1. Write a code for simple user registration for an event.

Code:

```
<!DOCTYPE html>
<html>

<head>
  <title>Registration Form</title>
  <link tyle="stylesheet" href="styles.css">
</head>

<body>
  <center class="bg">
    <h1>Registration Page</hi>
      <form action="registration" method="GET">
        <label>First Name </Label>
        <input type="text" name="name" placeholder="Enter Your name"><br>
        <label> Date Of Birth</label>
        <input type="date" name="dob"><br>
        <label>Email</label>
        <input type="email" name="email" placeholder="enter email"><br>
        <label> Gender</label>
        <input type="radio" name="Gender"> Male </input>
        <input type="radio" name="Gender"> female</input>
        <br>
        <label> Phone No</label>
        <input type="tel" name:"Phno"></input>
        <br>
        <label>Address</label>
        <input type="rextasen" rows=5 cols=5 name="adde"></input>
        <br>
        <input type="checkbox"> Accept terms and conditions
        <br>
        <label>Branch</label>
        <select name="Ops">
          <option value=05>CSE </option>
          <option value=66>csm</option>
          <option value=027> EEE </option>
          <option value=04> 042 ECE </option>
        </select>
        <br>
        <button type="submit">Submit </button>
        <button type="reset"> Reset</button>
      </form>
  </center>
</body>
</html>
```

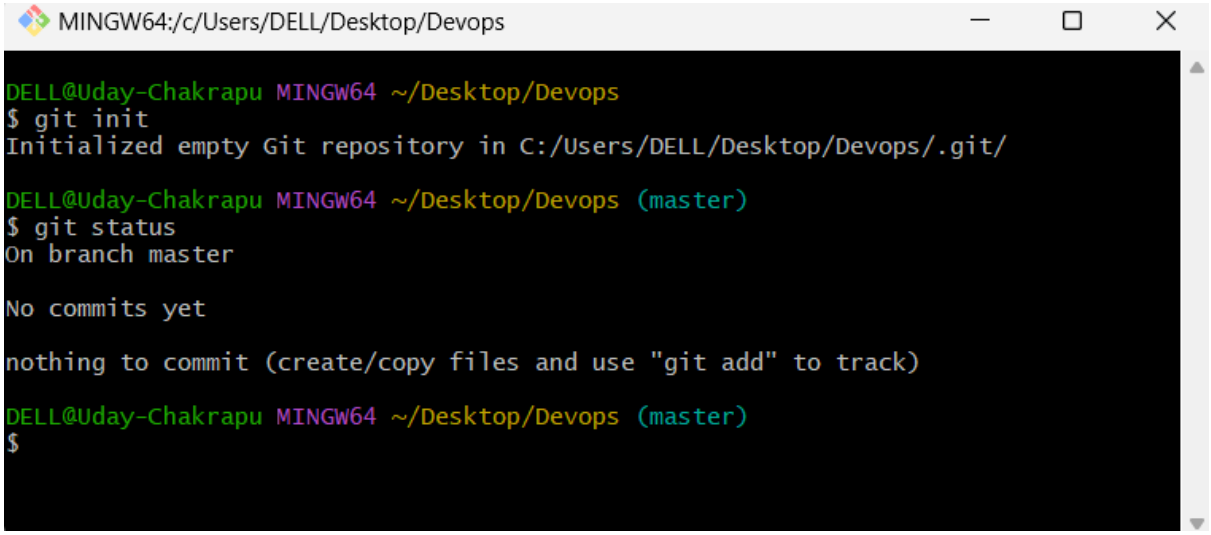
Output:

The following is the output when opened it in browser.

---

2. **Explore Git and Github Commands Aim: Explore git and github commands.**

- Initializing a Git repository: **~$ git init**
- Checking the status of your repository: **~$ git status**



- Creating new html file **: ~$ touch src**
- Creating another file **: ~$ touch index.html**

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ touch src

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ touch index.html

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$
```

- Checking the status of git: **~$ git status**

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html
        src

nothing added to commit but untracked files present (use "git add" to track)

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$
```

- Adding files to git : **~$ git add .**
- Checking status of git : **~$ git status**

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ git add .

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   src


DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$
```

- To remove file **: git rm –cached src**
- Checking status of git : **git status**

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ git rm --cached src
rm 'src'

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        src


DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$
```

- Editing the index.html file : **~$ vi index.html**

- To remote : ~**$ git remote add origin URL**
- Check remote : **~$ git remote**

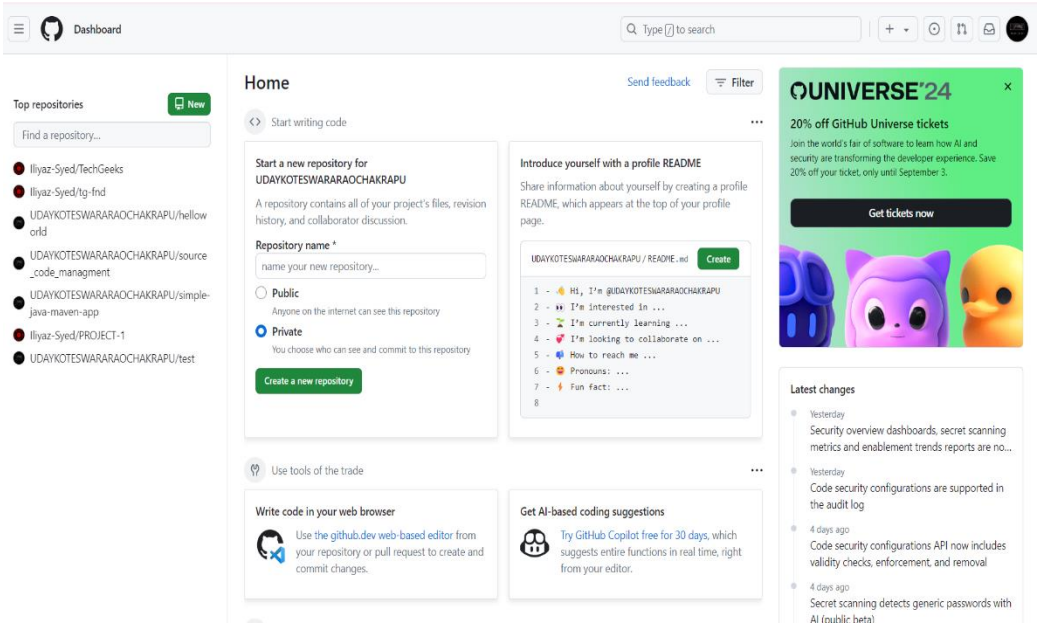
```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ git remote add origin https://github.com/UDAYKOTESWARARAOCHAKRAPU/test.git

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ git remote
origin

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$
```

3. Practice source code management in Github.

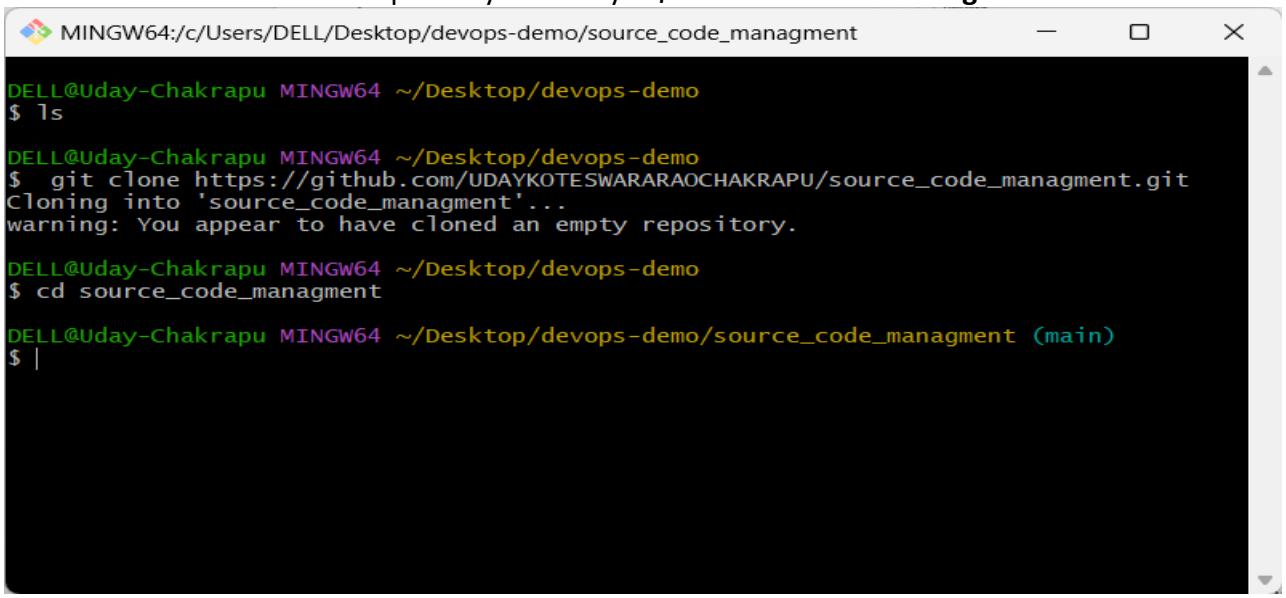
Description:

To practice source code management on GitHub, you can follow these steps:

- Create a GitHub account if you don't already have one
- Creat a new repository “source_code_managment”



- Clone the repository to your local machine: **~$ git clone https://github.com/UDAYKOTESWARARAOCHAKRAPU/source_code_managment.git**
- Move to the repository directory**: ~$ cd source-code-management**



- Initialize the directory : **~$ git init**
- Create a new file : **~$ touch registration.html**
- Add the source code to it **: ~$ vi registration.html**

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$ touch Registration.html

DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$ vim Registration.html

DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$
```

- Check status of git **: ~$ git status**

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Registration.html

nothing added to commit but untracked files present (use "git add" to track)

DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$
```

- Adding file to the stage : **~$ git add registration.html**
- Checking git status : **~$ git status**

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$ git add Registration.html

DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Registration.html


DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$
```

Now we can stage up the changes in the repo from the add area to commit area so that they can be pushed to any sort or remote git client such as ,
→ Github
→ Gitlab
→Bitbucket etc.

- Commiting changes : **~$ git commit -m " first commit"**
- Checking the status of git : **~$ git status**

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$ git commit -m "first commit"
[main (root-commit) 55b8caa] first commit
 1 file changed, 50 insertions(+)
 create mode 100644 Registration.html

DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$ git status
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean

DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$
```

- Pushing the : **~$ git push**

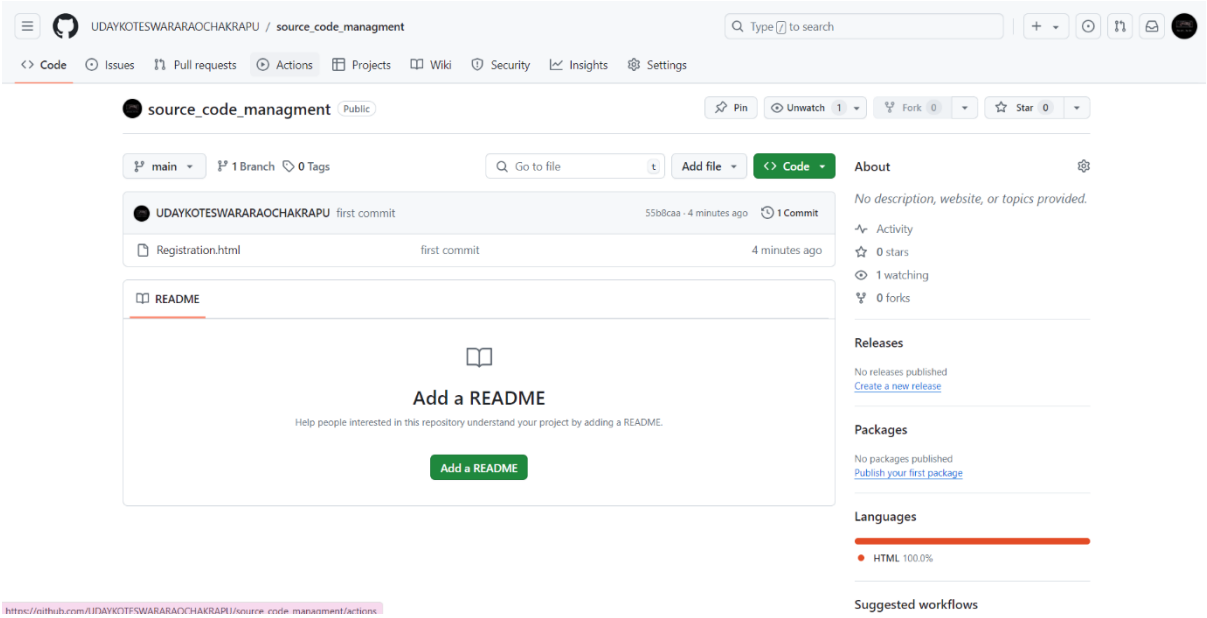
```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 776 bytes | 388.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/UDAYKOTESWARARAOCHAKRAPU/source_code_managment.git
 * [new branch]      main -> main

DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$
```

- Here we can see that the command is successful and the explanation is as follows.

- Initially it enumerates through all the changes and then compresses them unless they use git LFS and writes them to buffer which follows by sending them to the remote git repository and merges with main branch if needed.

- After successful pushing , Now go to Github and open your repository in it. Then you can see that your recent changes have been populated successfully to the repository.

- So this is how the source code is managed using Github and Git.



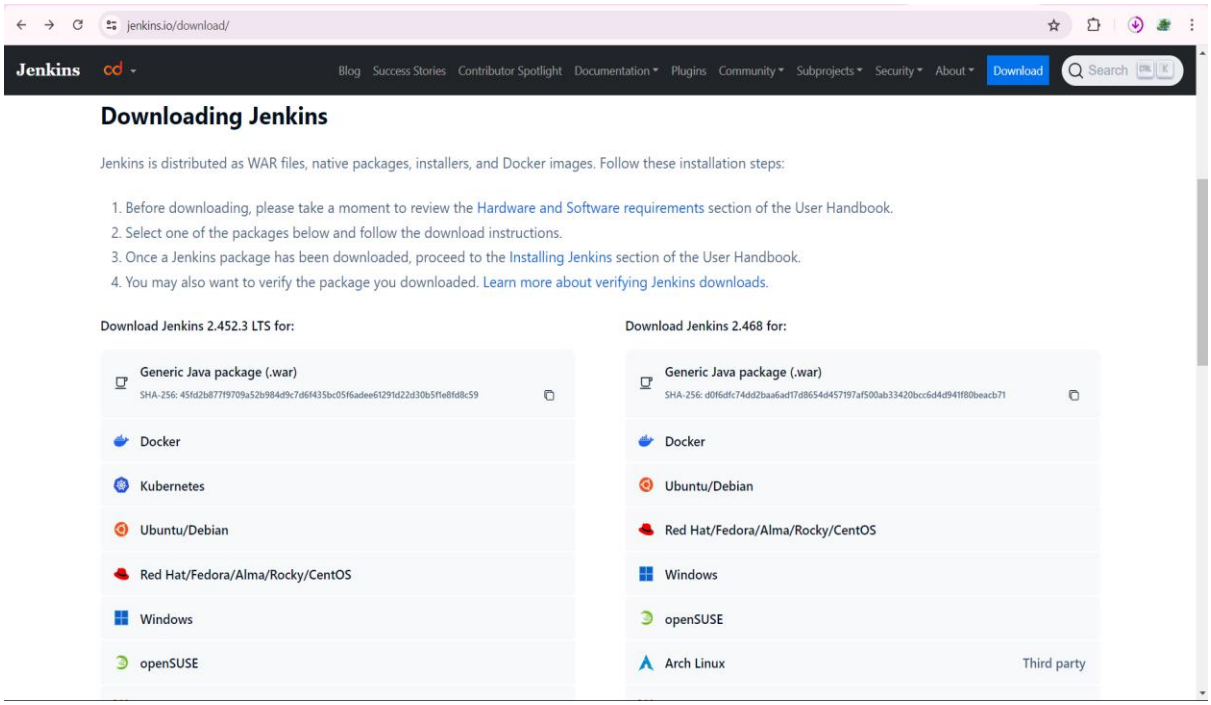
4. Jenkins Installations and setup, Explore the environment.

Jenkins: It is an open source optimization server developer which is used to automate building,testing and deploying application. It is available in Windows,Linux,Unix and Macos.

Installation:

Step 1:

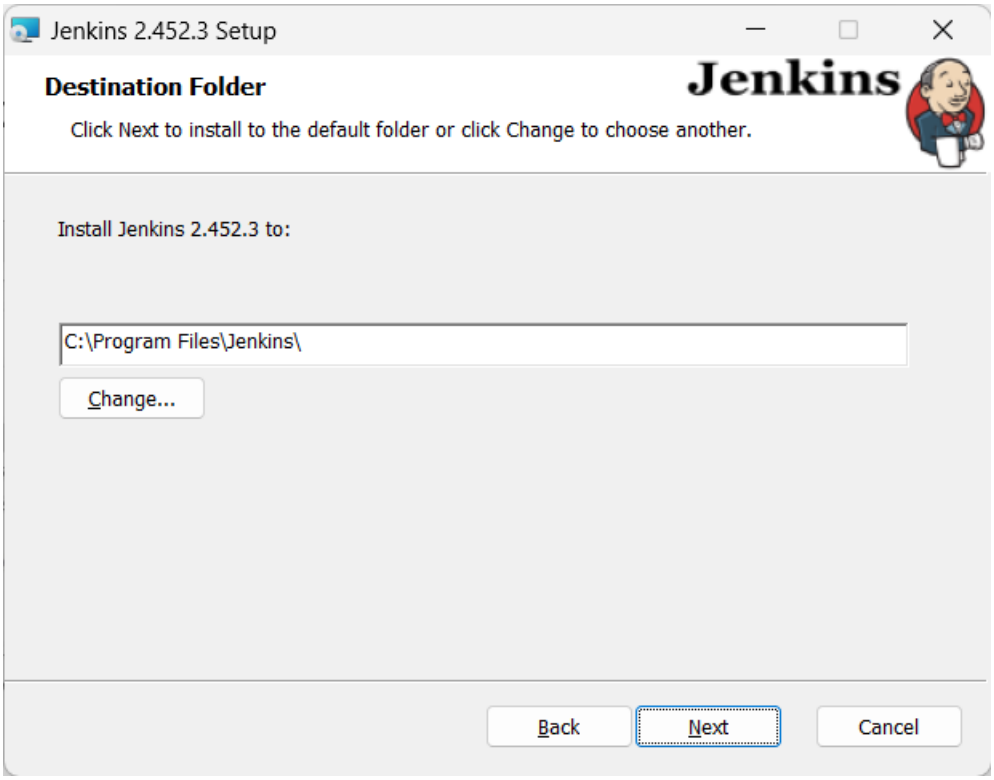
Browse to the Jenkins download page Under the **Downloading Jenkins** section is a list of installers for the long-term support (LTS) version of Jenkins. Click the **Windows** link to begin the download.



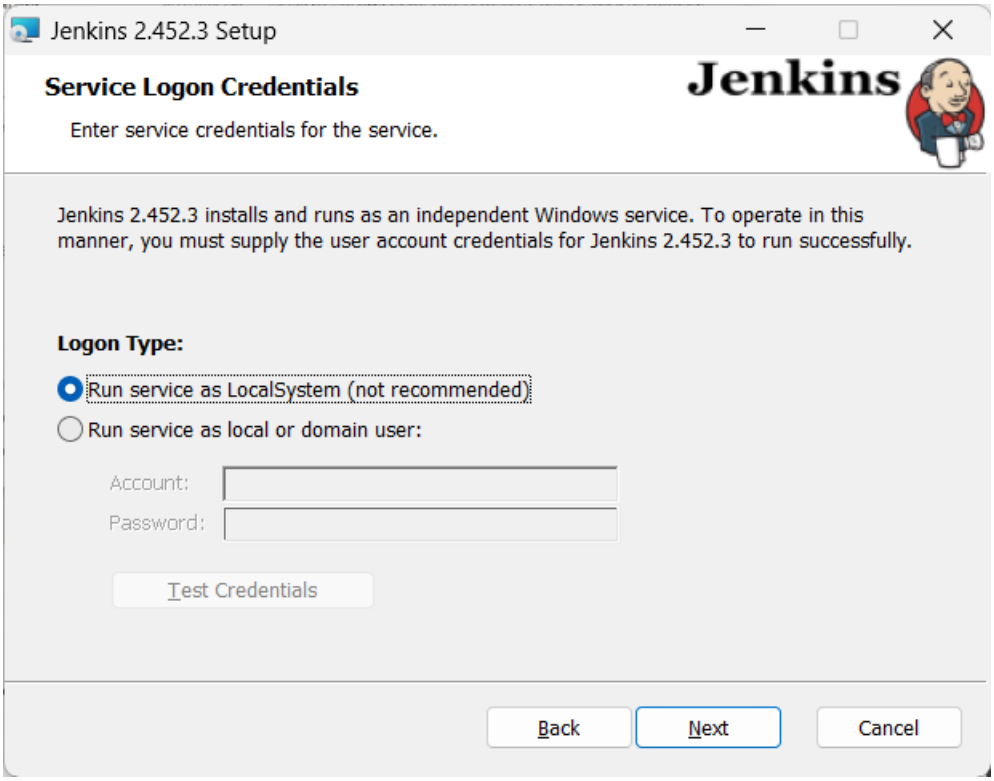
Step 2 : Once the download is complete, run the jenkins.msi installation file.

Step 3 : The setup wizard starts. Click Next to proceed.

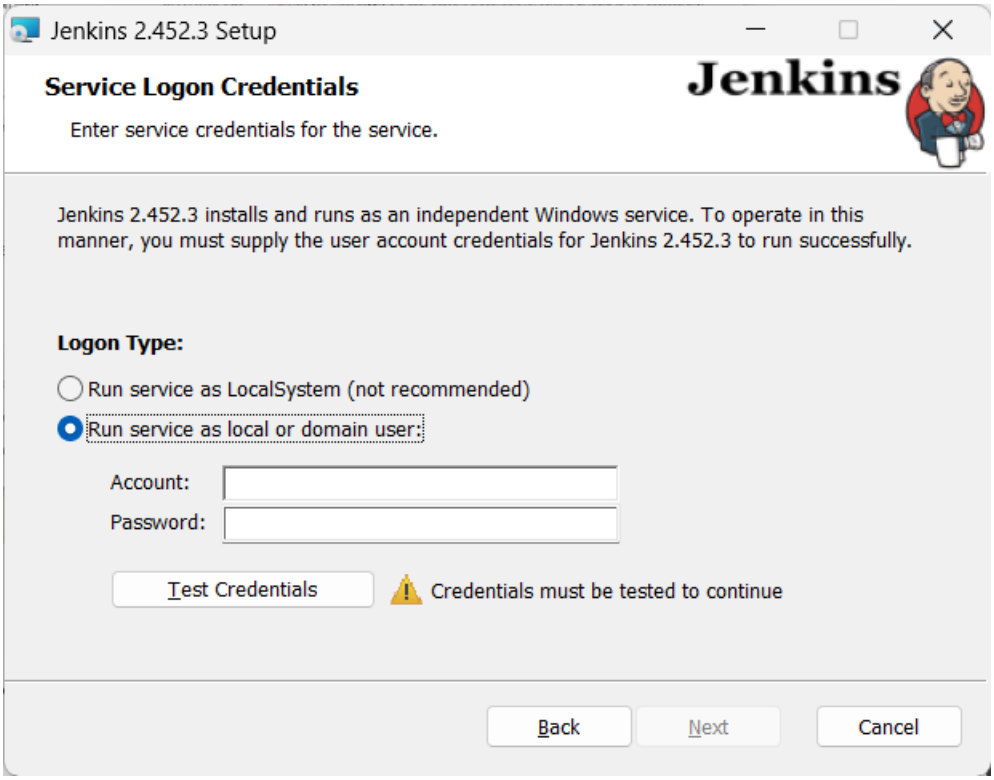
Step 4 : Select the install destination folder and click **Next** to continue.



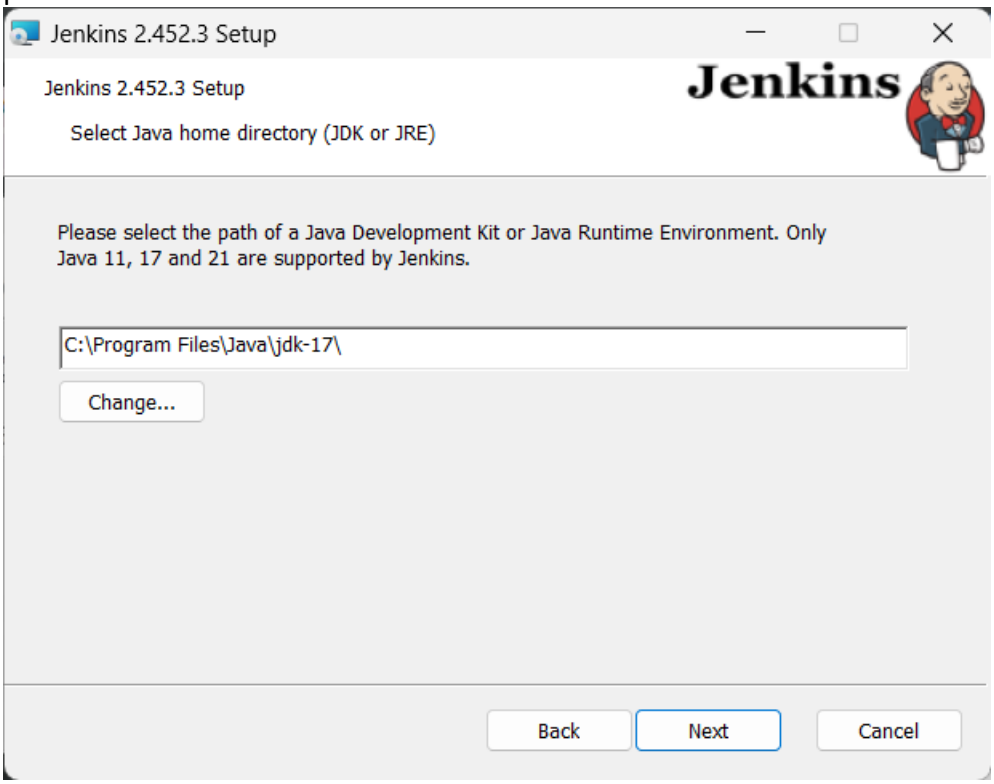
Step 5 : click on **Run service as a localsystem** option, and click **NEXT** to continue



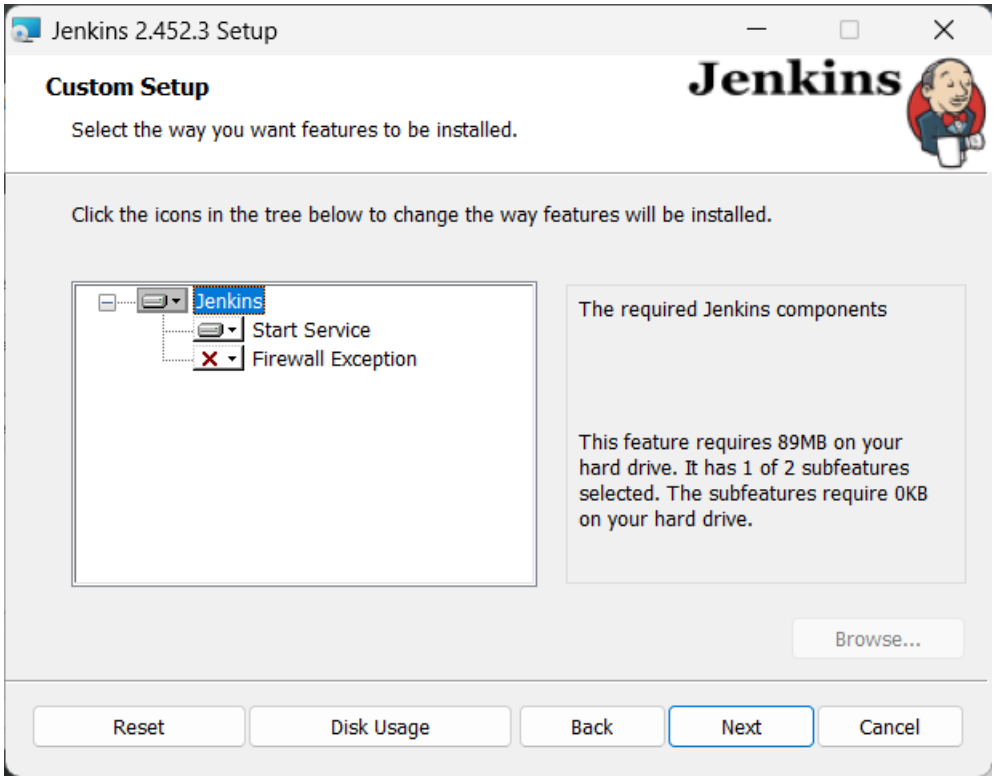
Step 6 : Enter the port number you want Jenkins to run on. Click **Test Port** to check if the selected port is available, then click **Next** to continue.



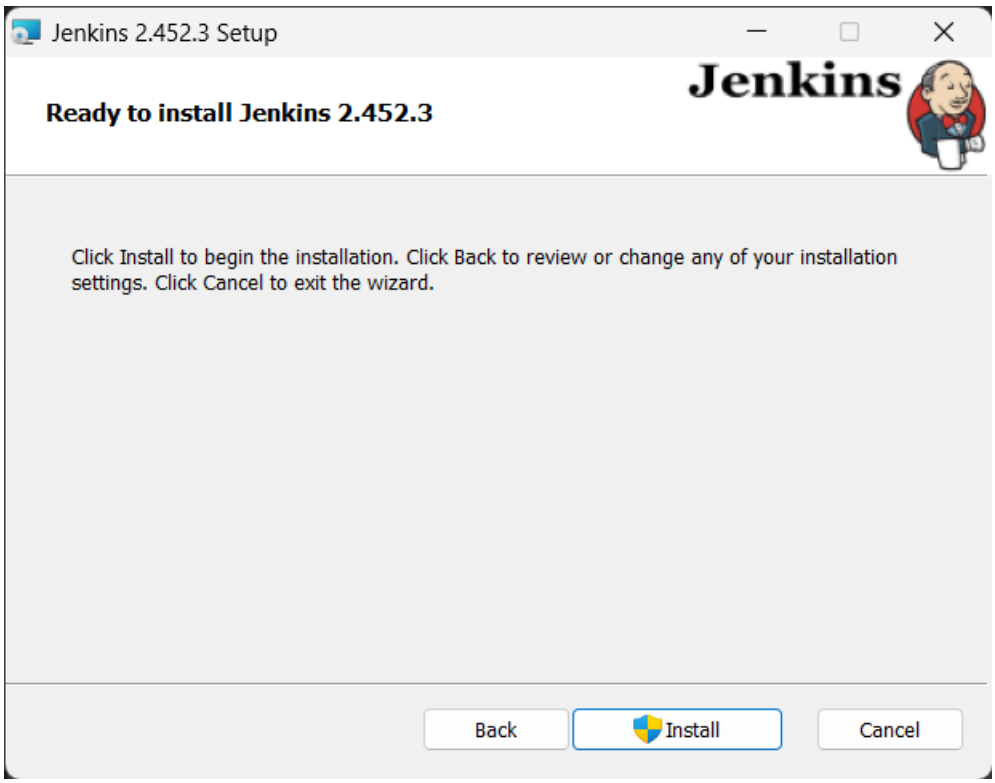
Step 7 : Select the directory where Java is installed on your system and click **Next** to proceed.



Step 8 : Select the features you want to install with Jenkins and click **Next** to continue



Step 9 : . Click **Install** to start the installation process.



Step 10 : Once the installation is complete, click **Finish** to exit the install wizard

Unblock Jenkins :

After completing the installation process, you have to unblock Jenkins before you can customize and start using it.

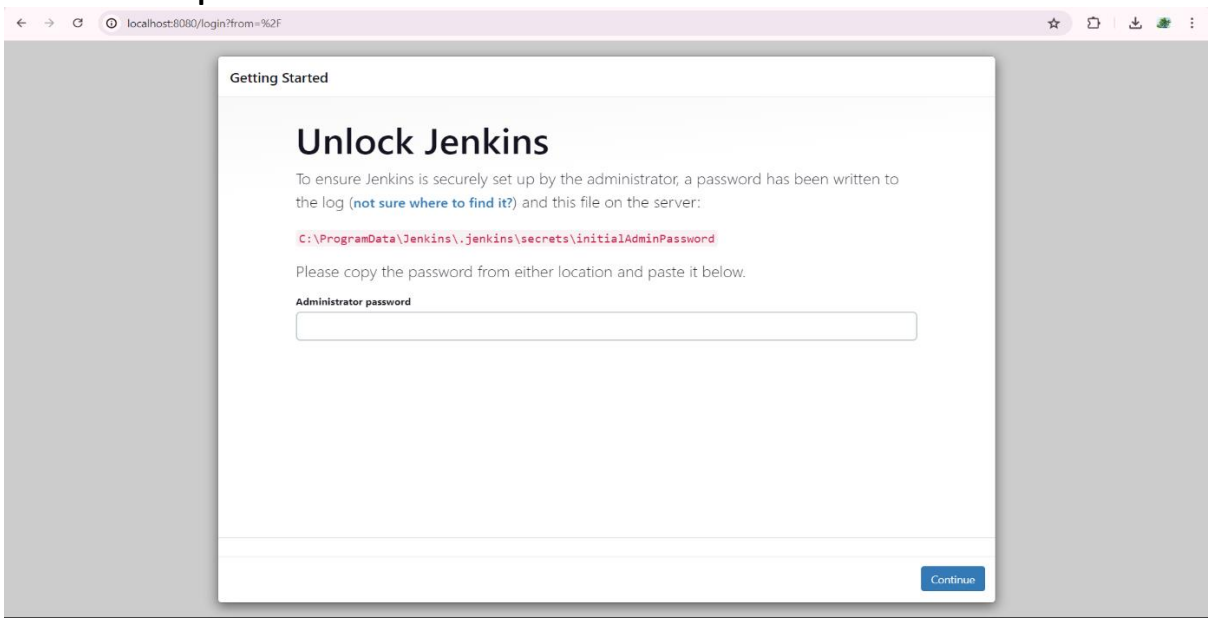
Step 1 : In your web browser, navigate to the port number you selected during the installation using the following address:

http://localhost:8080

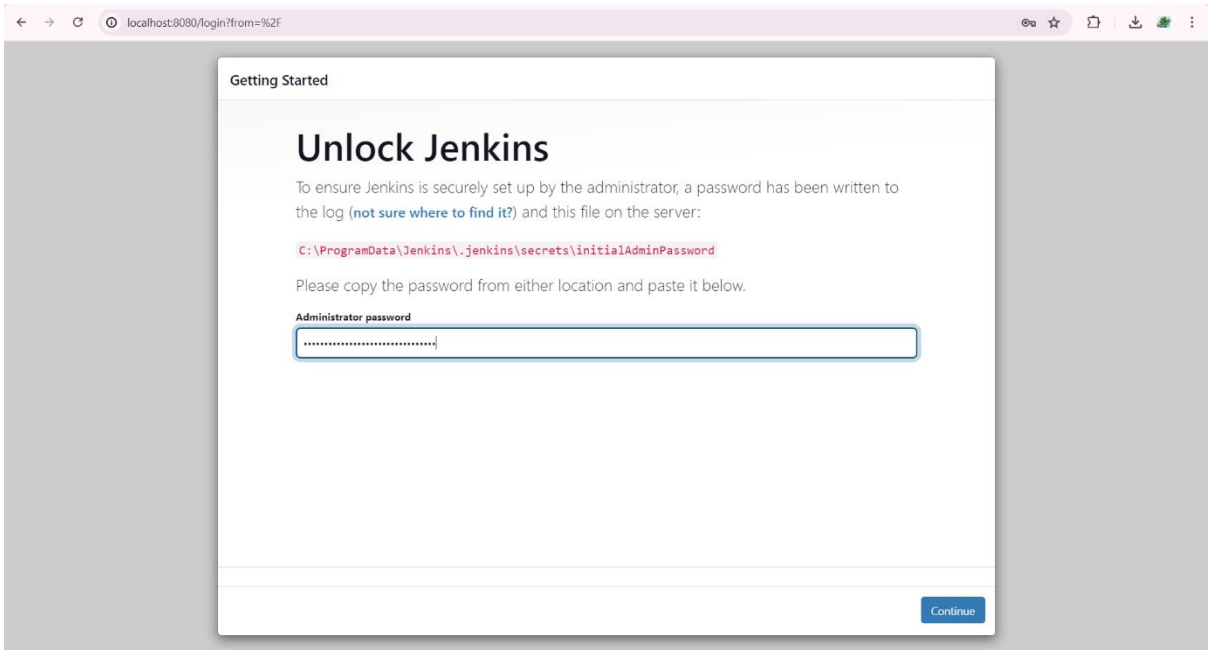
Step 2 : Navigate to the location on your system specified by the Unblock Jenkins page.

C:\ProgramData\Jenkins\.jenkins\secrets\initial admin. Password

Administrator password :

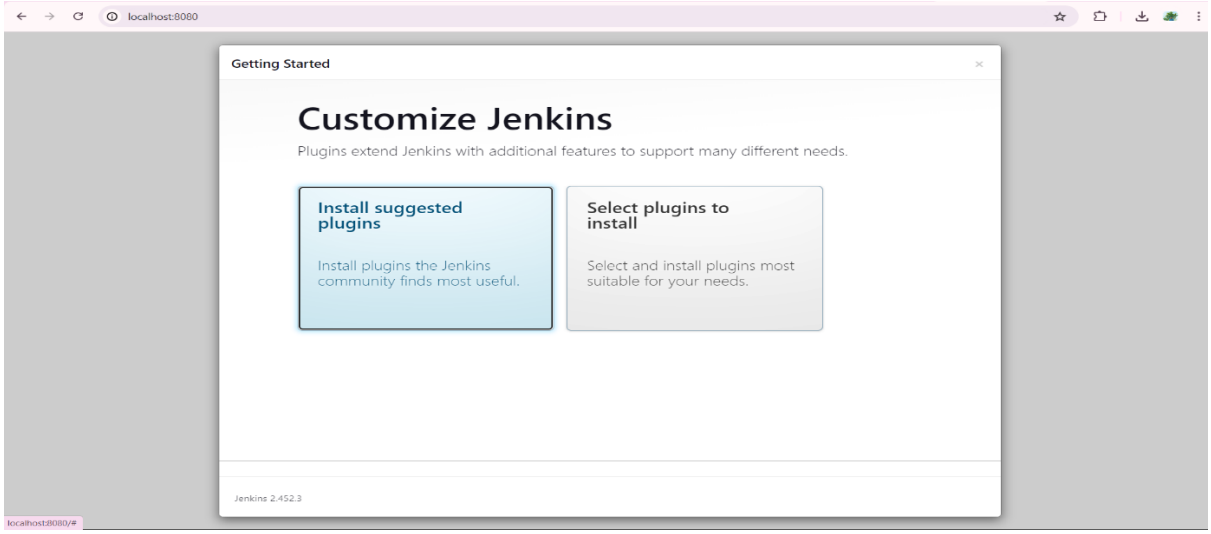


Step 3 : Open the **initialAdminPassword** file using a text editor such as Notepad.
Step 4 : Copy the password from the **initialAdminPassword** file.And paste it in the “administrator password” field on the unblock
Jenkins page and click on continue to proceed

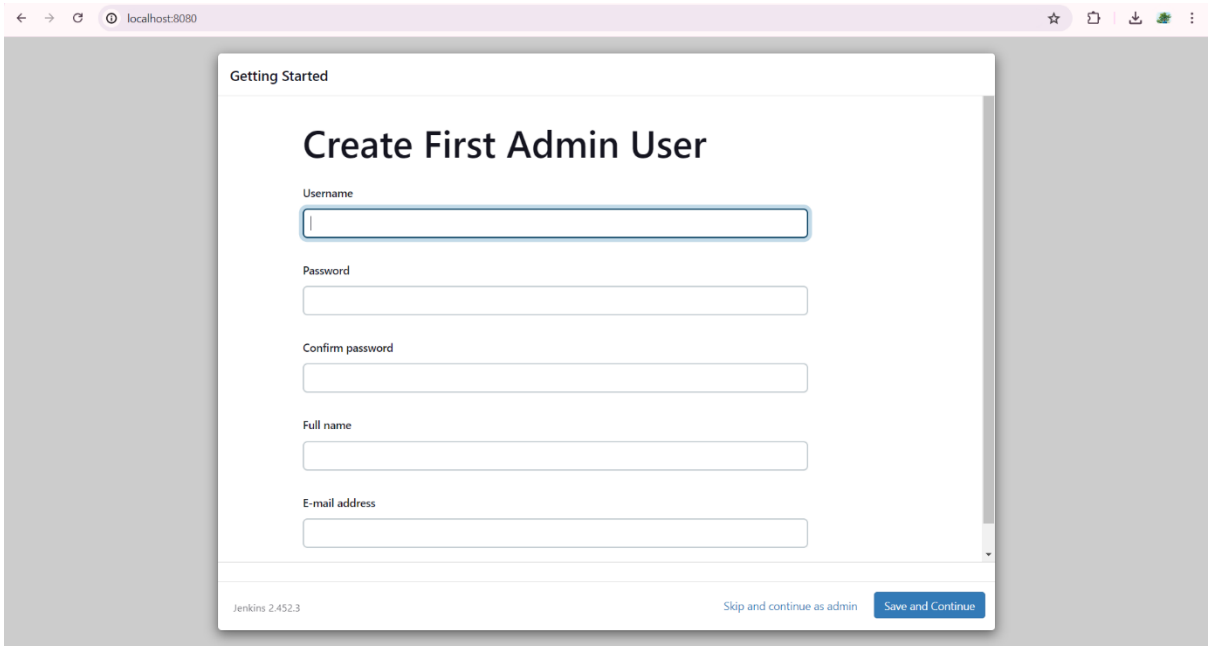


Customize Jenkins
Once you unlock Jenkins, customize and prepare the Jenkins environment.

