

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

Code:

```
<!DOCTYPE html>
<html>

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

<body>
    <center class="bg">
        <h1>Registration Page</h1>
        <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.



Registration Page

First Name

Date Of Birth dd-mm-yyyy

Email

Gender Male female

Phone No

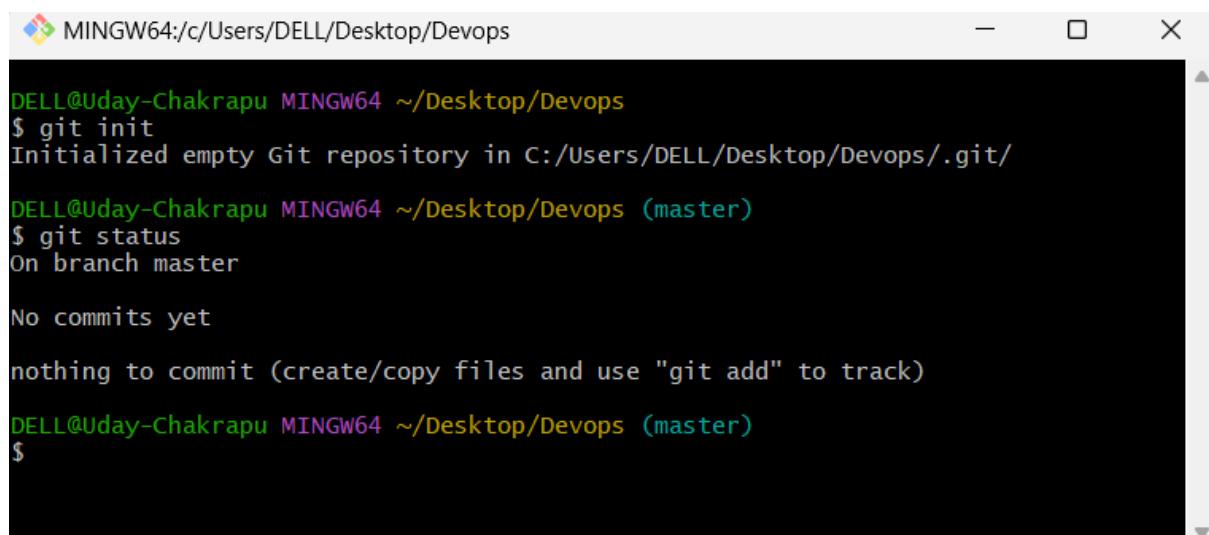
Address

Accept terms and conditions

Branch

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**



```
MINGW64:/c/Users/DELL/Desktop/Devops
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops
$ git init
Initialized empty Git repository in c:/Users/DELL/Desktop/Devops/.git/
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

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

- 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

```

<!DOCTYPE html>
<html>
<head>
<title>Registration Form</title>
<link type = "stylesheet" href="styles.css">
<style>
body {
background-color:red;
}
.bg {
background-color:green;
}
</style>
<body>
<center class="bg">
<h1>Registration Page</h1>
<form action="index.php" 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" value="Male" /><input type="radio" name="Gender" value="Female" />
<br>
<label> Phone No</label>
<input type="tel" name="Phno"><br>
<label>Address</label>
<input type="text" name="addde" rows=5 cols=5><br>
<input type="checkbox" checked="" value="Accept terms and conditions" />
<br>
<label>Branch</label>
<select name="Ops">
<option value="CSE" >CSE </option>
<option value="66 >cse" >66 >cse</option>
<option value="0275 EEE" >0275 EEE </option>
<option value="042 ECE" >042 ECE </option>
</select>
<br>
<button type="submit">Submit </button>
<button type="reset"> Reset</button>
</form>
</center>
</body>
</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/UDAYKOTESWARARAOCRAKRAPU/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
- Create a new repository "source_code_management"

The screenshot shows the GitHub Home page. On the left, there's a sidebar with 'Top repositories' and a search bar. The main area has a large 'Start a new repository' card for 'UDAYKOTESWARARAOCRAKAPU'. It asks to 'Introduce yourself with a profile README' and provides fields for 'Repository name' (with placeholder 'name your new repository...'), 'Public' (selected), and 'Private'. Below these are sections for 'Use tools of the trade' (github.dev editor) and 'Get AI-based coding suggestions' (GitHub Copilot). A 'UNIVERSE'24' promotional banner is visible on the right.

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

The terminal window shows the command `git clone https://github.com/UDAYKOTESWARARAOCRAKAPU/source_code_management.git` being run. The output indicates that the repository was cloned successfully, though it is noted as being empty. The user then moves into the repository directory with `cd source_code_management`.

```
MINGW64:/c/Users/DELL/Desktop/devops-demo/source_code_managment
DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo
$ ls
DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo
$ git clone https://github.com/UDAYKOTESWARARAOCRAKAPU/source_code_management.git
Cloning into 'source_code_management'...
warning: You appear to have cloned an empty repository.

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

- 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_magment (main)
$ touch Registration.html

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

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

```
<!DOCTYPE html>
<html>
<head>
<title>Registration Form</title>
<link type = "stylesheet" href="styles.css">
<style>
body {background-color:red;}
body {background-color:green; }
</style>
</head>
<body>
<center class="bg">
<h1>Registration Page</h1>
<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="number" name: "Phno"></input>
<br>
<label>Address</label>
<input type="text" rows=5 cols=5 name="adde"></input>
<br>
<input type="checkbox"> Accept terms and conditions
<br>
<label>Branch</label>
<select name="ops">
<option value=0>CSE </option>
<option value=60>EEE</option>
<option value=27> EEE </option>
<option value=04> 042 ECE </option>
</select>
<br>
<button type="submit">Submit </button>
<button type="reset"> Reset</button>
</form>
</center>
</body>
</html>
```

Registration.html [unix] (15:23 21/07/2024) 7,1 All

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

```
DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_magment (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_magment (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.
- Committing 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**

```

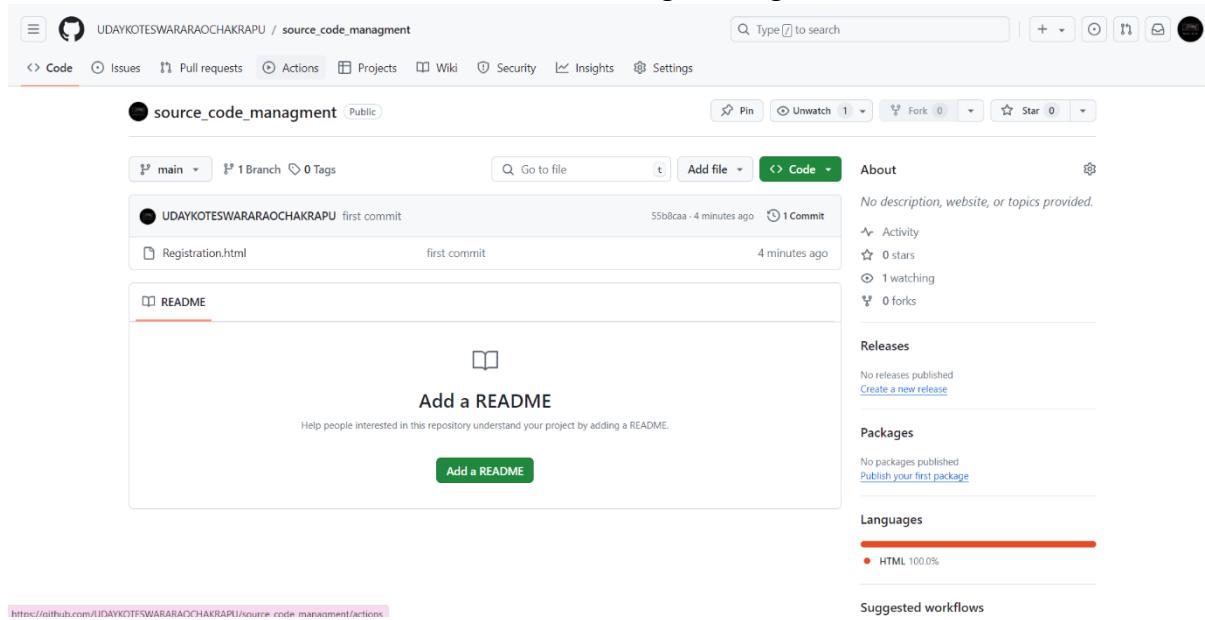
DELL@Uday-Chakrapu MINGW64 ~/Desktop/devops-demo/source_code_managment (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 776 bytes | 388.00 KiB/s, done.
Total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

jenkins.io/download/

Jenkins [cd](#) ▾

Blog Success Stories Contributor Spotlight Documentation ▾ Plugins Community ▾ Subprojects ▾ Security ▾ About ▾ [Download](#) Search [Search](#)

Downloading Jenkins

Jenkins is distributed as WAR files, native packages, installers, and Docker images. Follow these installation steps:

1. Before downloading, please take a moment to review the [Hardware and Software requirements](#) section of the User Handbook.
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the [Installing Jenkins](#) section of the User Handbook.
4. You may also want to verify the package you downloaded. [Learn more about verifying Jenkins downloads.](#)

