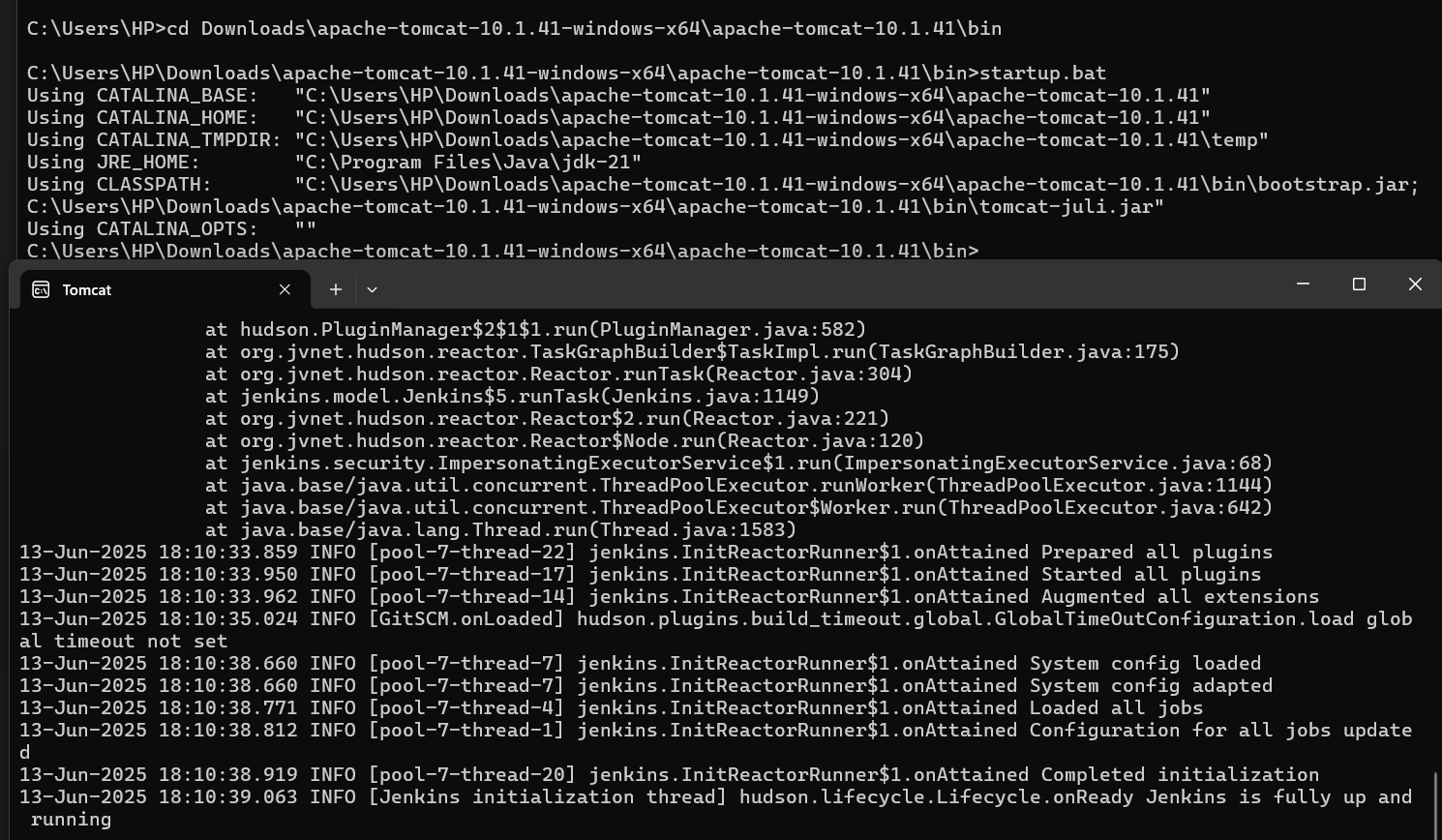


# Jenkins and Tomcat Setup

Copy the Jenkis.war file which was downloaded from the previous section and copy it to the webapps folder in the tomcat folder.

Now open the command prompt. From the command prompt, browse to the directory where the tomcat7 folder is location. Browse to the bin directory in this folder and run the start.bat file

C:\Users\HP\Downloads\apache-tomcat-10.1.41-windows-x64\apache-tomcat-10.1.41\bin>startup.bat

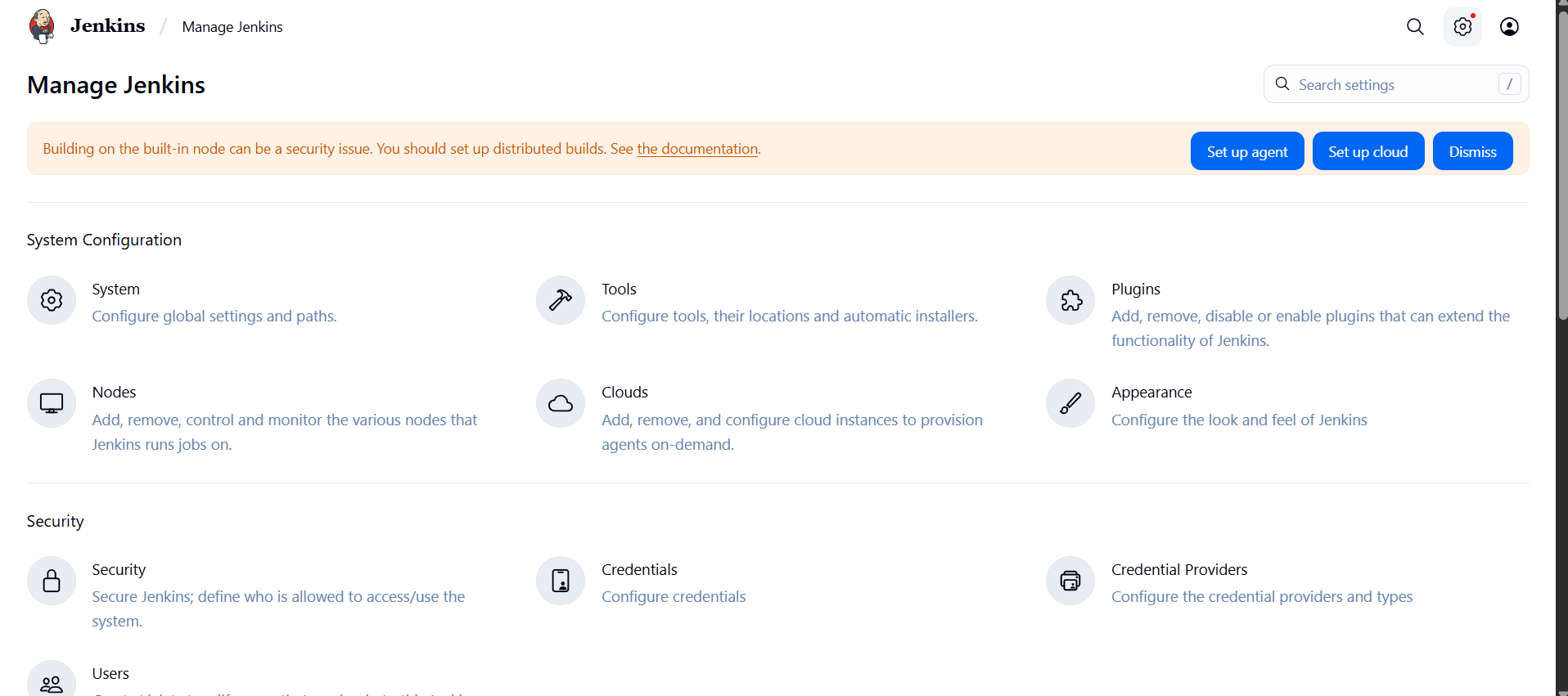


Open the browser and go to the link − **http://localhost:8080/jenkins.**

Jenkins will be up and running on tomcat.