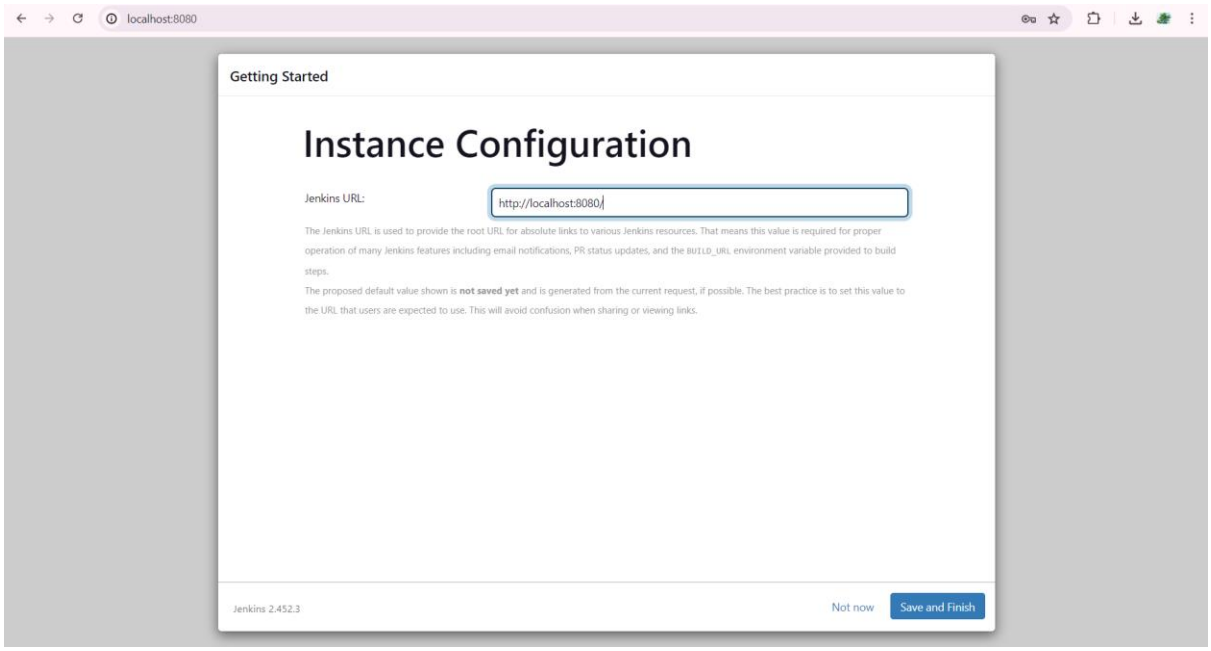
Step 1 : Click the **Install suggested plugins** button to have
Jenkins automatically install the most frequently used plugins



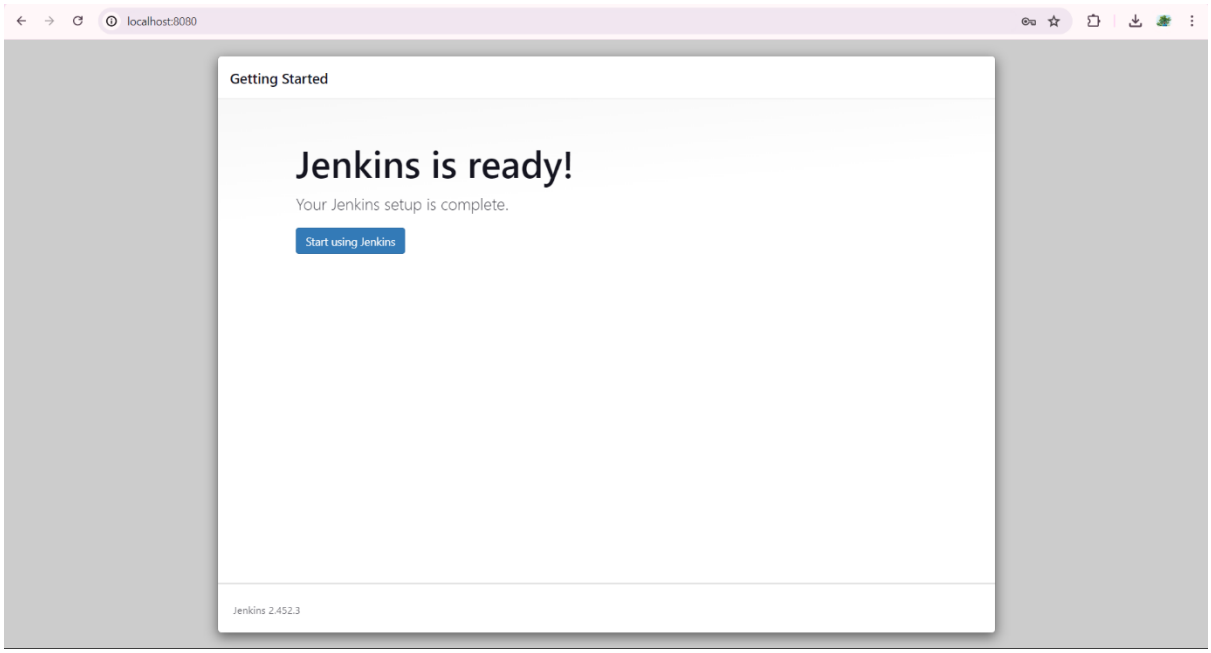
Step 2 : After Jenkins finishes installing the plugins, enter the required information on **the** Create First Admin **User** page.
Click **Save and Continue** to proceed



Step 3 : On the **Instance Configuration** page, confirm the port number you



Step 4 : Click the **Start using Jenkins** button to move to the Jenkins dashboad

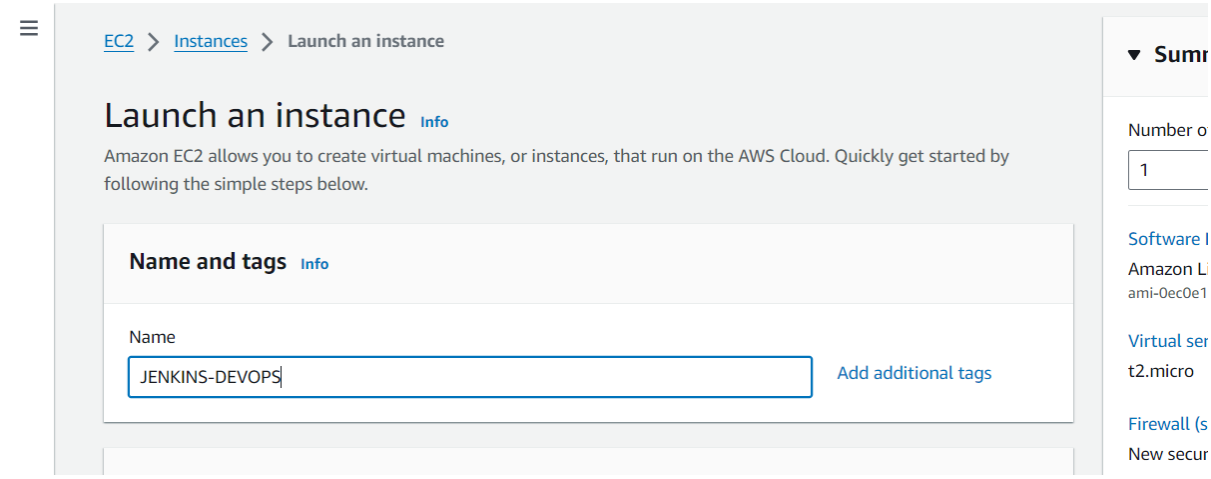


Jenkins installation in AWS instance :

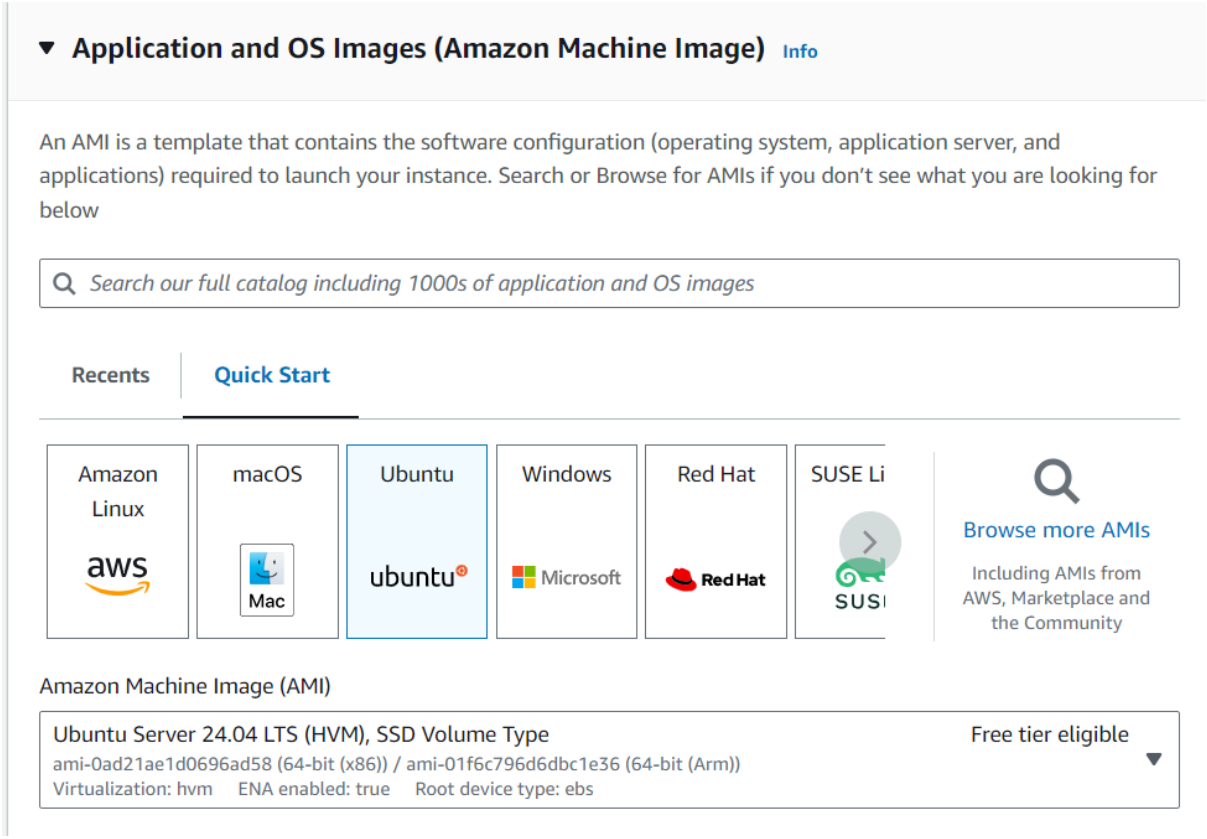
Steps to install Jenkins in aws :

- Login to AWS account with your email and password.
- opens a AWS home page and click on EC2.
- opens EC2 Dashboard,now launch instance.

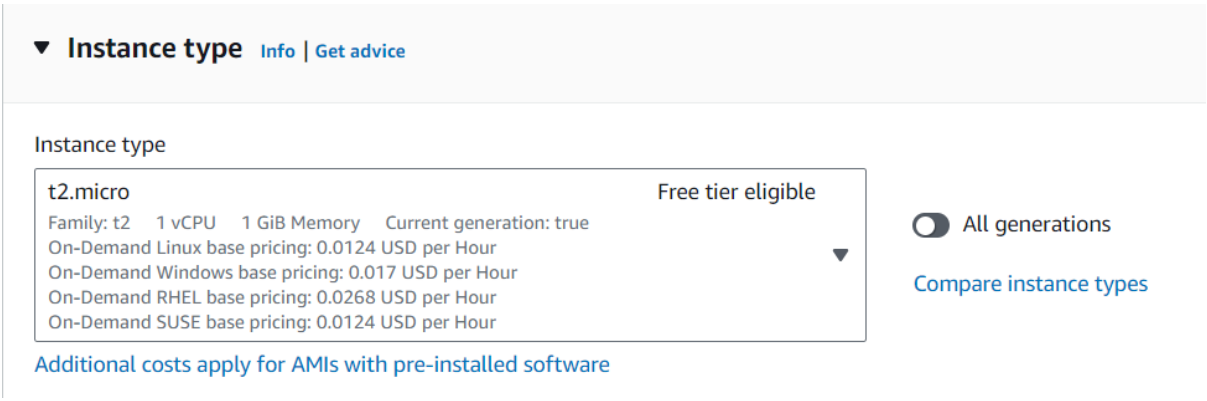
- Launch an instance and name it as **JENKINS**



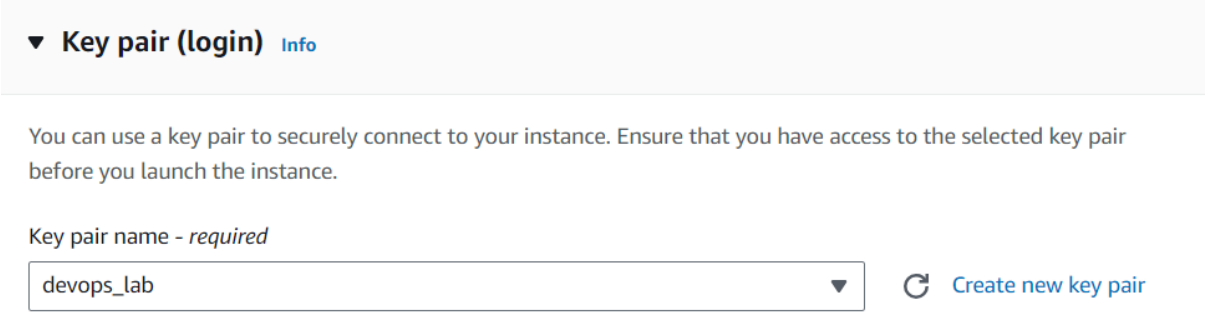
- Select application and OS images “UBUNTU”



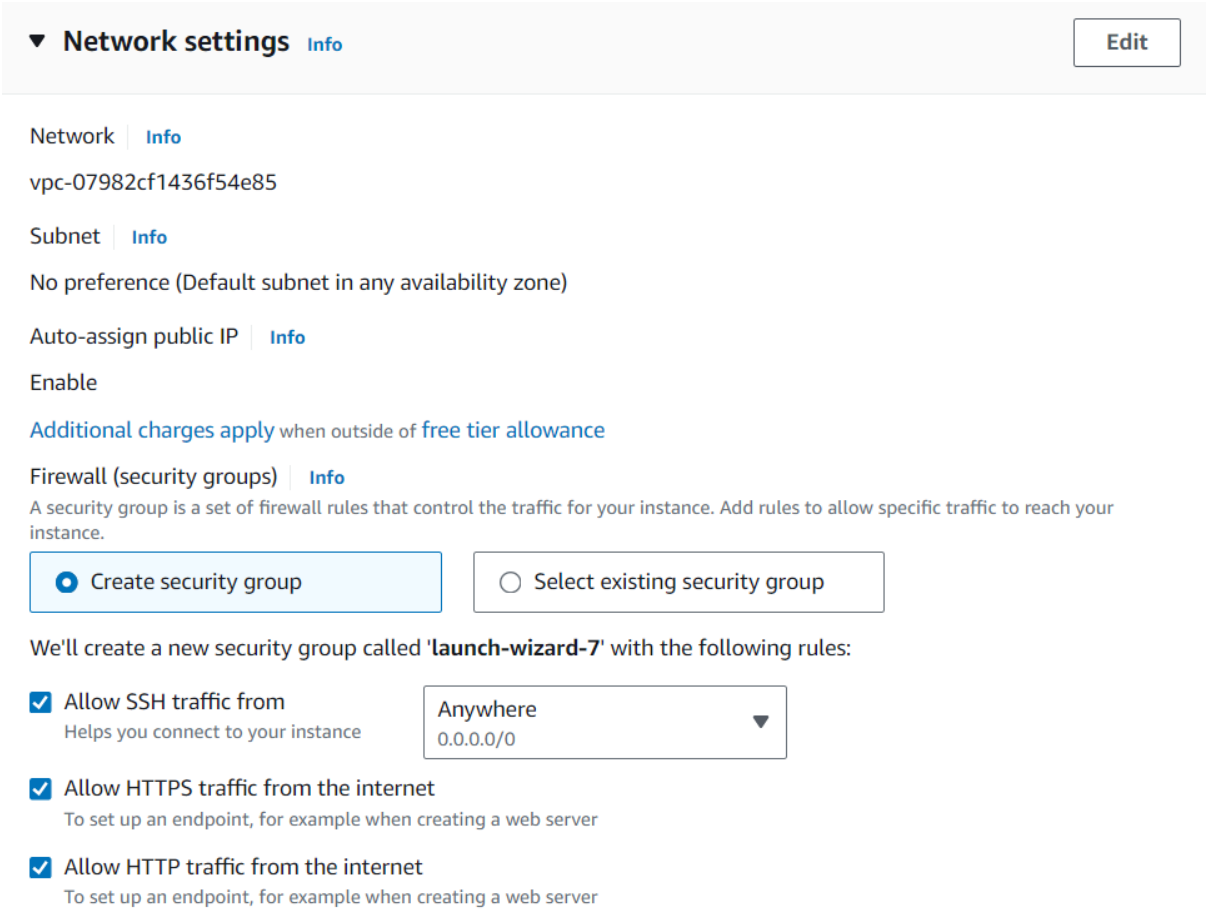
- Select instance type as t2-micro



- Select a Key Pair or create one If not for accessing your instance from remote places.

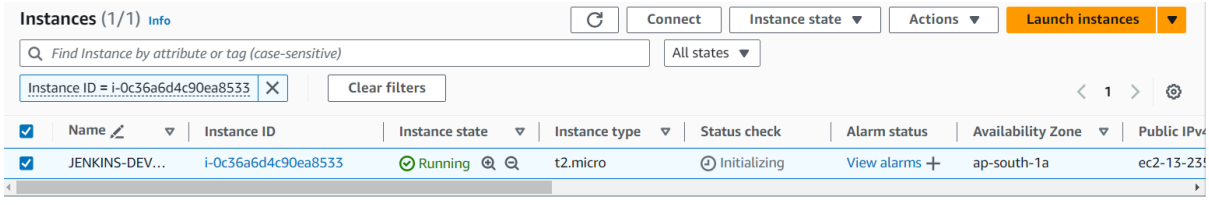


- Select network setting and allow all for accessing it through everywhere with keypair and letting ports open and used for serving web apps.

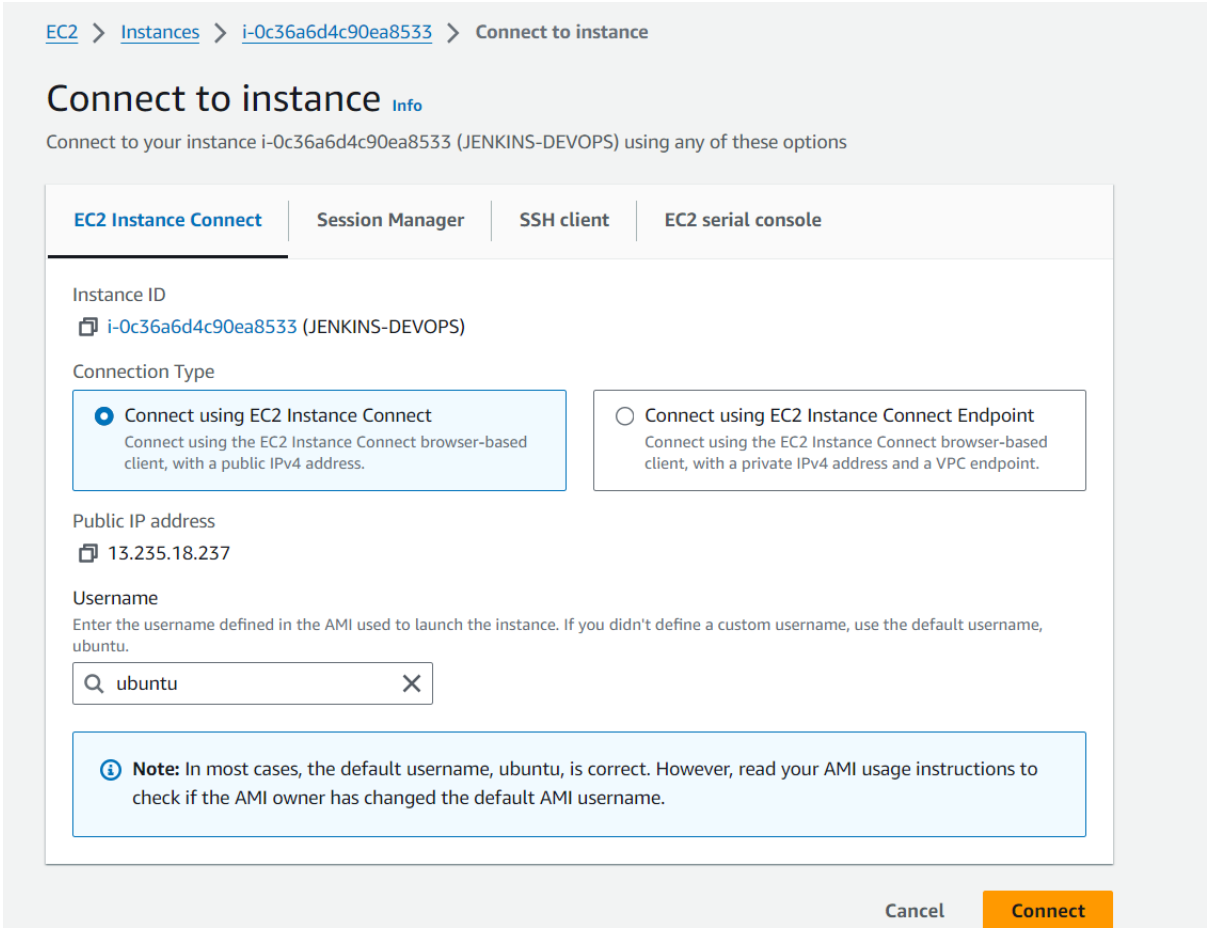


- Now click on Launch an instance.

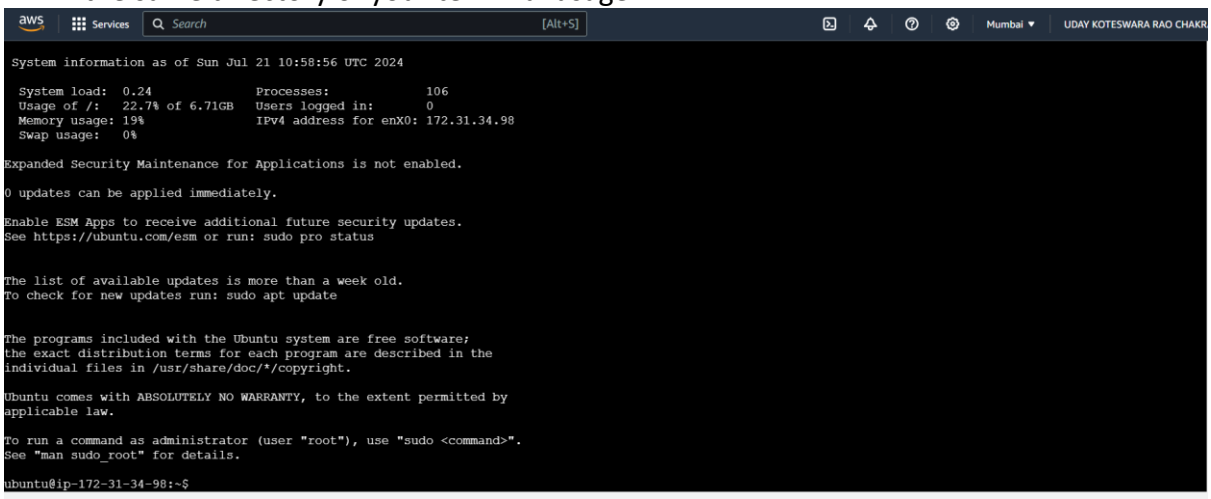
- **Now click on your created instance and click connect**



- **Connect to instance :**



- After successful connection to instance a new web page containing the shell access is opened for writing down the commands on your instance.
- This terminal is even accessible from the any computer if it supports the openssh and ssh command to begin with.
- If your system is a unix based machine then it will be built it , if it's a windows there might be a chance of missing ssh command.
- Then try installing git bash from chrome and use it for such instance related commands for even more simplicity.
- For you to login through ssh the previously created key pair file i.e., pem file must be in the same directory of your terminal usage.

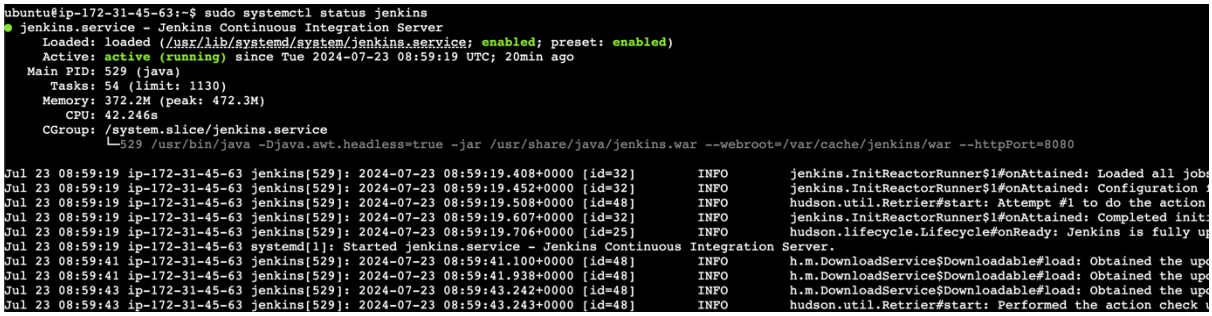


- Now enter the below commands in order to install Java , Maven and Jenkins

- **~$** sudo apt update
- **~$** sudo apt upgrade -y
- **~$** sudo apt install maven -y
- **~$** sudo apt-get install openjdk-11-jdk -y
- **~$** sudo curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

- **~$** sudo echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
- **~$** sudo apt update
- **~$** sudo apt install jenkins
- **~$** sudo systemctl start jenkins
- **~$** sudo systemctl status jenkins
- **~$**sudo systemctl enable Jenkins

- **Jenkins has installed successfully on your instance.**

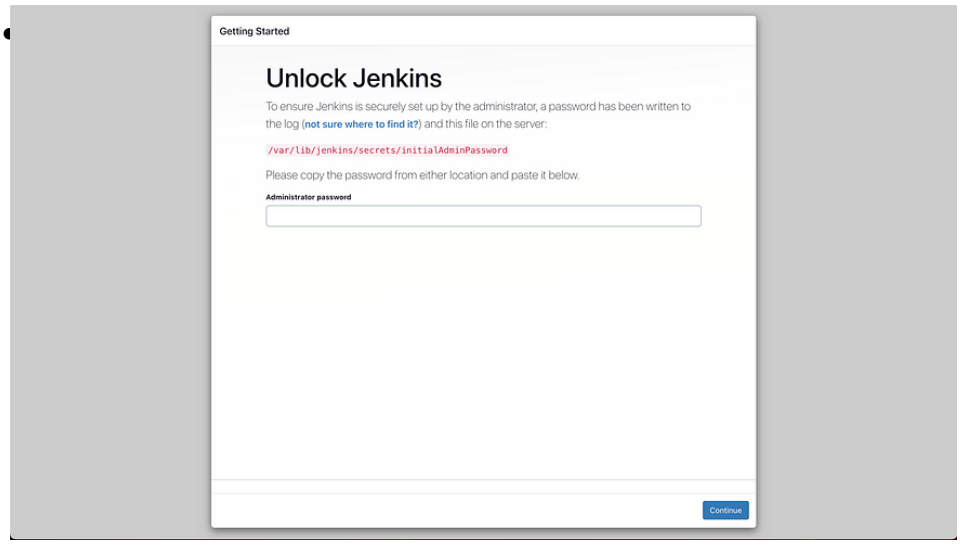


<u>Access Jenkins:</u>

Now, open your web browser and access Jenkins by navigating to:

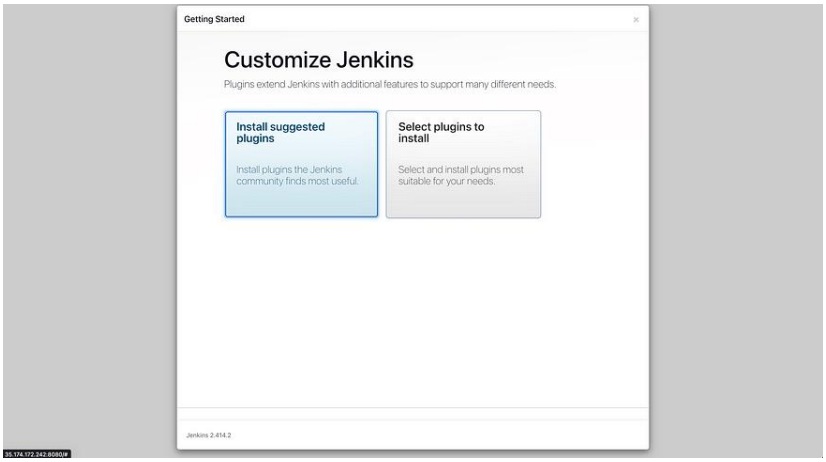
https://public-ip:8080

You will see a setup wizard and be prompted to enter the administrator password.

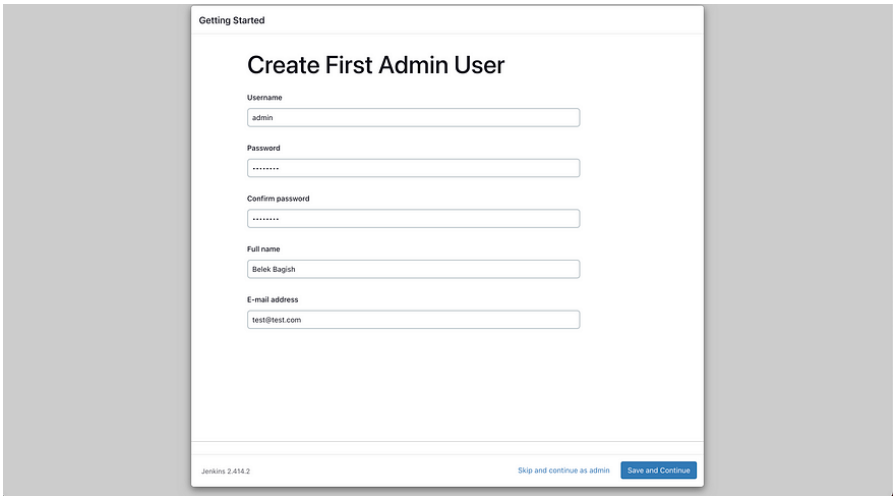


- Retrieve the password with the following command:
- $ sudo cat /var/Jenkins/secrets/intialAdminPassword

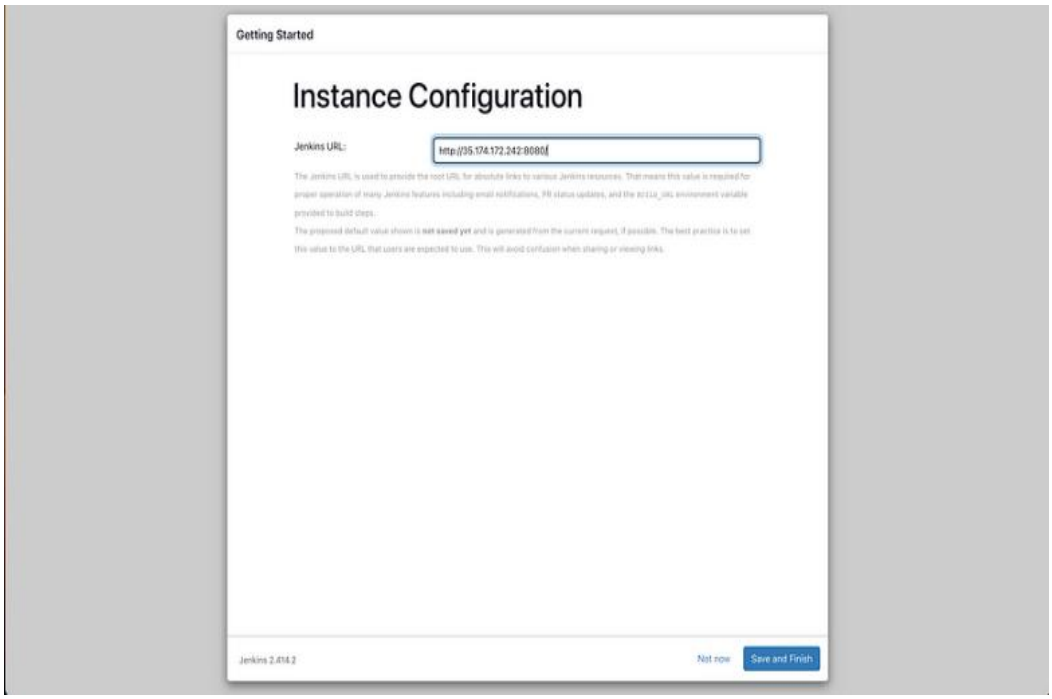
  - Copy the password and paste it into the Jenkins setup wizard to continue.
  - Complete the Setup Wizard:
  - Now for the fun part! After accessing Jenkins in your web browser, you'll be greeted by the Jenkins setup wizard. Let's walk through it together:
  - Install Plugins



- You'll be prompted to install plugins. You can opt for installing suggested plugins or select plugins to install based on your preference. The suggested plugins cover most general use cases.
- Create the First Admin User



- Next, you'll set up the first admin user. Fill in the username, password, name, and email fields as prompted.
- Instance Configuration



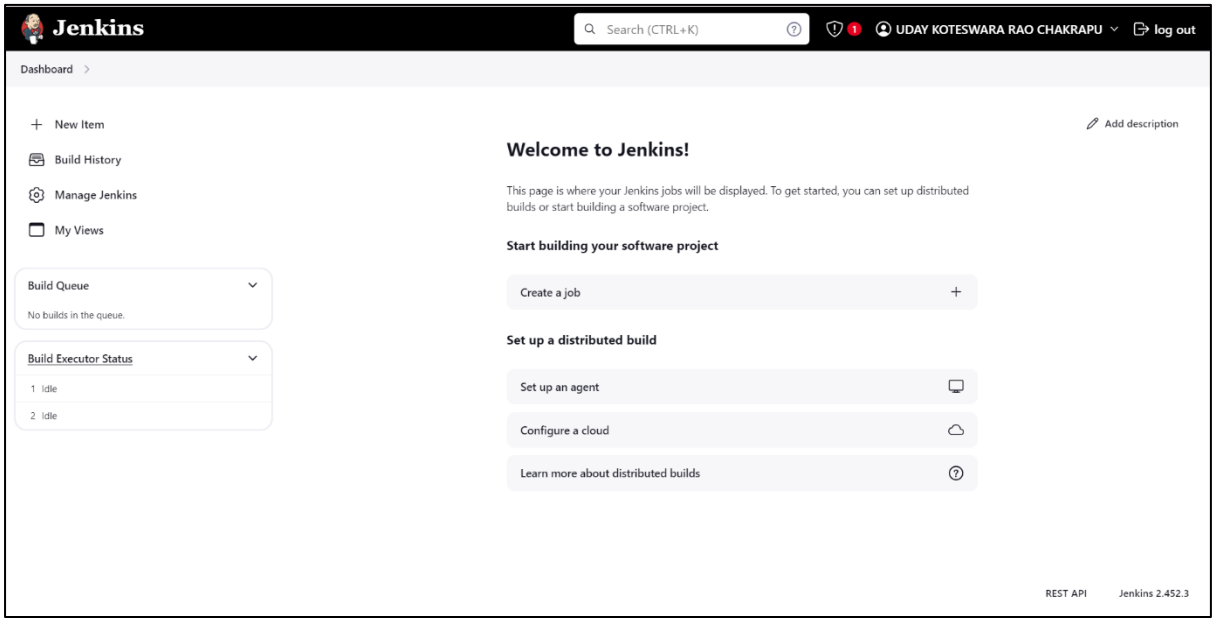
Finish the Setup

- click "Save and Finish." you've set up Jenkins!
- Start Using Jenkins

- Click on “Start using Jenkins” to be directed to the Jenkins dashboard, where you can begin creating your first Jenkins project.