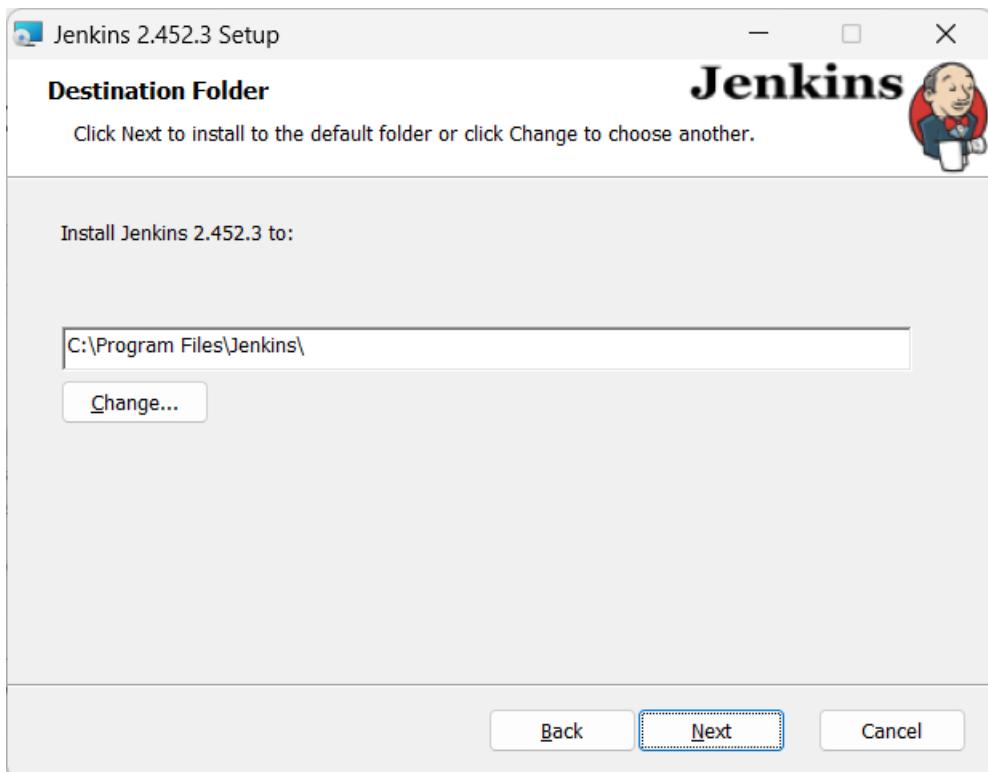
<p>Download Jenkins 2.452.3 LTS for:</p> <ul style="list-style-type: none">  Generic Java package (.war) SHA-256: 45fd2b877f9709a52b984d9c7d6f435bc05f6adee61291d22d30b5fe8fd8c59  Docker  Kubernetes  Ubuntu/Debian  Red Hat/Fedora/Alma/Rocky/CentOS  Windows  openSUSE 	<p>Download Jenkins 2.468 for:</p> <ul style="list-style-type: none">  Generic Java package (.war) SHA-256: d0f6dfc74dd2baaf5ad17d8654d457197af500ab33420bcc6d4d94ff80beac571  Docker  Ubuntu/Debian  Red Hat/Fedora/Alma/Rocky/CentOS  Windows  openSUSE  Arch Linux Third party
---	--

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

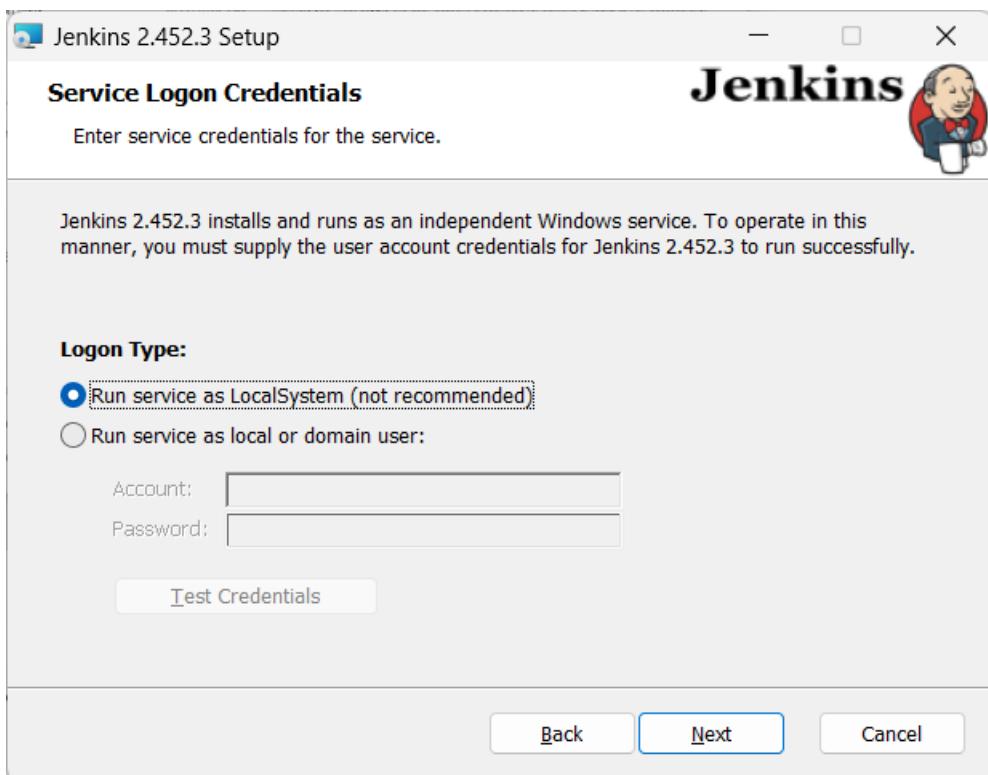
Step 3 : The setup wizard starts. Click Next to proceed.



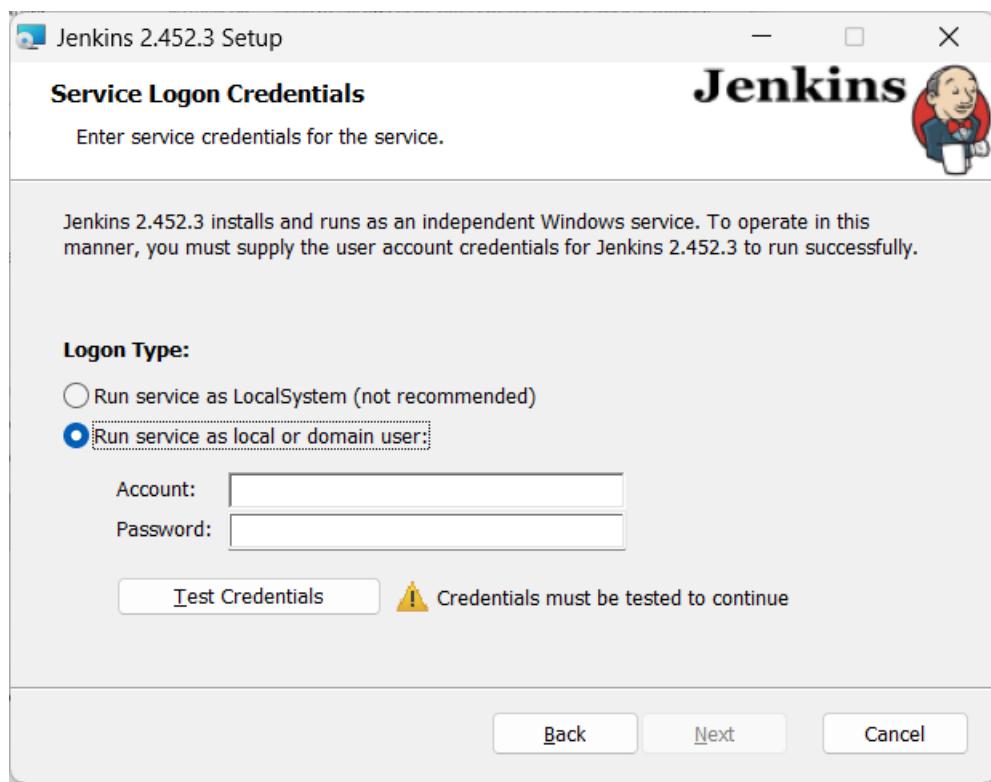
Step 4 : Select the install destination folder and click **Next** to continue.



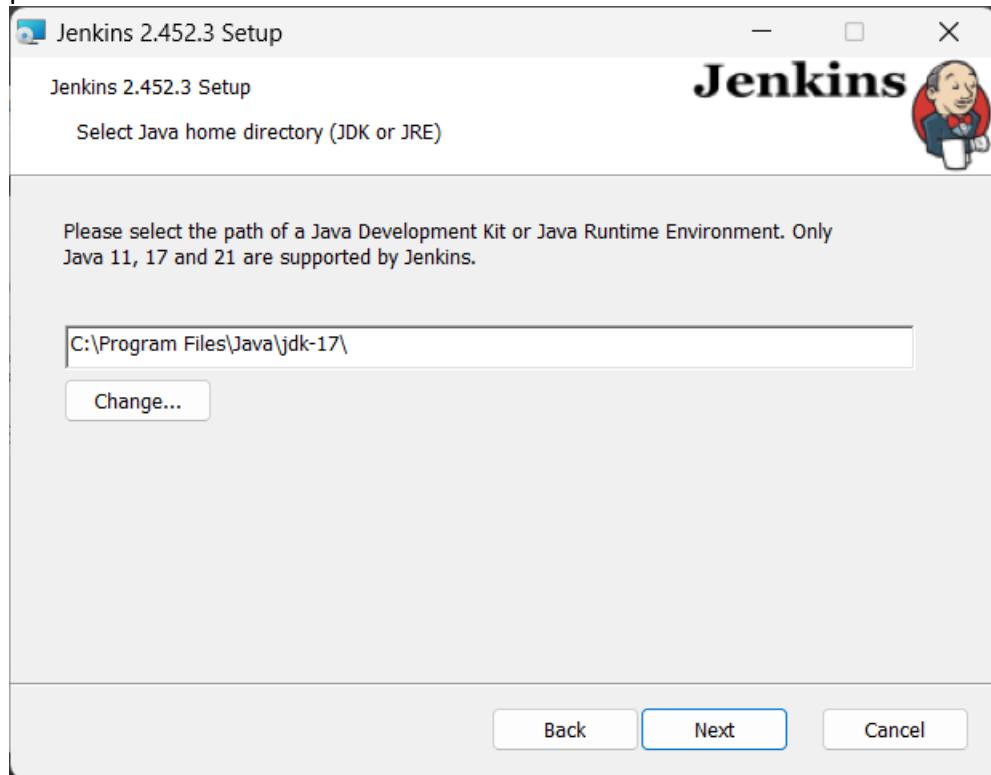
Step 5 : click on **Run service as a localsystem** option, and click **NEXT** to continue