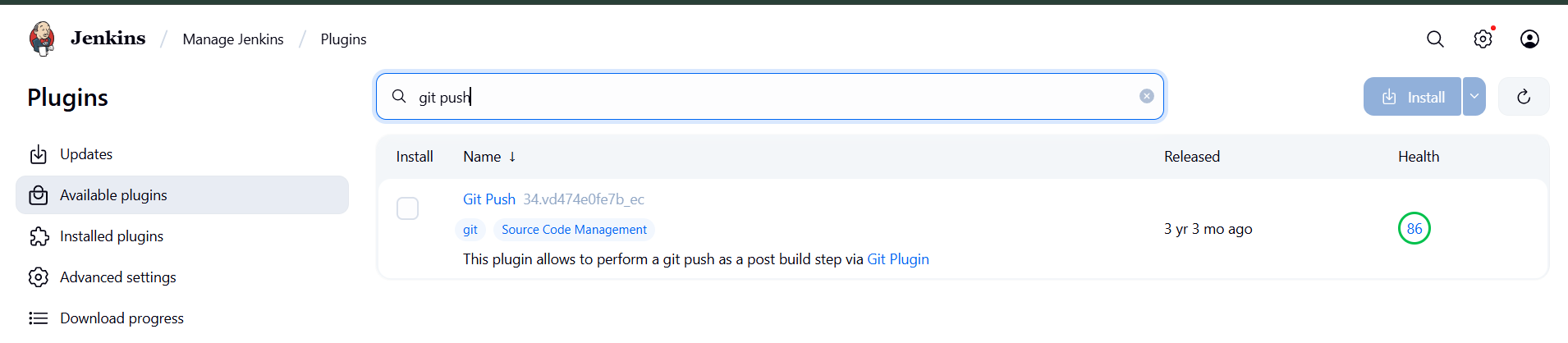
**Jenkins - Git Setup**

For this exercise, you have to ensure that Internet connectivity is present from the machine on which Jenkins is installed. In your Jenkins Dashboard (Home screen), click the Manage Jenkins option on the left hand side

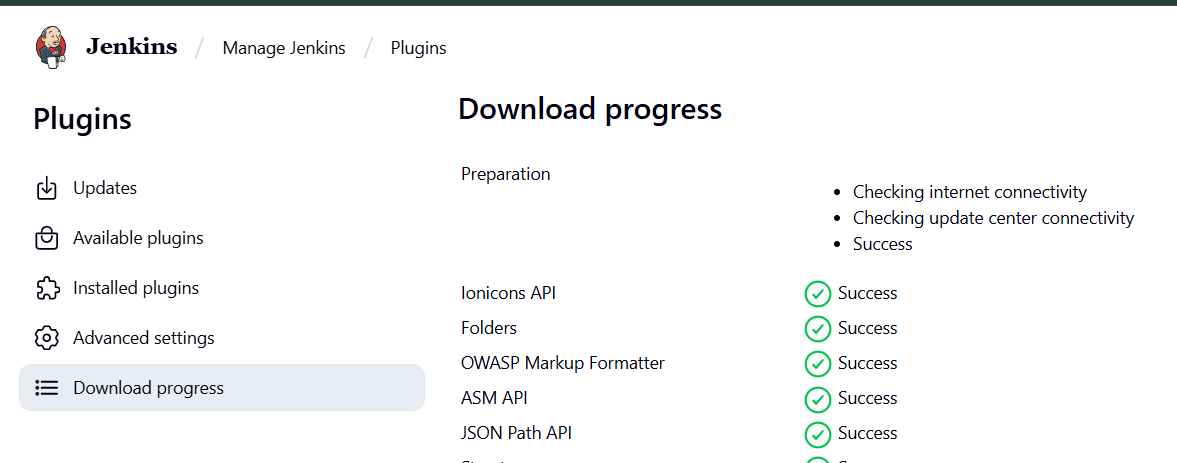


In the next screen, click the ‘Plugins’ option.

In the next screen, click the Available tab. This tab will give a list of plugins which are available for downloading. In the ‘Filter’ tab type ‘Git Push’

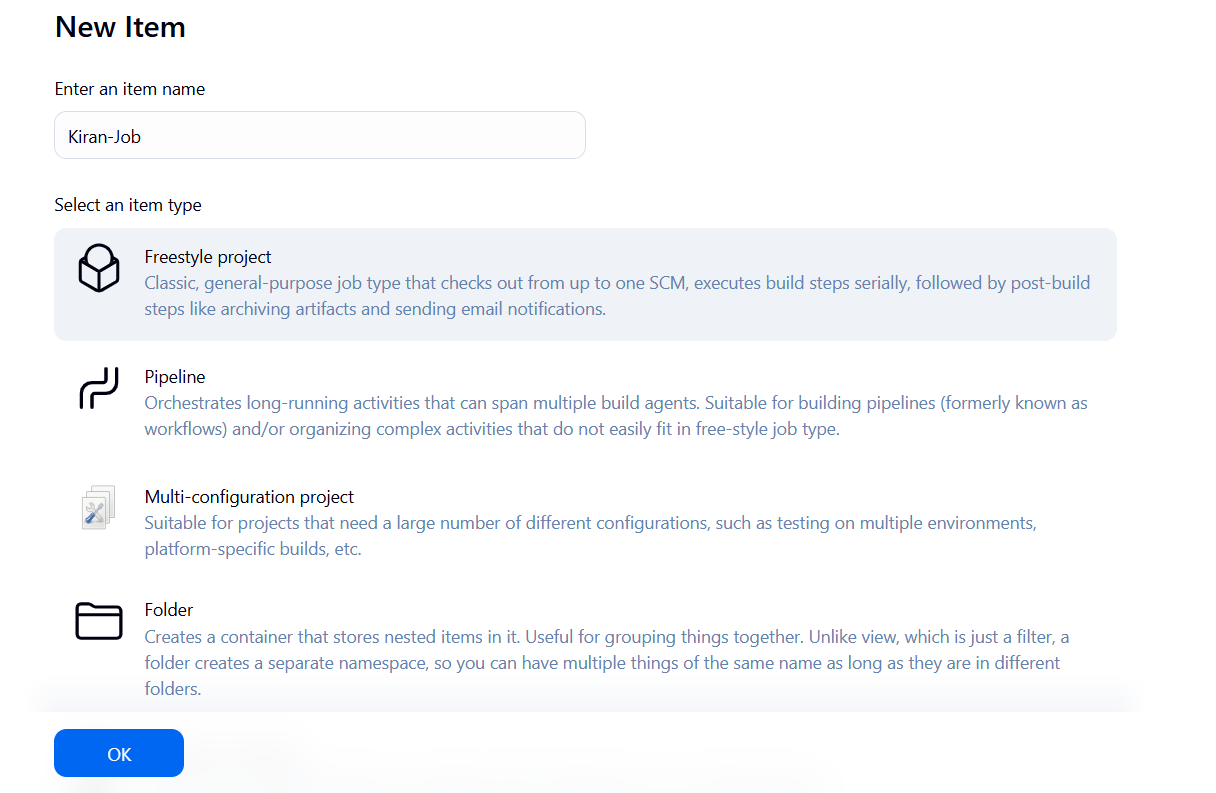


The installation will then begin and the screen will be refreshed to show the status of the download.