5. Demonstrate Continous Integration and Continous Development using Jenkins.

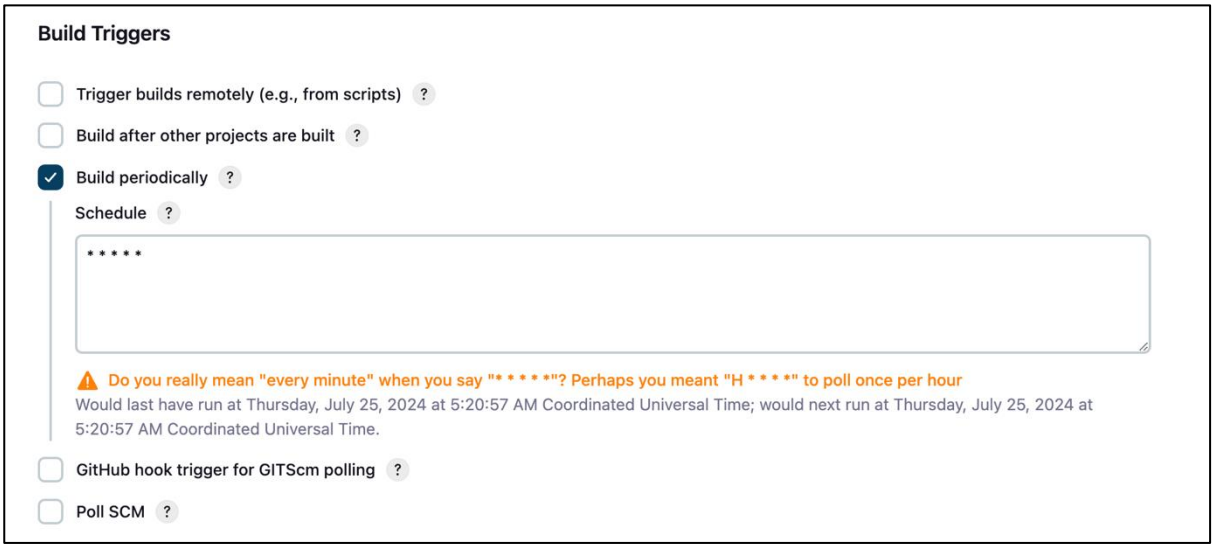
- The continuous Integration and continuous Development signifies the automatic build process of the software whenever the main branch is merged or a new commit is made to it.
- The following are the steps to setup the seamless flow of the CI/CD pipeline using github and Jenkins.
- Here we are going to use a sample web-app repo which is just a hello world program using jsp(Java server pages).
- After installing the Jenkins successfully , Login to your account and click on new item and enter a name of your desire and Select project style as free style project.



- Then scroll down to source code management and paste the url of your git repository in it.
- **Note:** Add credentials of your github if your git repo is a private repo to eradicate any build conflicts or Unauthorized errors.

- Now scroll down to Build Triggers and select Build periodically to set a schedule for build.

- Set the schedule vaue to * * * * *



- In the above image the * * * * * represents the following.
- **MINUTE**(0-59) **HOUR**(0-23) **DOM**(1-31) **MONTH**(1-12) **DOW**(0-7)

- Now click on add build step and select invoke top level maven target since our project is in java and uses maven to build.

- And then select maven version of your installation and mention goals as **clean install** and save the project.

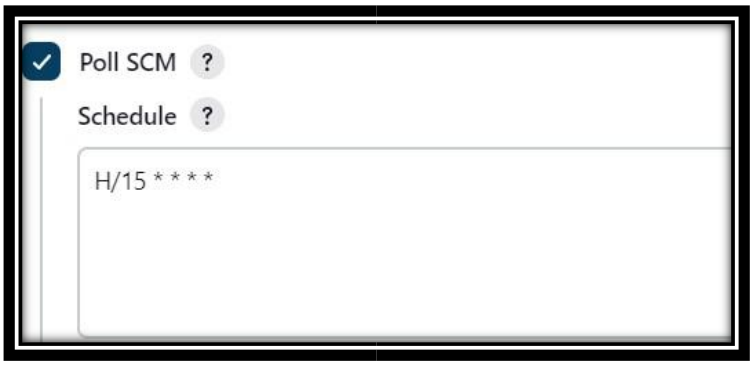
- Here the builds will be triggered at every minute of time and executes the **mvn clean install.**

- Now this will trigger the build irrespective of new commits. But if we need to trigger builds whenever there is a new commit and to achieve this we need to check whether there are new commits in it.

- Above can be achieved using the Poll SCM in build triggers.

- Click one build triggers and choose Poll SCM

- It is similar to build periodically but it will poll our git hub repository whenever there is a commit in repository it detect the change in that period or schedule.

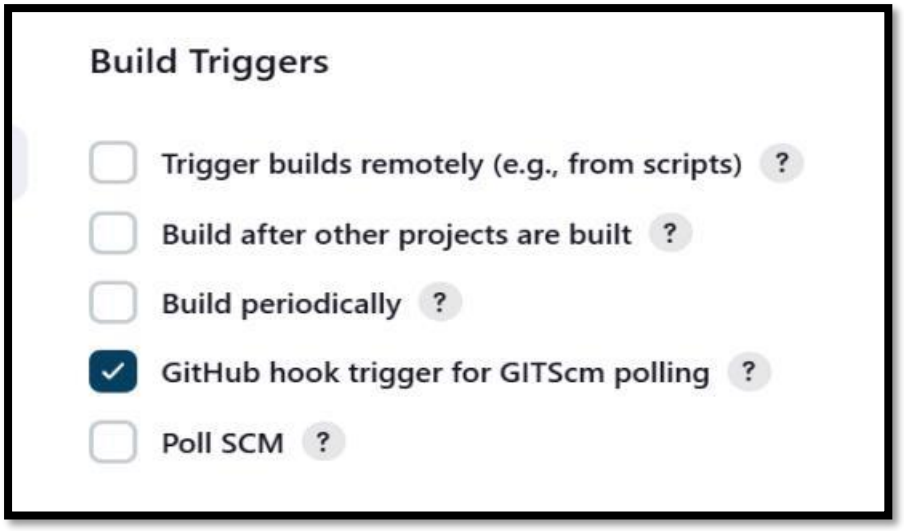


- **Save it**
- The above build trigger will commit at every 15min if any change occurs
- Make changes in any file amd convert it again in git bash

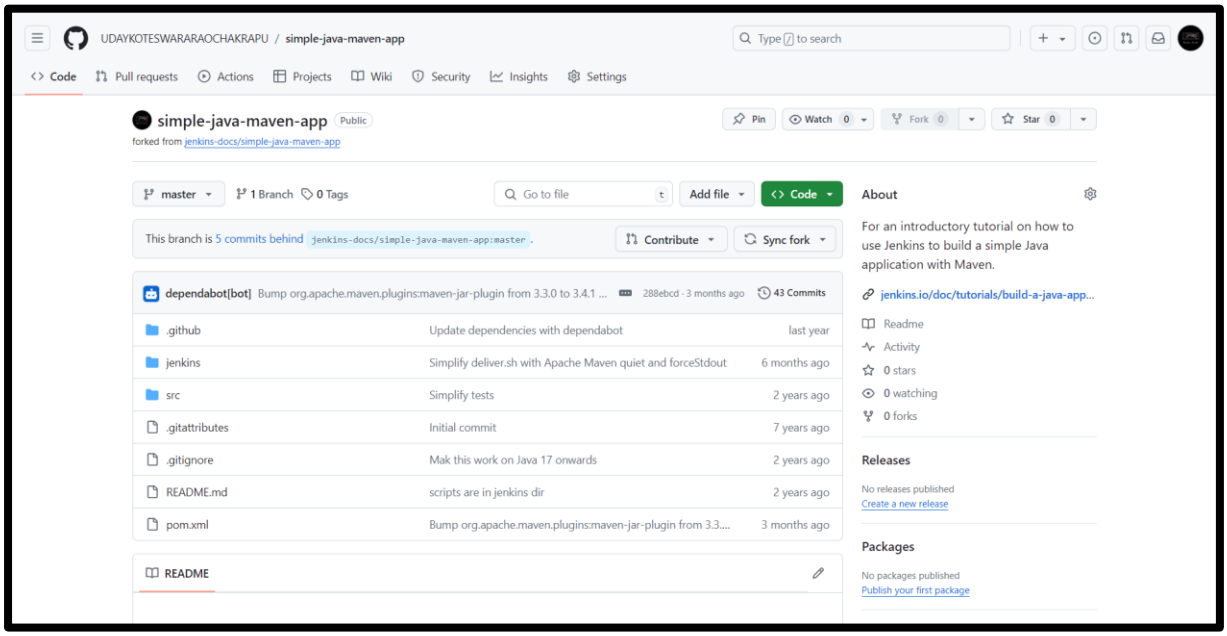
- Git commit -m “update”
- Git push
- Now build generate in Jenkins after 15min

WebHook:

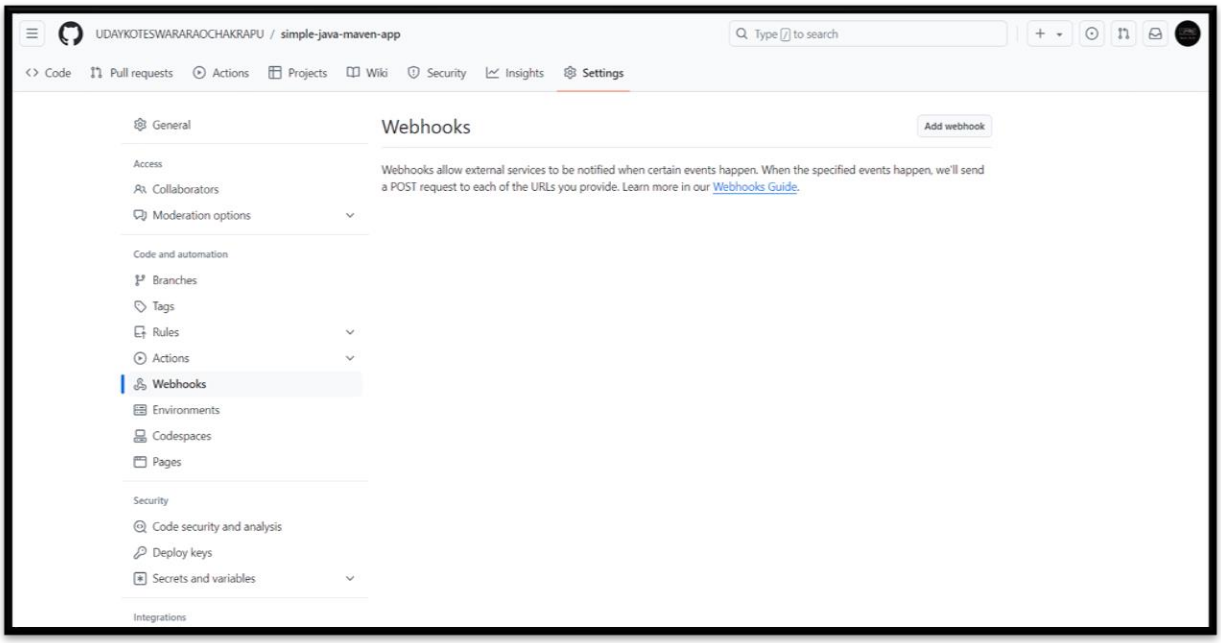
- In build triggers select “GitHub hook trigger for Gitscm polling?” .
- Click on “Apply”.
- Click on “save”.



- Open git hub accounrt and login to it.
- Choose “java -maven” repository
-



- Open setting and click on webhooks



- Paste the Jenkins URL in payload URL.
- In next step we must choose everything.
- This will involve in each, and every event would you like to trigger this webhook.
- Select the active radio button to active the webhook
- After this, select the button called “add webhook”.

Tomcat Installation in AWS EC2 with Amazon Linux as OS.

Step 1 :

- Open AWS account, Launch an instance
- Create instance name as QA-ENV
- Select amazon OS with T2 micro
- Select pem file and launch instance

Step 2 :

- Start the QA-ENV instance and set the port number
- By selecting **QA-ENV** open **SECURITY**
- Select **SECURITY GROUPS**
- Select **EDIT INBOUND RULE**
- Add a new rule by adding 8080 port number to it and save it

Step 3 : Connect to your EC2 Instance. Either using integrated terminal or SSH with your previously assigned pem file and ip address.

Steps to install and configure Tomcat

- Install Java on Machine
- Install Tomcat on Machine
- Add Execute Permission to startup.sh & shutdown.sh
- Create link files for Tomcat Server up and Down
- Change Settings to Manage Tomcat
- Update user information in tomcat-users.xml

Install Java

```
yum install java-1.8*
```

Install Tomcat

```
sudo su -

cd /

cd /opt

#Download tomcat binary
wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.55/bin/apache-
tomcat-9.0.55.tar.gz

#unzip tomcat binary
tar -zvxf apache-tomcat-9.0.55.tar.gz
```

Add Execute Permission to startup.sh & shutdown.sh

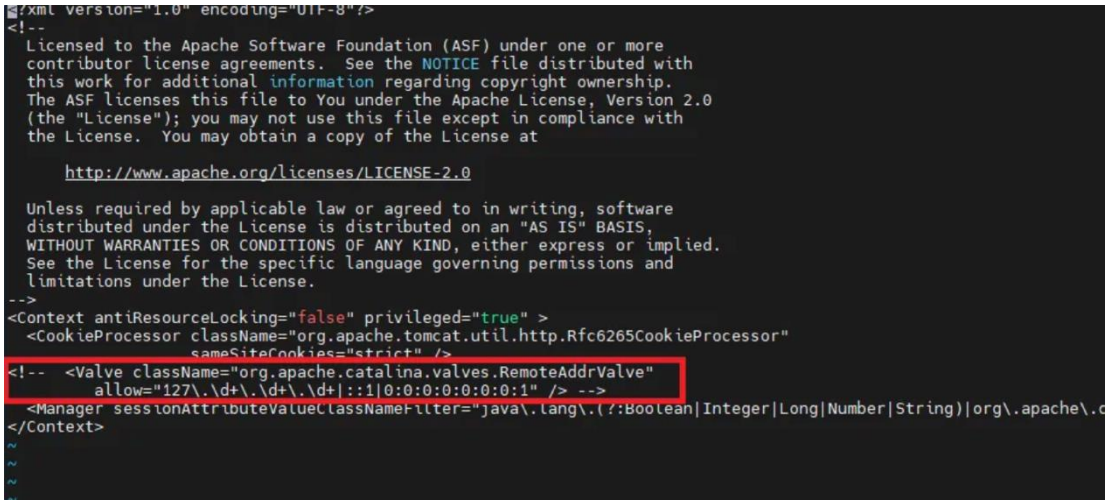
```
cd apache-tomcat-9.0.55

cd bin

chmod +x startup.sh
chmod +x shutdown.sh
```

Create link files for Tomcat Server up and Down

```
ln -s /opt/apache-tomcat-9.0.55/bin/startup.sh
/usr/local/bin/tomcatup

ln -s /opt/apache-tomcat-9.0.55/bin/shutdown.sh
/usr/local/bin/tomcatdown

tomcatup
```

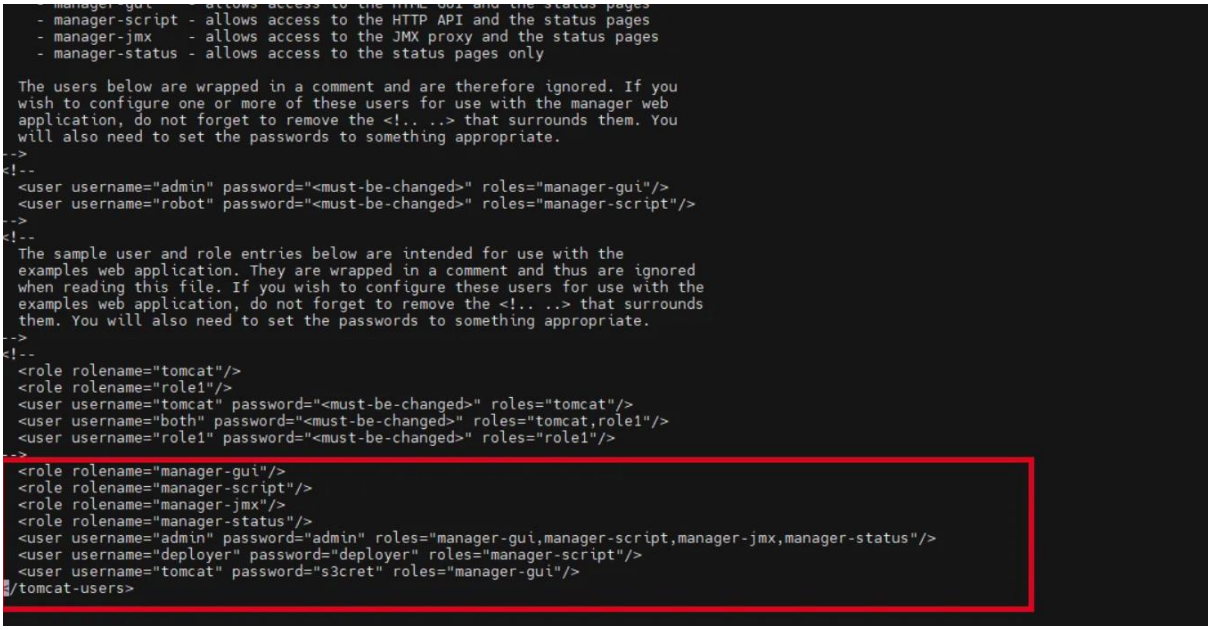
Change Settings to Manage Tomcat

```
cd apache-tomcat-9.0.55

find -name context.xml

./conf/context.xml
./webapps/examples/META-INF/context.xml
./webapps/host-manager/META-INF/context.xml
./webapps/manager/META-INF/context.xml

#comment value tag sections in below all files

vi ./webapps/examples/META-INF/context.xml
vi ./webapps/host-manager/META-INF/context.xml
vi ./webapps/manager/META-INF/context.xml
```



Update user information in tomcat-users.xml

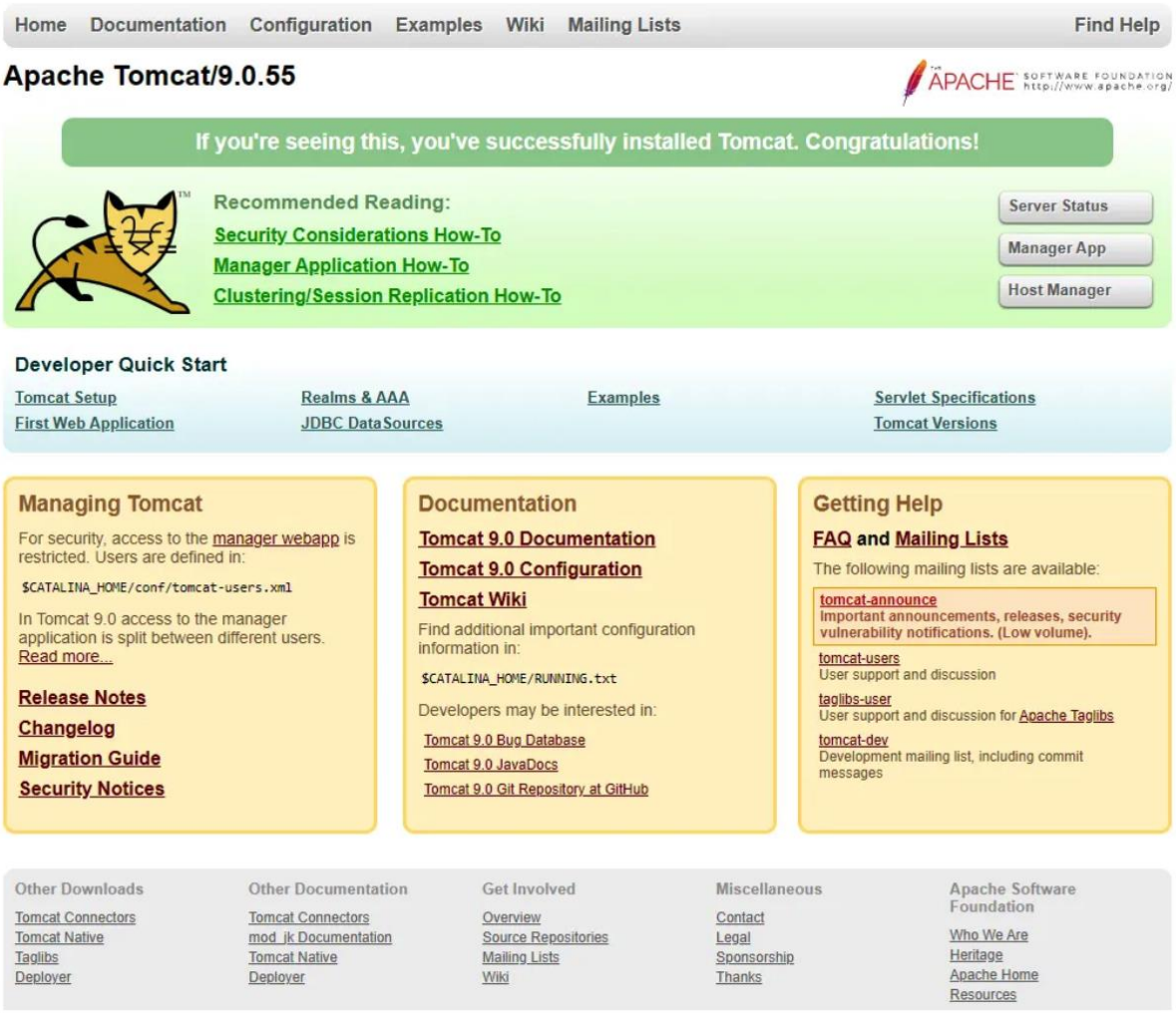
```
cd apache-tomcat-9.0.55
cd conf

vi tomcat-users.xml

#Add below lines between <tomcat-users> tag

<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="admin" password="admin" roles="manager-gui,manager-
script,manager-jmx,manager-status"/>
<user username="deployer" password="deployer" roles="manager-
script"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>
```



Access Tomcat Web Interface

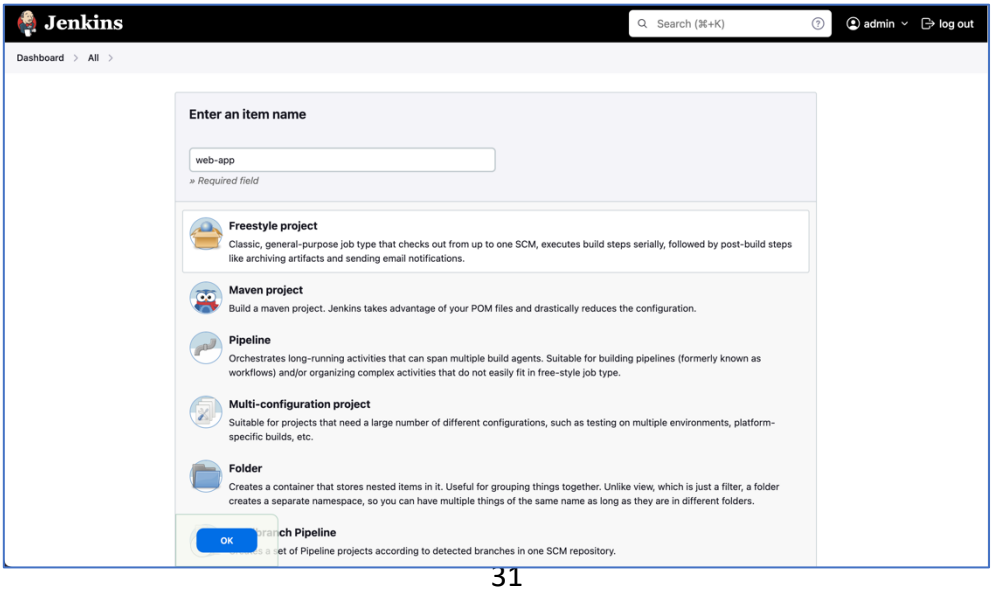
```
http://server_ip:8080/

You should see Tomcat welcome page.
```

- Replace server_ip with your actual EC2 public IP address.
- E.g. http://13.255.67.89:8080

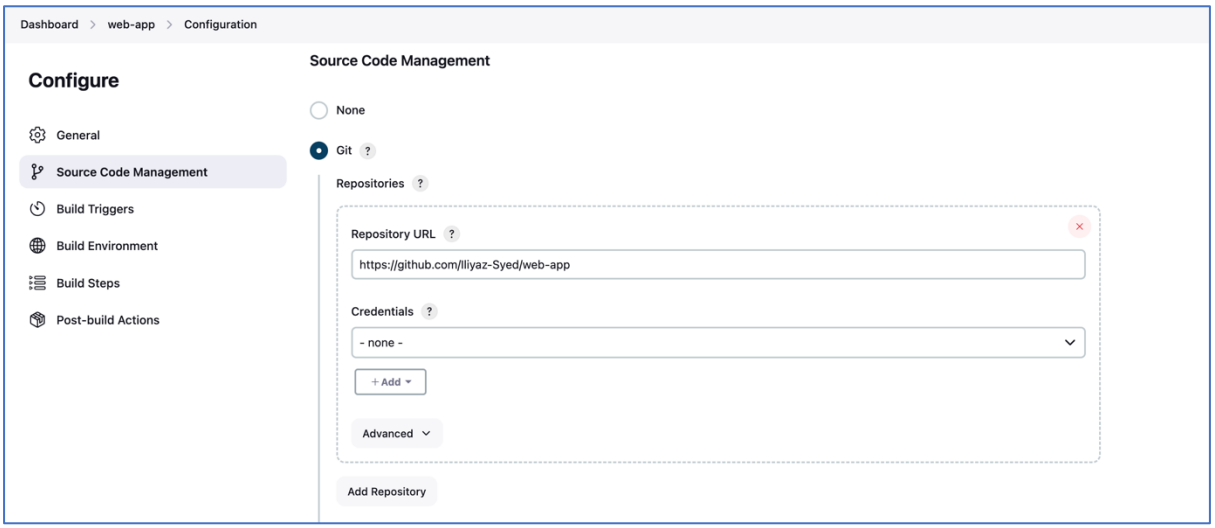


Production Environment:

- Open aws account and coonect Jenkins account
- Open Jenkins interface by using public ip
- Crerate a new project as “web-apps” by choosing free style project. Save it .

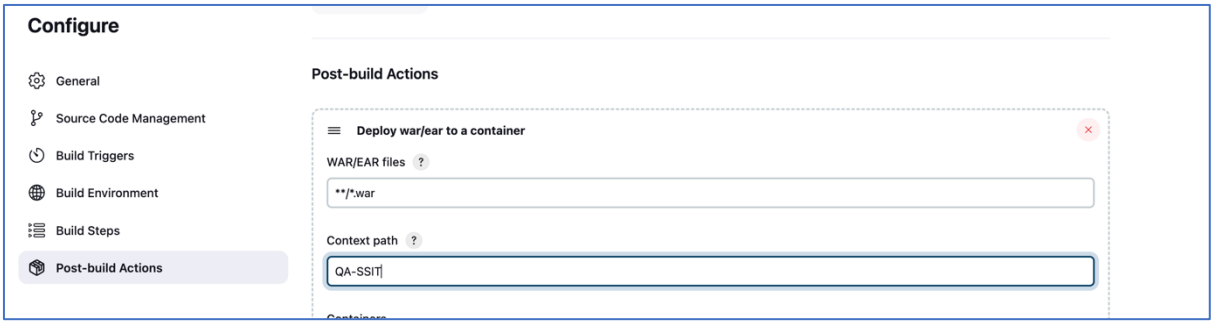


- Choose web apps and open it
- Select configure and open source code management
- Add git repository

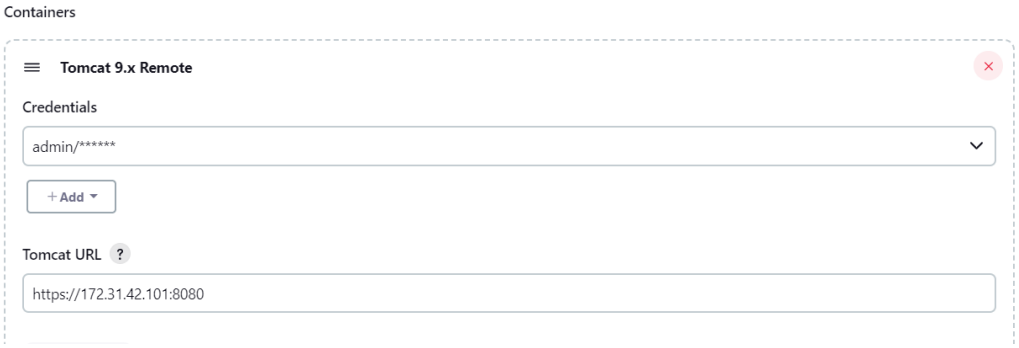


- For **Build Steps:** Select Invoke Top Level Maven targets and write goal as package.

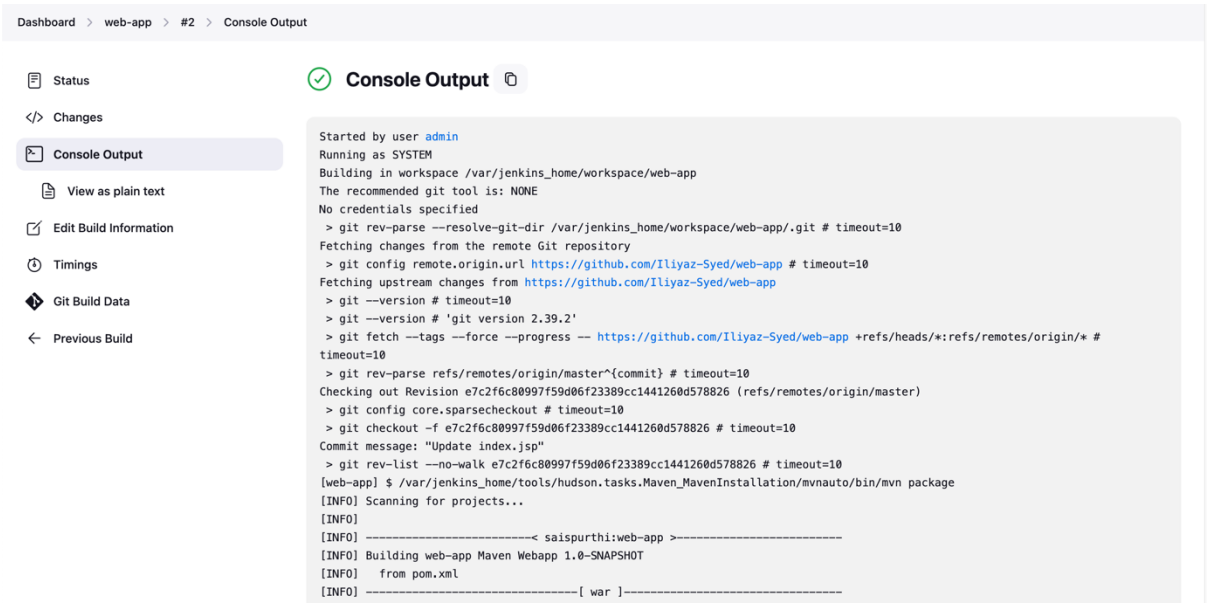
- Add **POST Build Steps :** Select deploy war/ear to container and Add **/*.war to it.



- Add Container



- Save it and build it.



- Login to github and open **web-apps** repository
- Open .src/main/web-apps click on it
- Open index.jsp file and update it with some code

- Now open aws instance and copy the public ip of tomcat instance
- Paste it in chrome new tab with 8080 port number
- By opening tomcat server select manager app and open it
- Now select QA-SSIT and check it

Tomcat Web Application Manager

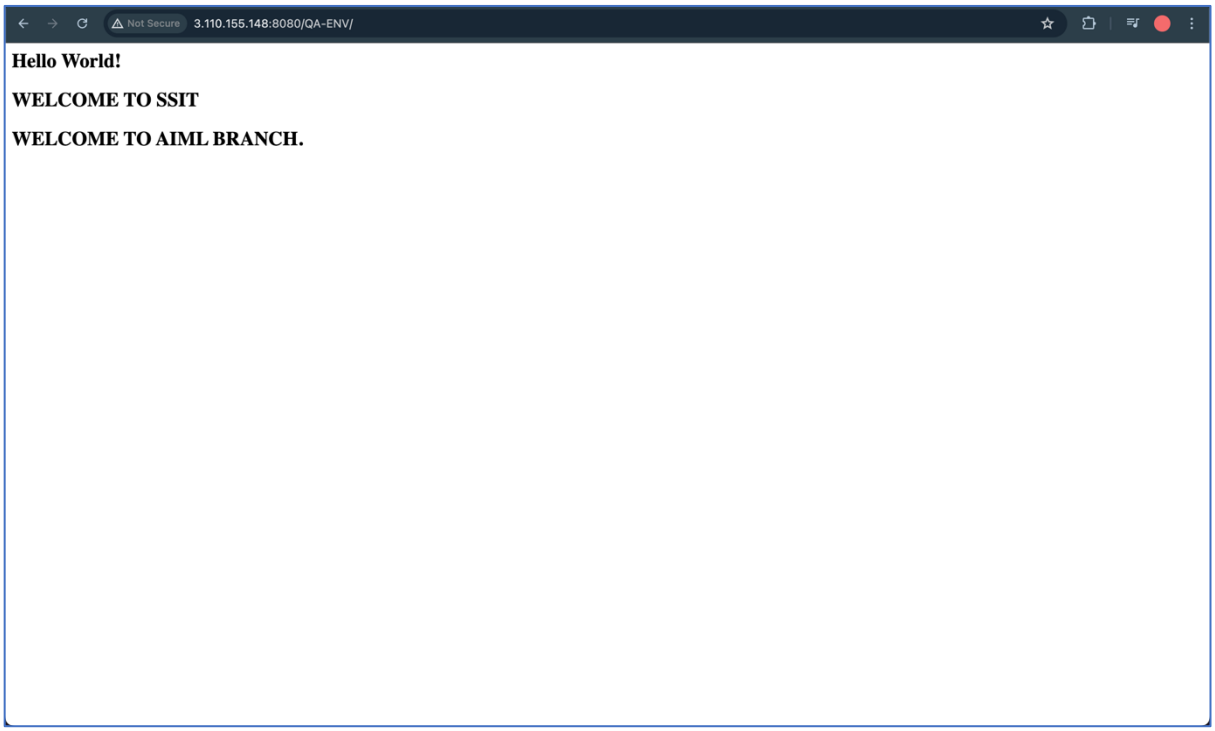
Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	*None specified*	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/QA-ENV	*None specified*	Archetype Created Web Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/QA-SSIT	*None specified*	Archetype Created Web Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	*None specified*	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	*None specified*	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	*None specified*	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	*None specified*	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/qaenv	*None specified*	Archetype Created Web Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

- The OutPut should be Like this when opened.

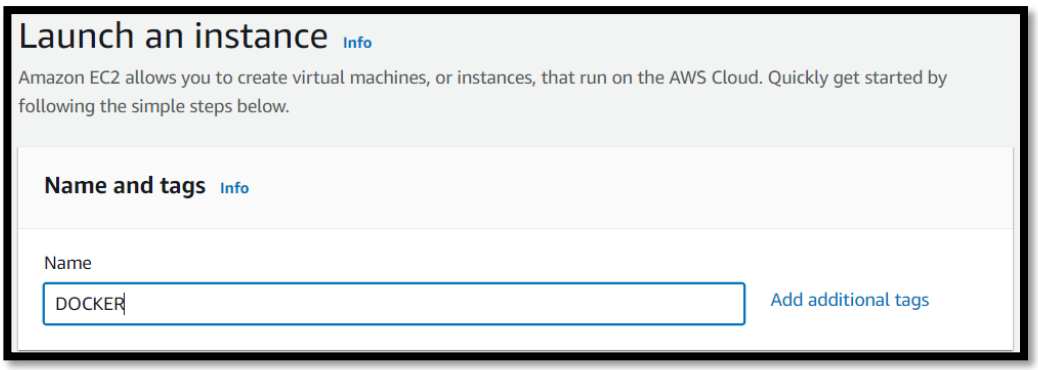


6.Explore Docker commands for content management.

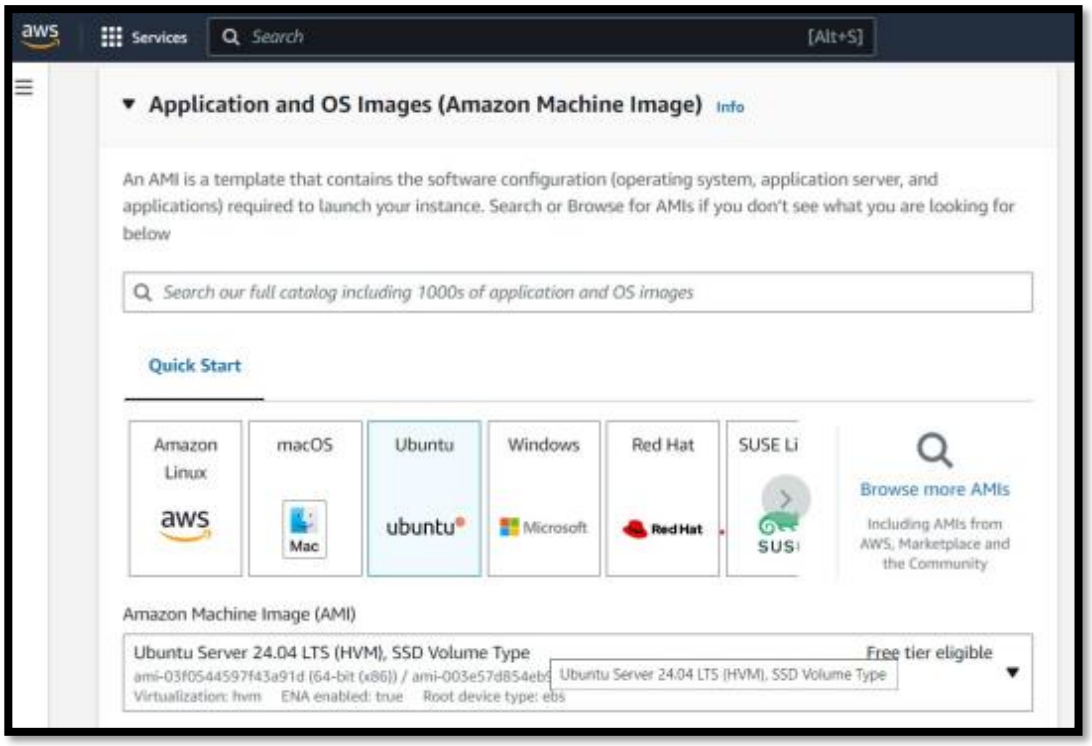
Docker : Docker is an open-source platform that allows you to package and run an application in a loosely isolated environment called containers. This security and isolation enable you to execute many containers simultaneously on a given host.

Steps to install DOCKER in aws :

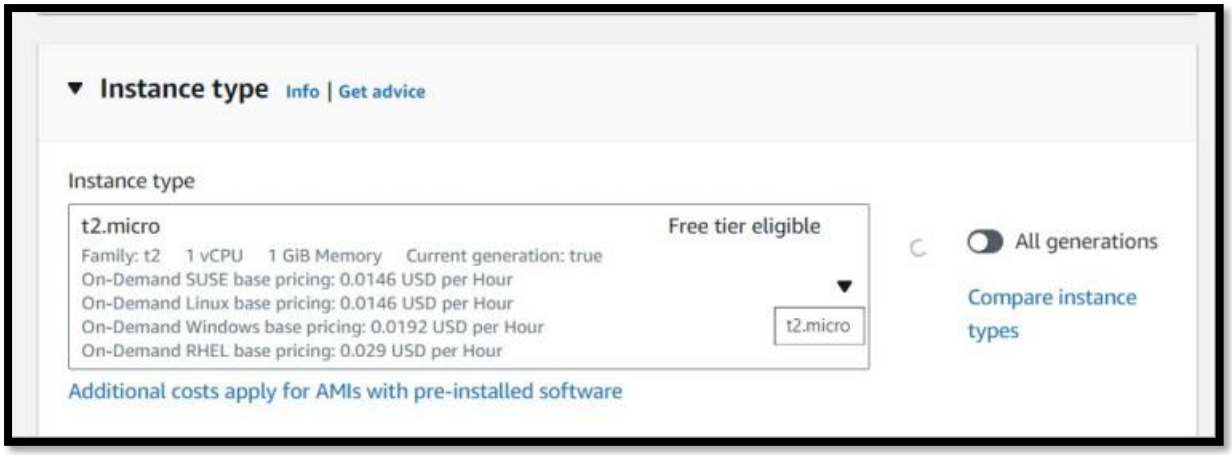
- Login to AWS account with your email and password.
- Open AWS home page and click on EC2.
- Opens EC2 Dashboard,now launch instance.
- Click on Launch an instance
- Name it as **DOCKER**



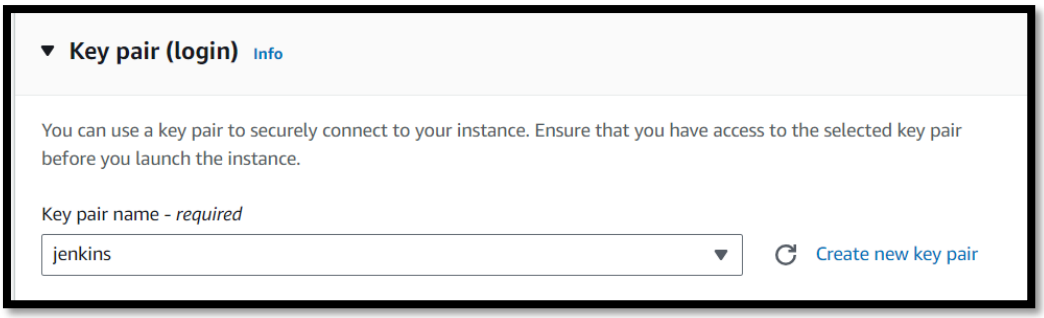
- Select application and OS images and Choose “Ubuntu” as operating system for running your Docker Containers.