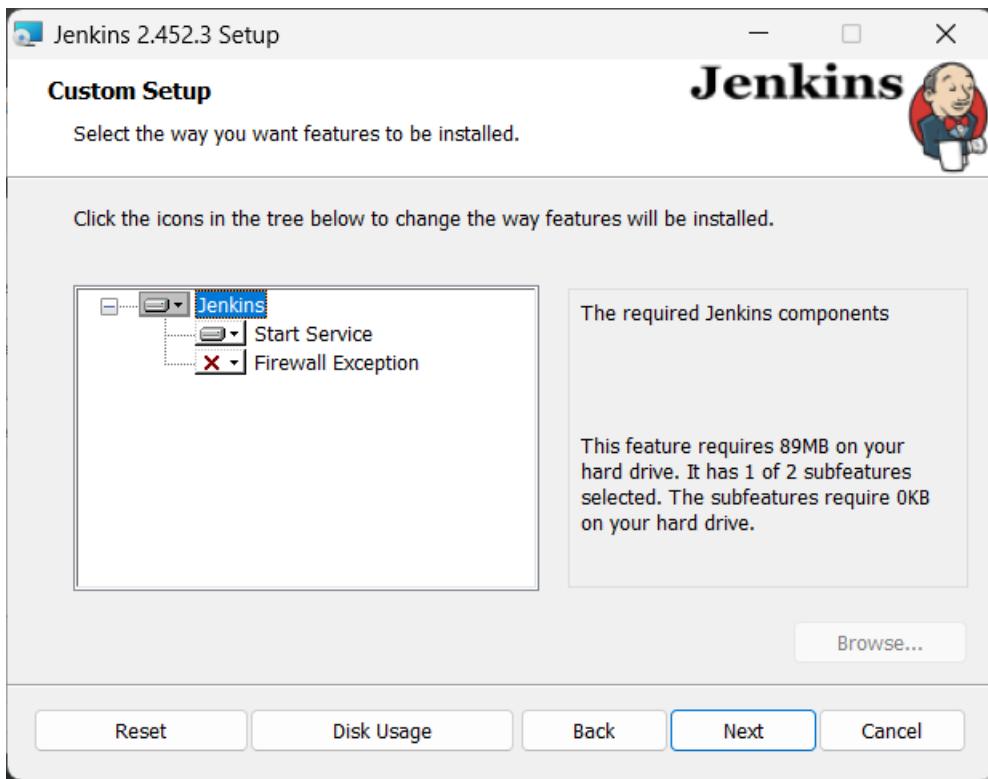
Step 6 : Enter the port number you want Jenkins to run on. Click **Test Port** to check if the selected port is available, then click **Next** to continue.



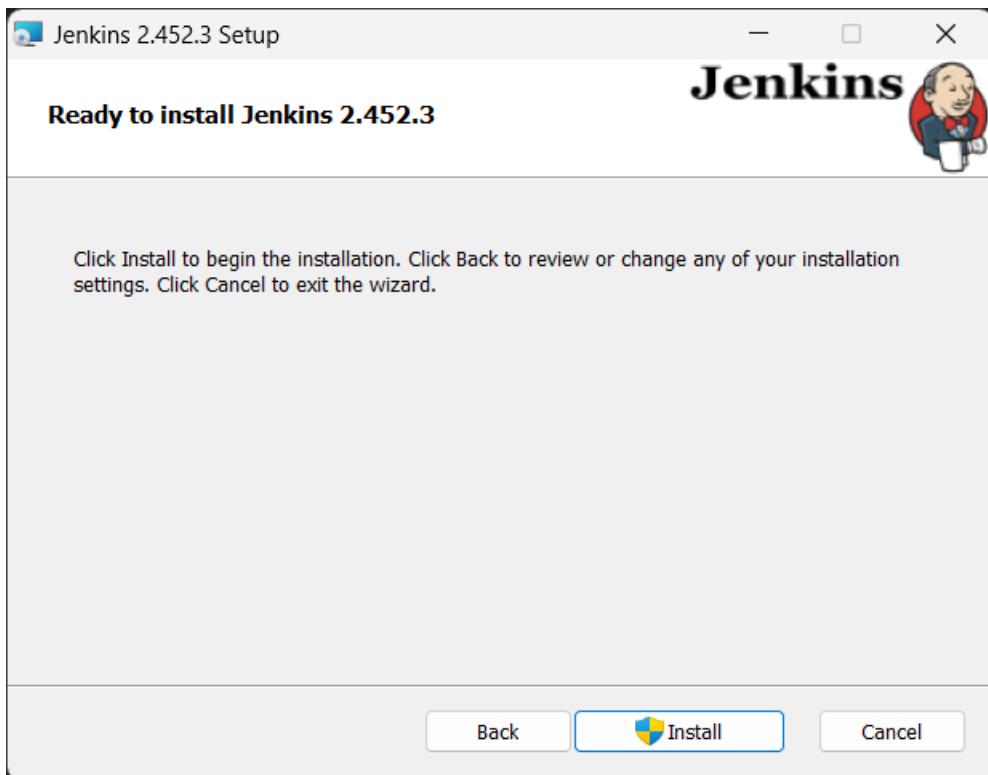
Step 7 : Select the directory where [Java is installed](#) on your system and click **Next** to proceed.



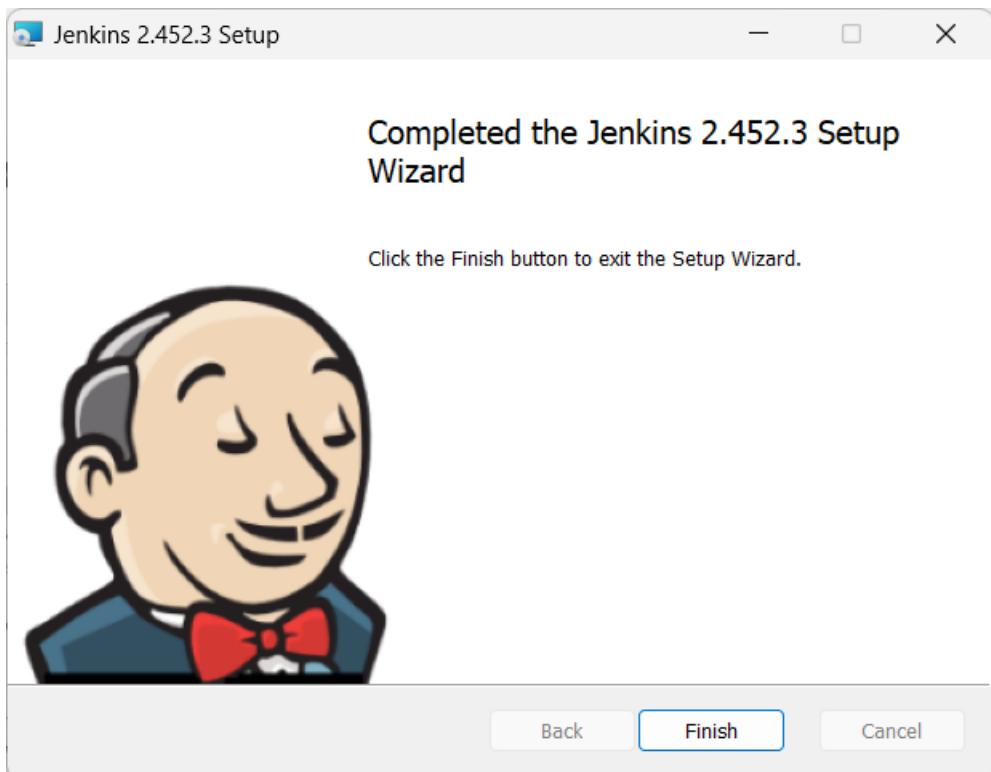
Step 8 : Select the features you want to install with Jenkins and click **Next** to continue



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.