Once all installations are complete, restart Jenkins by issue the following command in the

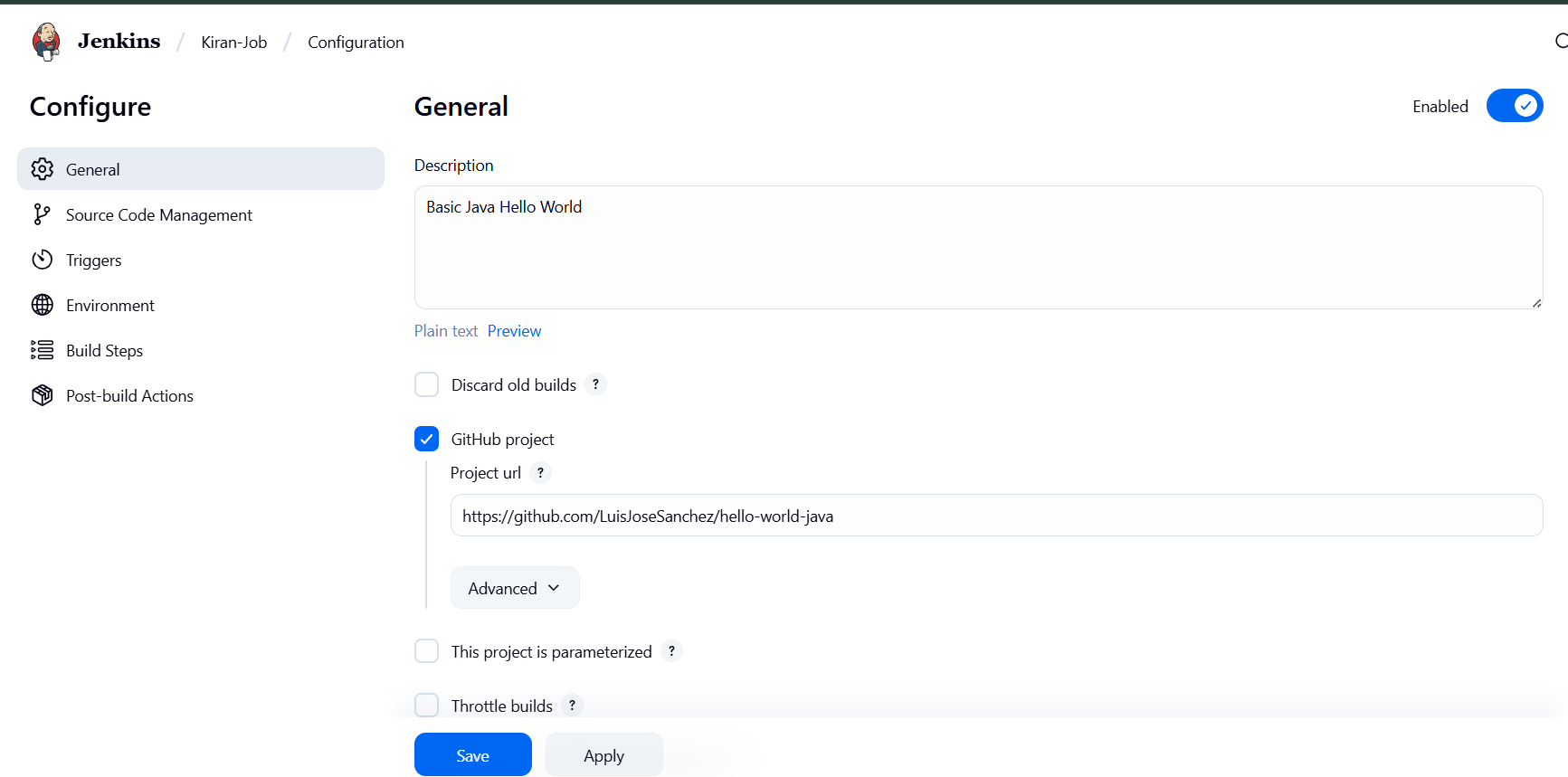
browser. http://localhost:8080/jenkins/restart



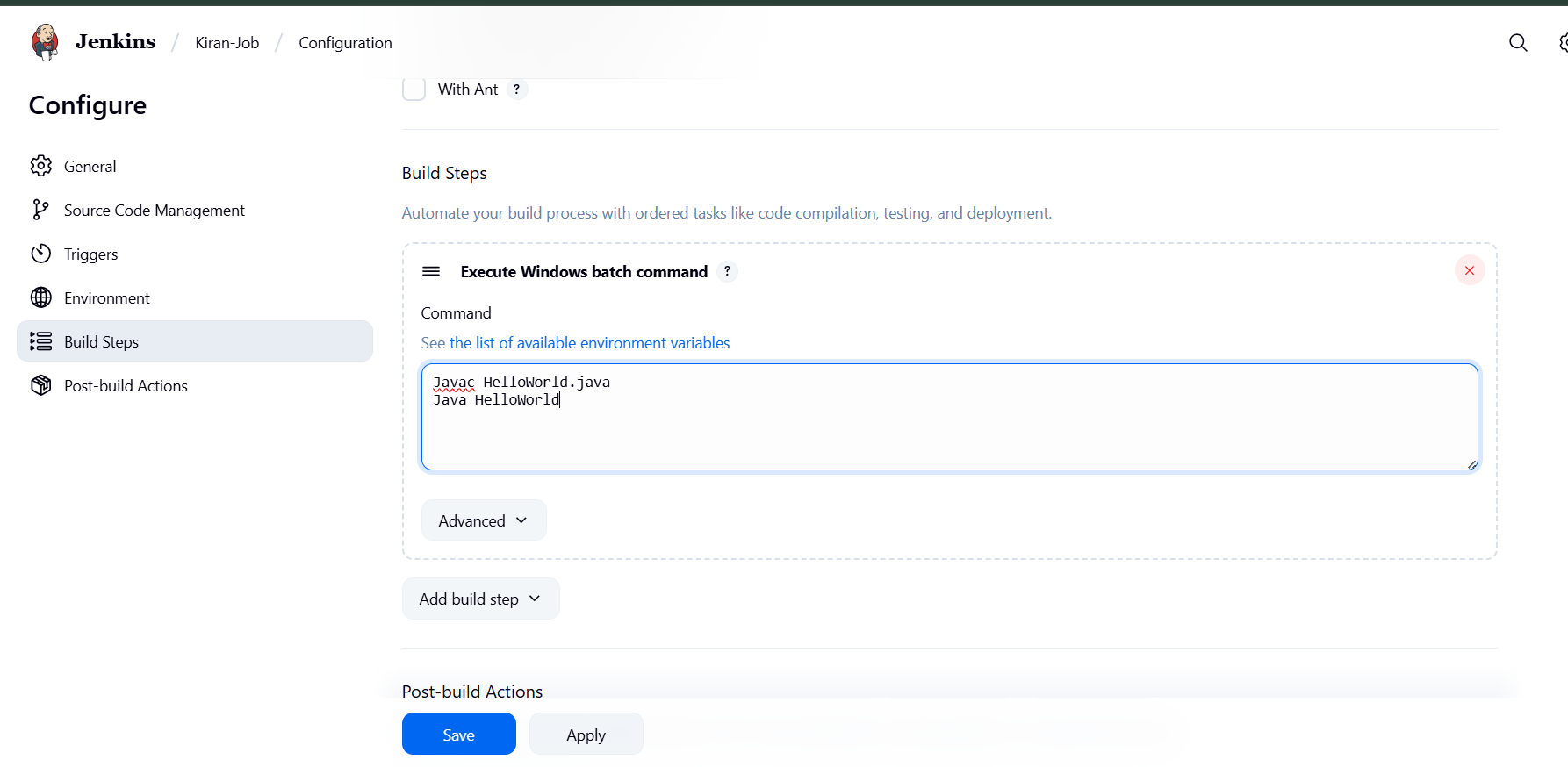
# Jenkins - Setup Build Jobs

For this exercise, we will create a job in Jenkins which picks up a simple HelloWorld application, builds and runs the java program.