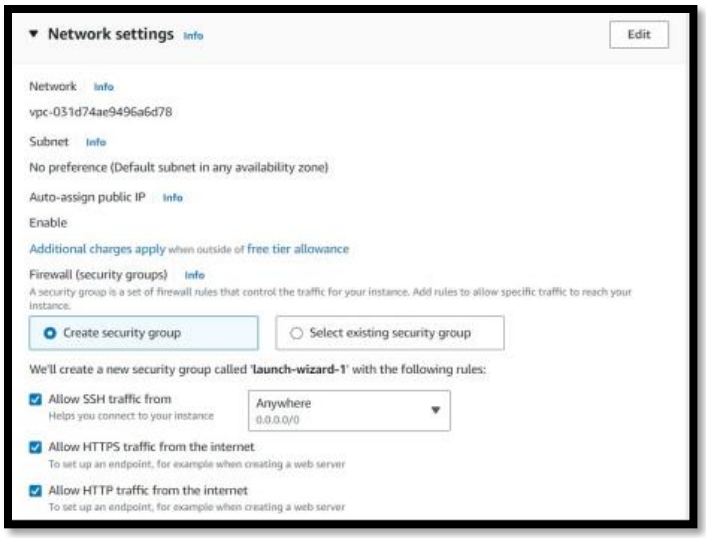
- After selecting the OS , Now select the Instance type as t3 Micro.



- Select a key pair for loggin in using SSH or Generate one If not.

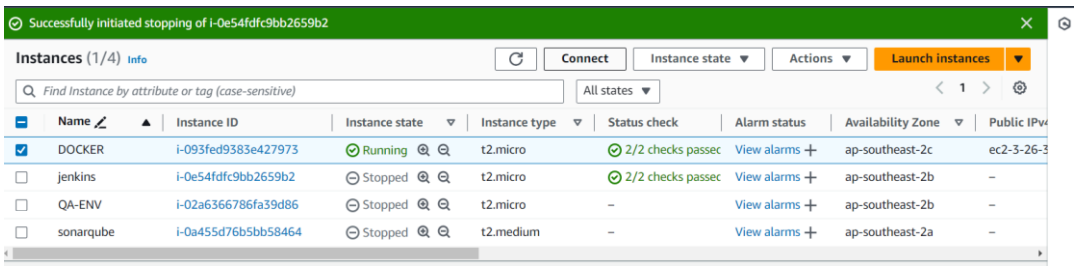


- In Network Settings Section , Use Existing security Group and Allow traffic from all protocols for accessing of web pages hosted.

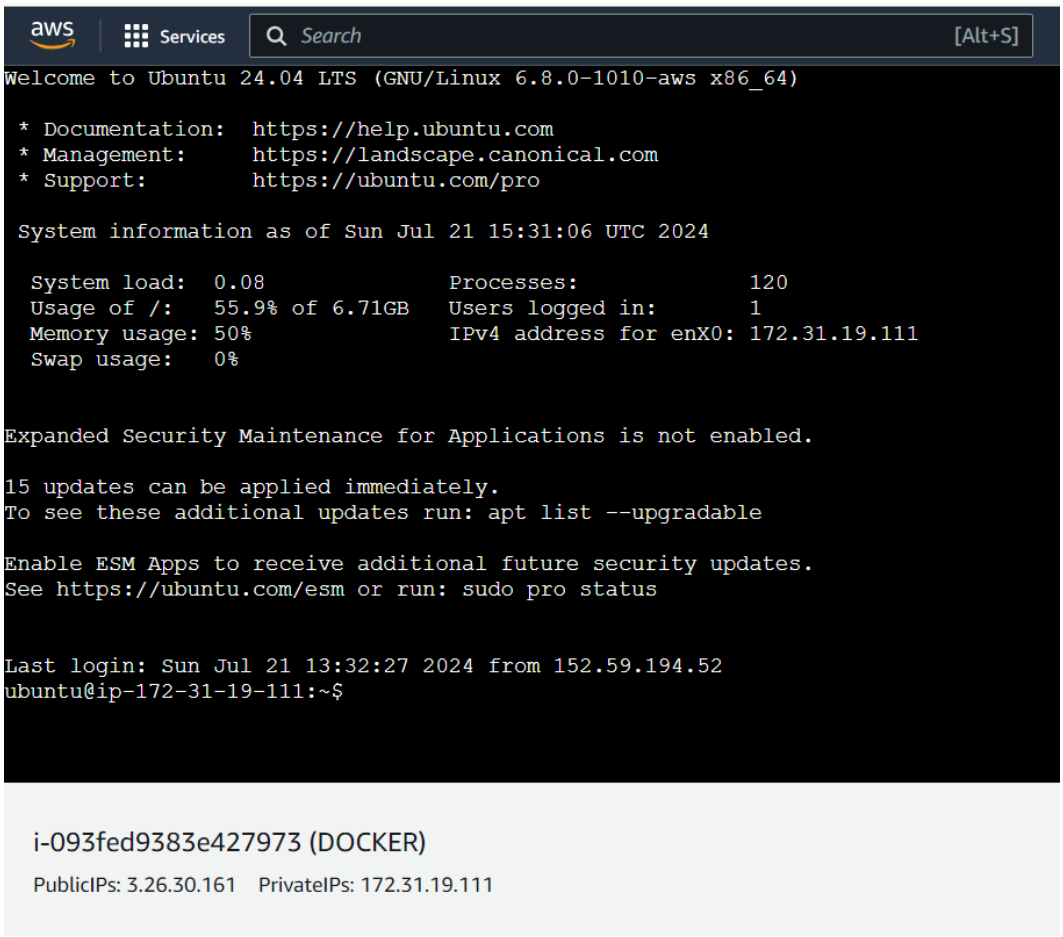


- Now with all the updated and modified Configurations Launch the instance

- Now connect to the **Docker** Instance that we have created just now via browser or SSH.

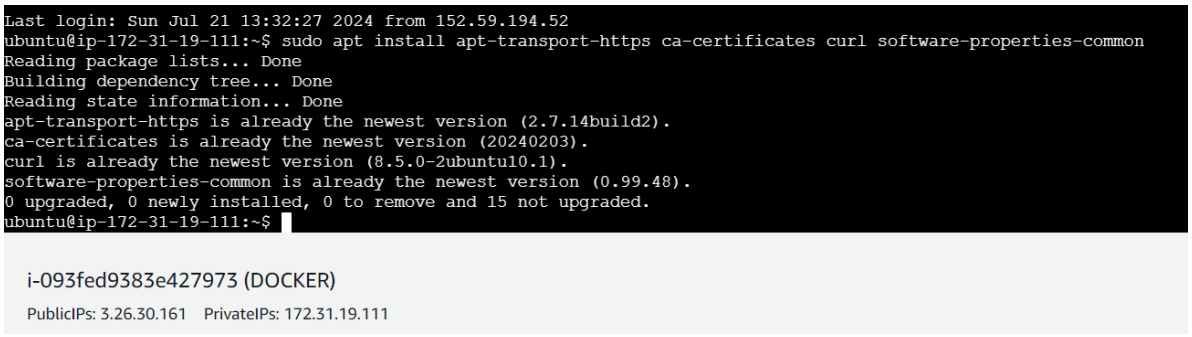


- On Successful connection a terminal will be available for us to access the Instance.



Step 1: Set-up prerequisite packages

~$ **sudo apt install apt-transport-https ca-certificates curl software-properties-common**



Step 2: Download the official Docker repository GPG key and add it to the apt-key trusted key manager:

~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

Step 3: And add the repository of Docker stable version for Ubuntu 20.04 (focal) to APT sources list:

~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

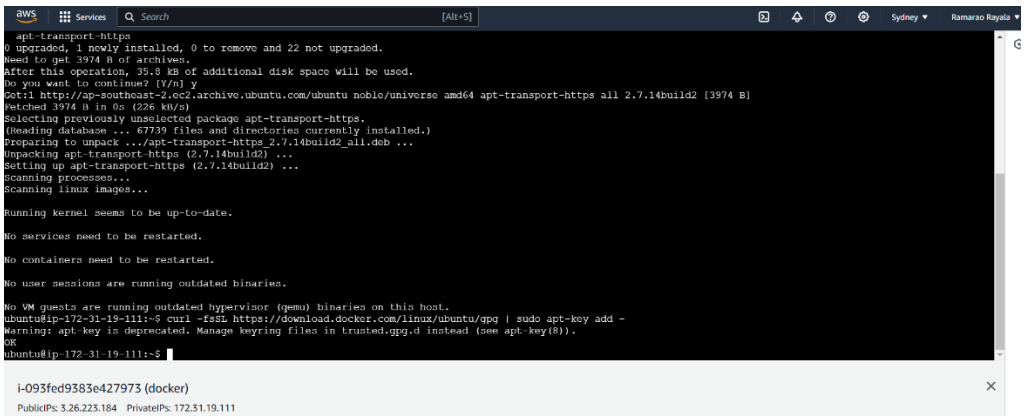
Step 4: Check the installation source priority for decker-ce

~$ apt-cache policy docker-ce

```
docker@t3st:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.8~3-0~ubuntu-focal
  Version table:
     5:20.10.8~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.7~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.6~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
```

step 5 : Install docker

~$ sudo apt install docker-ce



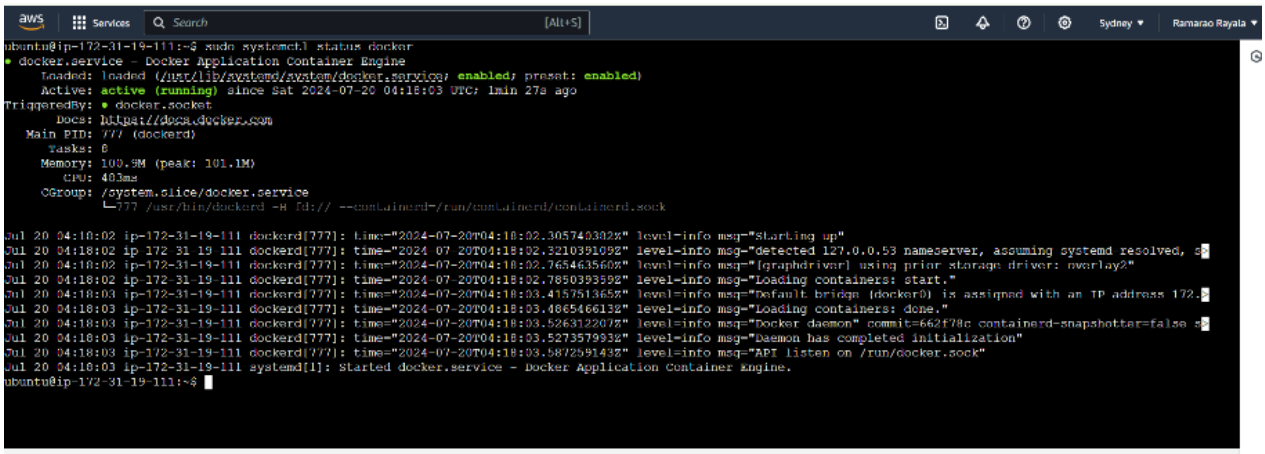
Step 6: Start the docker

~$ sudo systemctl start docker

```
ubuntu@ip-172-31-19-111:~$ sudo systemctl start docker
ubuntu@ip-172-31-19-111:~$
```

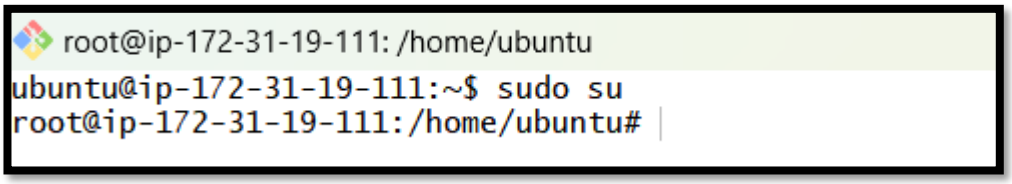
Step 7: Check the status of Docker

~$ sudo systemctl status docker

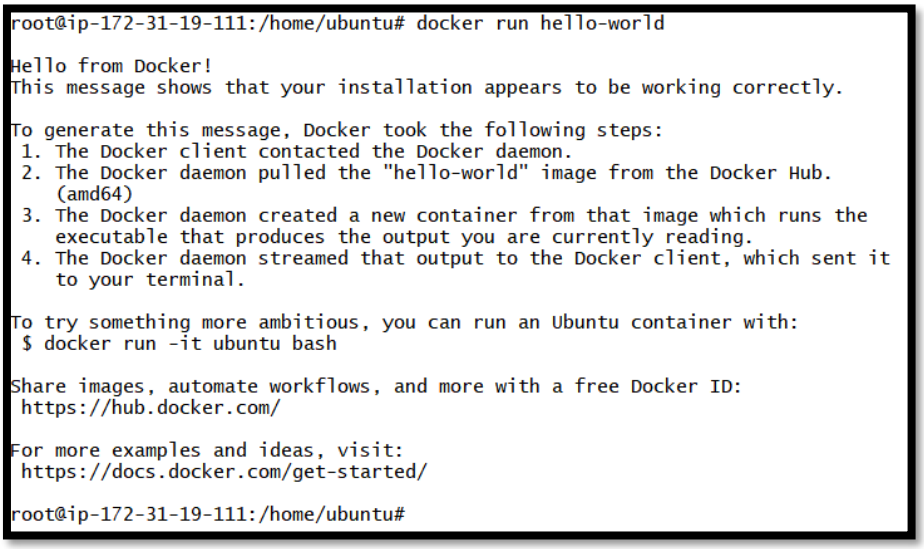


❖ Start using Docker

- Change the path from ubuntu to root user : **$ sudo su -**



- To download a docker image: **~$ docker run hello-world**



- To get list of docker images : **~$ docker ps**

```
root@ip-172-31-19-111:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-19-111:/home/ubuntu#
```

- To get list of all containers regardless of states : **~$ docker ps -a**

```
root@ip-172-31-19-111:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED        STATUS                     PORTS                                       NAMES
0912256e2f53   jenkins:1.651.3   "/bin/tini -- /usr/l…"   3 hours ago    Up 3 hours                 50000/tcp, 0.0.0.0:8081->8080/tcp, :::8081->8080/tcp   amazing_jones
f9fd62620921   hello-world       "/hello"                 3 hours ago    Exited (0) 3 hours ago                                                 hungry_mayer
ee76797e8809   myfirst:0.3       "echo echo google.com"   31 hours ago   Exited (0) 31 hours ago                                                cool_chatterjee
eb45369d506c   myfirst:0.3       "echo echo google.com"   33 hours ago   Exited (0) 33 hours ago                                                elegant_swanson
1021457cf4f8   myfirst:0.2       "echo google.com"        34 hours ago   Exited (0) 34 hours ago                                                mystifying_kapitsa
a87cd9ce97dc   myfirst:0.1       "echo 'hello world w…"   34 hours ago   Exited (0) 34 hours ago                                                agitated_antonelli
6f7b8a90fcb7   jenkins:1.651.3   "/bin/tini -- /usr/l…"   35 hours ago   Created                                                                condescending_villani
7470cb06c3f1   jenkins:1.651.3   "/bin/tini -- /usr/l…"   35 hours ago   Created                                                                gracious_jepsen
1c6dc4d12ae2   jenkins:1.651.3   "/bin/tini -- /usr/l…"   35 hours ago   Exited (143) 33 hours ago                                              serene_mccarthy
5b3f01dc0c50   hello-world       "/hello"                 35 hours ago   Exited (0) 35 hours ago                                                eloquent_williamson
root@ip-172-31-19-111:/home/ubuntu#
```

- To get list of IDs of all containers : **~$ docker ps -a -q**
- The IDs can be used for the following
    - To Start/Stop a particular container.
    - To get logs of a particular container.
    - To even tail(Follow) the logsof the container And
    - To perform various commands and operations on a container wherever possible.

```
root@ip-172-31-19-111:/home/ubuntu# docker ps -a -q
f9fd62620921
ee76797e8809
eb45369d506c
1021457cf4f8
a87cd9ce97dc
6f7b8a90fcb7
7470cb06c3f1
1c6dc4d12ae2
5b3f01dc0c50
root@ip-172-31-19-111:/home/ubuntu#
```

- To dowmload images from internet : **~$ docker pull nginx:latest**

```
root@ip-172-31-19-111:/home/ubuntu# docker pull nginx:latest
latest: Pulling from library/nginx
Digest: sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95df
Status: Image is up to date for nginx:latest
docker.io/library/nginx:latest
root@ip-172-31-19-111:/home/ubuntu#
```

- To see list of images in docker **: ~$ docker images**

```
root@ip-172-31-19-111:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-19-111:/home/ubuntu# docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
myfirst       0.1       29f370539dfd   31 hours ago     223MB
myfirst       0.3       51becbefa614   31 hours ago     223MB
myfirst       0.2       83a00e46a2a1   31 hours ago     223MB
nginx         latest    fffffc90d343   4 weeks ago      188MB
hello-world   latest    d2c94e258dcb   14 months ago    13.3kB
jenkins       1.651.3   d5c0410b1b44   8 years ago      737MB
root@ip-172-31-19-111:/home/ubuntu#
```

- To delete a image in docker : **~$ docker rmi nginx**

```
root@ip-172-31-19-111:~# docker rmi nginx
Untagged: nginx:latest
Untagged: nginx@sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95df
Deleted: sha256:fffffc90d343cbcb01a5032edac86db5998c536cd0a366514121a45c6723765c
Deleted: sha256:9f4b5d44149fd139616539e0ef5311a14338a970f25733ba95b4ae6d3becdf0d
Deleted: sha256:8e6e10fbcca1bb180c5cfc805a37240945a6ba703cb1f985d4d3b8a1c954fc5b
Deleted: sha256:dcec89b921d64a1d2f748c450832a4c5624219bbb1b696e1a606bff78b7afa60
Deleted: sha256:4994a8d5939af297abf594b7ab714dfb72e57217b94bf0a250a62903cbdb6d84
Deleted: sha256:8b2bc37f672b15bea06a204790da9f384ef7b06feccade3fd3b1945b63df5aef
Deleted: sha256:a4197512070a01764db77b424100c1f81f0ed696380b19bc26b6e72fafef0709
Deleted: sha256:32148f9f6c5aadfa167ee7b146b9703c59307049d68b090c19db019fd15c5406
root@ip-172-31-19-111:~#
```

- To upload a docker image into docker hub : **~$ docker push Jenkins**

7. Develop a simple containerized application using Docker.

- Create a new directory : **~$ mkdir DockerImage**
- Move to directory : **~$ cd DockerImage**
- Crerate another directoty :**~$ mkdir MyFirstImage**
- Move to new directory **: ~cd MyFirstImage**

```
Last login: Sun Jul 21 16:31:05 2024 from 152.59.194.34
ubuntu@ip-172-31-19-111:~$ ls
DockerImages
ubuntu@ip-172-31-19-111:~$ cd DockerImages
ubuntu@ip-172-31-19-111:~/DockerImages$ cd MyFirstImage/
ubuntu@ip-172-31-19-111:~/DockerImages/MyFirstImage$ vi Dockerfile
ubuntu@ip-172-31-19-111:~/DockerImages/MyFirstImage$
```

- Edit the Dockerfile : **~$ vi Dockerfile**

```
ubuntu@ip-172-31-19-111: ~/DockerImages/MyFirstImage
FROM ubuntu:24.04
RUN apt update -y && apt install apache2 -y
ENTRYPOINT [ "echo" ]
CMD ["echo" ,"google.com"]
~
~
~
~
~
~
~
~
~
~
~
~
```

- To see the content of file : **~$ cat Dockerfile**

```
root@ip-172-31-19-111:/home/ubuntu/DockerImages/MyFirstImage# cat Dockerfile
FROM ubuntu:24.04
RUN apt update -y && apt install apache2 -y
ENTRYPOINT [ "echo" ]
CMD ["echo" ,"google.com"]
root@ip-172-31-19-111:/home/ubuntu/DockerImages/MyFirstImage#
```

- To build an image from container : **~$ docker build -t myfirst:0.1 .**

```
root@ip-172-31-19-111:/home/ubuntu/DockerImages/MyFirstImage# docker build -t myfirst:0.1 .
[+] Building 1.9s (6/6) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 148B
 => [internal] load metadata for docker.io/library/ubuntu:24.04
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/2] FROM docker.io/library/ubuntu:24.04@sha256:2e863c44b718727c860746568e1d54afd13b2fa71b160f5cd9058fc436217b30
 => CACHED [2/2] RUN apt update -y && apt install apache2 -y
 => exporting to image
 => => exporting layers
 => => writing image sha256:51becbefa614dad19a82c8492e6766d7df6c00683c4cc61789dbf64c39ee8653
 => => naming to docker.io/library/myfirst:0.1
root@ip-172-31-19-111:/home/ubuntu/DockerImages/MyFirstImage#
```

- To run the image : **~$ docker run myfirst:0.1**

```
ubuntu@ip-172-31-19-111:~/DockerImages/MyFirstImage$ sudo su
root@ip-172-31-19-111:/home/ubuntu/DockerImages/MyFirstImage# docker run myfirst:0.1
echo google.com
root@ip-172-31-19-111:/home/ubuntu/DockerImages/MyFirstImage#
```

8. Integrate Kubernetes and Docker.

Docker

Docker is an open-source platform that automates the deployment, scaling, and management of applications in lightweight containers. These containers package applications with their dependencies, ensuring consistency across environments, and improving development efficiency, scalability, and resource utilization.

Kubernetes

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications across clusters of hosts. Developed by Google, it provides a robust framework for running distributed systems, featuring automated rollouts, rollbacks, service discovery, load balancing, storage orchestration, and self-healing capabilities.

Installing Minikube on Windows

- **Install Docker**:
  - Download and install Docker Desktop for Windows from their Official Website.
- **Install Minikube**:
  - Download Minikube installer from their Offcial website.
  - Run the installer and follow the on-screen instructions to complete the installation.

Installing Minikube on Linux

- **Install Docker**:
  - Follow the steps from the previous experiment to install Docker.
- **Install Minikube**:

Step 1: Download the Minikube binary:

~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

Step 2 : Install Minikube

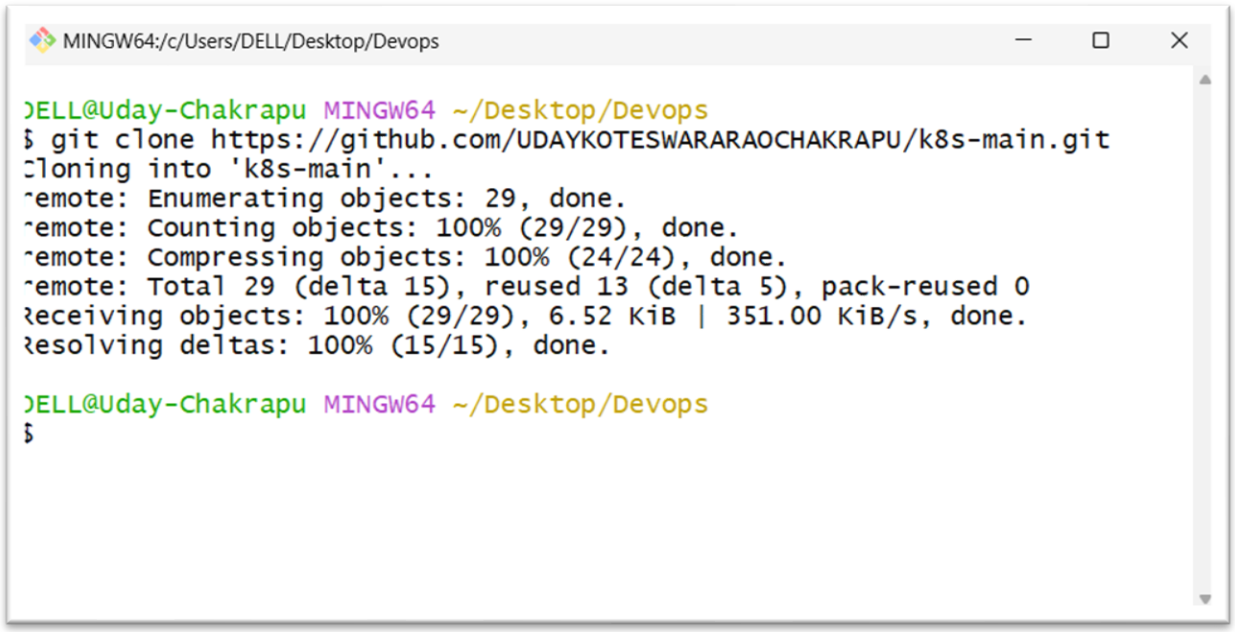
~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube

- These steps installs both Docker and minikube in your respective system and then allows you to use the Minikube to run your desired server.

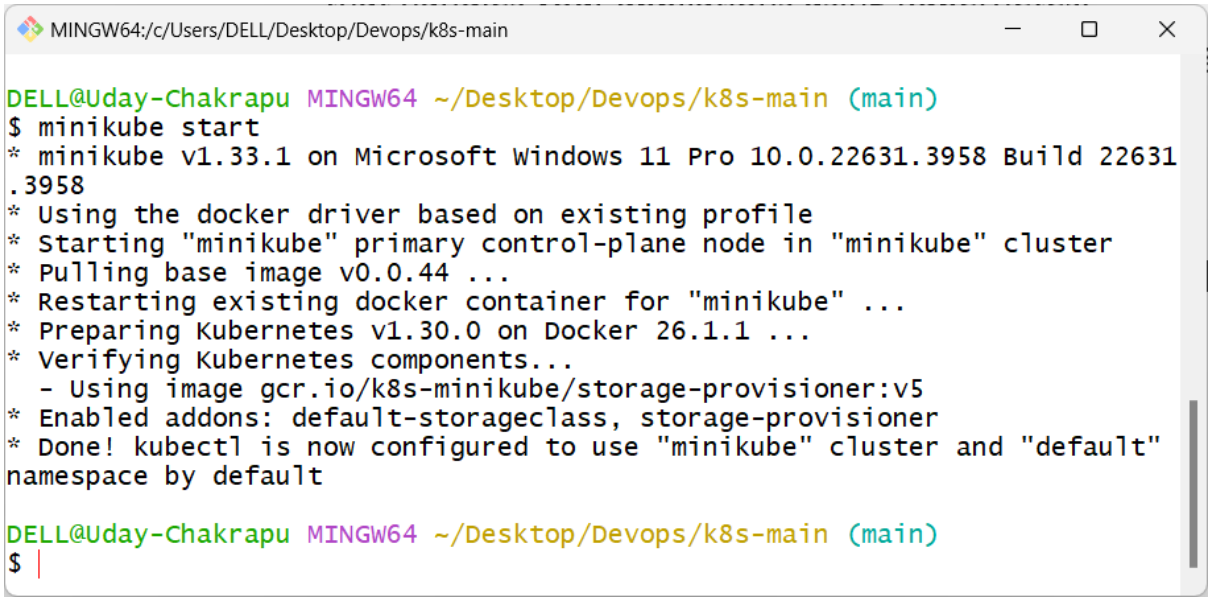
Running Flask through Kubernetes

- To Begin with Clone the repository that contains your flask application as shown below.

~$ git clone https://github.com/UDAYKOTESWARARAOCHAKRAPU/k8s-main.git

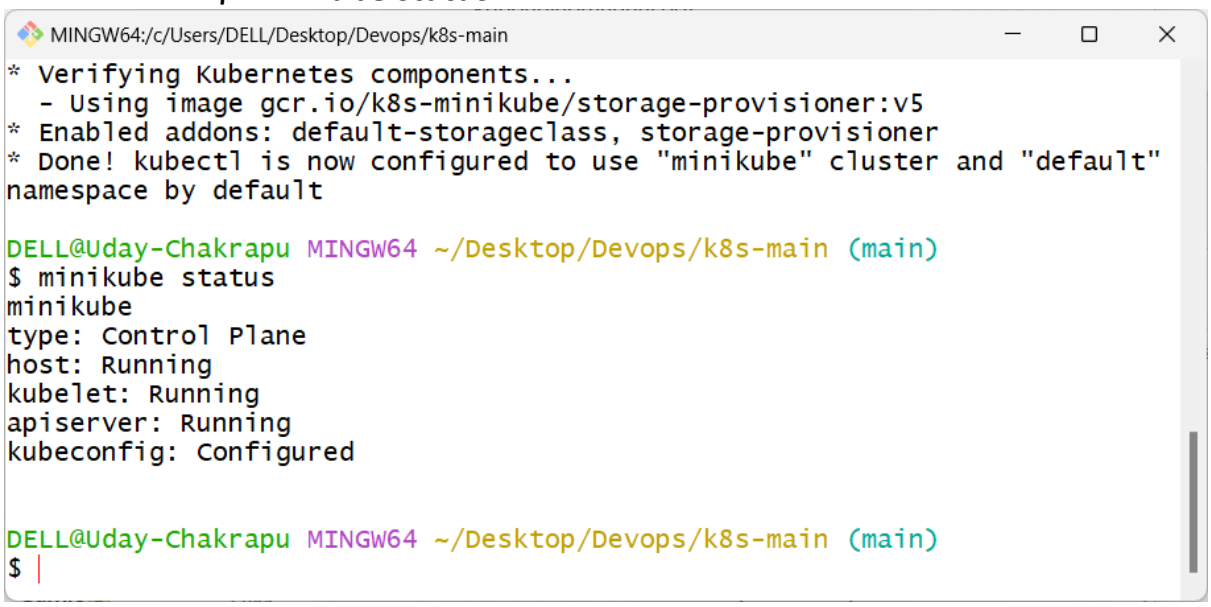


- Now go to its directory and run **minikube start** to start the minikube container that handles your appliations using Kubernetes.
- In order for Minikube to run, Make sure the docker engine is running.



- Now check the status of minikube using the following command.

  ~$ minikube status

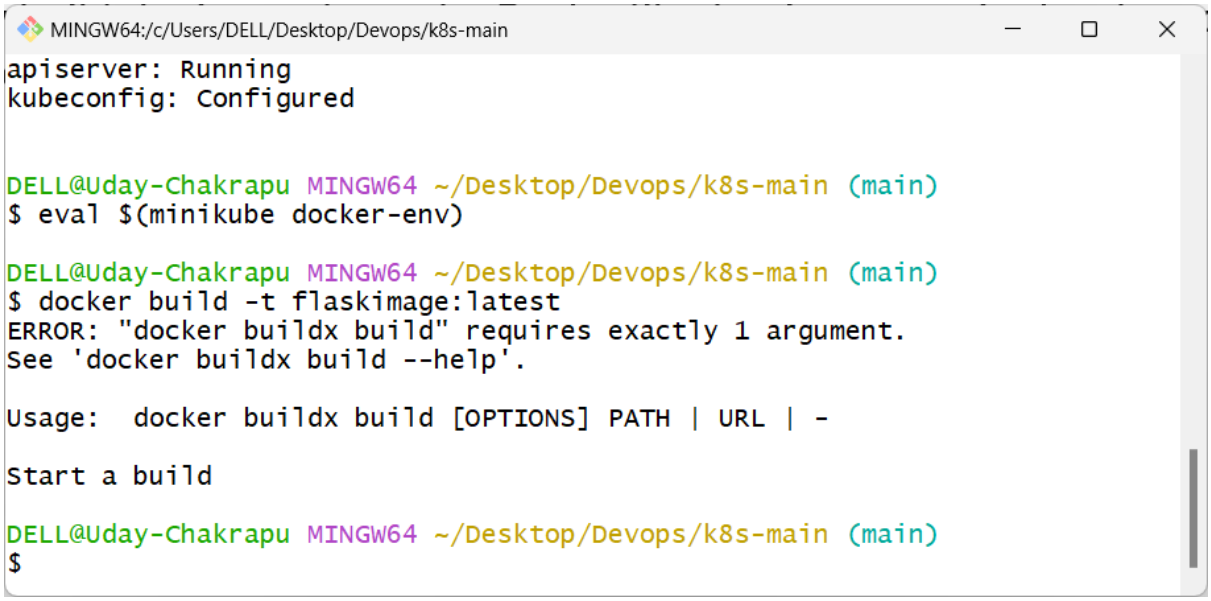


- Now the minikube is started successfully , Then activate the minikube environment using the following command.

  ~$ eval $(minikube docker-env)

- Now build the image from the Dockerfile that is present in the cloned repository in order to serve it using Kubernetes.

  ~$ docker build -t flaskimage:latest .



- Now Apply the Kubernetes deployment configuration using the following command.

  ~$ kubectl apply -f flask-app-deployment.yaml

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ kubectl apply -f flask-app-deployment.yaml
deployment.apps/flask-app unchanged
service/flask-app-service unchanged

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$
```

- Now type the following commands to check whether the pods are running or not.

  ~$ kubectl get po

- In order to list the all services running and check the port and external IP of the service run the following command.

  ~$ kubectl get svc

```
MINGW64:/c/Users/DELL/Desktop/Devops/k8s-main
        3d

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ kubectl get po
NAME                         READY   STATUS    RESTARTS      AGE
flask-app-5fb6cc9bd4-dbskb   1/1     Running   1 (17m ago)   3d

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ kubectl get svc
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
flask-app-service   LoadBalancer   10.106.208.132   127.0.0.1     6006:30416/TCP   3d
kubernetes          ClusterIP      10.96.0.1        <none>        443/TCP          3d

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ ^C

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ ^C

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ ^C

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ ^C
```

- Here my flask-app-service is running at port 6006. So in order to access the port of that service I need to enable a tunnel to that pod so that I can access the service using my local Ip **i.e., 127.0.0.1:6006**

  ~$ minikube tunnel

```
MINGW64:/c/Users/DELL/Desktop/Devops

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops
$ minikube tunnel
* Tunnel successfully started

* NOTE: Please do not close this terminal as this process must stay alive for th
e tunnel to be accessible ...