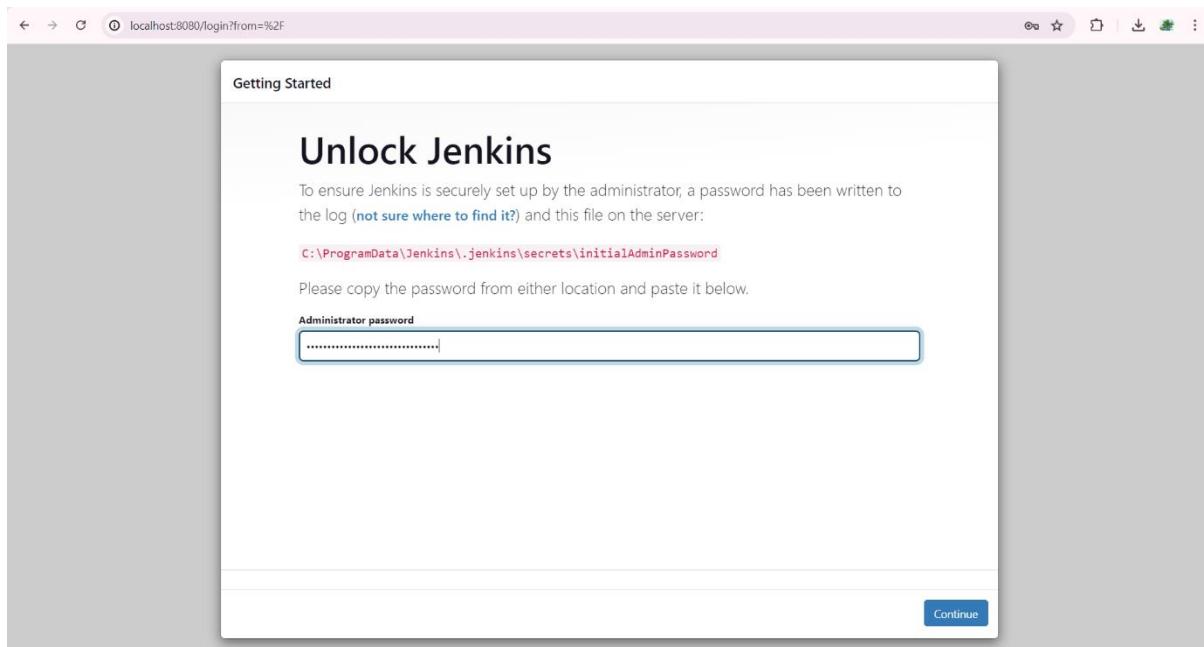
C:\ProgramData\Jenkins\.jenkins\secrets\initial admin. Password

Administrator password :

A screenshot of a "Unlock Jenkins" dialog box. The title bar says "Getting Started". The main content area has the heading "Unlock Jenkins". It explains that Jenkins is securely set up and a password has been written to the log and a file at "C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword". It instructs the user to copy the password from either location and paste it into the "Administrator password" input field. A "Continue" button is at the bottom right.

Step 3 : Open the **initialAdminPassword** file using a text editor such as Notepad.

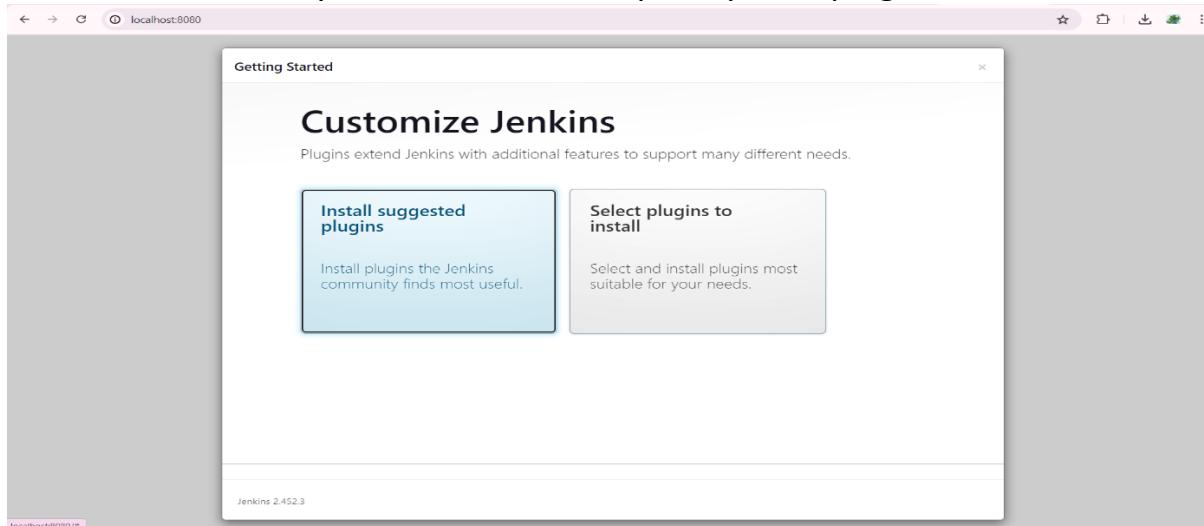
Step 4 : Copy the password from the **initialAdminPassword** file. And paste it in the “administrator password” field on the unblock Jenkins page and click on continue to proceed



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



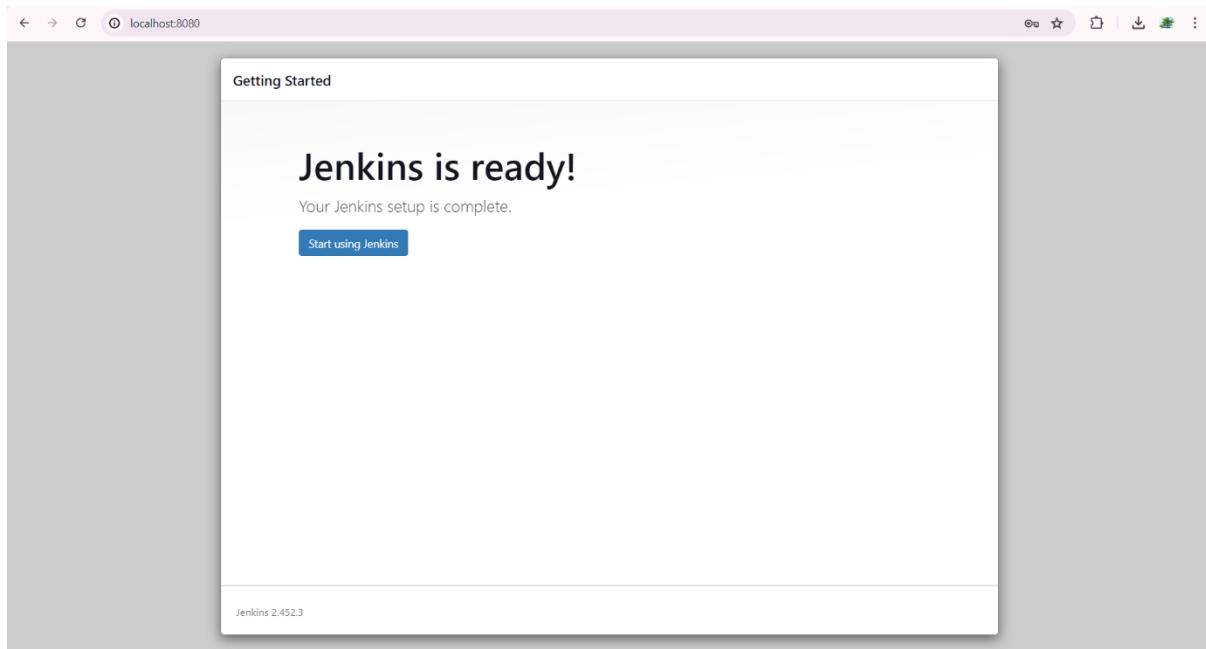
Step 2 : After Jenkins finishes installing the plugins, enter the required information on **the Create First Admin User** page.
Click **Save and Continue** to proceed

The screenshot shows the 'Create First Admin User' step of the Jenkins setup wizard. It contains five input fields: 'Username', 'Password', 'Confirm password', 'Full name', and 'E-mail address'. Below the fields is a note about Jenkins version (Jenkins 2.452.3). At the bottom right are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

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

The screenshot shows the 'Instance Configuration' step of the Jenkins setup wizard. It features a single input field for 'Jenkins URL' containing 'http://localhost:8080'. Below the field is a detailed note explaining its purpose. At the bottom right are two buttons: 'Not now' and 'Save and Finish'.

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



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**

A screenshot of the AWS EC2 'Launch an instance' wizard. The 'Name and tags' step is active, showing a text input field with 'JENKINS-DEVOPS' typed into it. To the right of the input field is a link 'Add additional tags'. On the right side of the screen, there is a sidebar with summary information:

- Number of instances: 1
- Software: Amazon Linux 2 (ami-0ec0e1)
- Virtual service type: t2.micro
- Firewall (security group: New security group)

- Select application and OS images "UBUNTU"

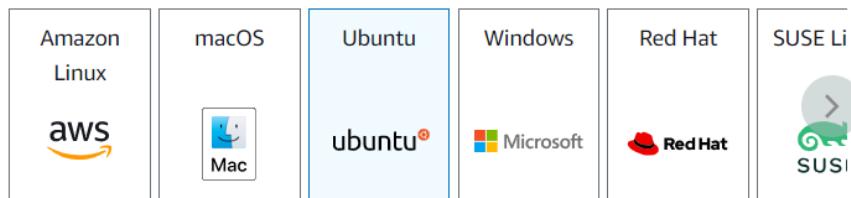
▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

 *Search our full catalog including 1000s of application and OS images*

Recents

Quick Start



Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-0ad21ae1d0696ad58 (64-bit (x86)) / ami-01f6c796d6dbc1e36 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

- Select instance type as t2-micro

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro	Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true	
On-Demand Linux base pricing: 0.0124 USD per Hour	
On-Demand Windows base pricing: 0.017 USD per Hour	
On-Demand RHEL base pricing: 0.0268 USD per Hour	
On-Demand SUSE base pricing: 0.0124 USD per Hour	

All generations

[Compare instance types](#)

[Additional costs apply for AMIs with pre-installed software](#)

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

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

devops_lab

 [Create new key pair](#)

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

▼ Network settings [Info](#)

[Edit](#)

Network | [Info](#)

vpc-07982cf1436f54e85

Subnet | [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP | [Info](#)

Enable

[Additional charges apply](#) when outside of [free tier allowance](#)

Firewall (security groups) | [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

We'll create a new security group called '**launch-wizard-7**' with the following rules:

Allow SSH traffic from
Helps you connect to your instance

Anywhere
0.0.0.0/0

Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

- Now click on Launch an instance.