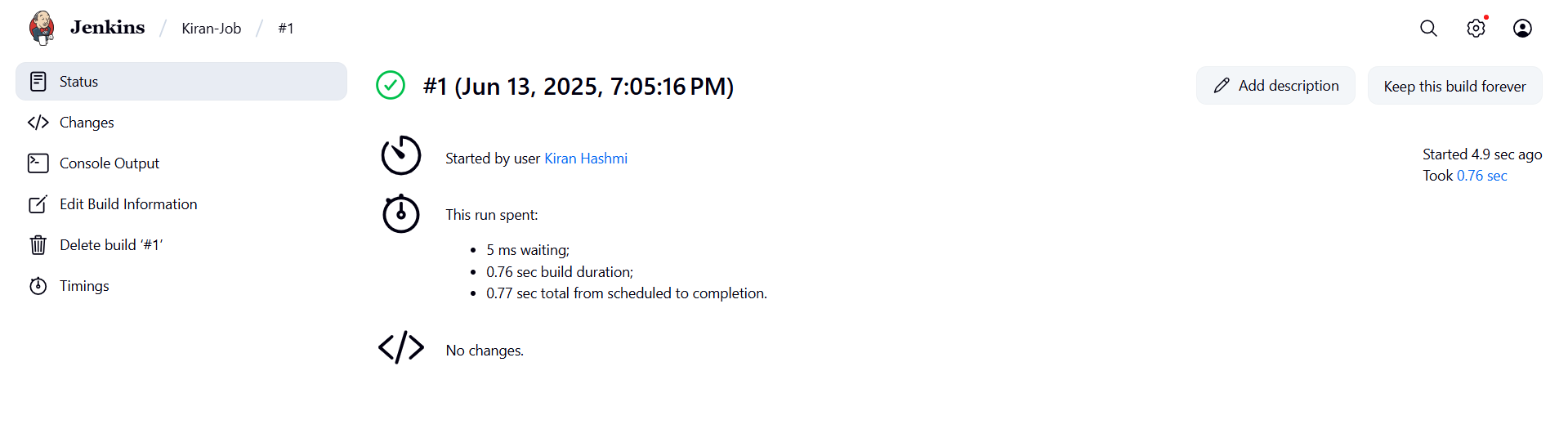
The following screen will come up in which you can specify the details of the job.

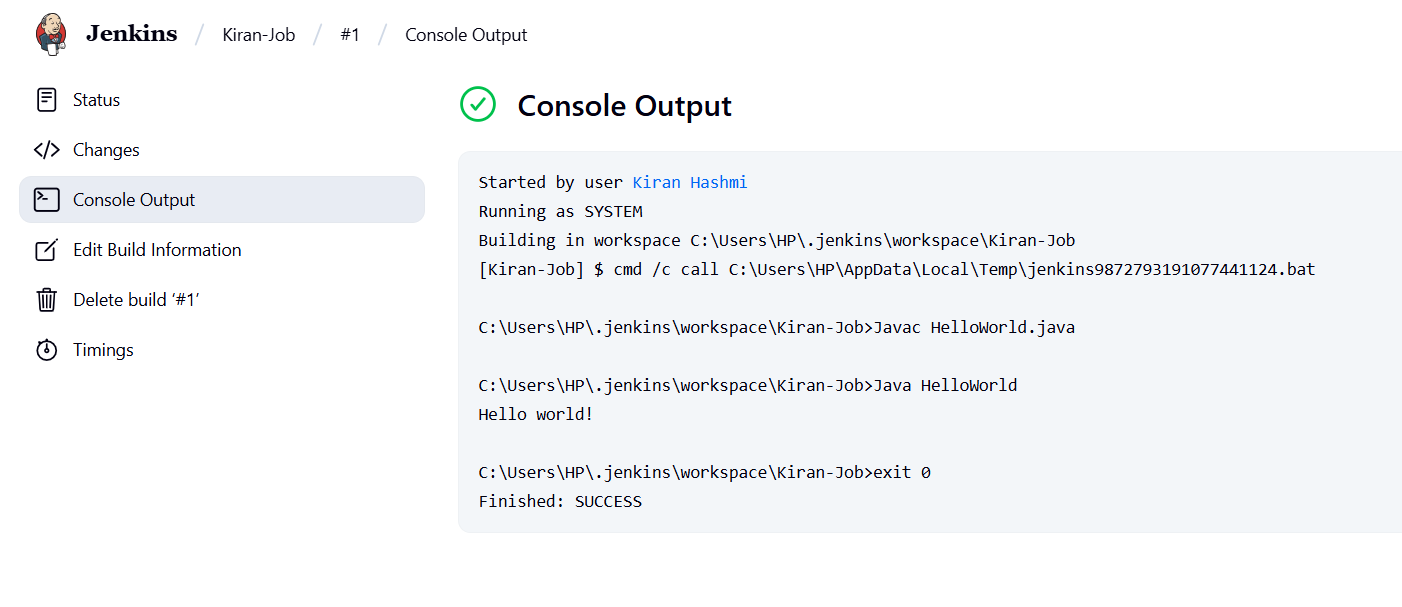


We need to specify the location of files which need to be built. If you repository if hosted on Github, you can also enter the url of that repository here:



Now go to the Build section and click on Add build step → Execute Windows batch command





successfully defined the job.

**What is Continuous Integration?**

Continuous Integration is a development practice that requires

developers to integrate code into a shared repository at regular

intervals. This concept was meant to remove the problem of finding

later occurrence of issues in the build lifecycle. Continuous

integration requires the developers to have frequent builds. The

common practice is that whenever a code commit occurs, a build

should be triggered.

**What is Jenkins File?**

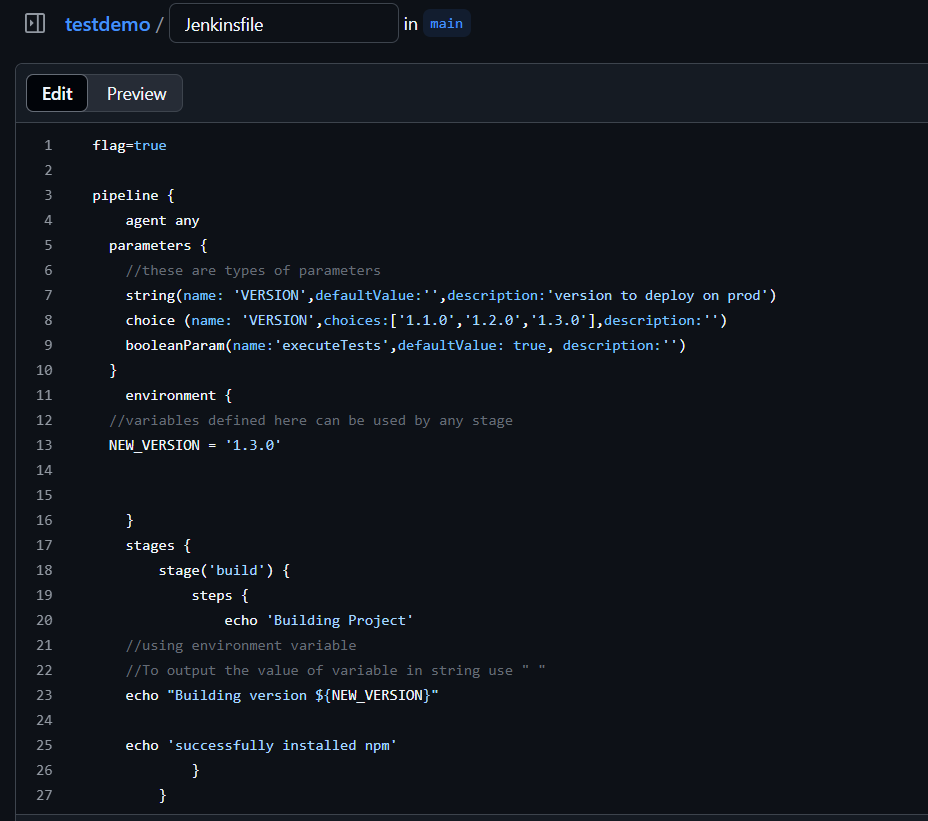
A Jenkins file is a script written in the Groovy programming language that defines the

steps to be executed by a Jenkins pipeline. The pipeline is a series of steps executed in

a particular order.