* Starting tunnel for service flask-app-service.
```

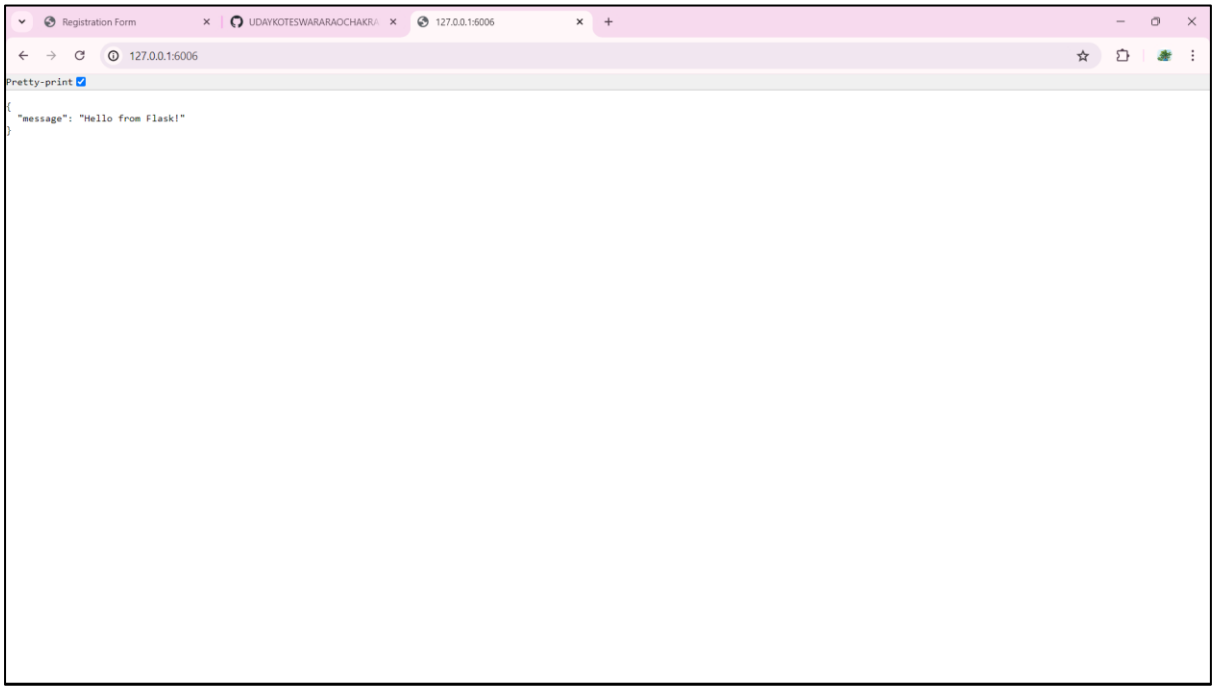
- Now we can access the server at **i.e., 127.0.0.1:6006.** The output should like this.
- By following these steps, you can successfully integrate Docker with Kubernetes, deploy a Flask application, and manage it using Minikube on both Windows and Linux environments.

9. Install and Explore Selenium for automated testing.

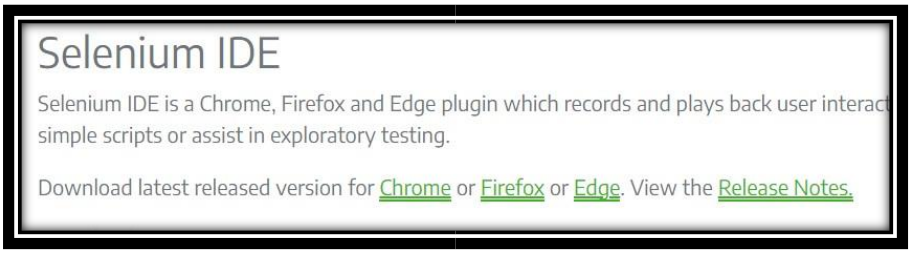
Selenium is a free, open-source automation testing suite for web applications across different browsers and platforms. It is somewhat similar to HP QuickTest Pro (QTP, currently UFT). However, Selenium focuses on automating web-based applications.

Steps :

- Open chrome and search for “selenium.dev” open selenium automated browse
- Click on downloads and select **java** for selenium client and webdriver language binding
- Selenium server (Grid)
- Download latest stable version 4-22.0

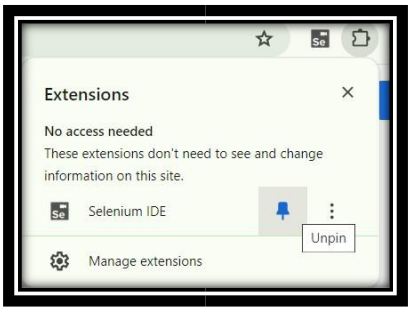


- **Selenium IDE :**
- At selenium ide click **chrome link**

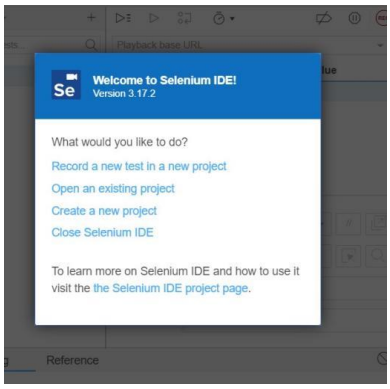


- It opens selenium ide,now click on Add to chrome

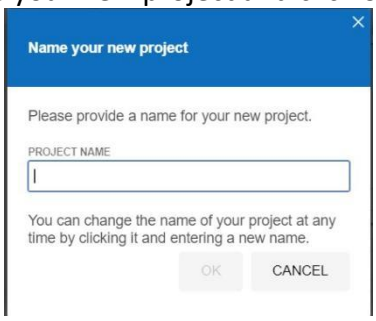
- Click on Extension and select selenium and pin it



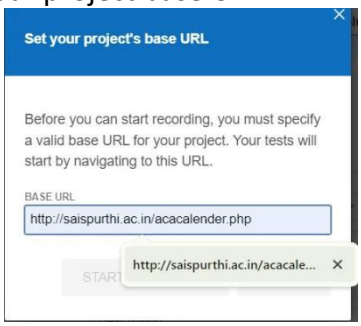
- Click on selenium ide and select record a new test in a new project



- Name your new project and click ok

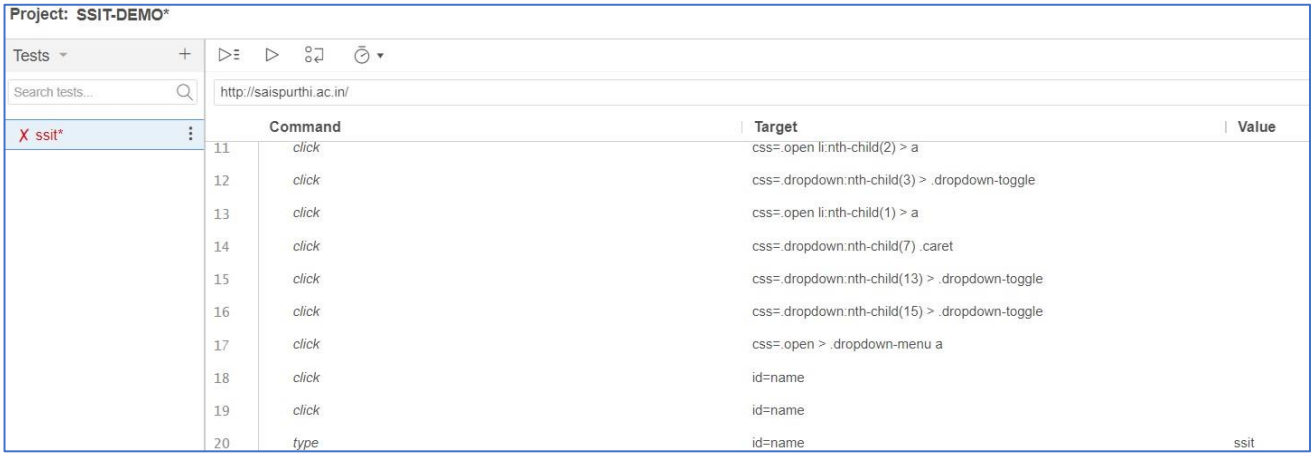


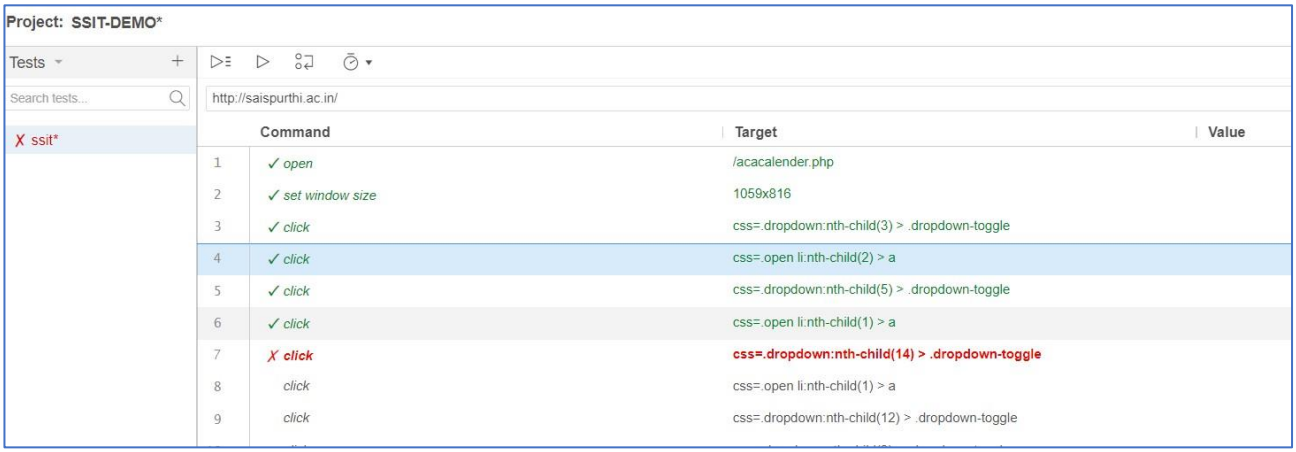
- Set your project base URL :



- Click on start recording
- Open ssit-demo project page and browse it and check the test cases
- Now check the recording in **ssit-demo** project page

- And save it as new test as **SSIT**





Selenium Environment Setup :

Installing Selenium requires going through the following steps:

Step 1: Install Java

Before moving further download and install Java Software Development Kit (JDK) in the first place.

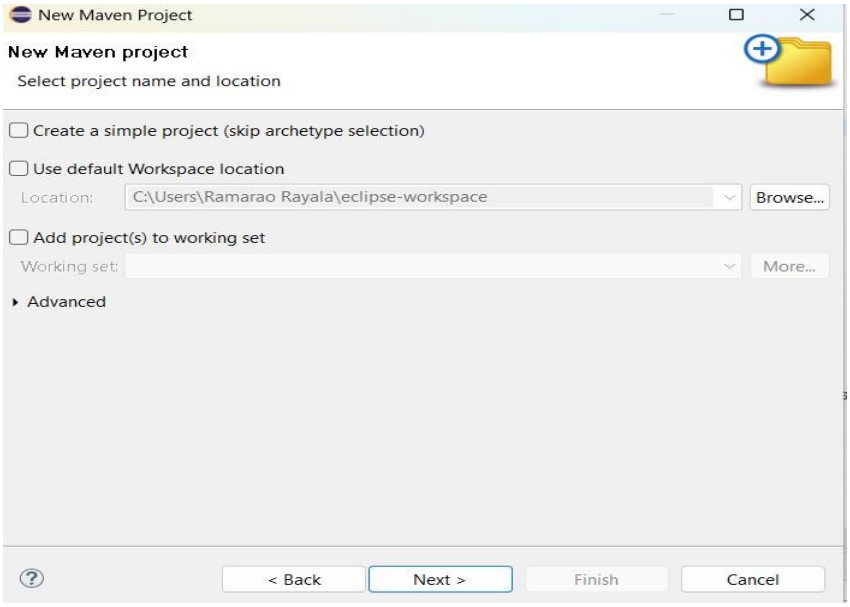
Step 2: Install Eclipse IDE

Next, download and install Eclipse IDE for Java developers keeping in mind the 32 or 64-bit windows versions.

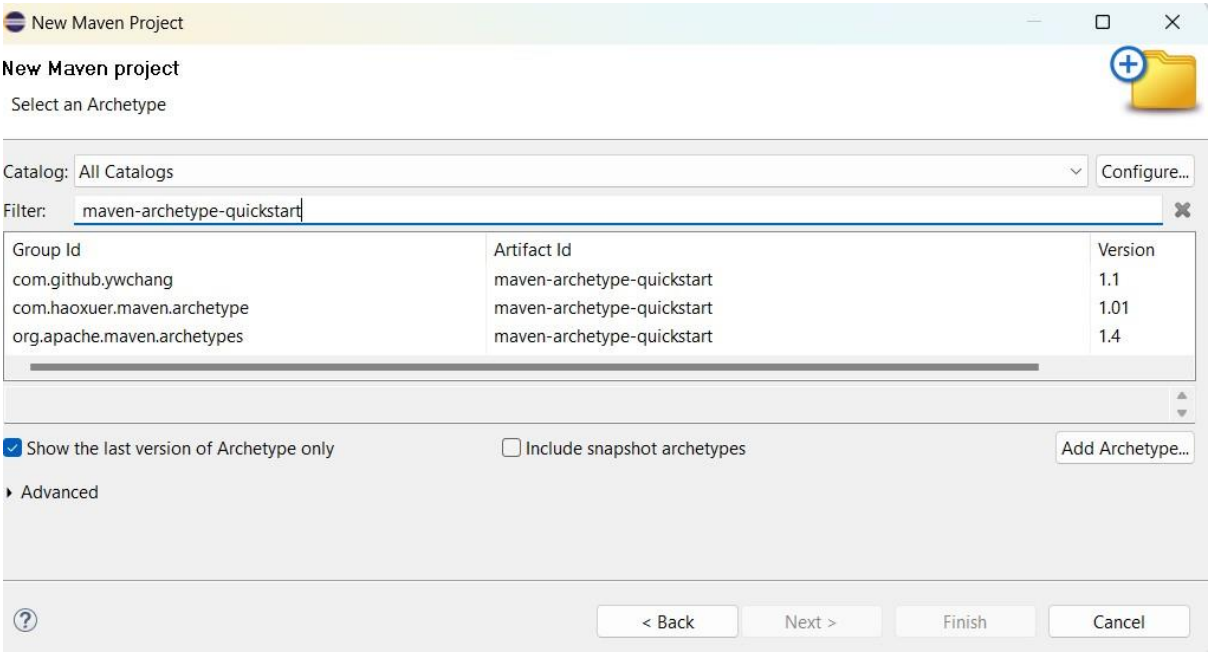
→ After completing the installation of eclipse open eclipse

- **Create a new maven Project**

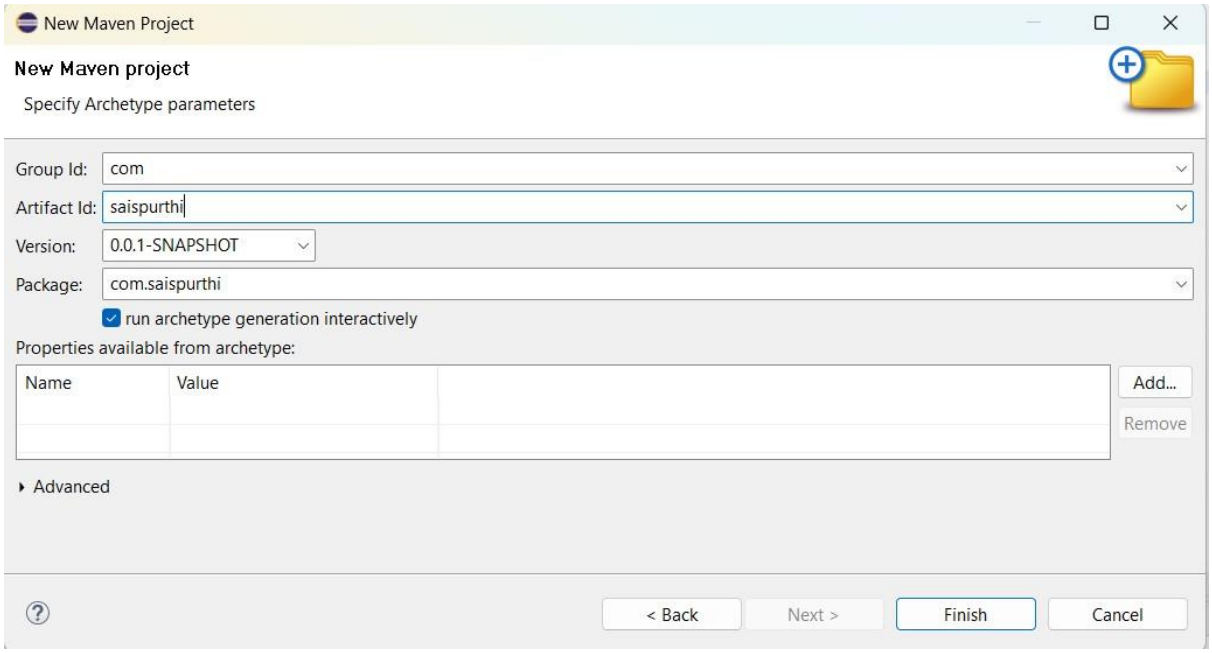
Click on the File menu, then go to New and select maven Project selct maven project name and location



- Select an archetype
- By using maven-archetype-qickstart-1.4



- Set archetype parameters and finish it



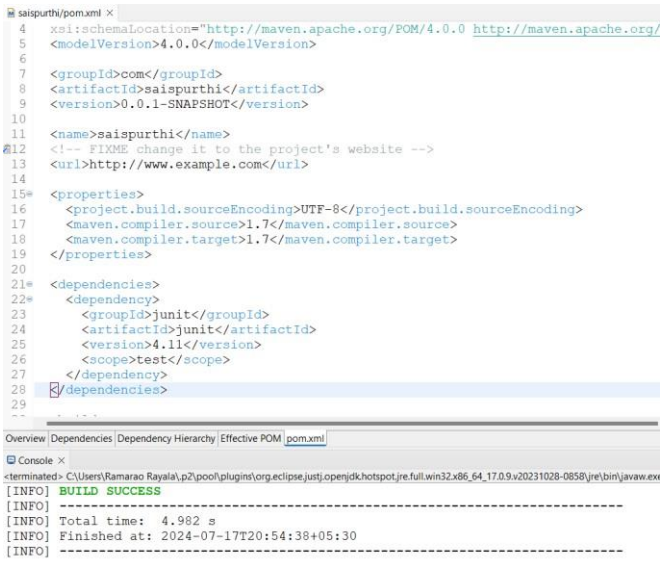
- Open pom.xml file from eclipse page itself by using file tab
- Now open chrome and serach for maven repository
- Search for **selenium** and **webdrivermanager** mvn repoistory
- **Selenium dependency :**

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java
->
<dependency>
   <groupId>org.seleniumhq.selenium</groupId>
   <artifactId>selenium-java</artifactId>
   <version>4.22.0</version>
</dependency>
```

- **WEBDRIVERMANAGER dependency :**

```
<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
<dependency>
   <groupId>io.github.bonigarcia</groupId>
   <artifactId>webdrivermanager</artifactId>
   <version>5.9.1</version>
</dependency>
```

- Copy the both above dependency and paste it on pom.xml file in **dependencies tag**
- Run it as **maven test**

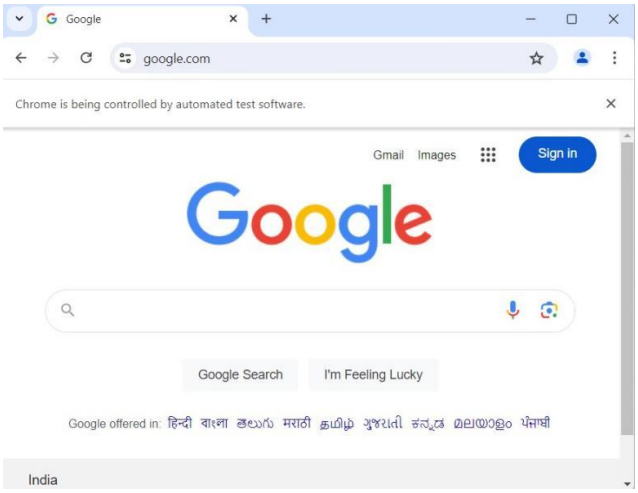


- Now open package explorer and select saispurthi file
- Testing code for automated testing :

Code :

```
package csm.testselinium;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;

public class TestLibraries{
 public static void main(String[] args) {
            // TODO Auto-generated method stub
    WebDriverManager.chromedriver().setup();
    WebDriver driver=new ChromeDriver();
    driver.get("http://google.com");
  }
}
```

Save the code and run it The output is :

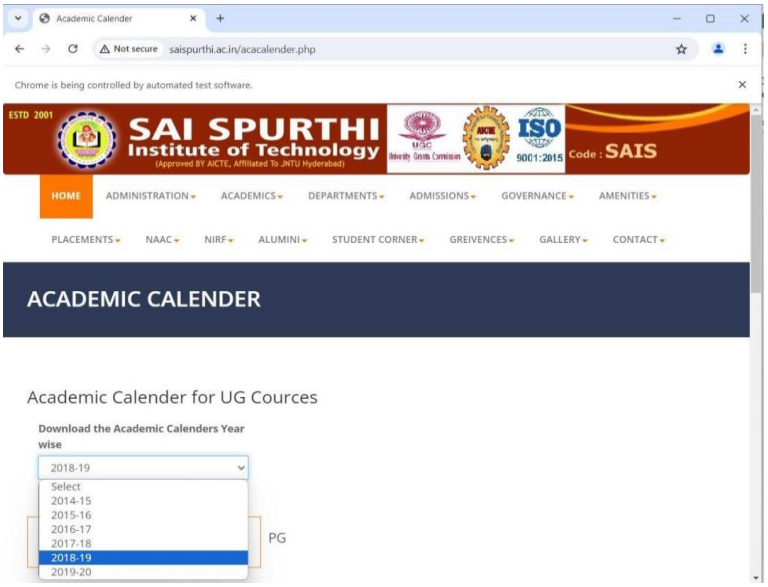


DropDown testing :

Code :

```
package csm.testselinium;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;
import io.github.bonigarcia.wdm.WebDriverManager;
public class TestDropDownLib {
public static void main(String[] args) throws InterruptedException {
                        WebDriverManager.chromedriver().setup();
                        WebDriver driver = new ChromeDriver();
                        driver.get("http://saispurthi.ac.in/acacalender.php");
                        Thread.sleep(3000);
                        WebElement calElement = driver.findElement(By.id("calyear"));
                        Select calNameDropdown = new Select(calElement);
                        List<WebElement> calNameDropdownOptions =
calNameDropdown.getOptions();
                        for (WebElement option : calNameDropdownOptions) {
                                System.out.println(option.getText());
                        }
                        calNameDropdown.selectByIndex(1);
                        calNameDropdown.selectByValue("2018-19");
                        calNameDropdown.selectByVisibleText("2019-20");
                        String selectedText =
                        calNameDropdown.getFirstSelectedOption().getText();
                        System.out.println("Selected visible text - " +selectedText);
    }
}
```

Save the code and run it The output is :



10. Write a simple program in JavaScript and perform testing using Selenium.

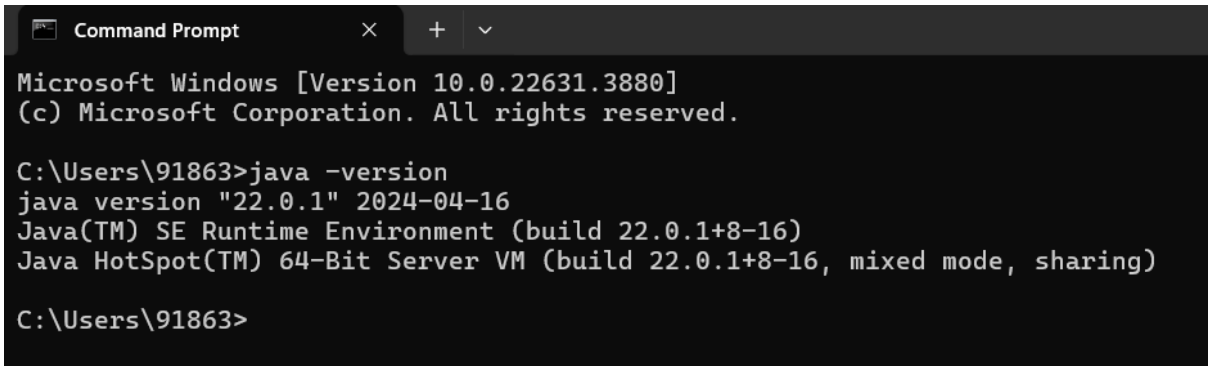
JMeter is an open-source testing tool that analyses and measures the performance of software and products. The tool is entirely Java-based and ensures performance testing, load testing, and functional testing.

Earlier, JMeter was originally developed for web applications but has now expanded to several other functions.

Steps to install JMeter:

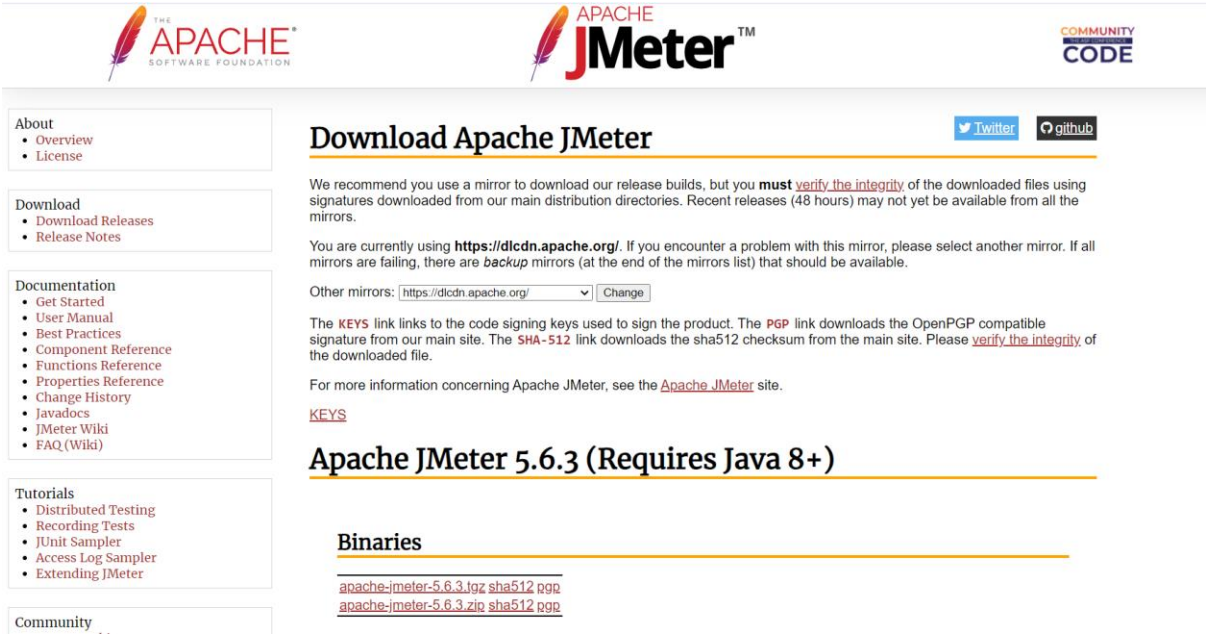
1.Check if Java Is Installed

- Open the command prompt
- Put the command java -version



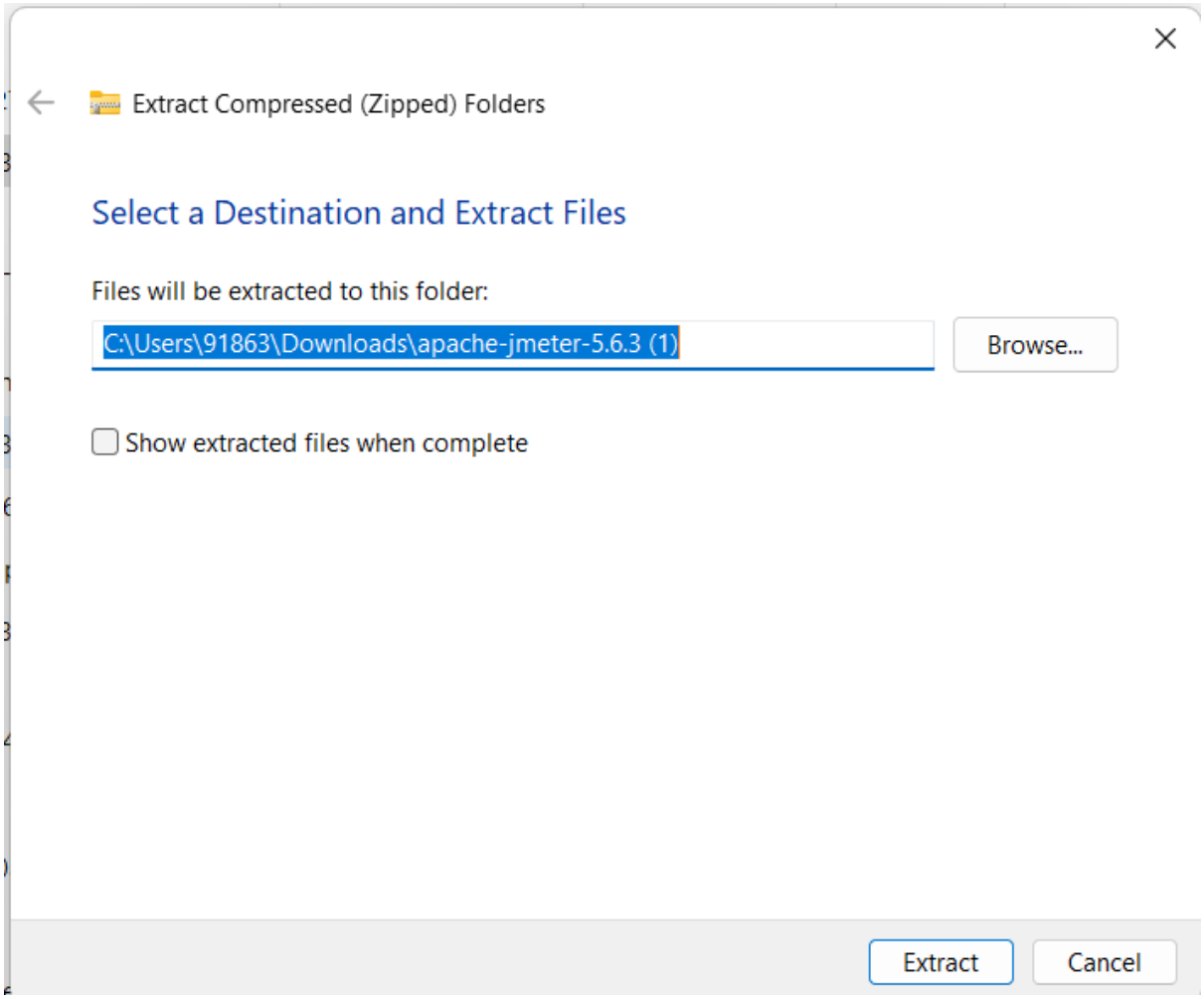
2.Download JMeter

- To download JMeter go to the Apache JMeter Website: https://jmeter.apache.org/download_jmeter.cgi
- On the website go to the binaries section and download the zip file
- Wait for the zip folder to be downloaded



3.Install JMeter

• Once the zip folder is downloaded, go to the folder location, and then extract the zip folder



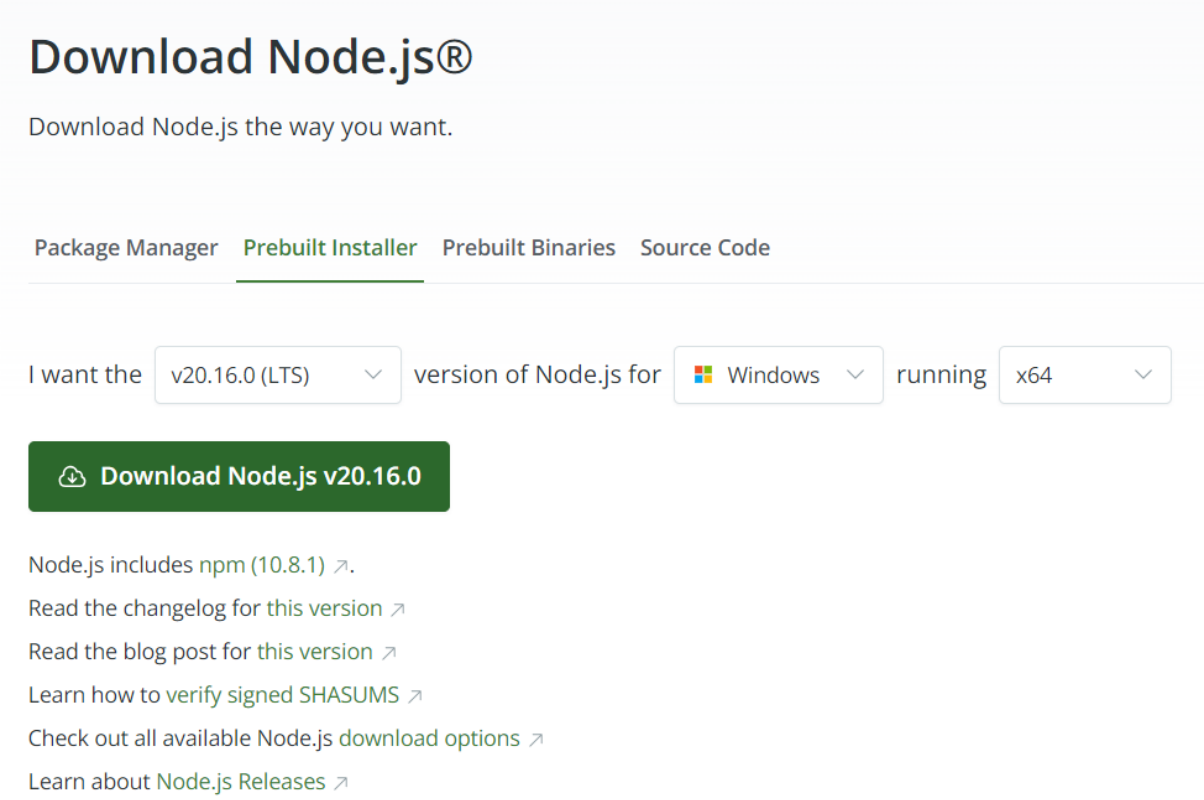
• After completing the extraction open bin file then apacheJmeter.jar

•Start the process.

• After we must create a folder with the name “**Sonar”**, again in that we have to create another folder called “**JMeter-demo”.**

•After that we must open it with git bash.

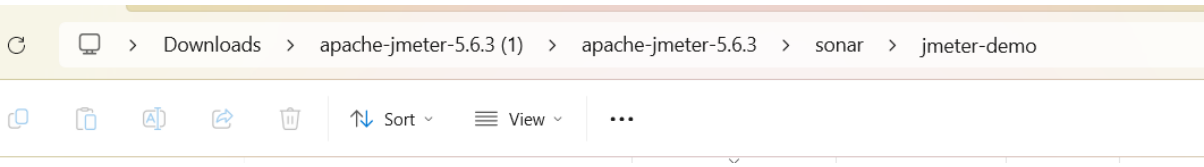
•For the continuation of the next process here, we must install **“node js”.**



• After that extract the node js also.

• Create a folder in apache-jmeter-5.6.3 in that create a new folder called sonar

• In sonar folder create a new folder called jmeter-demo



•For automated testing we must install some dependencies in the JMeter-demo folder via git bash.

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sona
r/jmeter-demo
$ mkdir googleTest

91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo
$ cd googleTest
```

- **Check the version of the node js and npm version**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ node -v

v20.16.0

91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ npm --version
10.8.1
```

- **Initialising empty "npm" repository using gitbash.**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (googletest)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\91863\Downloads\apache-jmeter-5.6.3 (1)\apache-jmeter-5.6.3\sonar\jmeter-demo\googleTest\pa
ckage.json:

{
  "name": "googletest",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
Is this OK? (yes) yes

npm notice
npm notice New patch version of npm available! 10.8.1 -> 10.8.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.2
npm notice To update run: npm install -g npm@10.8.2
npm notice
```

- **List the files and directories-ls**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ ls
package.json
```

- **Installing selenium WebDriver**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ npm install selenium-webdriver


added 17 packages, and audited 18 packages in 2m

found 0 vulnerabilities
```

- **Displaying the content of the files-cat**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ cat package.json
{
  "name": "googletest",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "selenium-webdriver": "^4.23.0"
  }
}
```

- **Installing chrome driver**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ npm install chromedriver

added 63 packages, and audited 81 packages in 1m

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

- **ls**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ ls
node_modules/  package-lock.json  package.json
```

- **Cat command**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ cat package.json
{
  "name": "googletest",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "chromedriver": "^127.0.0",
    "selenium-webdriver": "^4.23.0"
  }
}
```

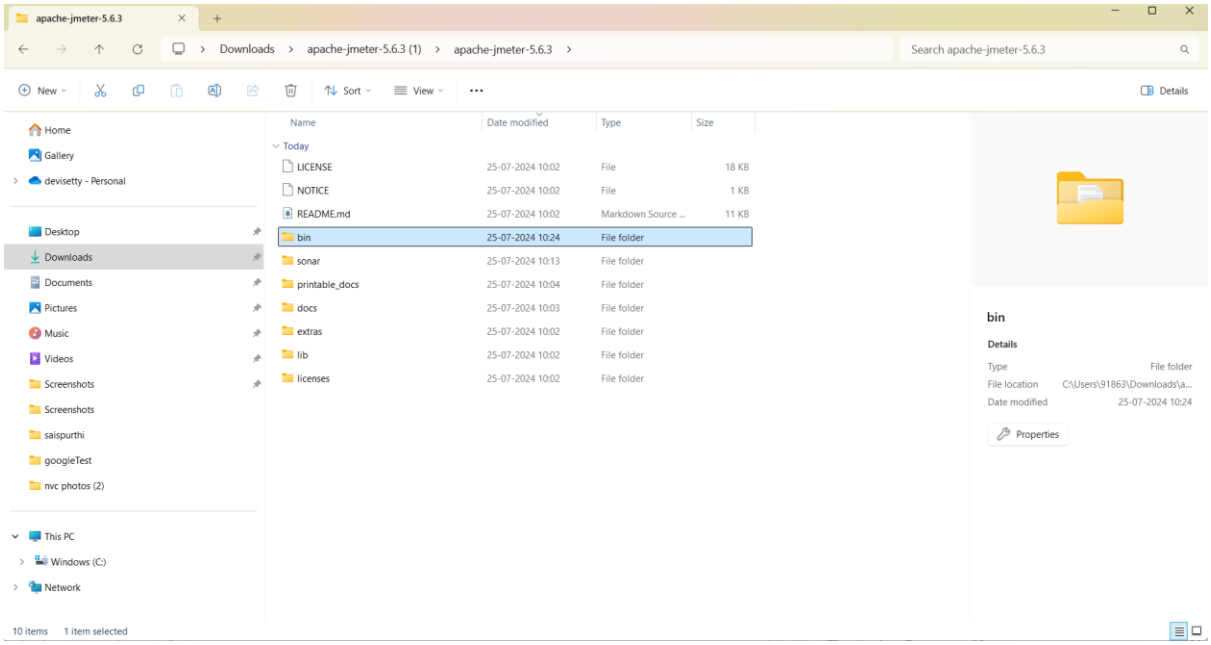
- **Installing gecko driver**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ npm install geckodriver

added 29 packages, and audited 110 packages in 10s

10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

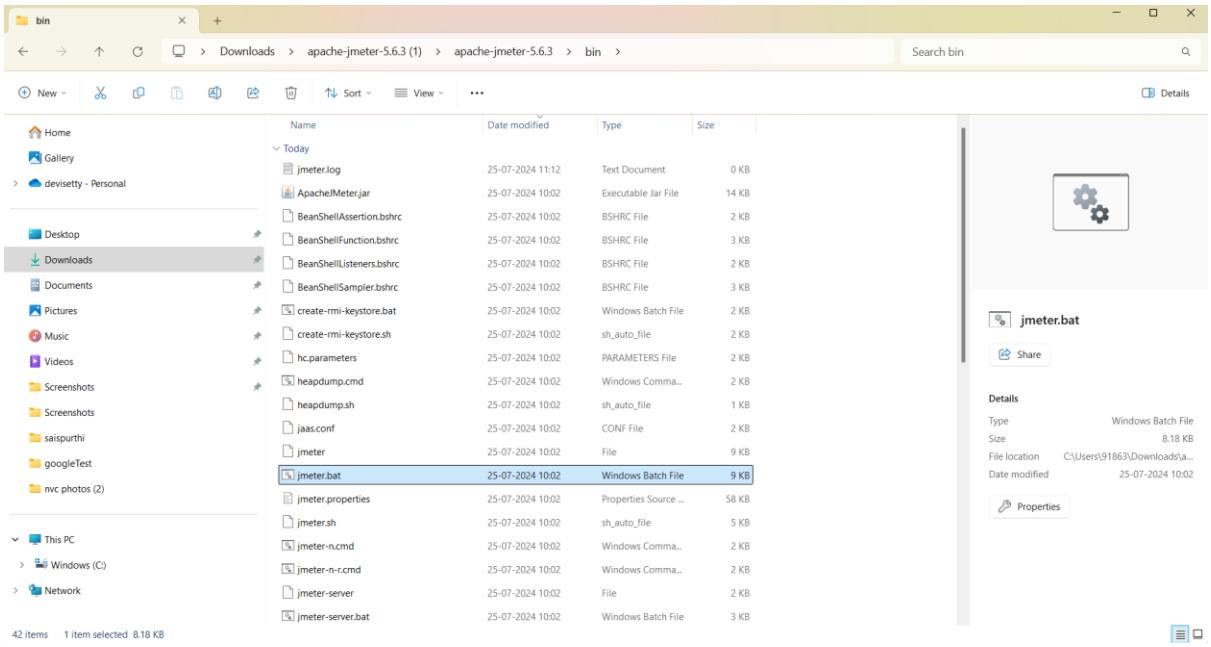
- **Cat command**

```
91863@sreya MINGW64 ~/Downloads/apache-jmeter-5.6.3 (1)/apache-jmeter-5.6.3/sonar/jmeter-demo/googleTest
$ cat package.json
{
  "name": "googletest",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "chromedriver": "^127.0.0",
    "geckodriver": "^4.4.2",
    "selenium-webdriver": "^4.23.0"
  }
}
```