- Now click on your created instance and click connect

Instances (1/1) Info									
G Connect Instance state Actions Launch instances									
<input type="text"/> Find Instance by attribute or tag (case-sensitive) All states									
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4	
<input checked="" type="checkbox"/>	JENKINS-DEV...	i-0c36a6d4c90ea8533	Running	t2.micro	Initializing	View alarms	ap-south-1a	ec2-13-23...	

- Connect to instance :

[EC2](#) > [Instances](#) > [i-0c36a6d4c90ea8533](#) > Connect to instance

Connect to instance [Info](#)

Connect to your instance i-0c36a6d4c90ea8533 (JENKINS-DEVOPS) using any of these options

EC2 Instance Connect	Session Manager	SSH client	EC2 serial console
--------------------------------------	---------------------------------	----------------------------	------------------------------------

Instance ID
 [i-0c36a6d4c90ea8533 \(JENKINS-DEVOPS\)](#)

Connection Type
 [Connect using EC2 Instance Connect](#)
 Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

[Connect using EC2 Instance Connect Endpoint](#)
 Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

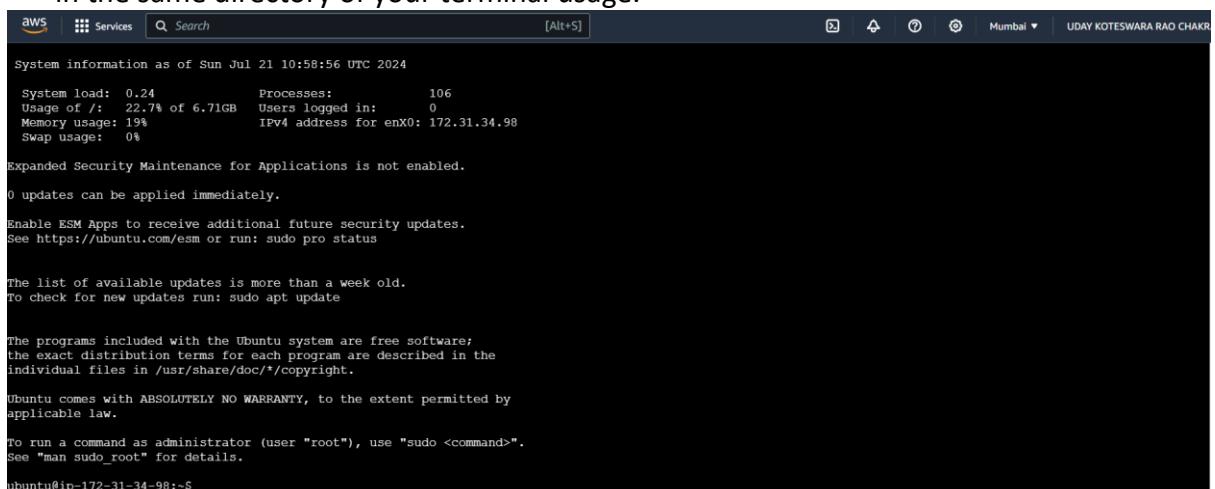
Public IP address
 [13.235.18.237](#)

Username
 Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.

Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

[Cancel](#) [Connect](#)

- 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.



```

aws | Services | Search | [Alt+S] | Mumbai | UDAY KOTESWARA RAO CHAKRABORTY
System information as of Sun Jul 21 10:58:56 UTC 2024
System load: 0.24 Processes: 106
Usage of /: 22.7% of 6.71GB Users logged in: 0
Memory usage: 19% IPV4 address for enx0: 172.31.34.98
Swap usage: 0%
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-34-98:~$
```

- 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.**

```
ubuntu@ip-172-31-45-63:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Tue 2024-07-23 08:59:19 UTC; 20min ago
     Main PID: 529 (java)
        Tasks: 54 (limit: 1130)
       Memory: 372.2M (peak: 472.3M)
          CPU: 42.246s
         CGroup: /system.slice/jenkins.service
             └─529 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

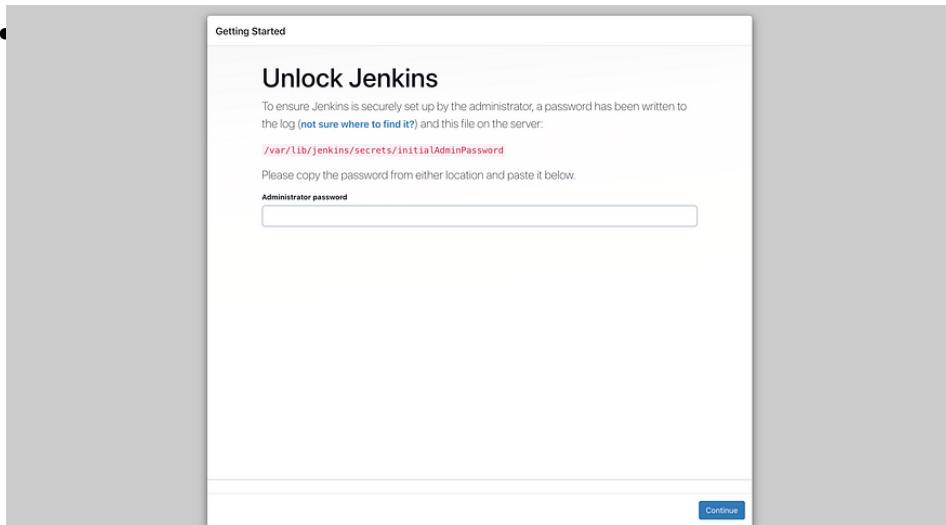
Jul 23 08:59:19 ip-172-31-45-63 jenkins[529]: 2024-07-23 08:59:19.408+0000 [id=32]      INFO      jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
Jul 23 08:59:19 ip-172-31-45-63 jenkins[529]: 2024-07-23 08:59:19.452+0000 [id=32]      INFO      jenkins.InitReactorRunner$1#onAttained: Configuration
Jul 23 08:59:19 ip-172-31-45-63 jenkins[529]: 2024-07-23 08:59:19.508+0000 [id=48]      INFO      hudson.util.Retriger#start: Attempt #1 to do the action
Jul 23 08:59:19 ip-172-31-45-63 jenkins[529]: 2024-07-23 08:59:19.607+0000 [id=32]      INFO      jenkins.InitReactorRunner$1#onAttained: Completed init
Jul 23 08:59:19 ip-172-31-45-63 jenkins[529]: 2024-07-23 08:59:19.706+0000 [id=25]      INFO      hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up
Jul 23 08:59:19 ip-172-31-45-63 systemd[1]: Started Jenkins service - Jenkins Continuous Integration Server.
Jul 23 08:59:41 ip-172-31-45-63 jenkins[529]: 2024-07-23 08:59:41.100+0000 [id=48]      INFO      h.m.DownloadService$Downloadable#load: Obtained the up
Jul 23 08:59:41 ip-172-31-45-63 jenkins[529]: 2024-07-23 08:59:41.938+0000 [id=48]      INFO      h.m.DownloadService$Downloadable#load: Obtained the up
Jul 23 08:59:43 ip-172-31-45-63 jenkins[529]: 2024-07-23 08:59:43.242+0000 [id=48]      INFO      h.m.DownloadService$Downloadable#load: Obtained the up
Jul 23 08:59:43 ip-172-31-45-63 jenkins[529]: 2024-07-23 08:59:43.243+0000 [id=48]      INFO      hudson.util.Retriger#start: Performed the action check v
```

Access Jenkins:

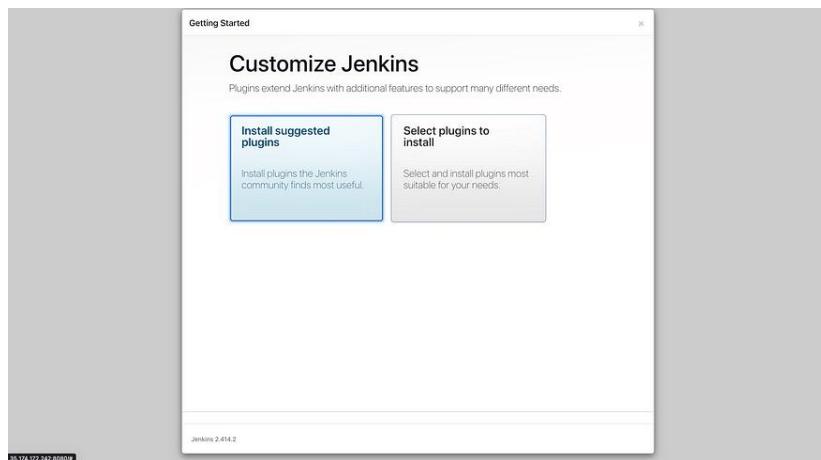
Now, open your web browser and access Jenkins by navigating to:

<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

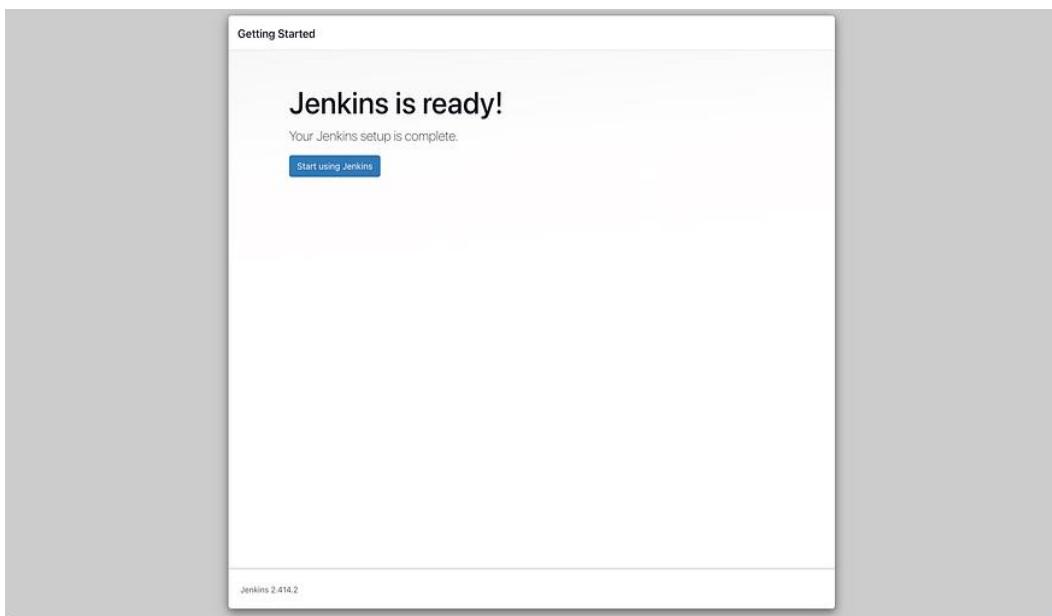
The screenshot shows a 'Create First Admin User' dialog box. It has several input fields: 'Username' (admin), 'Password' (redacted), 'Confirm password' (redacted), 'Full name' (Belik Bagisic), and 'E-mail address' (test@test.com). At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

- 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

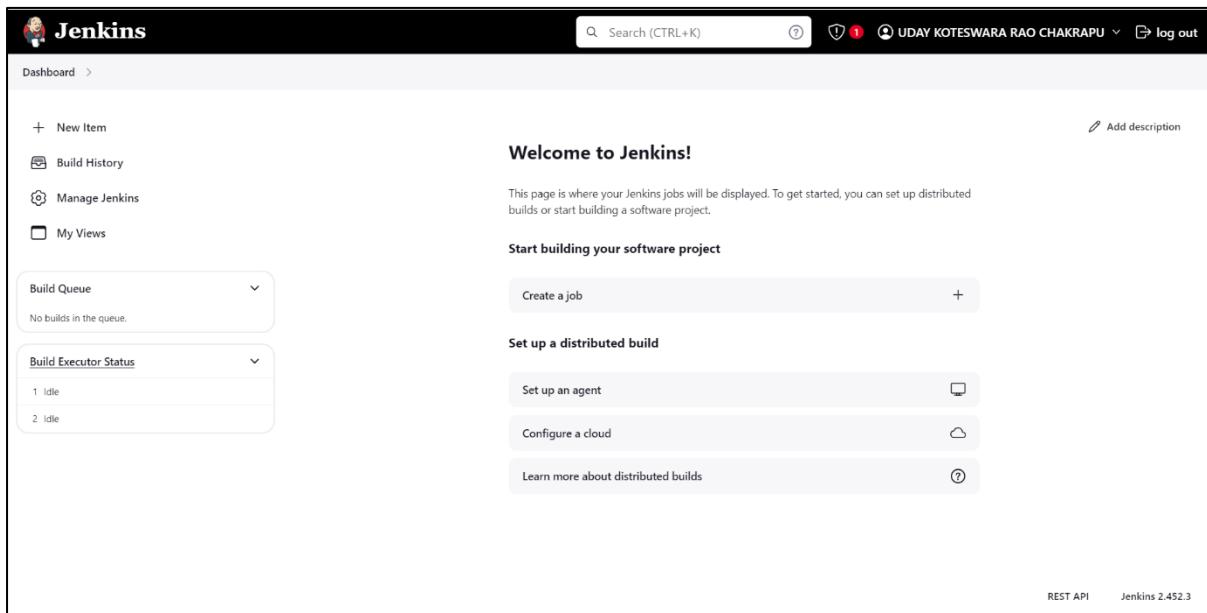
- click “Save and Finish.” you’ve set up Jenkins!
- Start Using Jenkins



- 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.

Source Code Management

None

Git [?](#)

Repositories [?](#)

Repository URL [?](#) ✖

Credentials [?](#)

[+ Add ▾](#)

Advanced ▾

[Add Repository](#)

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

Build Triggers

Trigger builds remotely (e.g., from scripts) [?](#)

Build after other projects are built [?](#)

Build periodically [?](#)

Schedule [?](#)

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour
 Would last have run at Thursday, July 25, 2024 at 5:20:57 AM Coordinated Universal Time; would next run at Thursday, July 25, 2024 at 5:20:57 AM Coordinated Universal Time.

GitHub hook trigger for GITScm polling [?](#)

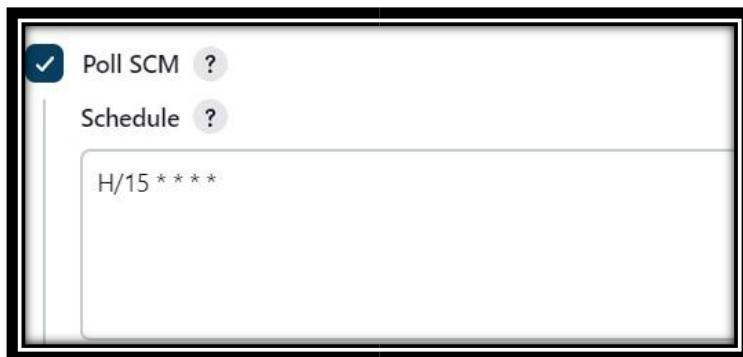
Poll SCM [?](#)

- 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.

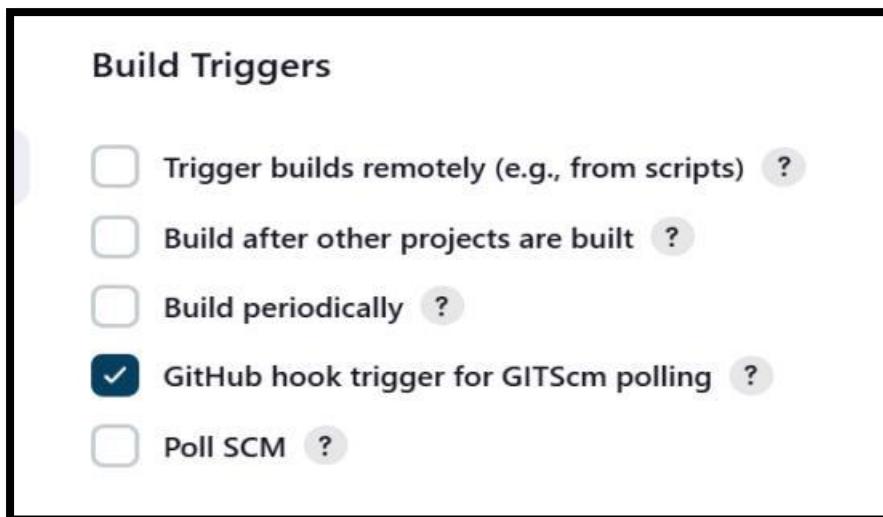


- Save it**
- The above build trigger will commit at every 15min if any change occurs
- Make changes in any file and convert it again in git bash

- 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”.



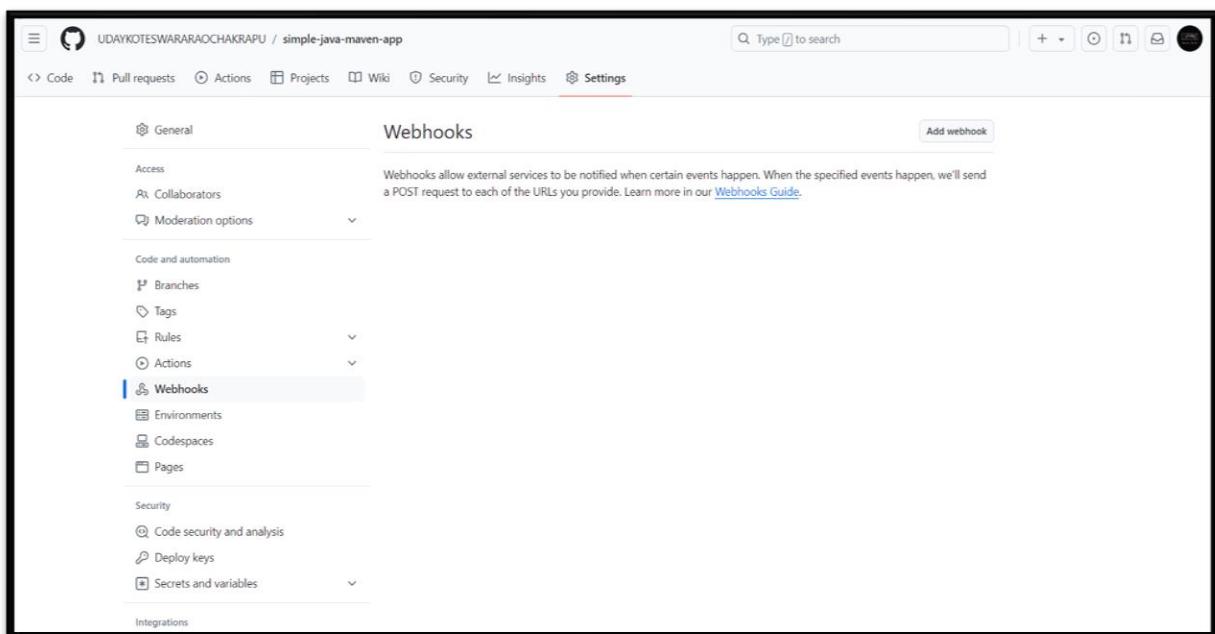
- Open git hub account and login to it.
- Choose “java -maven” repository
-