Make a new file named Jenkinsfile in any of your existing GitHub repository and

paste the code given above:



flag=true

pipeline {

agent any

parameters {

//these are types of parameters

string(name: 'VERSION',defaultValue:'',description:'version to deploy on prod')

choice (name: 'VERSION',choices:['1.1.0','1.2.0','1.3.0'],description:'')

booleanParam(name:'executeTests',defaultValue: true, description:'')

}

environment {

//variables defined here can be used by any stage

NEW\_VERSION = '1.3.0'

}

stages {

stage('build') {

steps {

echo 'Building Project'

//using environment variable

//To output the value of variable in string use " "

echo "Building version ${NEW\_VERSION}"

echo 'successfully installed npm'

}

}

stage('test') {

when {

expression {

params.executeTests

}

}

steps {

echo 'Testing Project'

}

}

stage('deploy') {

steps {

echo 'Deploying Project'

echo "DEploying version ${params.VERSION}"

}

}

}

post {

// the conditions here will execute after the build is done

always {

//this action will happen always regardless of the result of build

echo 'Post build condition running'

}

failure {

//this action will happen only if the build has failed

echo 'Post Action if Build Failed'

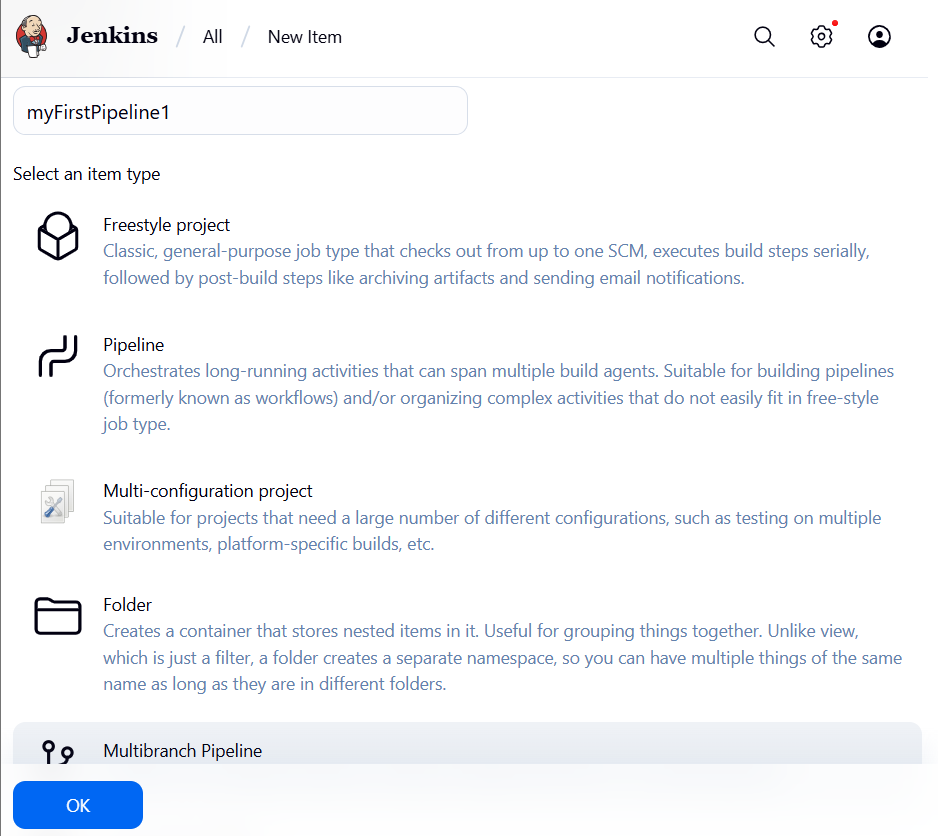
}

}

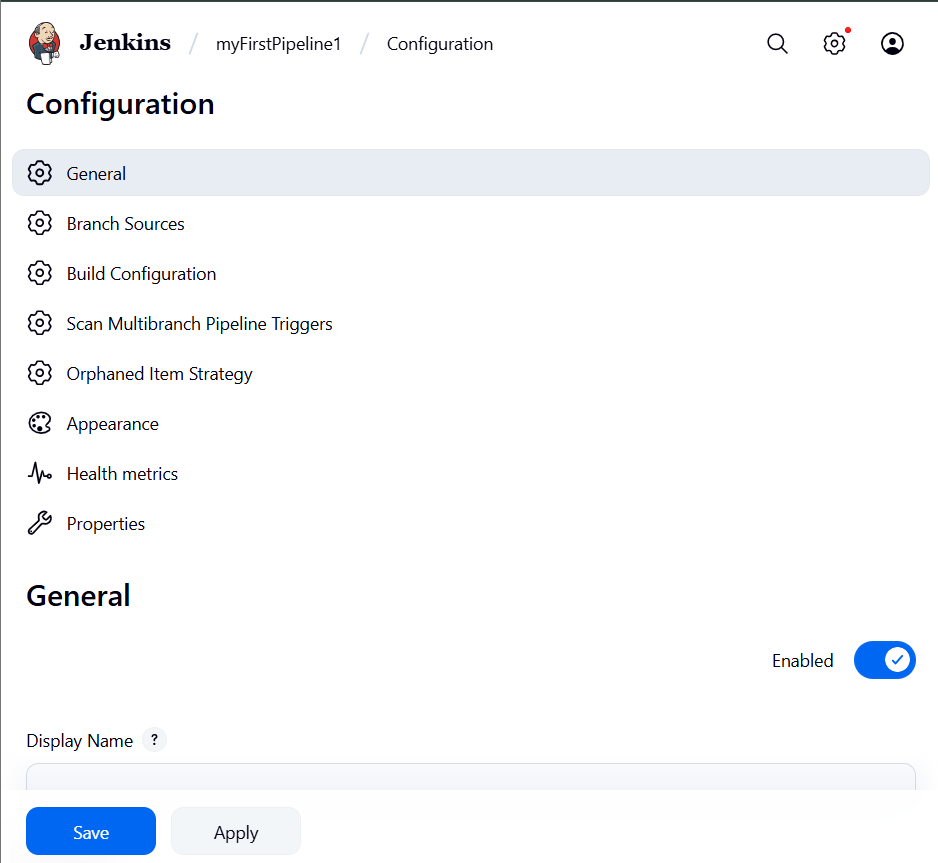
}

Now start Jenkins with this command:

* Java -jar Jenkins.war
* Go to Jenkins dashboard.
* Click on new item.
* Add name of your pipeline as myfirstpipeline.
* Select multibranch pipeline option:



* A new pipeline will be created:



Now click on add source option in branch sources select github

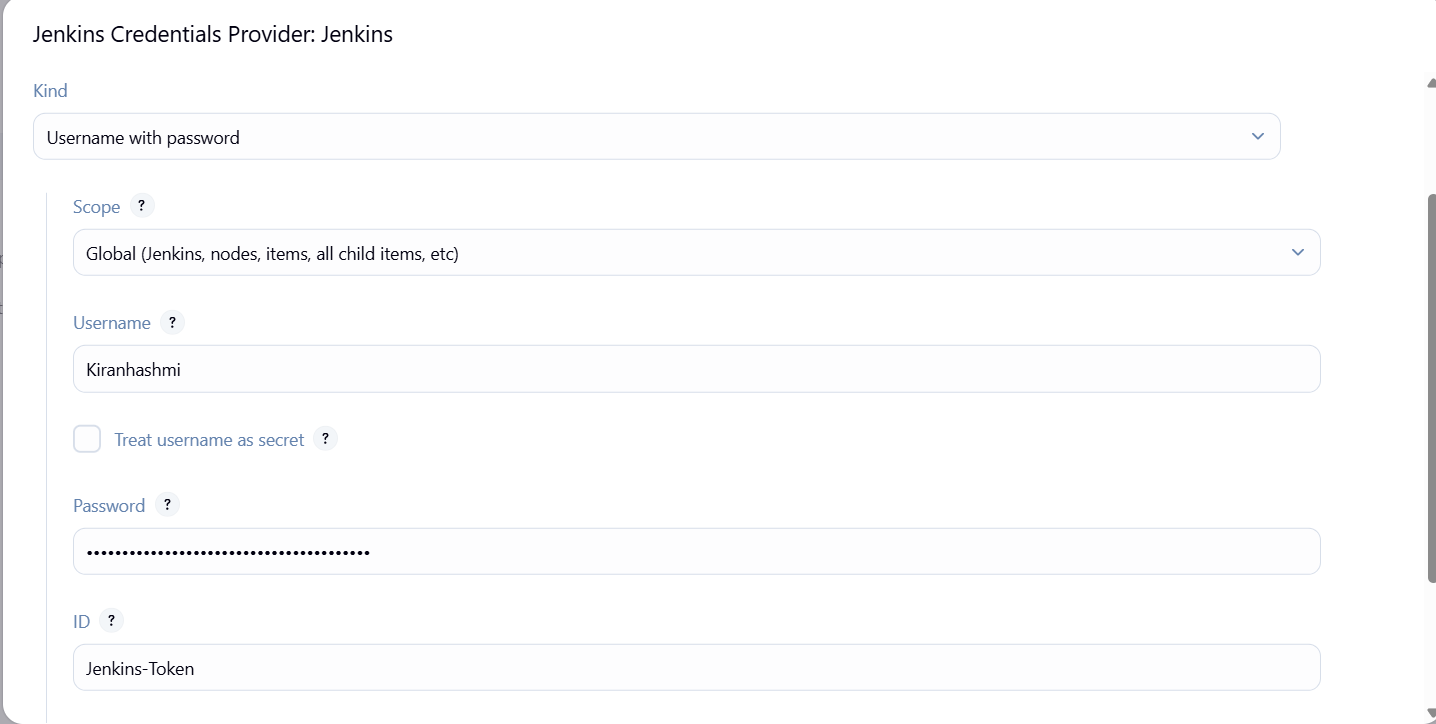
Generate a github token

1. Generate a GitHub Personal Access Token (PAT)

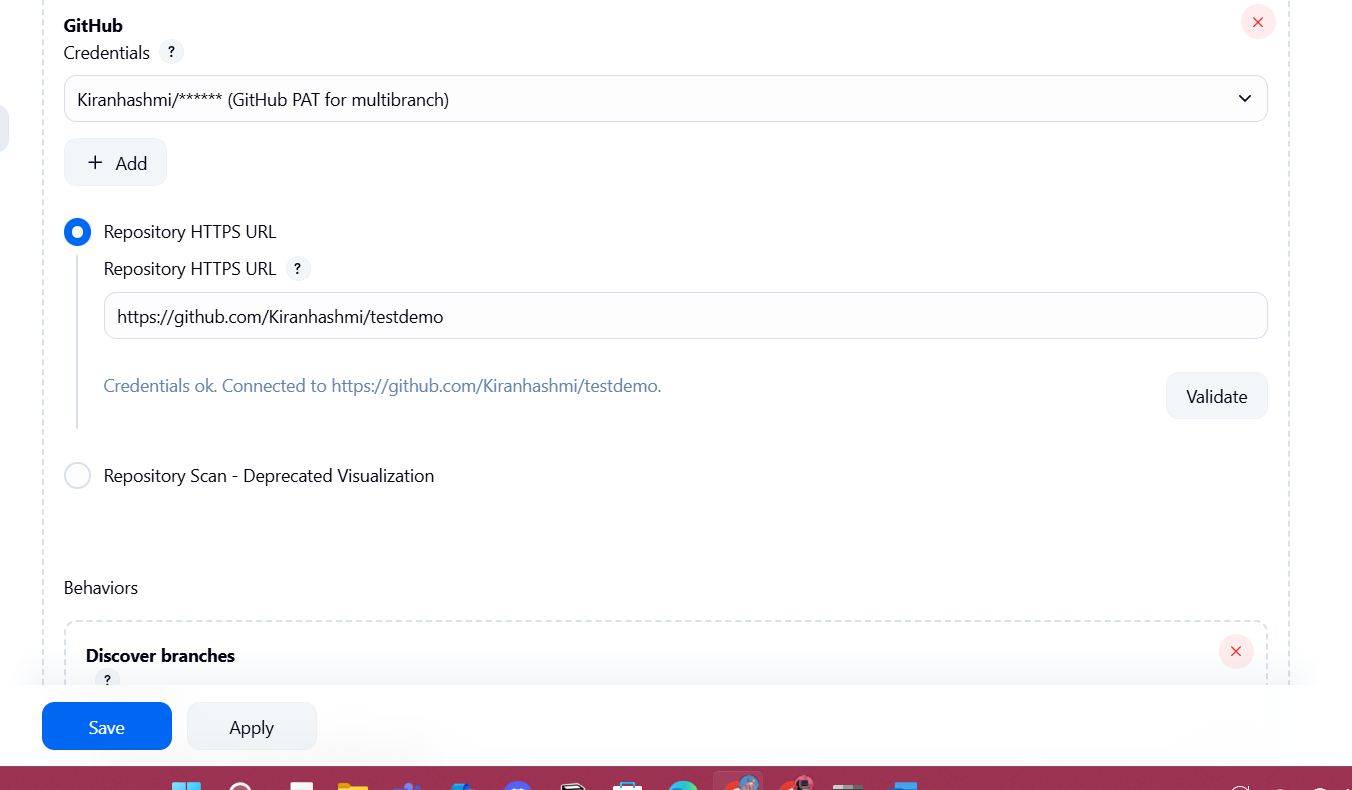
1. Go to: <https://github.com/settings/tokens>
2. Click "Generate new token" → Select Classic token
3. Name it like: JenkinsIntegrationToken
4. Select scopes:
   * repo (to allow Jenkins to read your private/public repos)
   * admin:repo\_hook (to allow webhooks)
5. Generate the token
6. Copy the token

Add Credentials in Jenkins

1. In Jenkins → in the screen you're on → click + Add
2. Choose “Jenkins” scope (global is fine)
3. In the credentials form:
   * Kind: Username with password
   * Username: Your GitHub username (e.g., kiranhashmi)
   * Password: Paste the GitHub token here
   * ID (optional): e.g., github-token
   * Description: e.g., GitHub PAT for multibranch