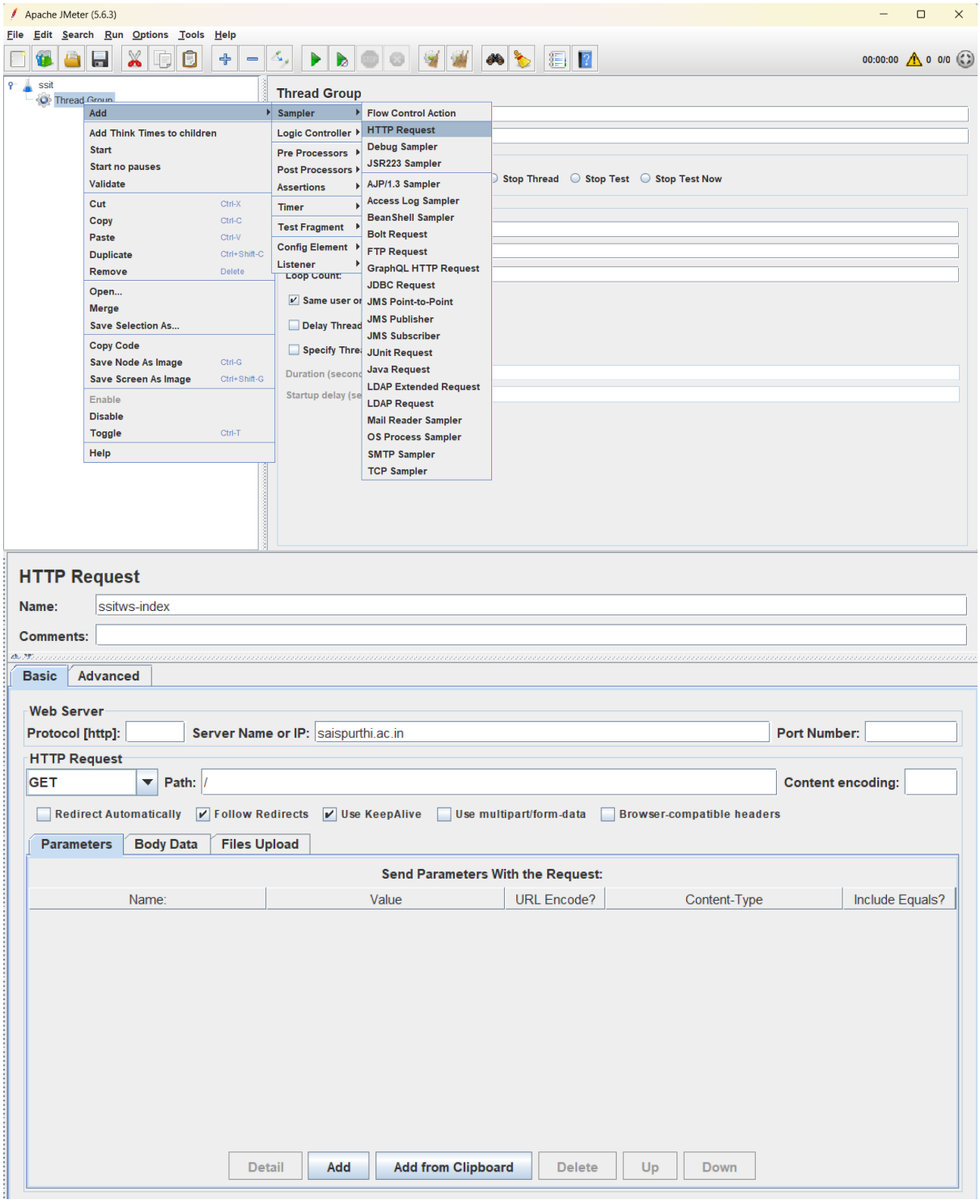
- **Open bin file in JMeter-folder**



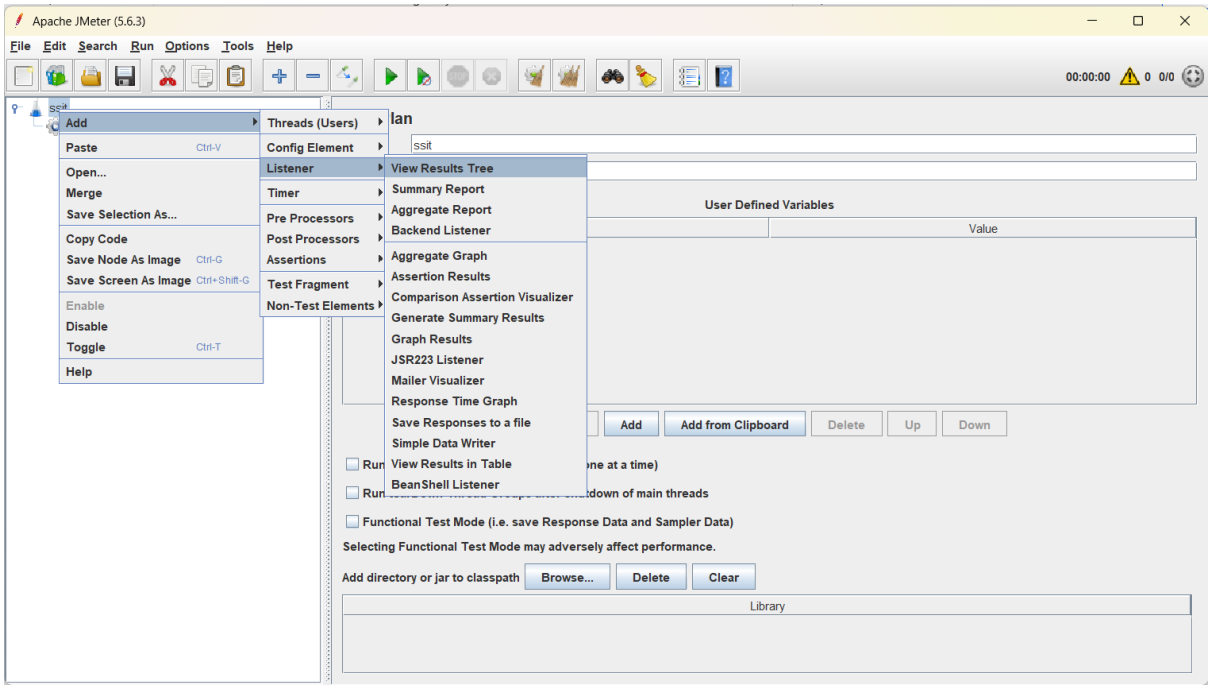
- **Open jmeter.bat in JMeter-folder after bin**



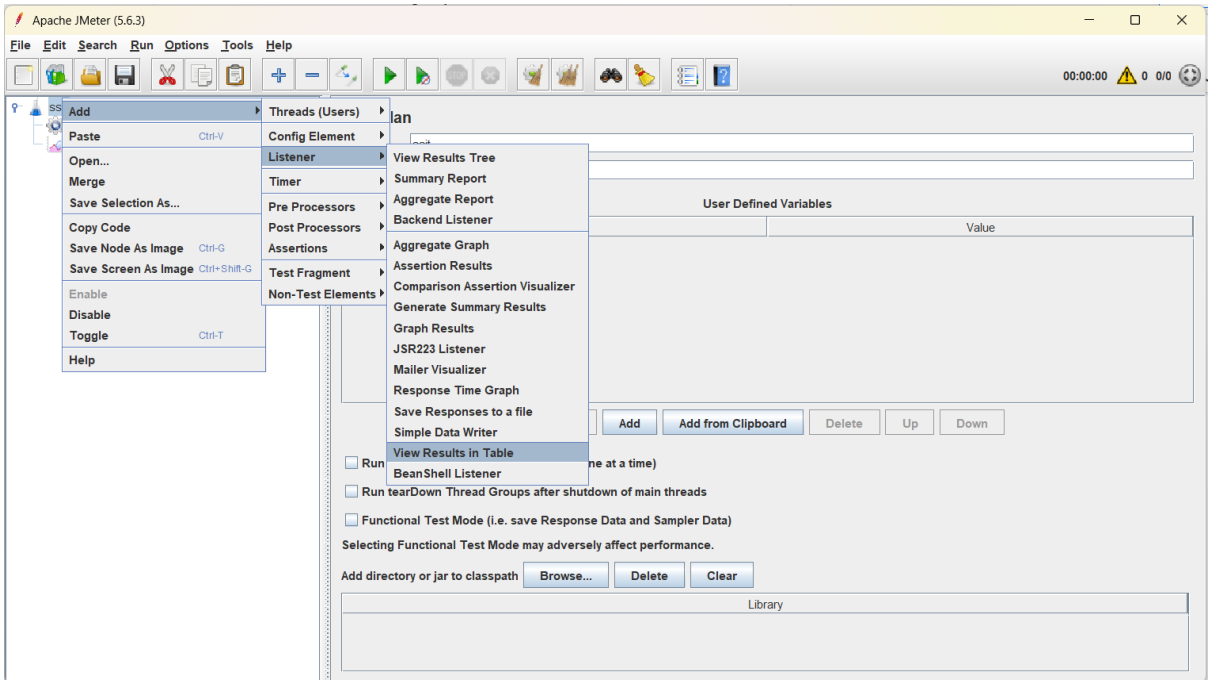
- **From that bin folder open Apache-JMeter and select all the options displaying in the image**



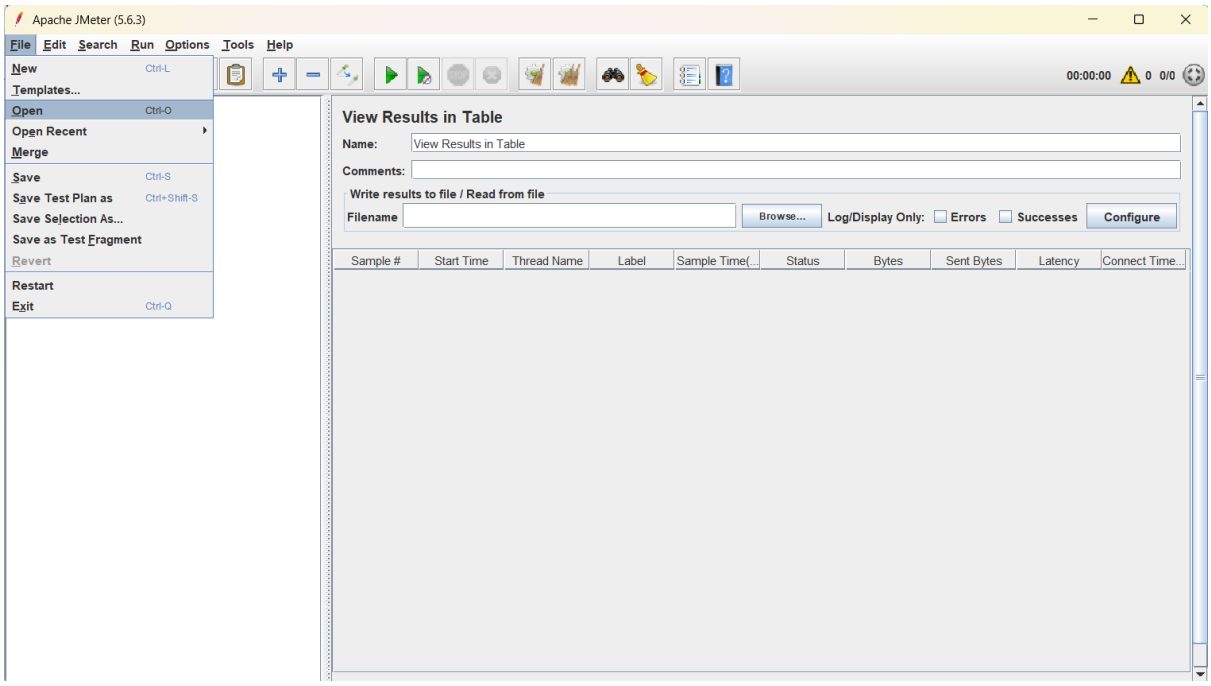
- **Viewing the results in the tree**



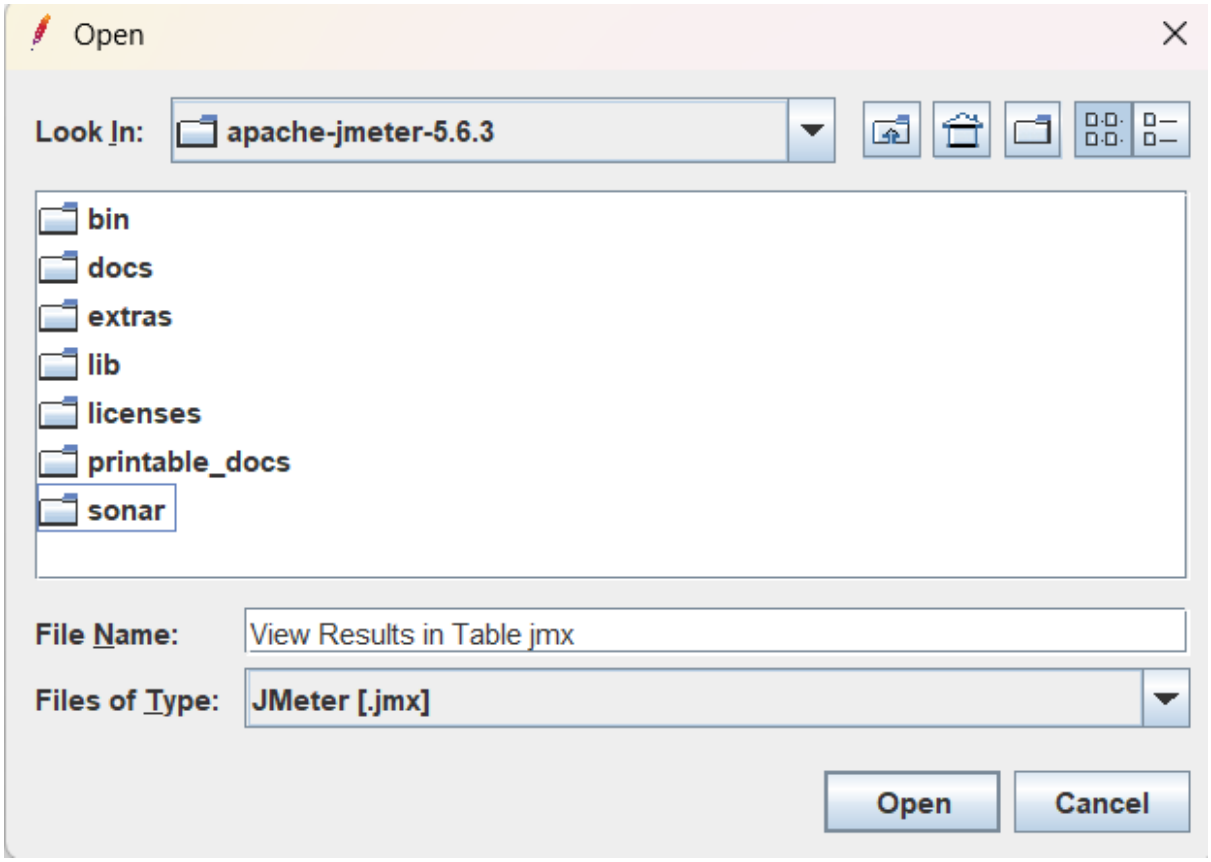
- **Viewing the results in the table**



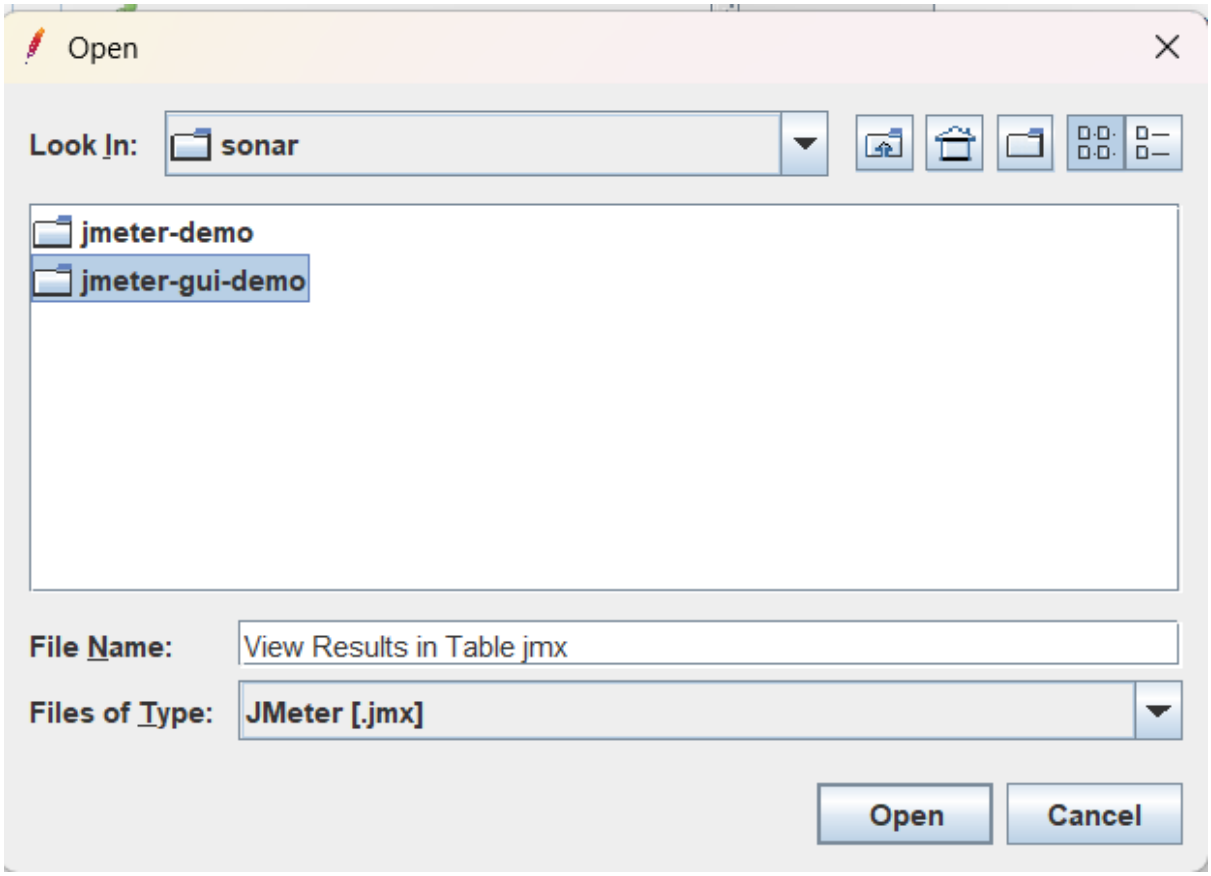
- **Click on file and in that click on open**



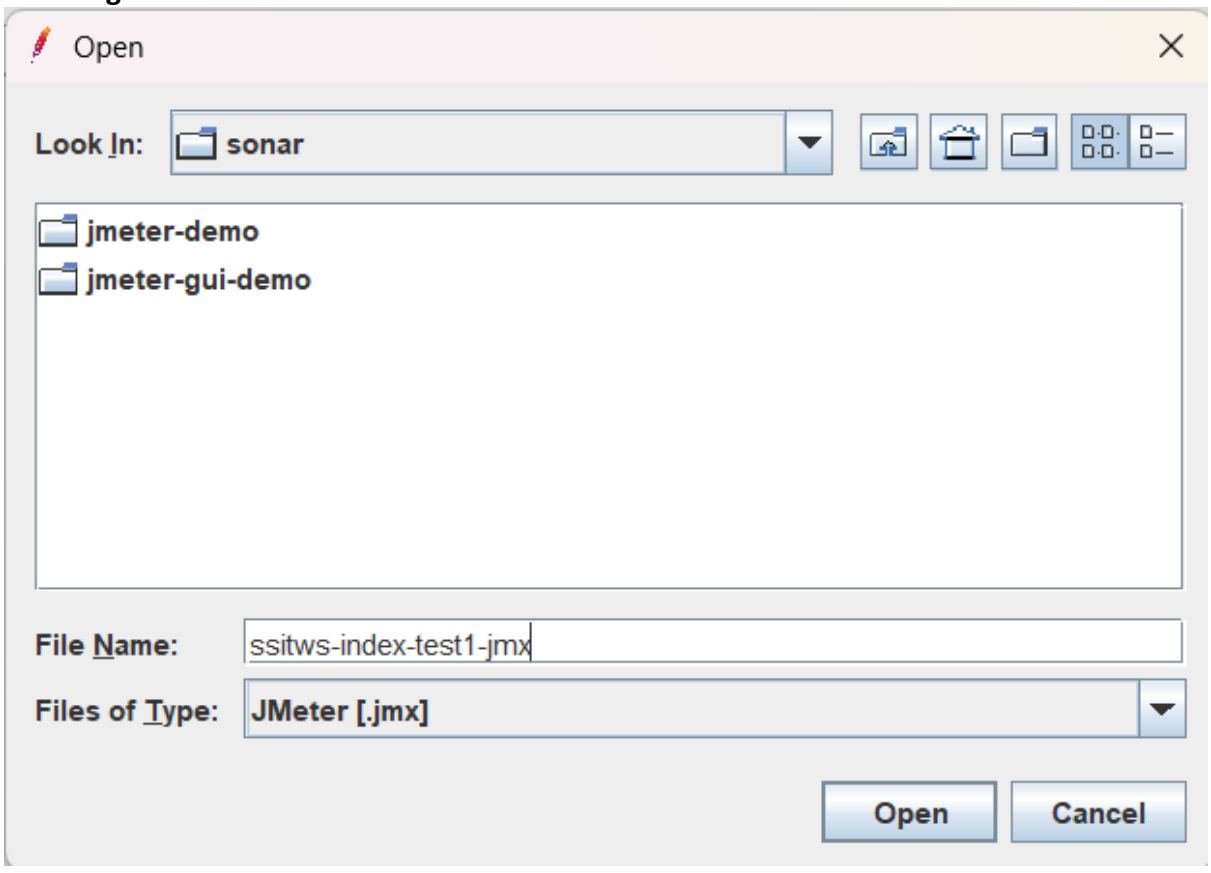
- **Click on sonar**



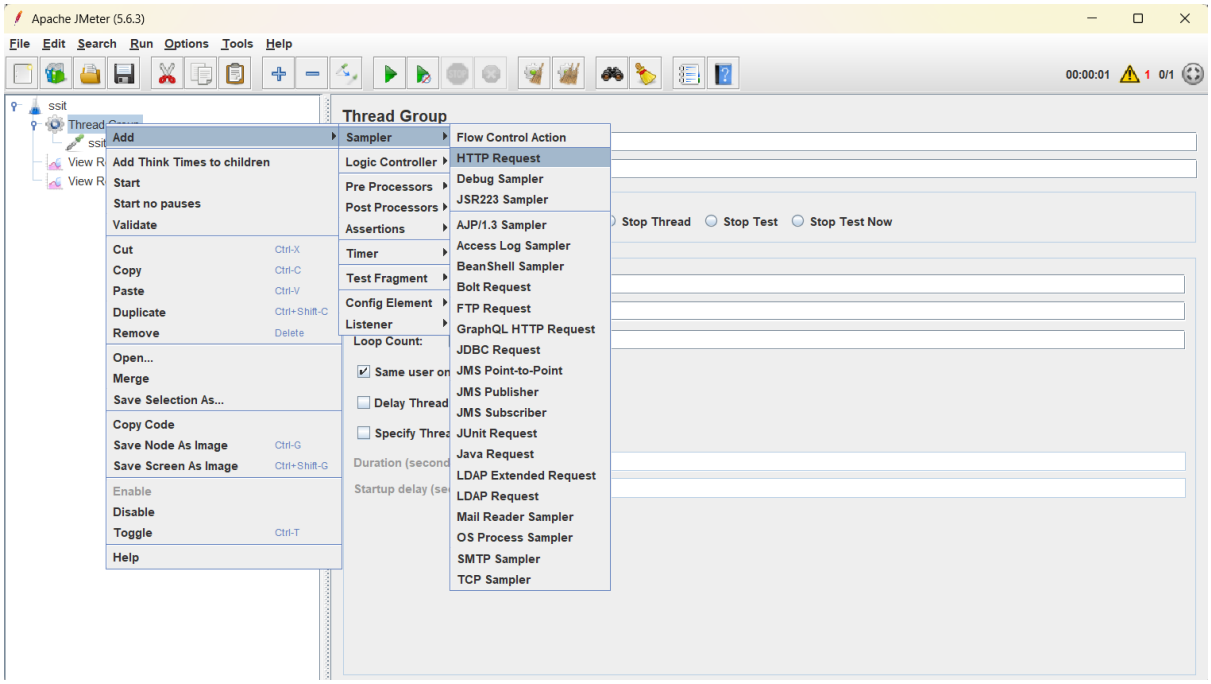
- **Selecting JMeter-demo and JMeter-Gui-demo**



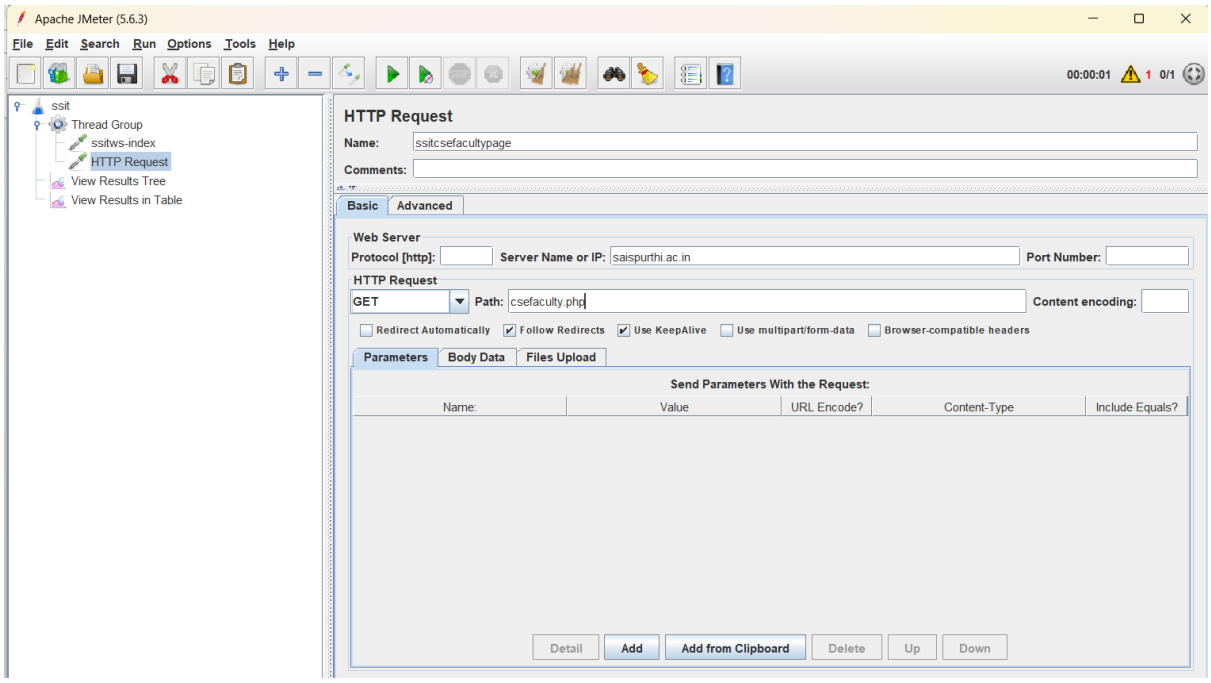
- **Setting the file name**



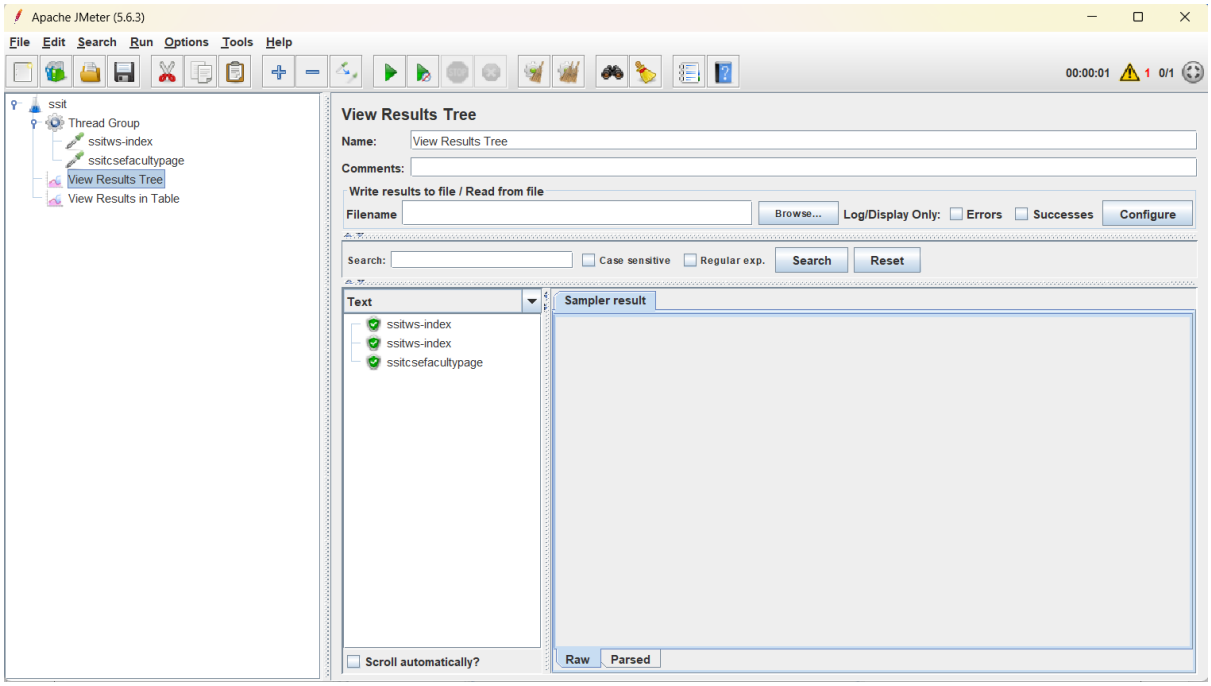
- **Then select the http request**



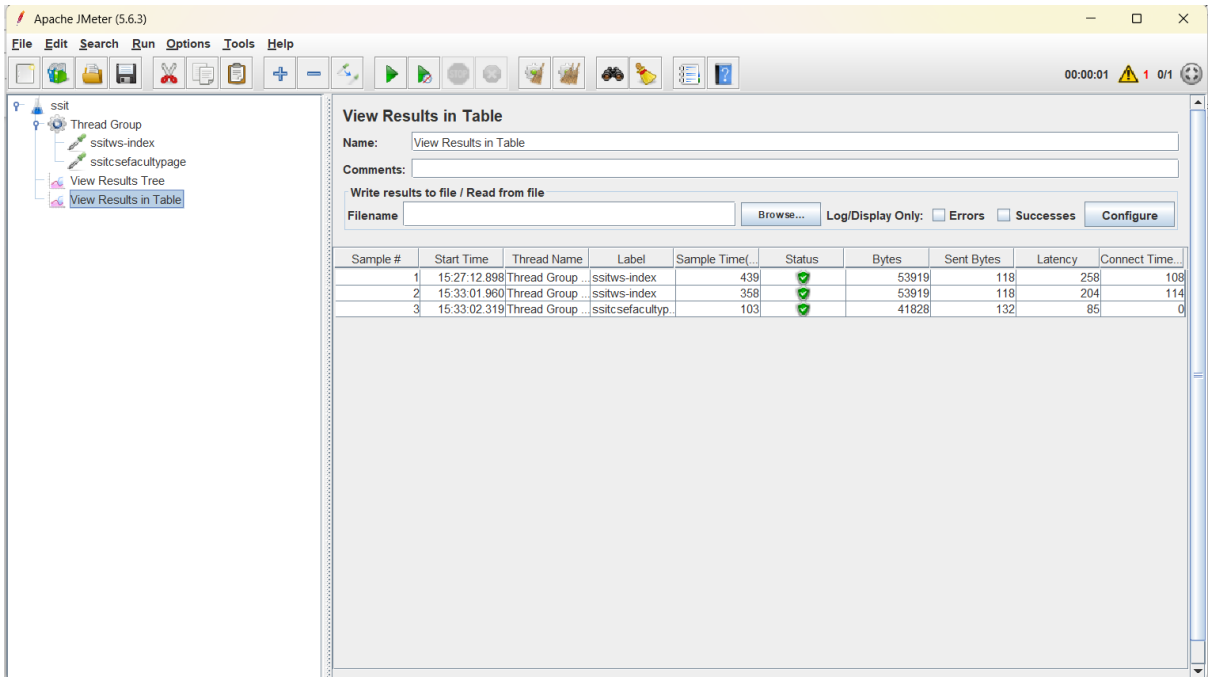
• In http request place the server's name or Ipas saispurthi.ac.in and path as csefaculty.php



• **Viewing the results in tree**

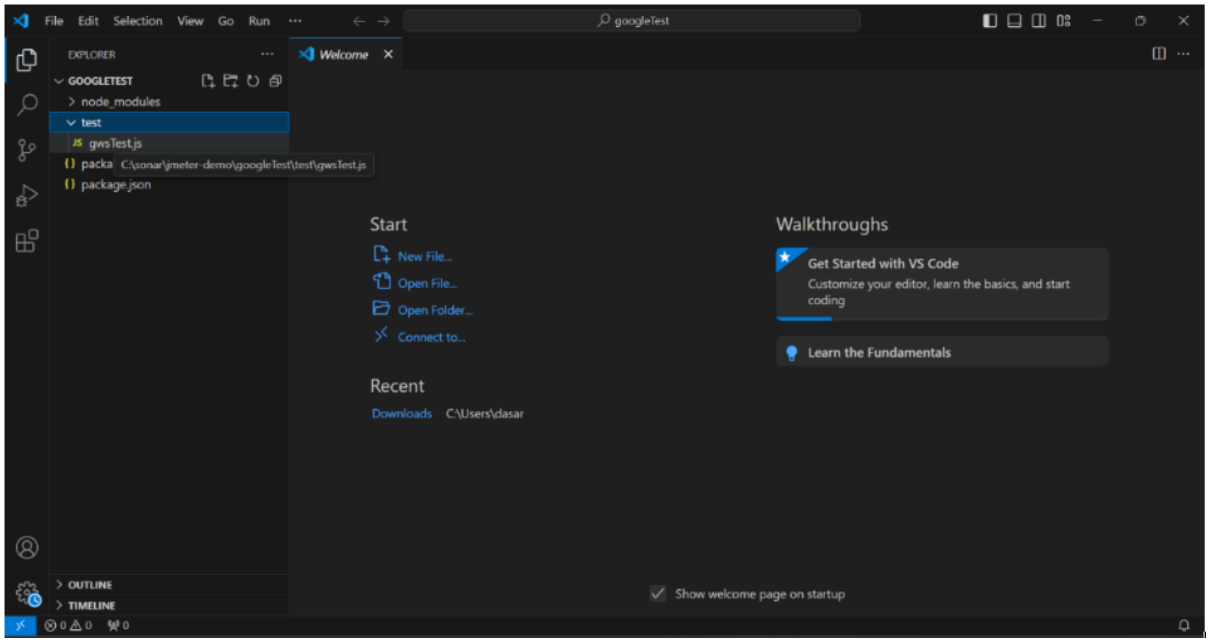


• **Viewing the results in table**



• **After completing above installation process in the google Test directory we create a folder “Test” in that we must create a test document with the name “gwsTest.js”**

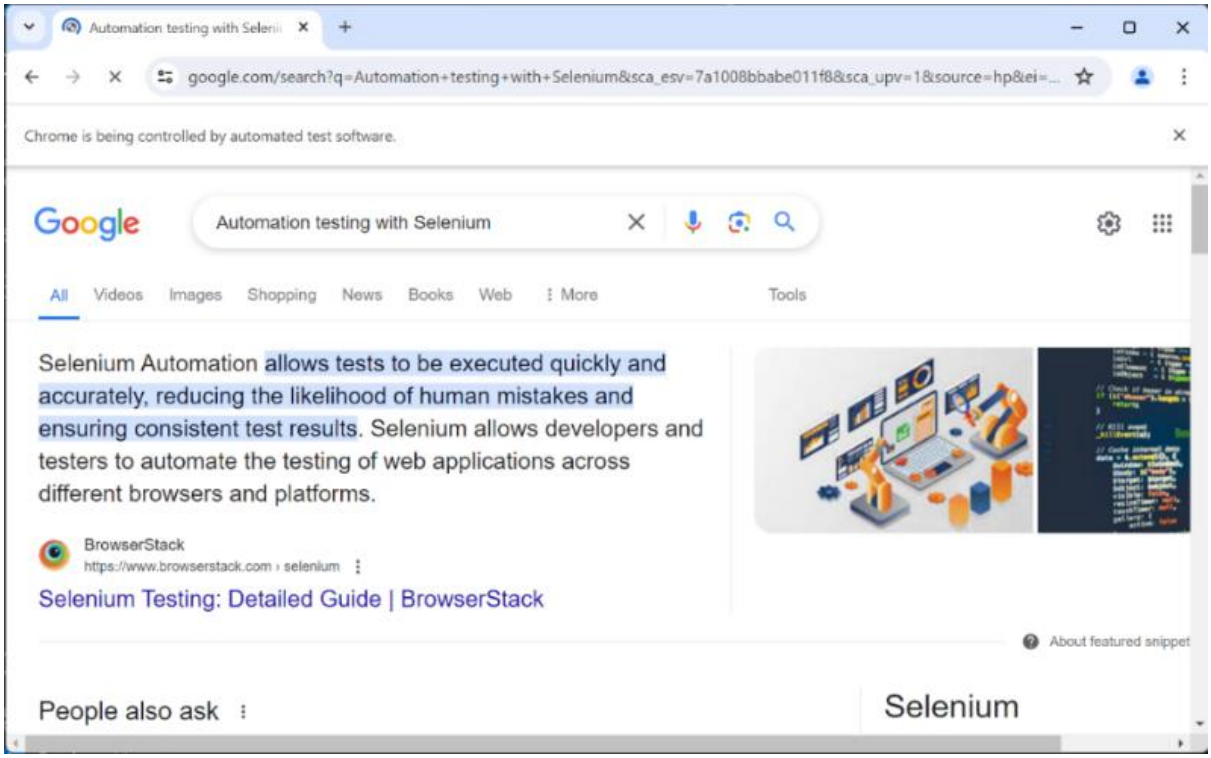
• **After that go to VISUAL STUDIO Code, and then open it with the test document and in that we must write the code for the automated testing.**



Code:

```
const {By, Key, Builder} = require("selenium-WebDriver");

require("chromedriver");

 async function example () {

        var searchString = "Automation testing with Selenium";

        let driver = await new Builder (). for Browser("chrome"). build ();

        await driver. Get("http://google.com");

        await driver. findElement (By.name("q")). sendKeys (searchString, Key. RETURN);

        var title = await driver. getTitle ();

        console.log ('Title is:', title);

    await driver. Quit ();

}
```

Output:


Automation testing with Selenium
google.com/search?q=Automation+testing+with+Selenium&sca_esv=7a1008bbabe011f8&sca_upv=1&source=hp&ei=...
Chrome is being controlled by automated test software.
Google
Automation testing with Selenium
All
Videos
Images
Shopping
News
Books
Web
More
Tools
Selenium Automation allows tests to be executed quickly and accurately, reducing the likelihood of human mistakes and ensuring consistent test results. Selenium allows developers and testers to automate the testing of web applications across different browsers and platforms.
BrowserStack
https://www.browserstack.com › selenium
Selenium Testing: Detailed Guide | BrowserStack
About featured snippet
People also ask
Selenium

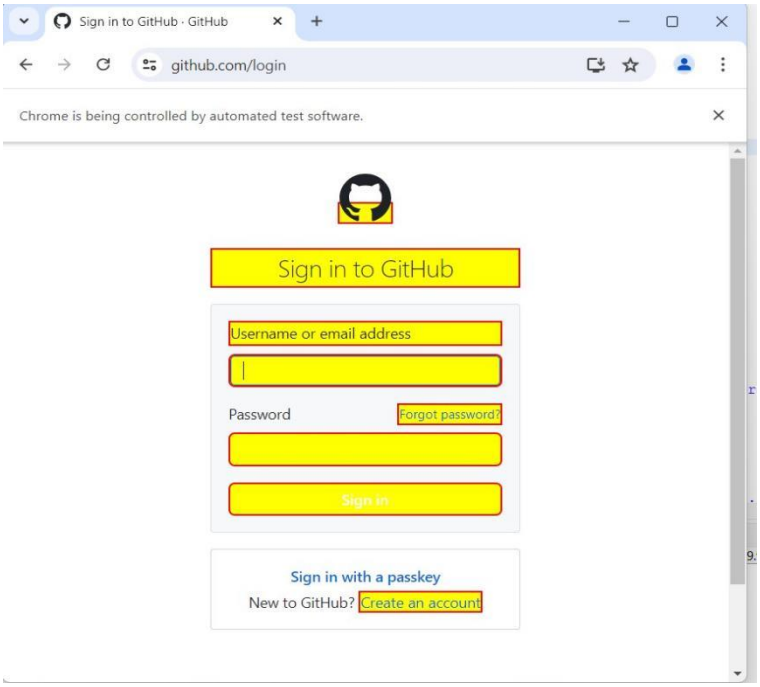
11. Develop test cases for the above containerized application using selenium.