The screenshot shows a GitHub repository page for "simple-java-maven-app". The repository is public and was forked from "jenkins-docs/simple-java-maven-app". The "Code" tab is selected, showing the master branch with 43 commits. The commits are listed as follows:

- dependabot[bot] Bump org.apache.maven.plugins:maven-jar-plugin from 3.3.0 to 3.4.1 ... 288ebcd - 3 months ago 43 Commits
- .github Update dependencies with dependabot last year
- .jenkins Simplify deliver.sh with Apache Maven quiet and forceStdout 6 months ago
- src Simplify tests 2 years ago
- .gitattributes Initial commit 7 years ago
- .gitignore Mak this work on Java 17 onwards 2 years ago
- README.md scripts are in jenkins dir 2 years ago
- pom.xml Bump org.apache.maven.plugins:maven-jar-plugin from 3.3... 3 months ago

The "About" section includes a description: "For an introductory tutorial on how to use Jenkins to build a simple Java application with Maven.", a link to "jenkins.io/doc/tutorials/build-a-java-app...", and statistics: 0 stars, 0 watching, and 0 forks. The "Releases" section indicates "No releases published" and "Create a new release". The "Packages" section indicates "No packages published" and "Publish your first package".

- 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
```

```
<?xml version="1.0" encoding="UTF-8"?>  
<!--  
Licensed to the Apache Software Foundation (ASF) under one or more  
contributor license agreements. See the NOTICE file distributed with  
this work for additional information regarding copyright ownership.  
The ASF licenses this file to You under the Apache License, Version 2.0  
(the "License"); you may not use this file except in compliance with  
the License. You may obtain a copy of the License at  
  
http://www.apache.org/licenses/LICENSE-2.0  
Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.  
-->  
<Context antiResourceLocking="false" privileged="true" >  
    <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"  
        sameSiteCookies="strict" />  
    <!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"  
        allow="127\\.\\d+\\.\\d+\\.\\d+(:\\d+)?(\\.\\d+)*(:\\d+)?(\\.\\d+)*" /> -->  
    <Manager sessionAttributeValueClassNameFilter="java\\.lang\\.\\{Boolean|Integer|Long|Number|String\\}|org\\.apache\\.c  
</Context>  
~  
~  
~
```

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"/>
```

```
-->
<!--
  - manager-gui      - allows access to the HTML GUI and the status pages
  - manager-script   - allows access to the HTTP API and the status pages
  - manager-jmx      - allows access to the JMX proxy and the status pages
  - manager-status   - allows access to the status pages only
-->
The users below are wrapped in a comment and are therefore ignored. If you
wish to configure one or more of these users for use with the manager web
application, do not forget to remove the <!... ..> that surrounds them. You
will also need to set the passwords to something appropriate.
-->
<!--
<user username="admin" password="<must-be-changed>" roles="manager-gui"/>
<user username="robot" password="<must-be-changed>" roles="manager-script"/>
-->
<!--
  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!... ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
<user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<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"/>
-->
</tomcat-users>
```

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>

Apache Tomcat/9.0.55



If you're seeing this, you've successfully installed Tomcat. Congratulations!



Recommended Reading:

- [Security Considerations How-To](#)
- [Manager Application How-To](#)
- [Clustering/Session Replication How-To](#)

[Server Status](#)[Manager App](#)[Host Manager](#)

Developer Quick Start

[Tomcat Setup](#)[First Web Application](#)[Realms & AAA](#)[JDBC DataSources](#)[Examples](#)[Servlet Specifications](#)[Tomcat Versions](#)

Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

`$CATALINA_HOME/conf/tomcat-users.xml`

In Tomcat 9.0 access to the manager application is split between different users.
[Read more...](#)

[Release Notes](#)[Changelog](#)[Migration Guide](#)[Security Notices](#)

Documentation

[Tomcat 9.0 Documentation](#)[Tomcat 9.0 Configuration](#)[Tomcat Wiki](#)

Find additional important configuration information in:

`$CATALINA_HOME RUNNING.txt`

Developers may be interested in:

[Tomcat 9.0 Bug Database](#)[Tomcat 9.0 JavaDocs](#)[Tomcat 9.0 Git Repository at GitHub](#)

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

[tomcat-announce](#)

Important announcements, releases, security vulnerability notifications. (Low volume).

[tomcat-users](#)

User support and discussion

[taglibs-user](#)

User support and discussion for [Apache Taglibs](#)

[tomcat-dev](#)

Development mailing list, including commit messages

[Other Downloads](#)

[Tomcat Connectors](#)
[Tomcat Native](#)
[Taglibs](#)
[Deployer](#)

[Other Documentation](#)

[Tomcat Connectors](#)
[mod_jk Documentation](#)
[Tomcat Native](#)
[Deployer](#)

[Get Involved](#)

[Overview](#)
[Source Repositories](#)
[Mailing Lists](#)
[Wiki](#)

[Miscellaneous](#)

[Contact](#)
[Legal](#)
[Sponsorship](#)
[Thanks](#)

Apache Software Foundation
[Who We Are](#)
[Heritage](#)
[Apache Home](#)
[Resources](#)

Production Environment:

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

The screenshot shows the Jenkins dashboard with the URL [http://jenkins:8080](#). A search bar at the top right shows 'Search (F+K)' and the user 'admin'. Below the search bar, there's a log out link. The main area shows a form titled 'Enter an item name' with the value 'web-app'. Below the input field, a required field message is displayed. A list of project types is shown:

- Freestyle project**: Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

A blue 'ok' button is at the bottom left of the form, and a note at the bottom right says 'Branch Pipeline is a set of Pipeline projects according to detected branches in one SCM repository.'

- 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

Containers

≡ Tomcat 9.x Remote

Credentials

+ Add ▾

Tomcat URL ?

- Save it and build it.

Dashboard > web-app > #2 > Console Output

✓ **Console Output**

Status Changes Console Output View as plain text Edit Build Information Timings Git Build Data Previous Build

```
Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/web-app
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/web-app/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Iliyaz-Syed/web-app # timeout=10
Fetching upstream changes from https://github.com/Iliyaz-Syed/web-app
> git --version # timeout=10
> git --version # 'git version 2.39.2'
> git fetch --tags --force --progress -- https://github.com/Iliyaz-Syed/web-app +refs/heads/*:refs/remotes/origin/*
timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision e7c2f6c80997f59d06f23389cc1441260d578826 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f e7c2f6c80997f59d06f23389cc1441260d578826 # timeout=10
Commit message: "Update index.jsp"
> git rev-list --no-walk e7c2f6c80997f59d06f23389cc1441260d578826 # timeout=10
[web-app] $ /var/jenkins_home/tools/hudson.tasks.Maven_MavenInstallation/mvnauto/bin/mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< saispurthi:web-app >-----
[INFO] Building web-app Maven Webapp 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] ----- [ war ] -----
```

- 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

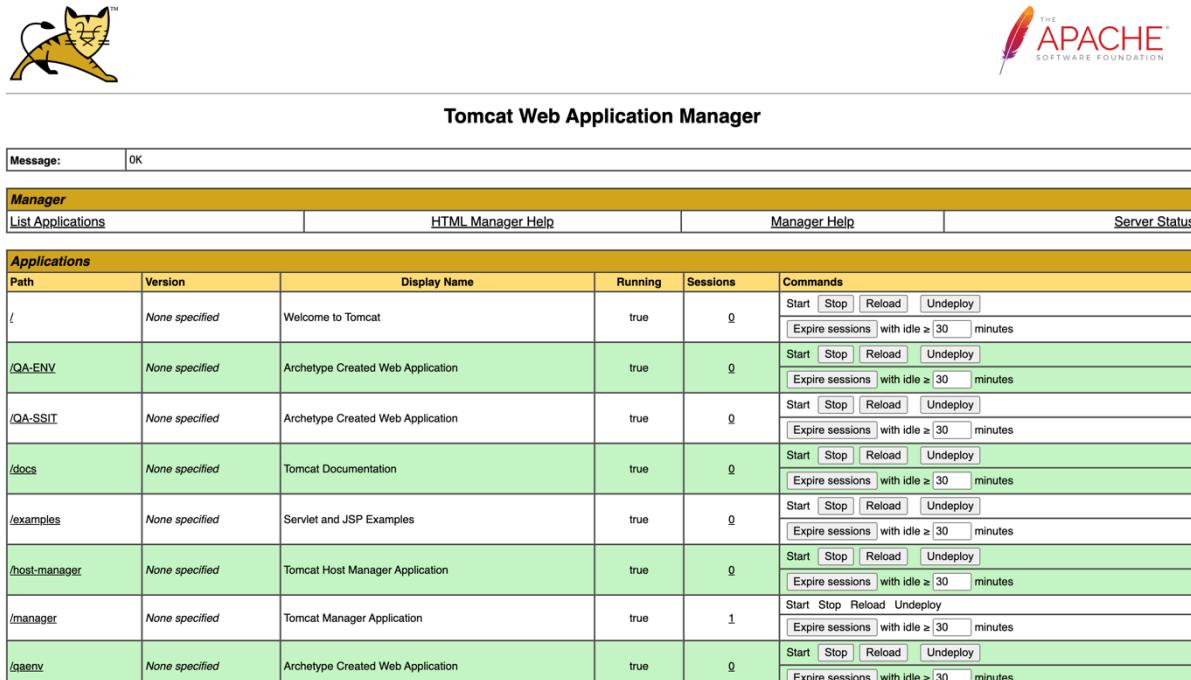
web-apps / src / main / webapp / index.jsp

vvsprasad1242 Update index.jsp

Code Blame 6 lines (6 loc) · 79 Bytes Code 55% faster with GitHub Copilot

```
1 <html>
2 <body>
3 <h2>Hello World! </h2>
4 <h2>WELCOME TO SSIT </h2>
5 </body>
6 </html>
```