* Paste the link of your github repository where you created the JenkinsFile



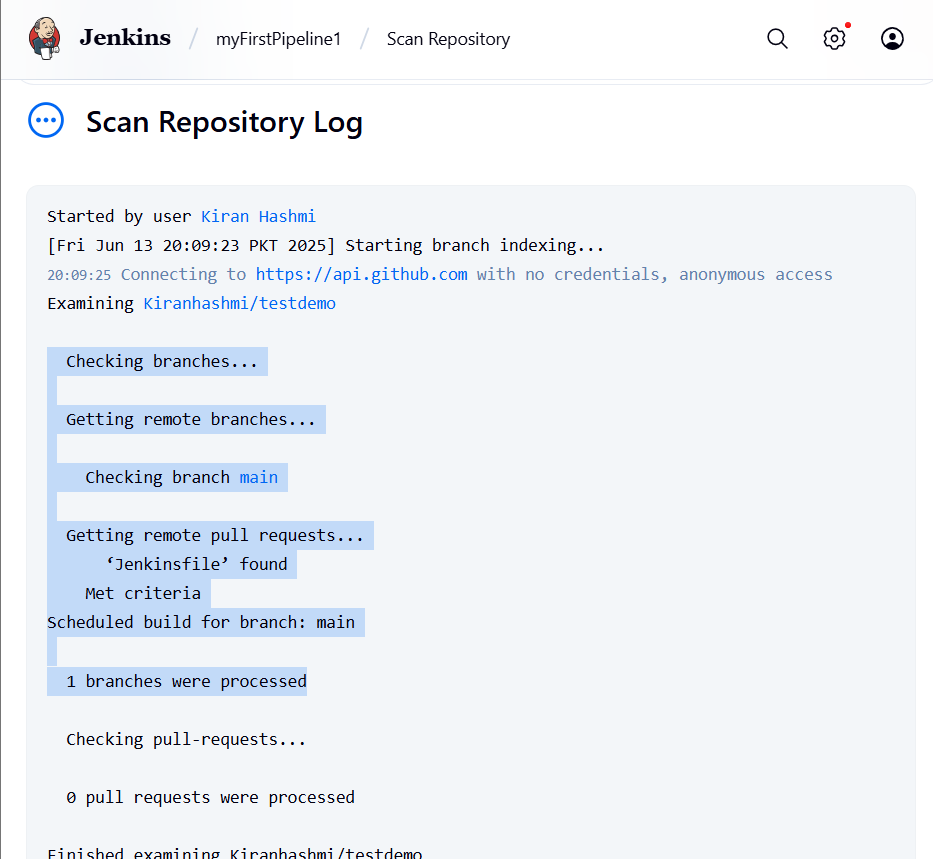
Validate the link to see if it is correct.

And click on save.

It will start scanning the repository.

It will scan all the branches one by one to look for Jenkinsfile

When it is found, the scanning stops and the exits successfully.



Why Did We Install Git for Jenkins?

Because Jenkins needs Git to clone your GitHub repository (i.e., download your code and Jenkinsfile) before it can run the pipeline.

1. Install Git (if not already installed)

* Go to: <https://git-scm.com/download/win>
* Download and install Git for Windows.
* After installing, verify it's available:
  + Open Command Prompt
  + Type: git --version
  + You should see something like: git version 2.xx.x

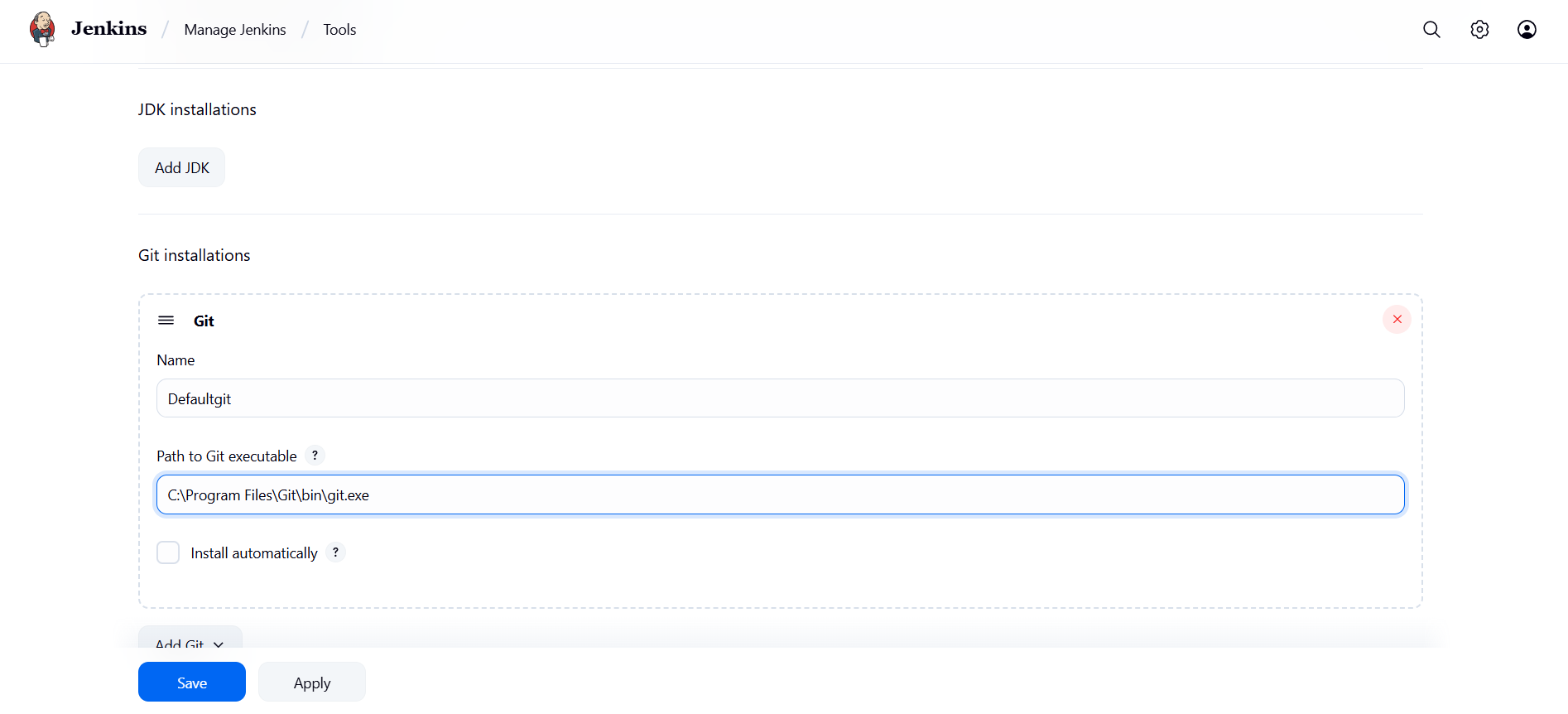
2. Configure Git in Jenkins

Step A: Open Jenkins settings

* Go to Jenkins Dashboard
* Click “Manage Jenkins”
* Click “Global Tool Configuration”