- Testing the code for highlighting the specific words :

Code:

```
package csm.testselinium;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;
public class TestLib {
public static void main(String[] args) throws InterruptedException {
		WebDriverManager.chromedriver().setup();
		WebDriver driver = new ChromeDriver();
		driver.get("https://github.com/login");
		Thread.sleep(3000);
		highlight(driver, driver.findElement(By.id("login_field")));
		Thread.sleep(3000);
		highlight(driver, driver.findElement(By.name("password")));
		Thread.sleep(3000);
		highlight(driver, driver.findElement(By.className("header-logo")));
		Thread.sleep(3000);
		highlight(driver, driver.findElement(By.linkText("Forgot password?")));
		Thread.sleep(3000);
		highlight(driver, driver.findElement(By.partialLinkText("Create an")));
		Thread.sleep(3000);
		highlight(driver, driver.findElement(By.tagName("h1")));
		Thread.sleep(3000);
		highlight(driver,
		driver.findElement(By.xpath("//label[contains(text(),'Username or email
	address')]")));
		Thread.sleep(3000);
		highlight(driver,
		driver.findElement(By.cssSelector("input[name='commit']")));
	}

	public static void highlight(WebDriver driver, WebElement element) {
		JavascriptExecutor jsExecutor = (JavascriptExecutor) driver;
		jsExecutor.executeScript("arguments[0].setAttribute('style', 'border:2px solid
red; background:yellow')", element);
} }
```

- Save the code and run the output is :



12. Demonstrate the Integration of SonarQube with Jenkins for code analysis.

Step 1: Install SonarQube and SonarQube Scanner

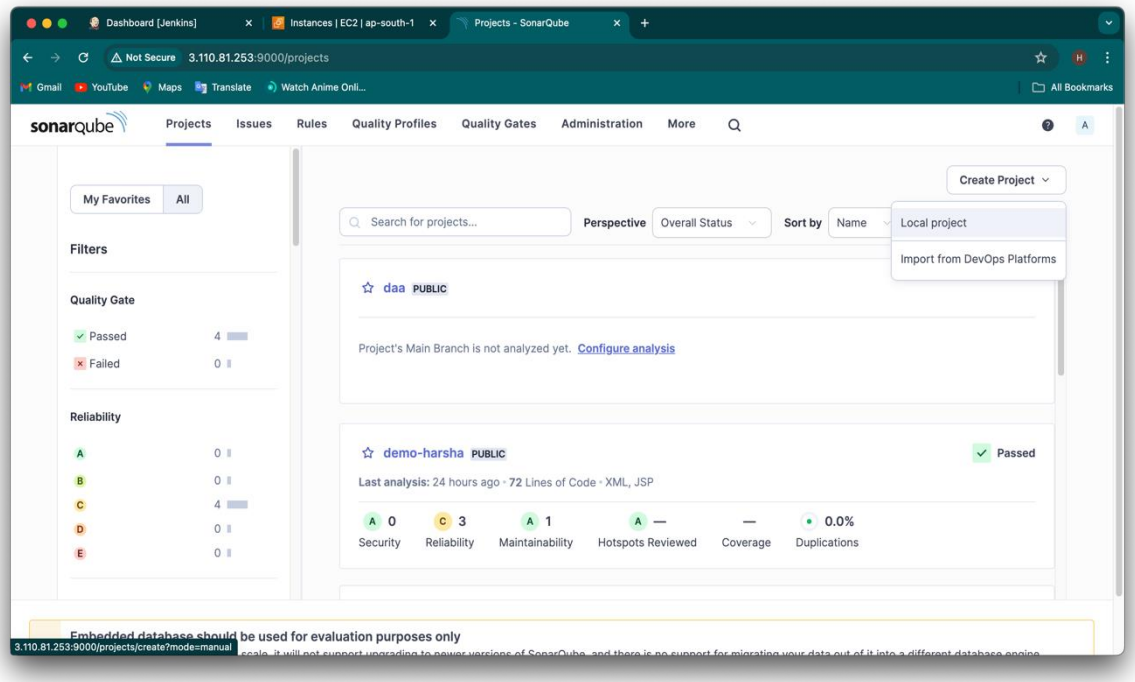
Download and Install SonarQube:

1. **Download SonarQube from the official website:** Visit the SonarQube download page and download the appropriate version for your operating system.

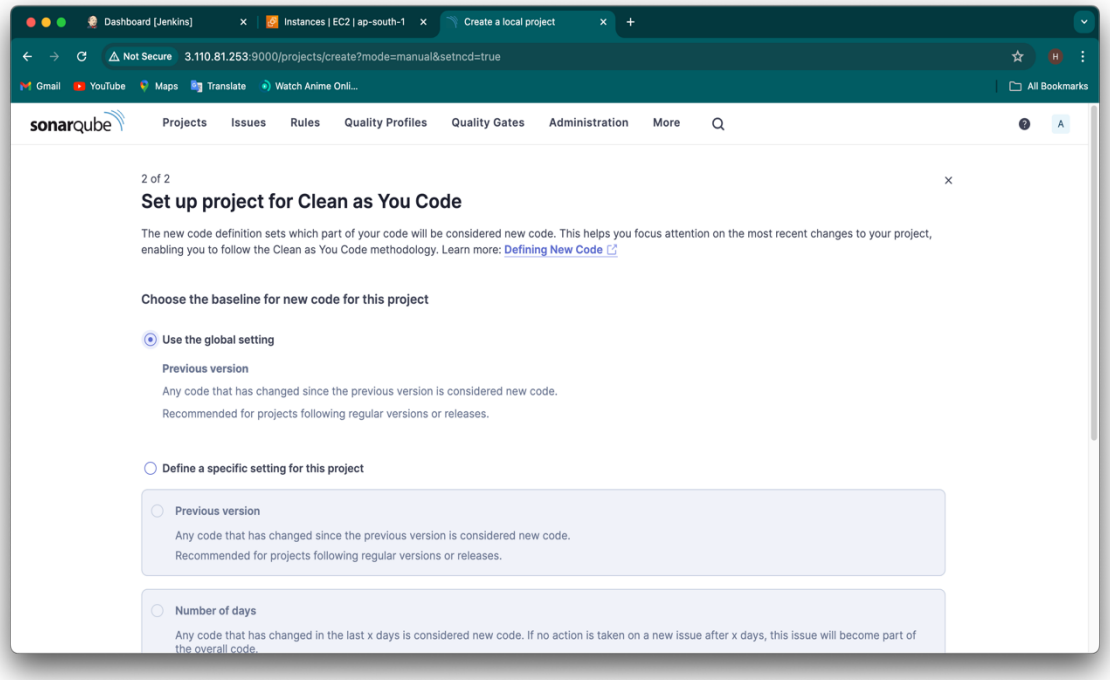
2. **Unzip the file and follow the installation instructions for your operating system:** Unzip the downloaded file. For Windows, unzip it and run StartSonar.bat from the bin\windows-x86-64 directory. For macOS or Linux, unzip it and run sonar.sh start from the bin/<system> directory.

3. **Start the SonarQube server:** Navigate to the bin/<system> directory in the terminal and execute ./sonar.sh start.
4. **Create project in SonarQube:**

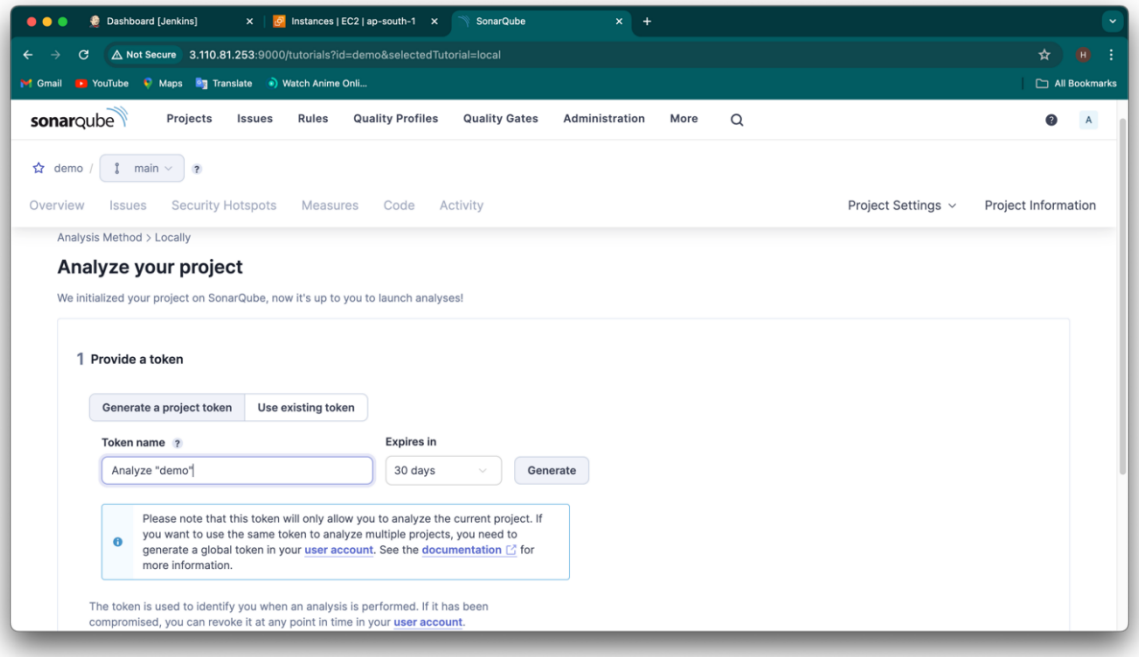
→ **Open SonarQube:** Now create a local project by clicking on create project select (use the global setting) and create project



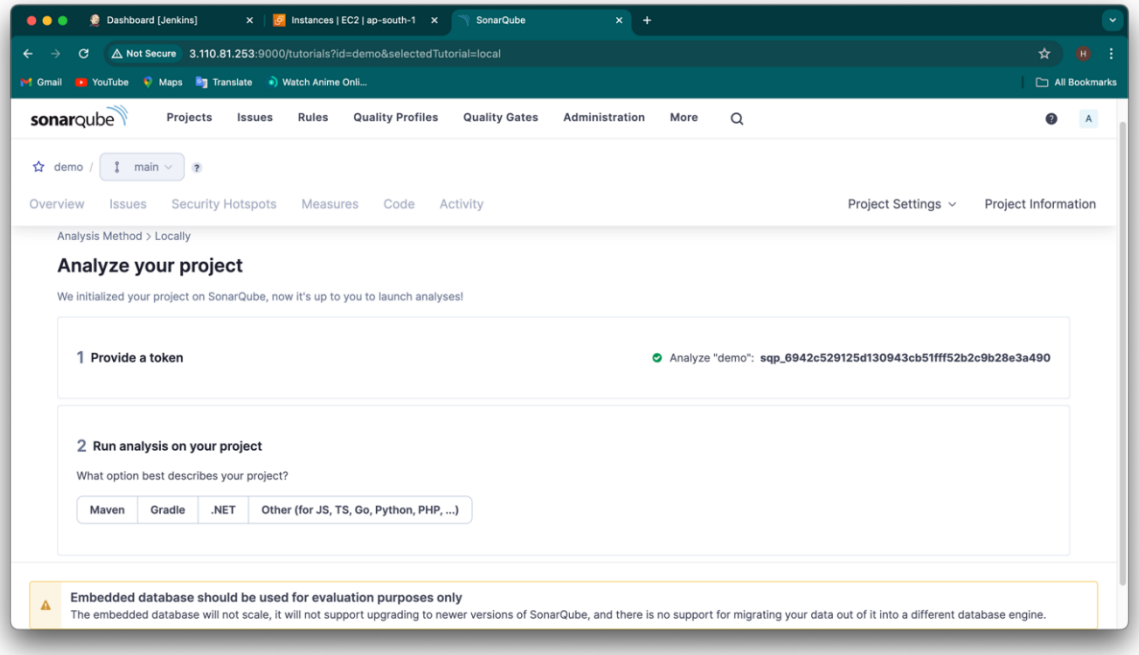
→ Select (use the global setting) and create project



→ **Generate a SonarQube Authentication Token:** After creating a local project just click on the LOCAL which pops below and give a name to your token ,set expiration of the token and click GENARATE



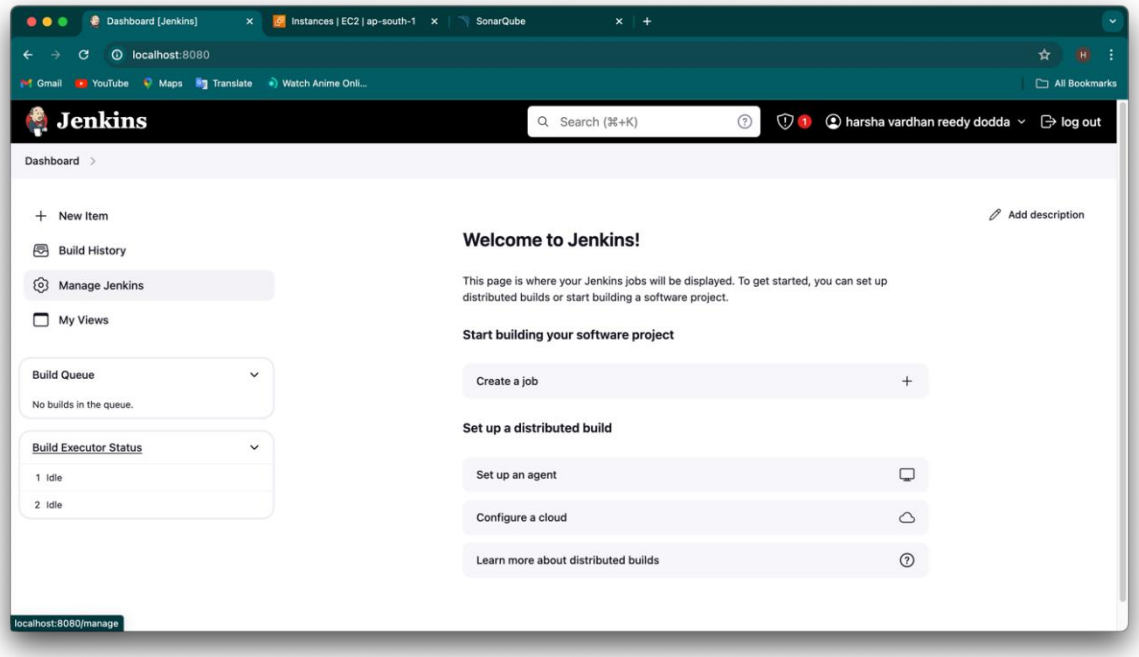
→ **Copy this token:** This token will be used for Jenkins integration.



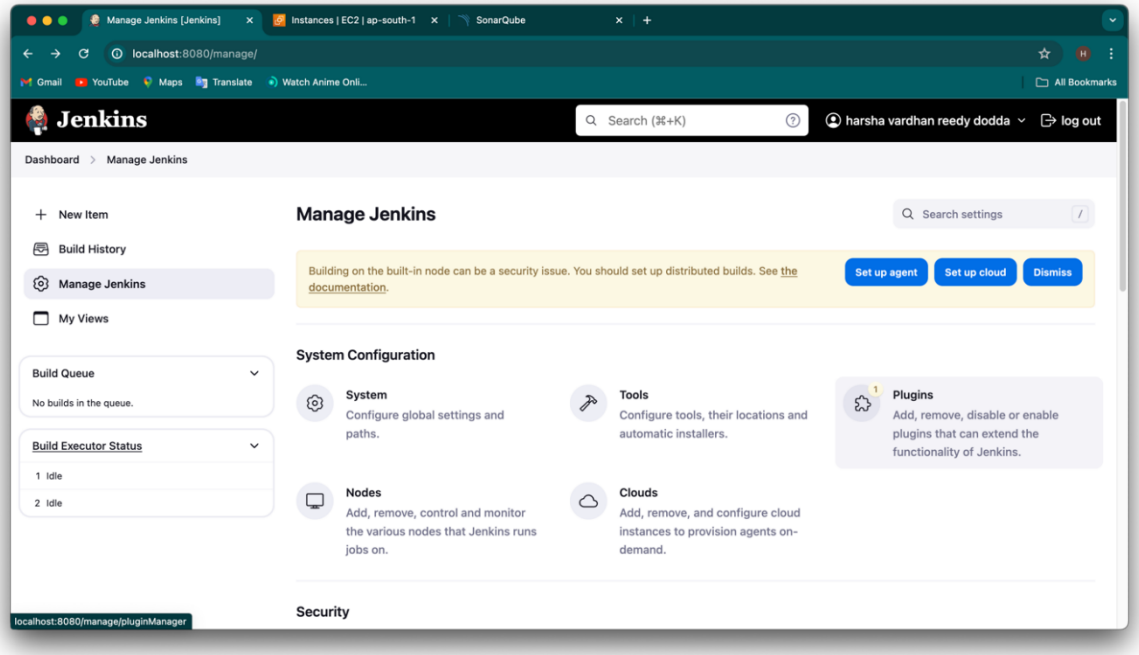
Step 2: Install SonarQube Plugin in Jenkins

Install the SonarQube Plugin:

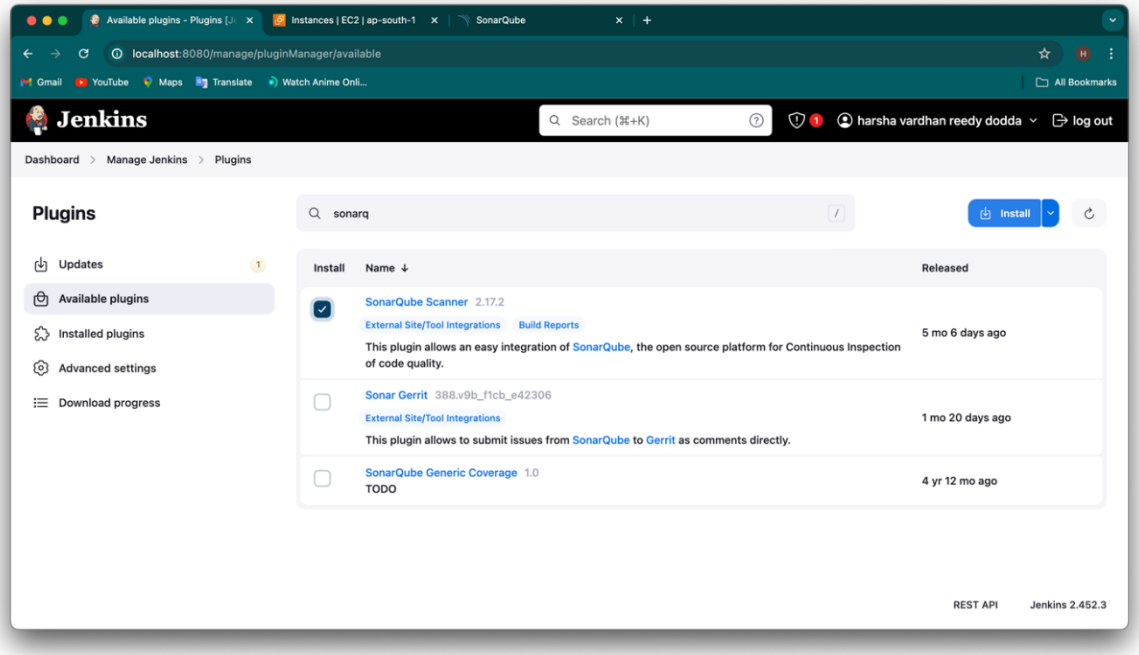
→ **Go to the Jenkins Dashboard:** Open your Jenkins instance in a web browser.
→ **Click on "Manage Jenkins":** This will take you to the Jenkins management page.



→ **Click on "Plugins":** Navigate to the plugin management section.



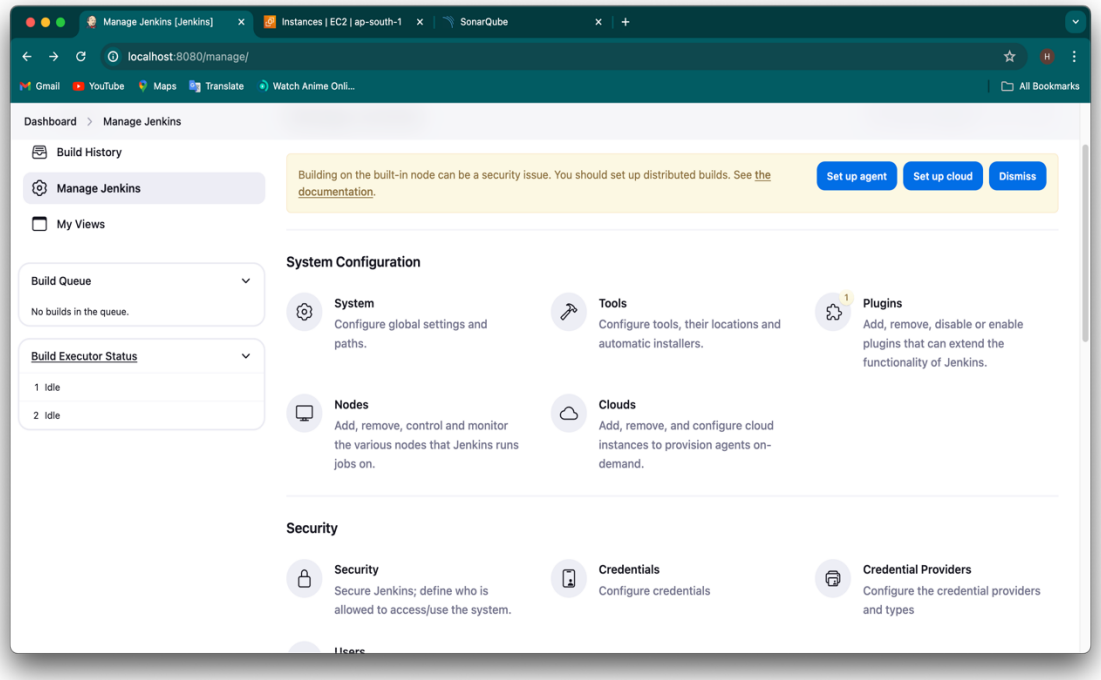
→ **Go to the "Available" tab:** Search for Sonarqube scanner plugin.



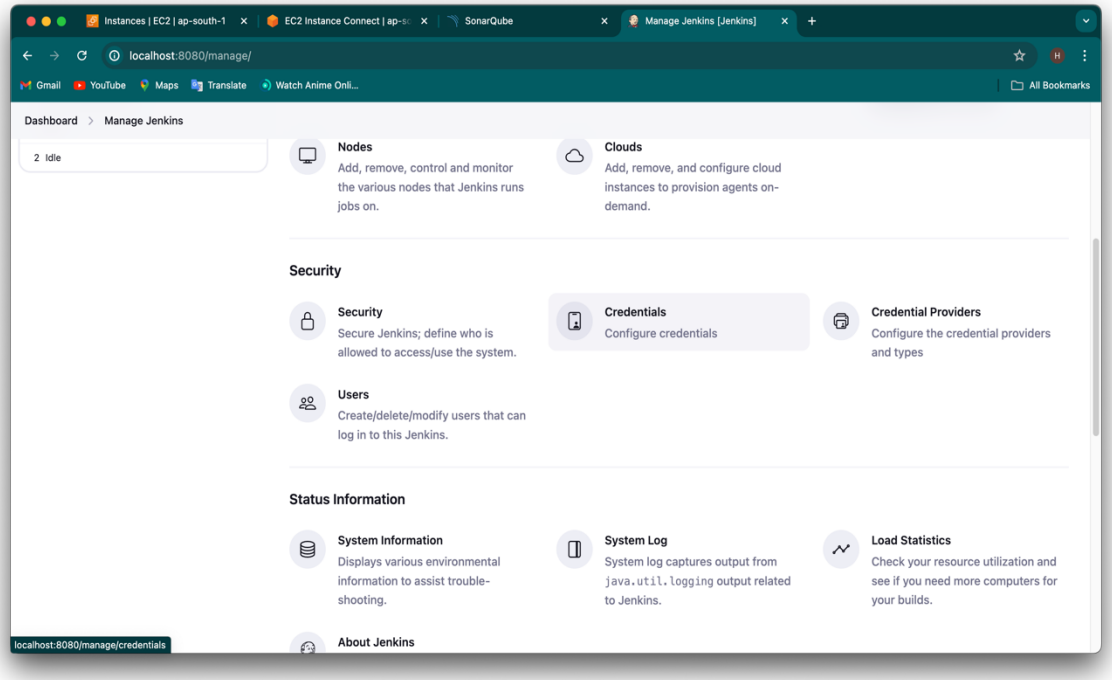
→ **Restart Jenkins if required:** Some plugin installations require a Jenkins restart to take effect.

Step 3: Configure SonarQube in Jenkins

→ **Click on "Manage Jenkins":** This will take you to the Jenkins management page.

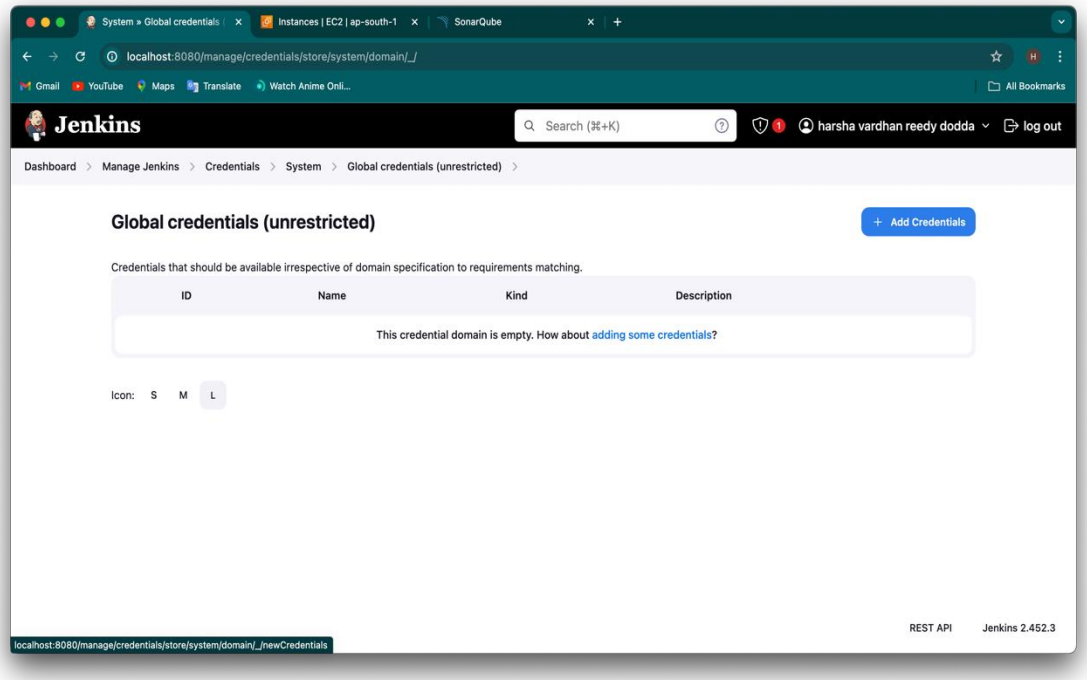


→ **Click on "credentials":** Navigate to the credentials.

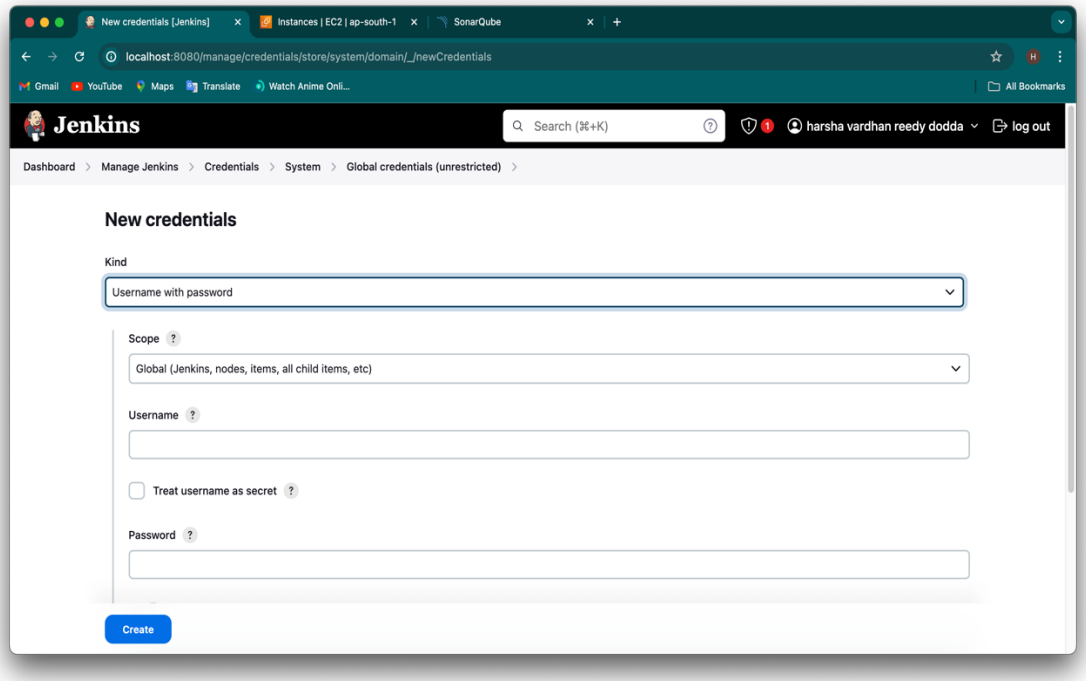


→ **Scroll down to the "Stores scoped to jenkins" section:** Here click on global to add global credentials.

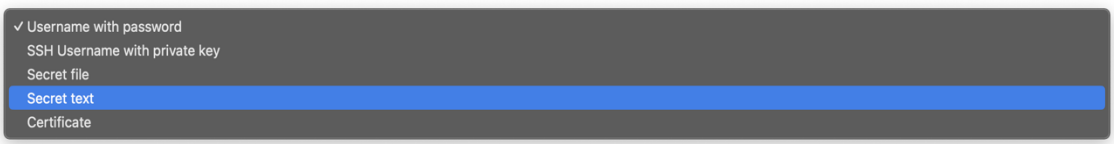
→ Later on that page , **Click "Add Credentials":** Here you can add the id of the token you have generated .

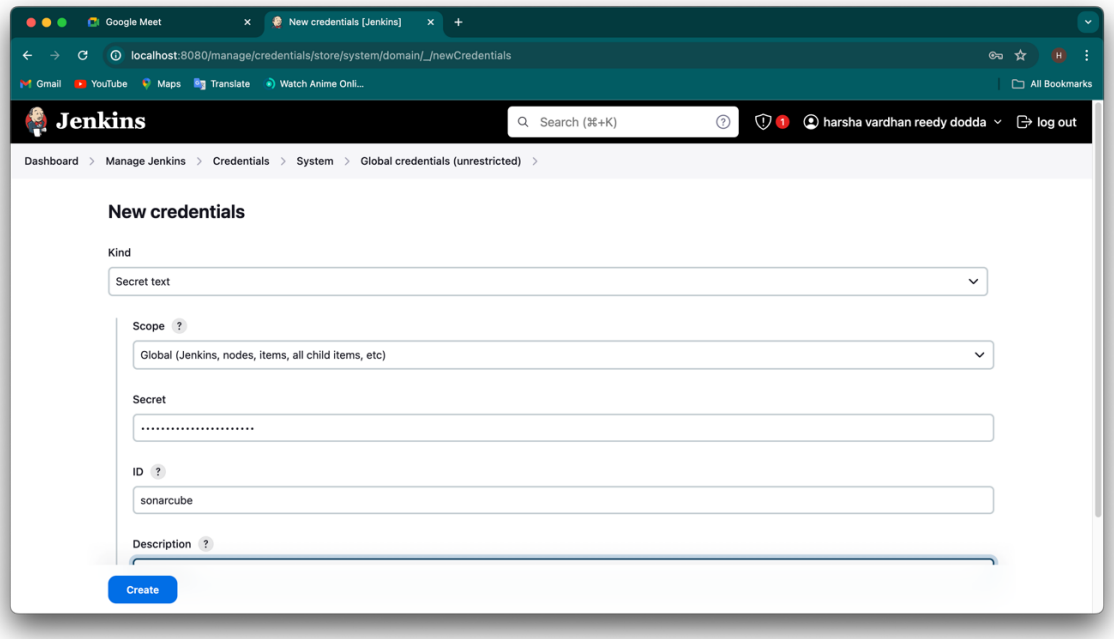


→ Scroll down to ‘new credentials’ section and hover near KIND (where you can add the secret text).

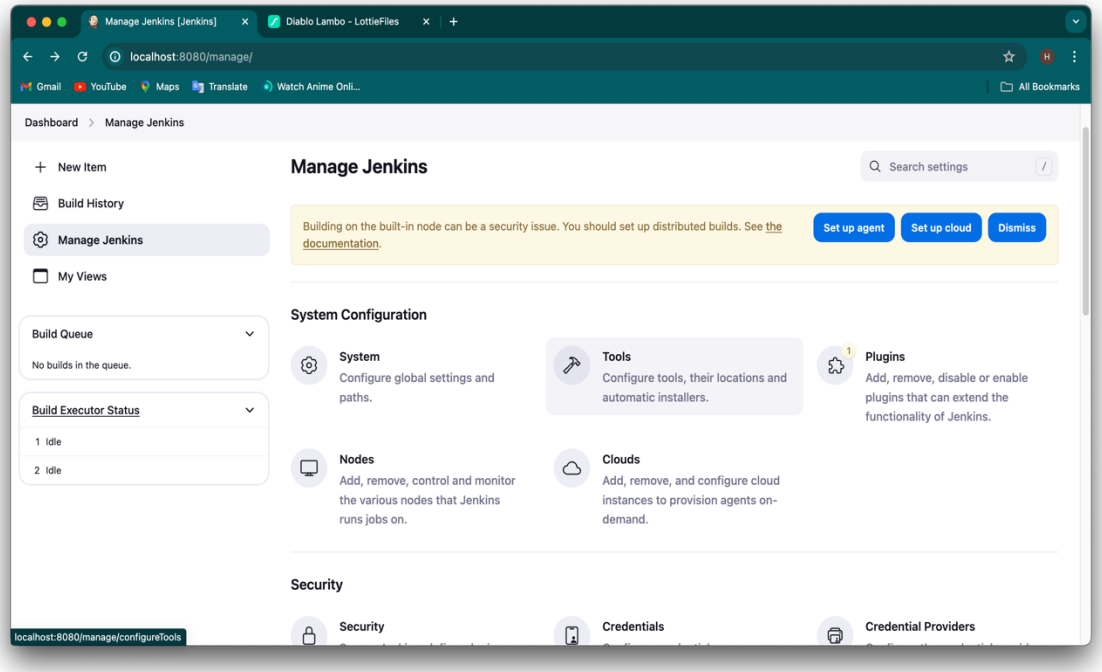


→ Select the Secret Text





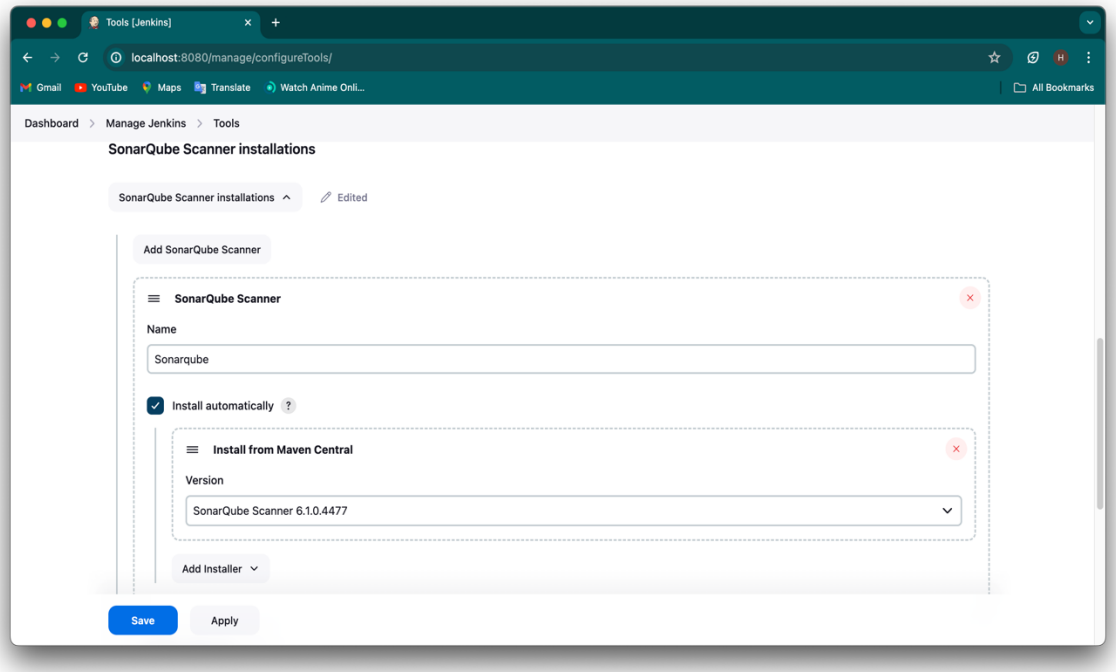
→ Click On **Tools**



→ **Scroll down to the "Sonarqube scanner installations"**

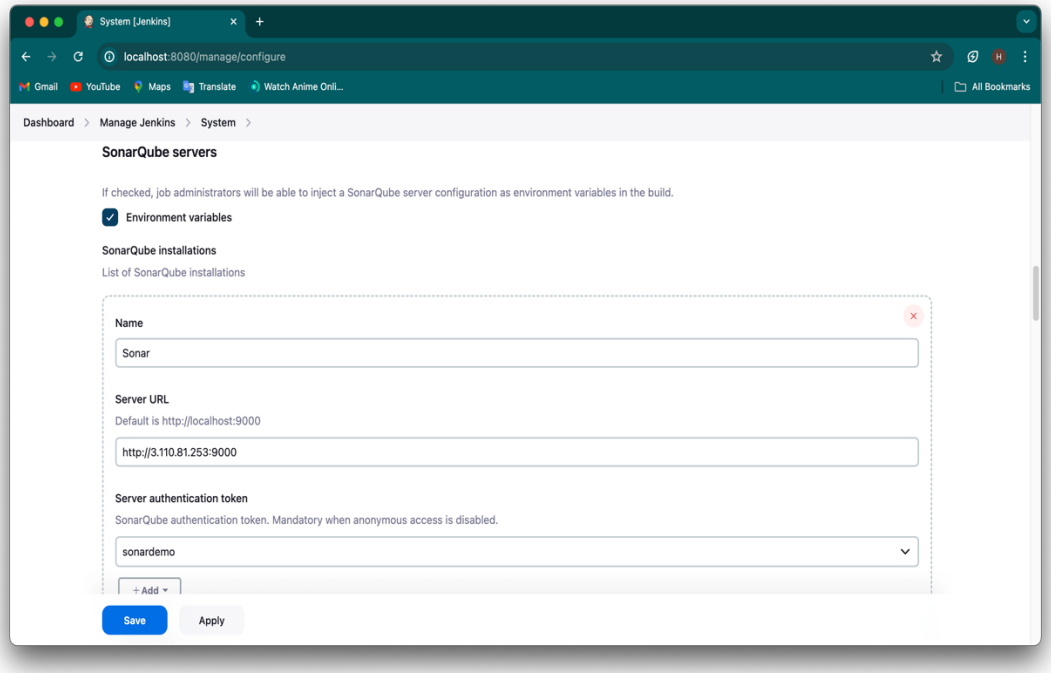
→ **Now,** Fill the name with installed plugin name i.e, Sonarqube and select the specific version of the plugin then click on '**Save'**.