Step B: Add Git installation

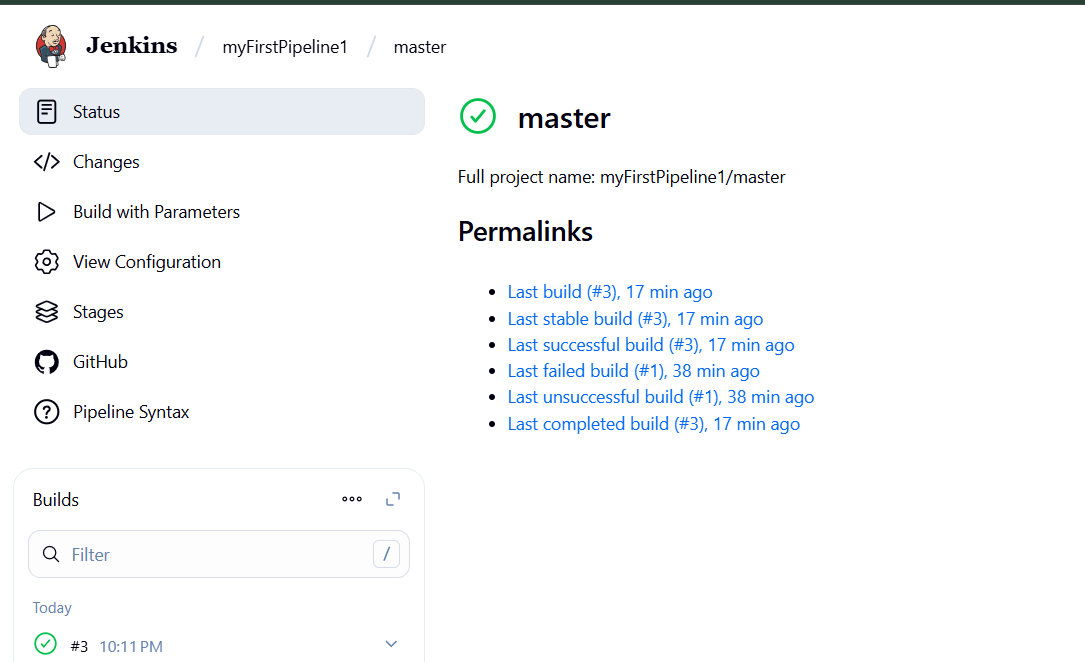
1. Scroll to Git section
2. Click “Add Git”
3. Give it a name (e.g., DefaultGit)
4. Uncheck "Install automatically"
5. In the Path to Git executable, enter:

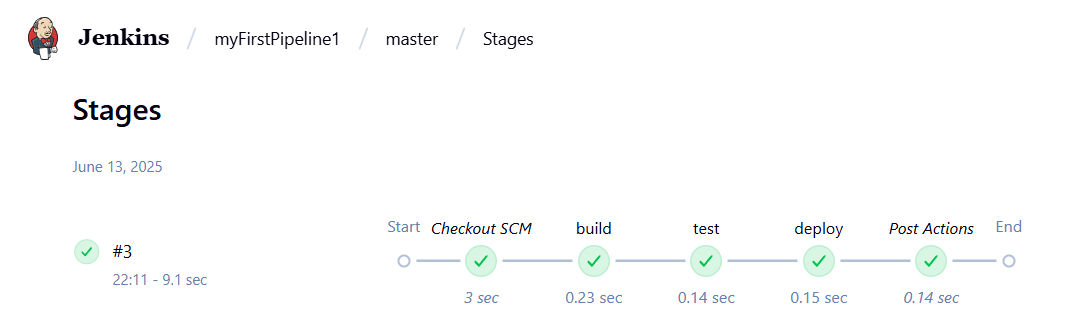


Now go back to the pipeline page:



* Click on master and you will be able to all the stages successfully build.





* From build history, click on Console Output:

Here you will see how the whole pipeline has been executed from building till

Deploying





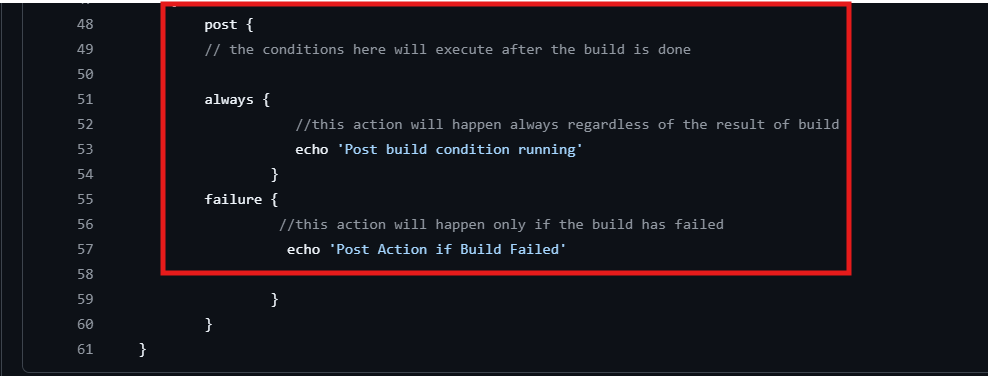
Post Build Actions:

 You can also perform some actions after the build is complete.

You just need to add another attribute named post in Jenkins file.

So, edit your Jenkins file from GitHub and write the post attribute like this, and

commit changes.



* Build the pipeline again and see the console output:

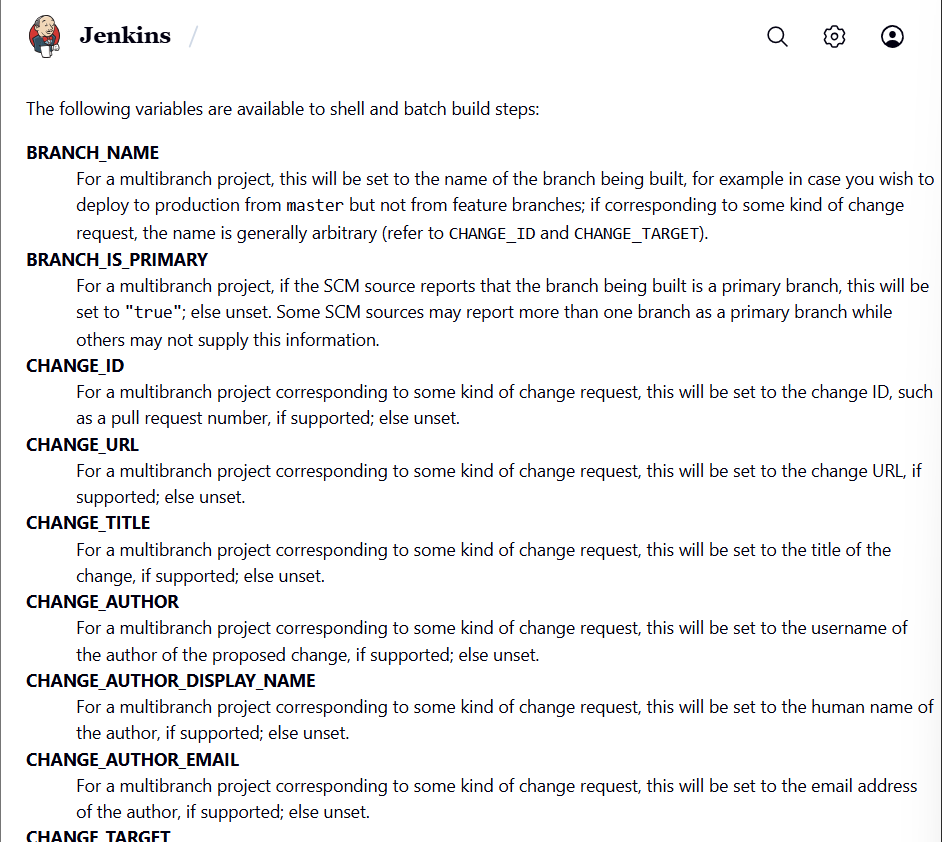


Environment Variables:

* You can also use or change the environment variable available in Jenkins.
* What variables are available in Jenkins?

Type this: http://localhost:8080/env-vars.html/. You will see a list of all the

environment variables.



You can also define your own environment variables in the Jenkins file.

The attribute for this purpose is named as environment. It is defined before the

stages so that it is accessible by all stages.

Suppose we need a specific version of something in all stages:



Here first we defined the variable in the environment attribute and the used it in

the build stage.

 Now build the pipeline again and see the console output.



Tools attribute for build tools:

Using tools attribute, you can access different tools for your projects.

* Jenkins file support 3 tools right now that are:
* Maven, gradle, and JDK
* We will include Maven (A build automation tool used primarily for Java projects, mainly in
* downloading dependencies etc.)
* Edit your Jenkinsfile to add the following lines.
* Here first we listed Maven in the tools list.
* Then we used a command to install maven during build.