→ Now the SonarQube Scanner has been Setted Up Succesfully for using in projects and pipelines etc.



Configure SonarQube Scanner:

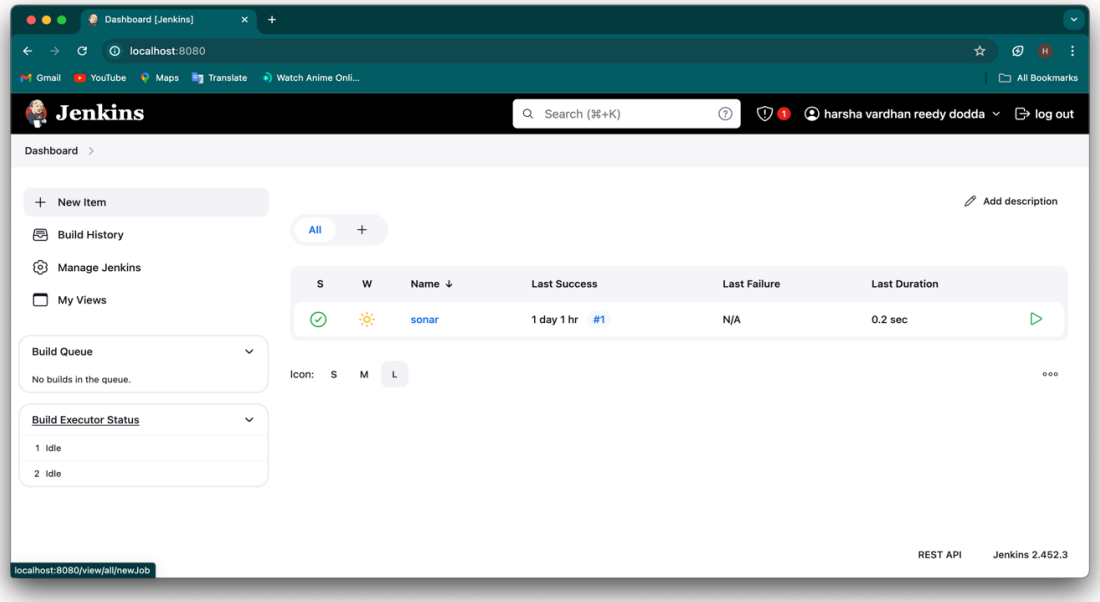
1. **Go to the Jenkins Dashboard:** Open your Jenkins instance.
2. **Click on "Manage Jenkins": This will take you to the Jenkins management page.**
3. **Click on "System":**
4. **Scroll down to the "SonarQube Server" section:** Fill the details and save



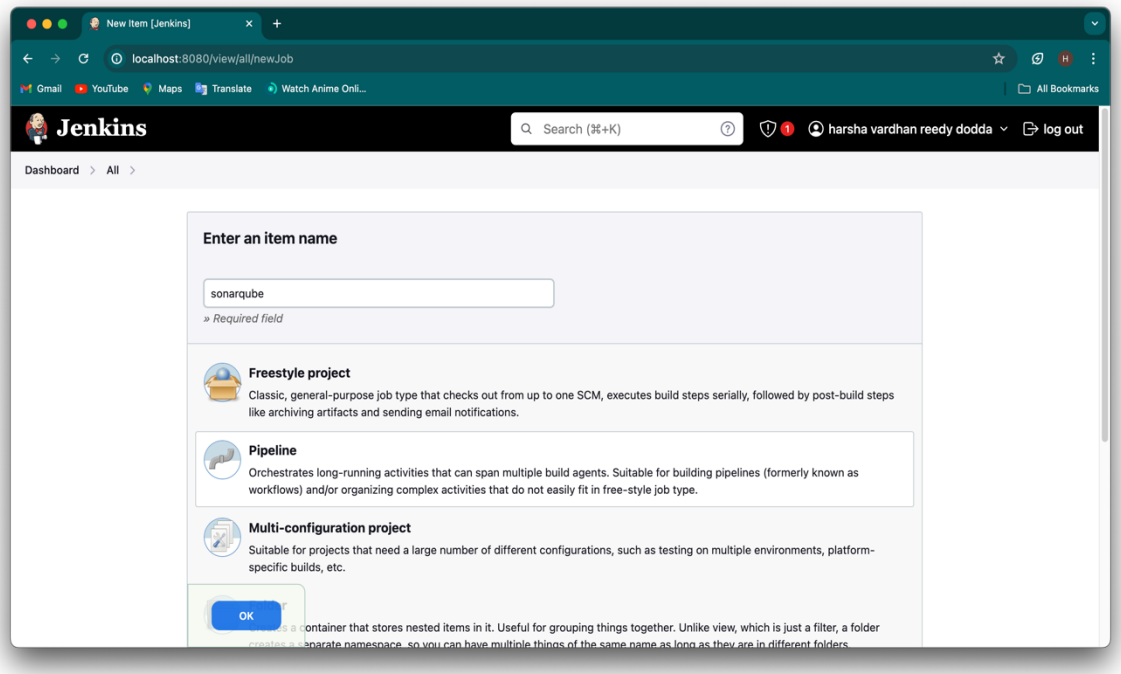
Step 4: Create a Jenkins Pipeline for SonarQube Analysis

Create a New Pipeline Job:

→ Go to the Jenkins Dashboard: Open your Jenkins instance.
Click "New Item": Create a new Jenkins item.



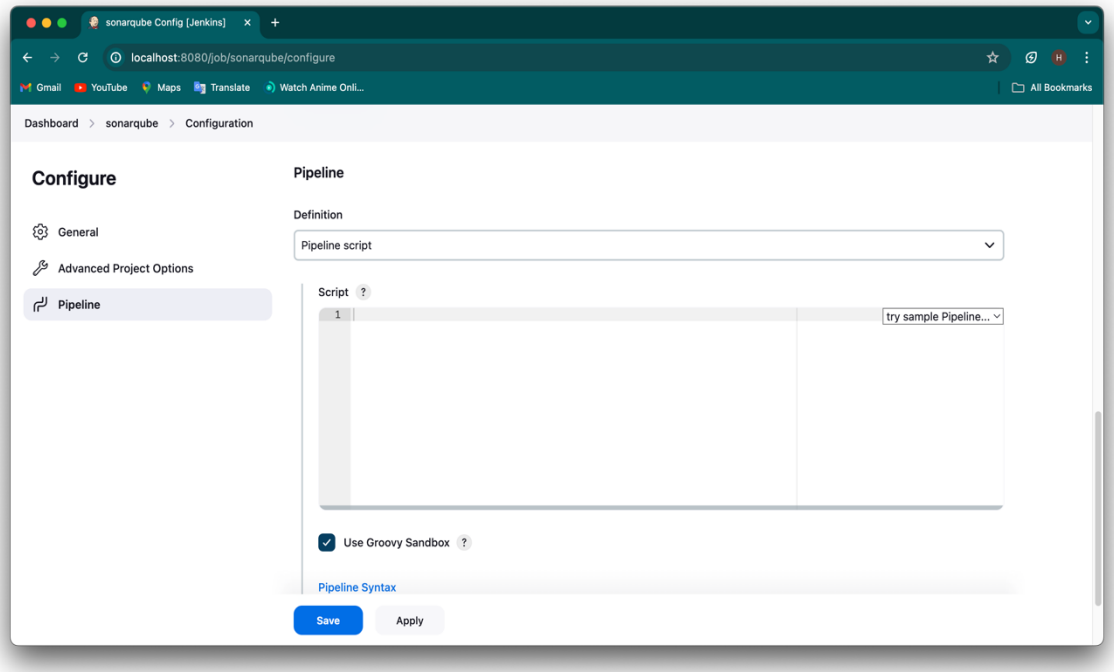
→ Enter a name for your job and select "Pipeline": Provide a name for your pipeline job and select the pipeline option.



→ Click "OK": Create the pipeline job.

Configure the Pipeline Script:

→ **In the pipeline configuration page, scroll down to the "Pipeline" section:** Configure the pipeline script.



→ **Select "Pipeline script" and enter your pipeline script:** Use the following example script:

```
pipeline {
    agent any
    stages {
        stage('Fetch Code') {
            steps {
                git "https://github.com/harsha5969/web-app"
            }
        }
        stage('Code Analysis') {
            environment {
                scannerHome = tool name: 'Sonarqube', type:
'hudson.plugins.sonar.SonarRunnerInstallation'
            }
            steps {
                script {
                    withSonarQubeEnv('Sonar') {
                        sh """
                            ${scannerHome}/bin/sonar-scanner \
                            -Dsonar.projectKey=test-sonar \
                            -Dsonar.projectName=test-sonar \
                            -Dsonar.projectVersion=1.0 \
                            -Dsonar.sources=.
                        """
                    }
                }
            }
        }
```

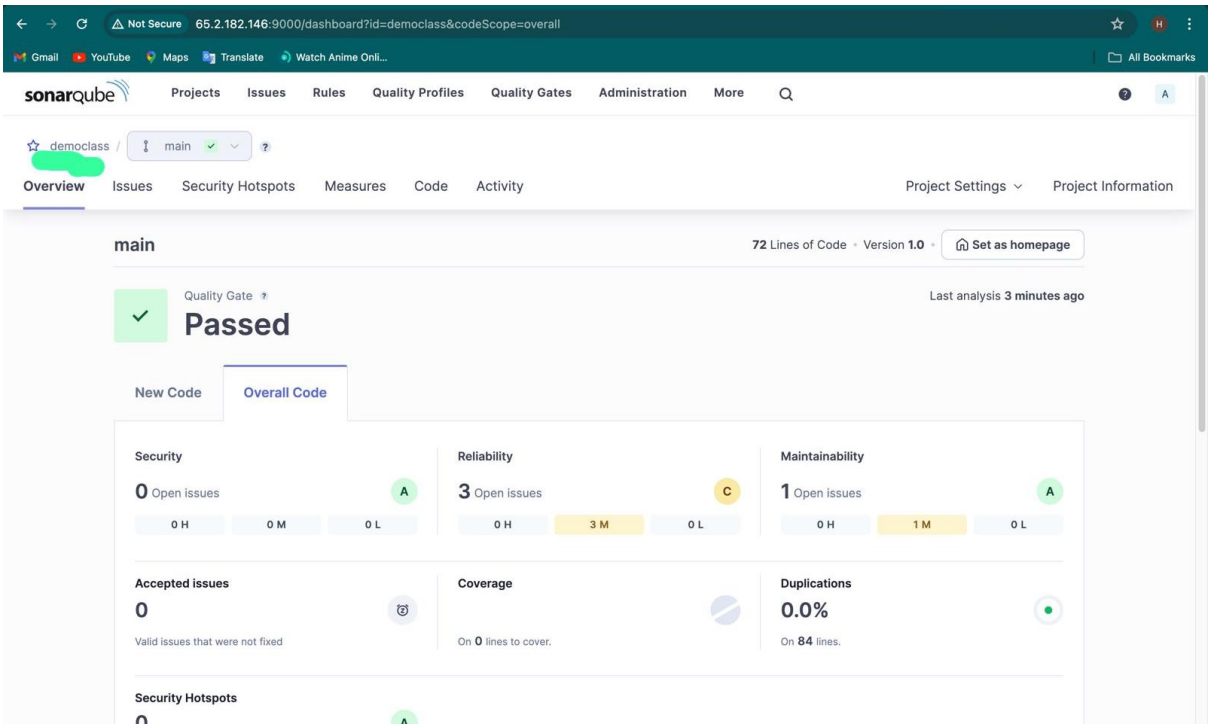
```
    }
}
```

Step 5: Run the Jenkins Job
Build the Pipeline:

1. Save the pipeline configuration: Ensure your configurations are saved.

2. Click "Build Now" to run the pipeline: Start the pipeline job.

3. Monitor the build log to ensure the SonarQube analysis runs successfully: Check the build logs to confirm successful execution.



13. Demonstrate Docker Swam Commands by deploying a Multi-Container Flask Server .

Docker Swarm:

Docker Swarm is a native clustering and orchestration tool for Docker containers, allowing you to manage a group of Docker hosts as a single virtual system. It simplifies the deployment, scaling, and management of containerized applications by enabling you to define and control a cluster of Docker engines, distribute workloads across different nodes, and ensure high availability and fault tolerance. With Docker Swarm, you can easily create, deploy, and manage multi-container applications using a declarative service model.

Prerequisites:

→ Make sure you installed both docker and docker-compose using the previous experiments.

→ Ensure that the docker engine is running perfectly and you have Good Internet Connection to pull larger Images.

Using Docker Swarm:

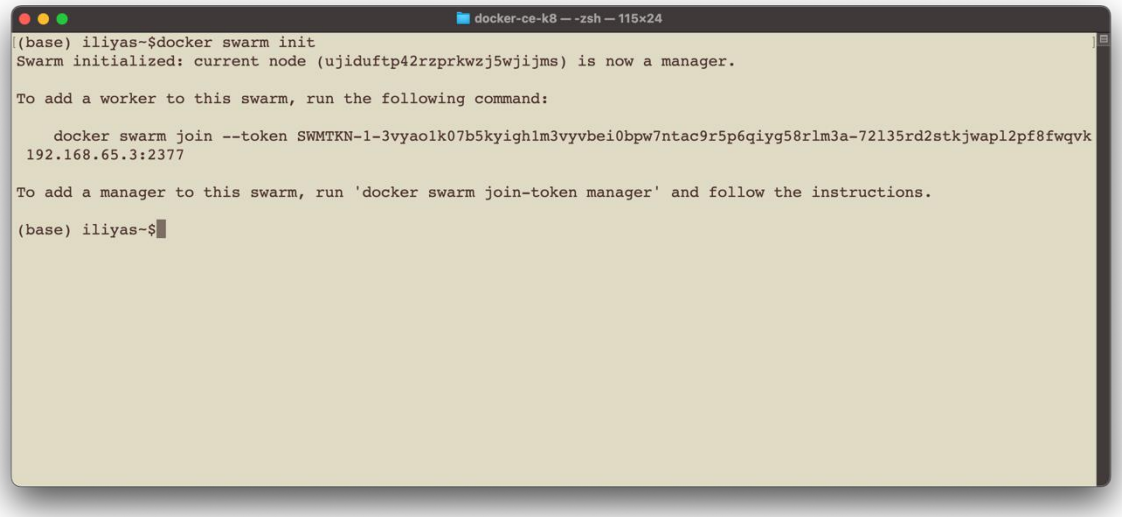
→ Firstly , Clone the git repository and cd into it using the following commands.

~$ git clone https://github.com/Iliyaz-Syed/docker-ce-k8

~$ cd docker-ce-k8

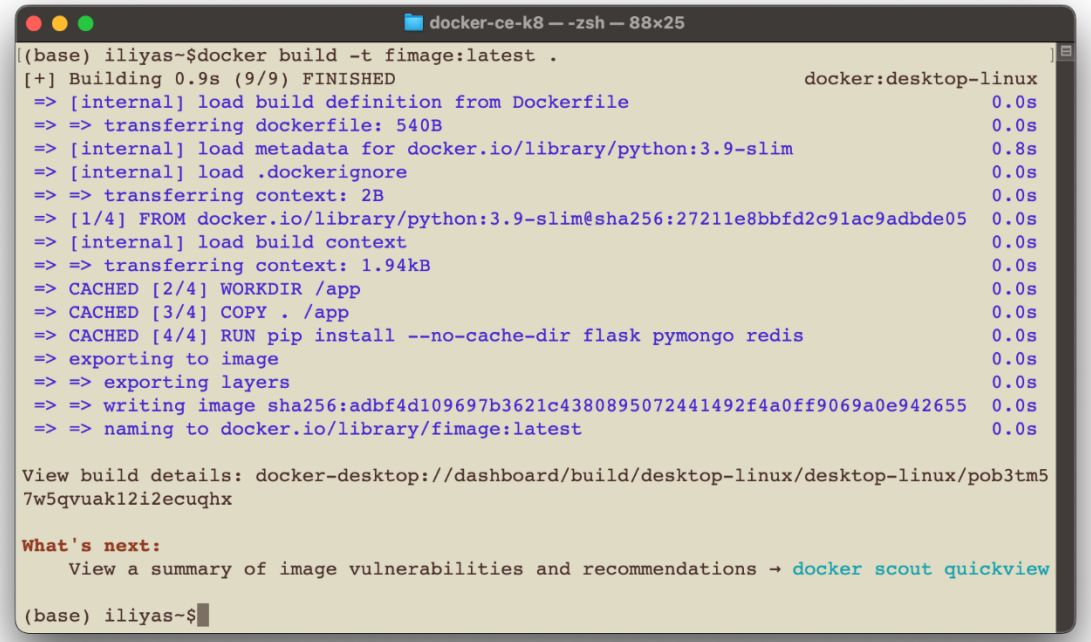
→ Now, Initialize the docker engine into swarm mode. Using

~$ docker swarm init



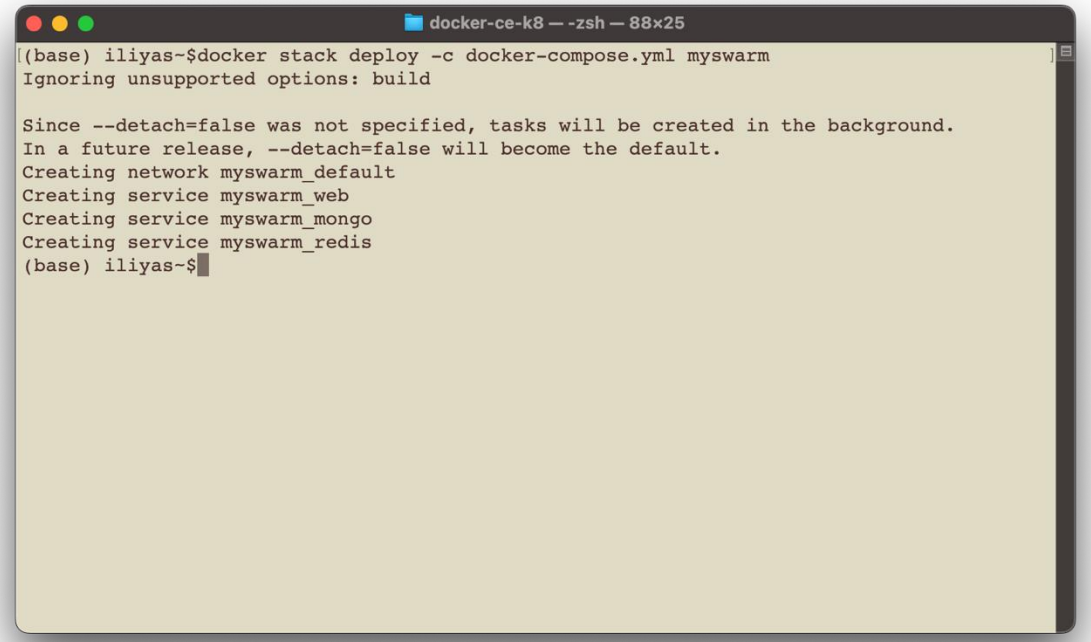
→ Now build the image in order to start the containers using the Docker swarm using the command.

~$ docker build -t fimage:latest .



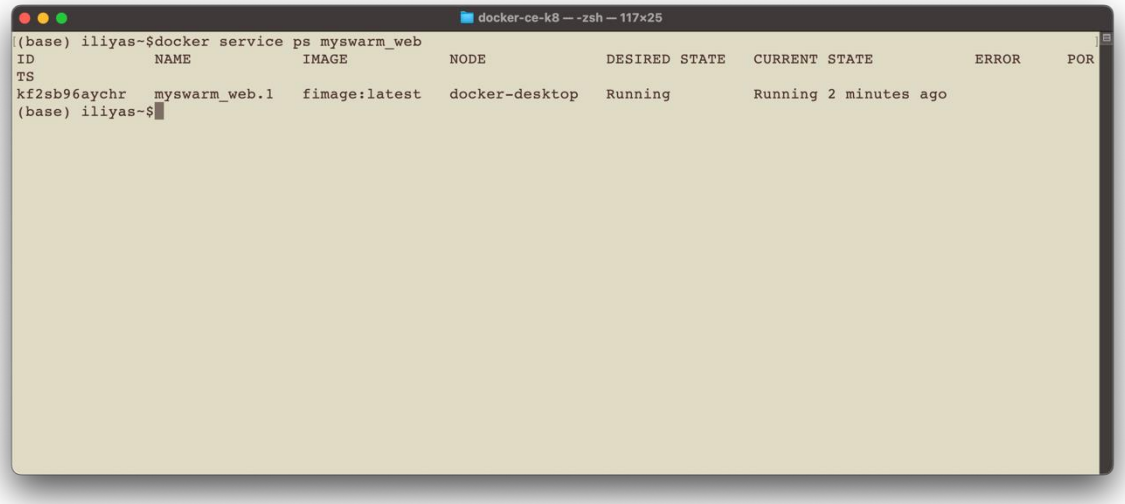
→ Now the image is built and we can deploy the whole stack that is written in docker-compose.yml file containing flaks,redis and mongo using the following command.

~$ docker stack deploy -c docker-compose.yml myswarm



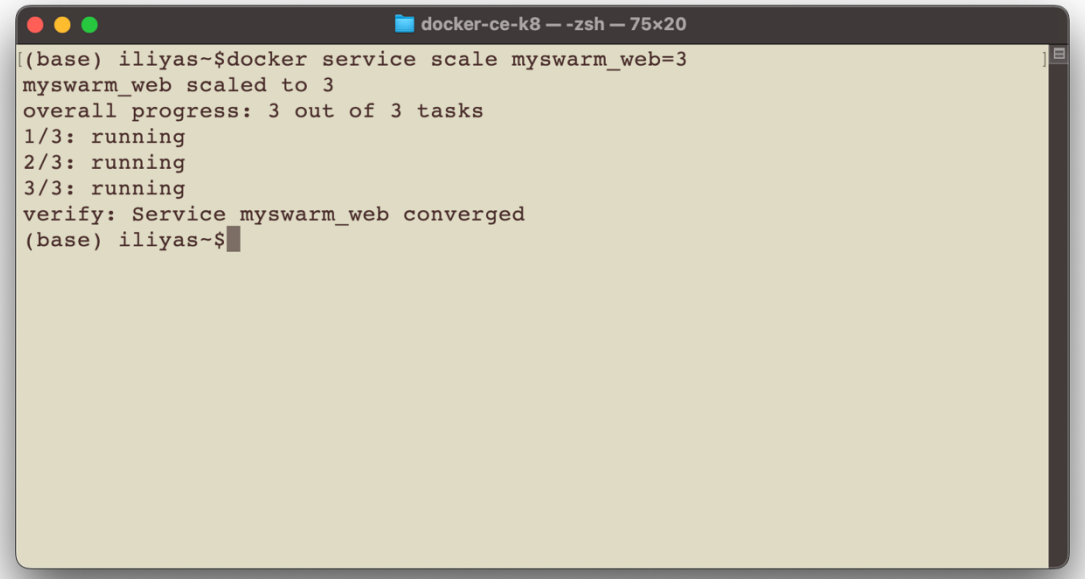
→ Now to monitor the running swarm nodes you can use the following command.

~$ docker service ps myswarm_web



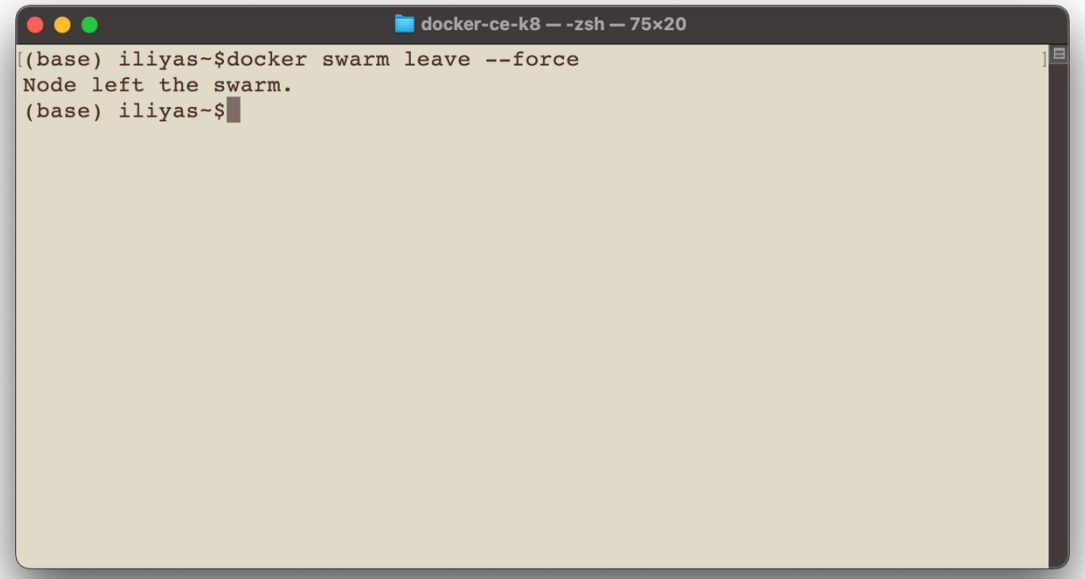
→ Since this is a Orch tool we can scale the application to desired number of replicas using the following command.

~$ docker service scale myswarm_web=3



→ Now for exiting the swarm mode and stop the applications we can use the following command.

~$ docker swarm leave –force

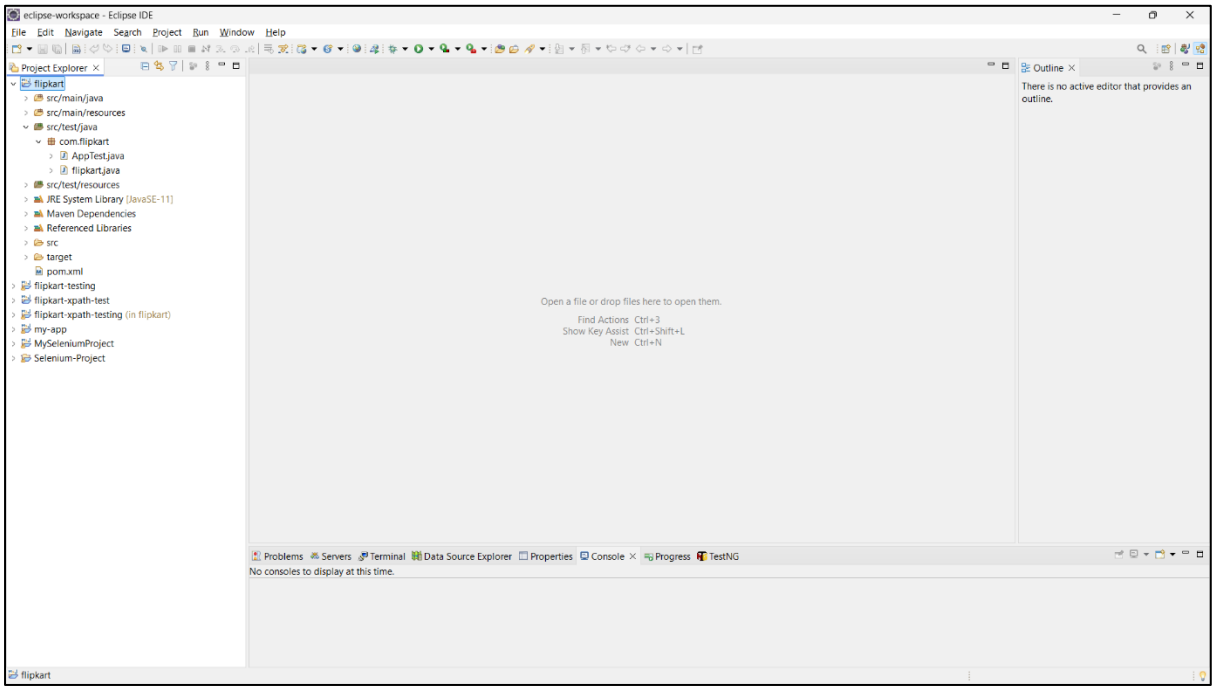


14. Demonstration of XPath using Java Maven Project using Flipkart website.

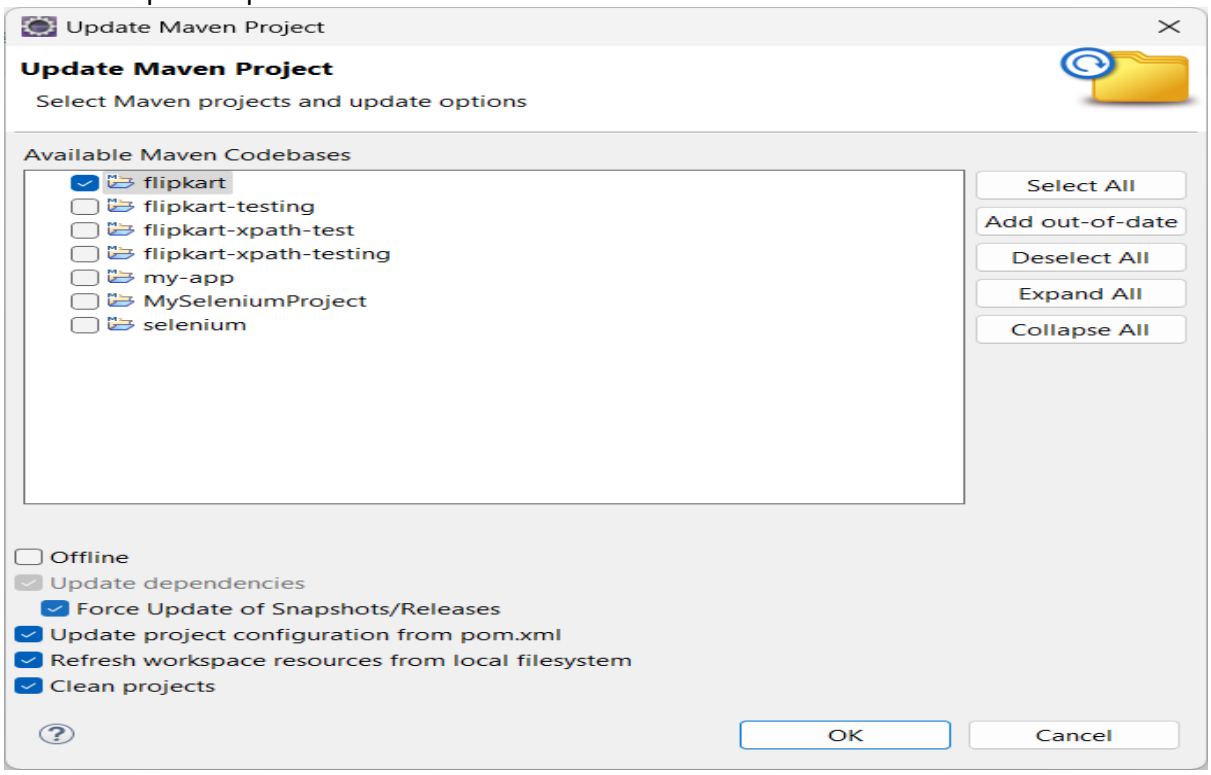
→ Initially clone the given repo using the following command.

~$ git clone https://github.com/UDAYKOTESWARARAOCHAKRAPU/flipkart-xpath.git

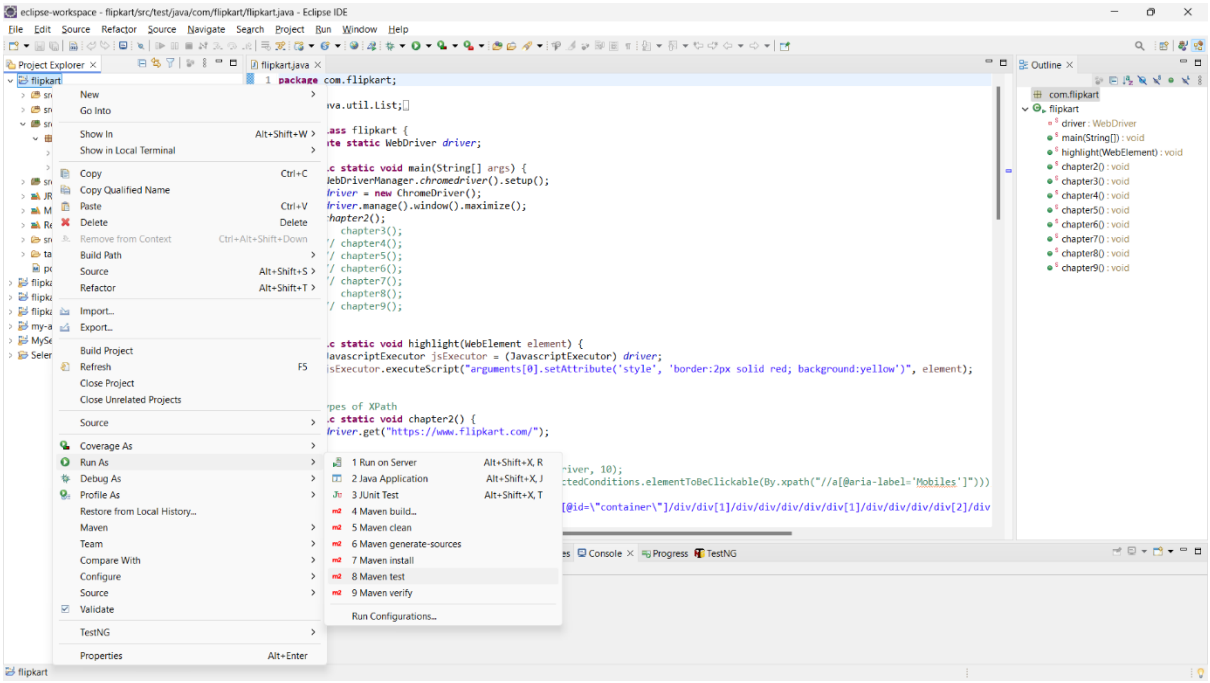
→ Now open the repo using Eclipse IDE Java Enterprise Edition.
→ The Home Page should Look like this after opening the project.



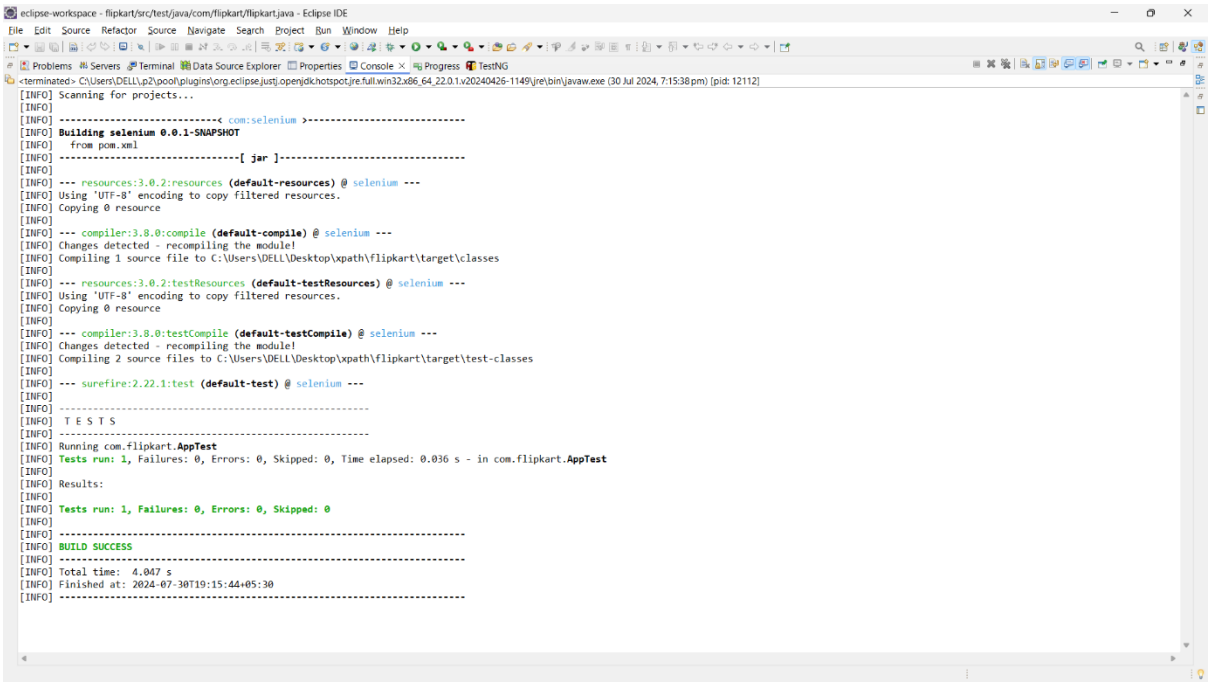
→ Now click on Project in Top bar and click on **Update Maven Project.** Make sure to check the force update option. And click on **ok**



→ Now right click on flipkart and choose **Run As** Followed by 7th option **Maven Test.** The output in the bottom console comes as **BUILD SUCCESS**

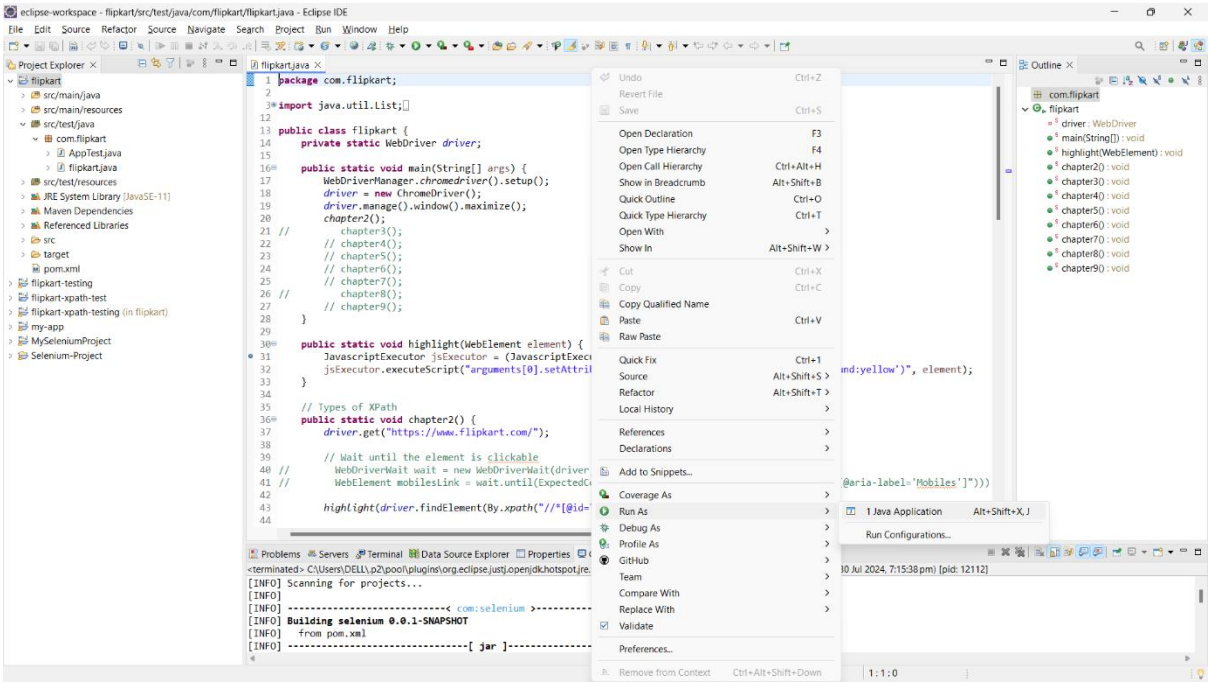


→ Here the build is successful and now we can proceed with running the application.



→ To run the application open the following file
flipkart>src/test/java>com.flipkart>flipkart.java

→ Then right click inside the file or program and click on **Run As** and Then Click on **Java Application** to run the program.



→ The Output should be as follows where Mobile Section is Highlighted.