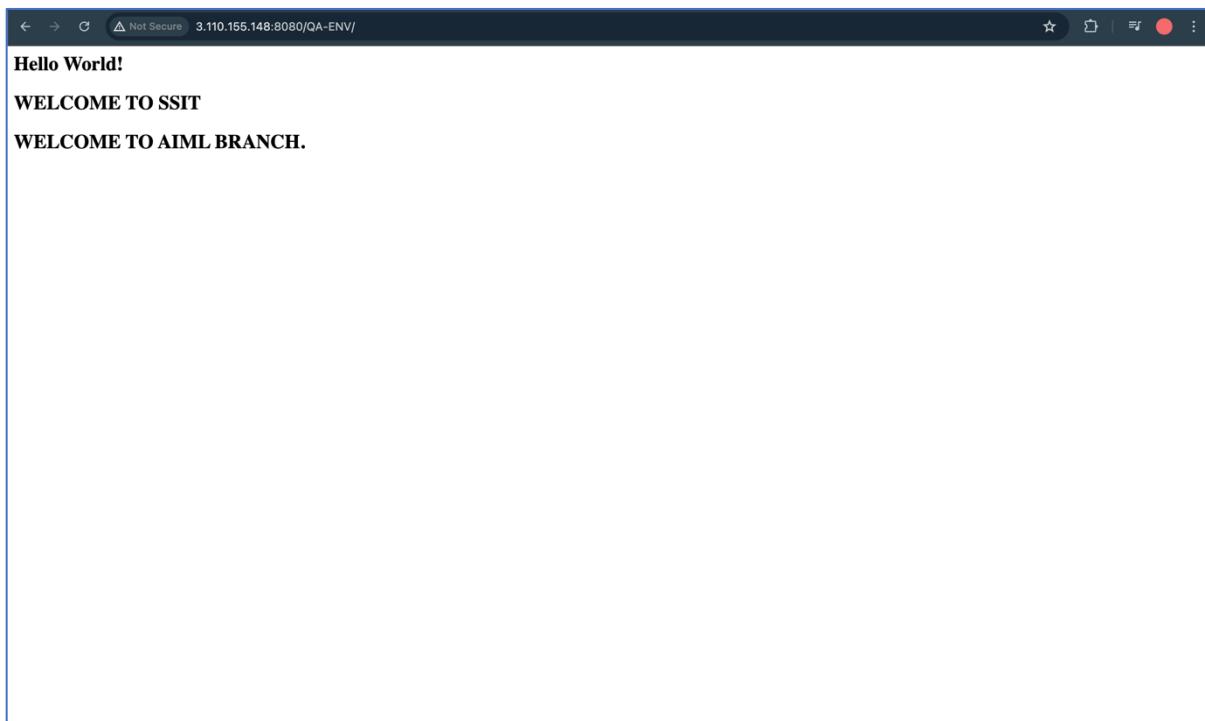
- Now open aws instance and copy the public ip of tomcat instance
- Paste it in chrome new tab with 8080 port number
- By opening tomcat server select manager app and open it
- Now select QA-SSIT and check it



The screenshot shows the Tomcat Web Application Manager interface. At the top left is a cartoon cat logo, and at the top right is the Apache Software Foundation logo. Below the header is a message box containing "Message: OK". The main area is titled "Tomcat Web Application Manager" and contains a table of applications:

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

- The OutPut should be Like this when opened.



The screenshot shows a browser window displaying the output of a Tomcat application. The URL bar shows "Not Secure 3.110.155.148:8080/QA-ENV". The page content includes:

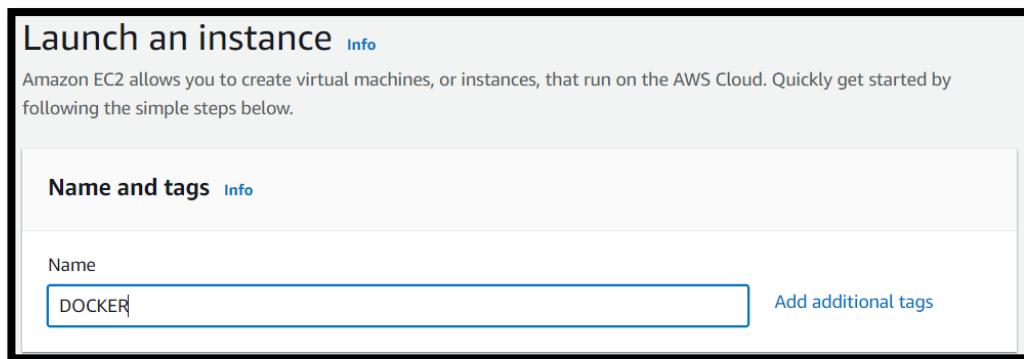
Hello World!
WELCOME TO SSIT
WELCOME TO AIML BRANCH.

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.

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-03f0544597f43a91d (64-bit (x86)) / ami-003e57d854ebc
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

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

Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro	Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true	
On-Demand SUSE base pricing: 0.0146 USD per Hour	
On-Demand Linux base pricing: 0.0146 USD per Hour	
On-Demand Windows base pricing: 0.0192 USD per Hour	
On-Demand RHEL base pricing: 0.029 USD per Hour	

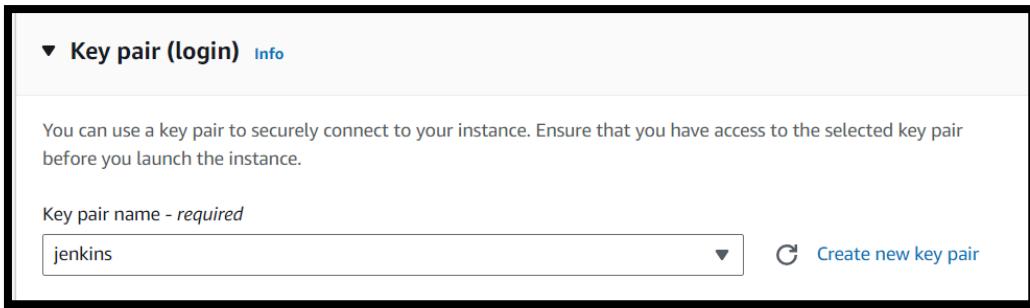
t2.micro

All generations

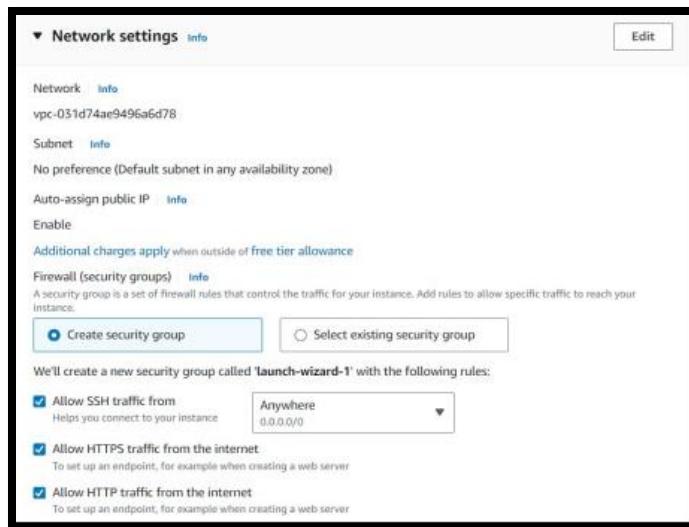
Compare instance types

Additional costs apply for AMIs with pre-installed software

- 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

Launch instance

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

Successfully initiated stopping of i-0e54fdfc9bb2659b2								
Instances (1/4) Info								
Connect Instance state Actions Launch instances								
Find Instance by attribute or tag (case-sensitive) All states								
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4	
<input checked="" type="checkbox"/> DOCKER	i-093fed9383e427973	Running View details Edit	t2.micro	2/2 checks passed View alarms Edit	View alarms Edit	ap-southeast-2c	ec2-3-26-3	
<input type="checkbox"/> jenkins	i-0e54fdfc9bb2659b2	Stopped View details Edit	t2.micro	2/2 checks passed View alarms Edit	View alarms Edit	ap-southeast-2b	-	
<input type="checkbox"/> QA-ENV	i-02a6366786fa39d86	Stopped View details Edit	t2.micro	-	View alarms Edit	ap-southeast-2b	-	
<input type="checkbox"/> sonarqube	i-0a455d76b5bb58464	Stopped View details Edit	t2.medium	-	View alarms Edit	ap-southeast-2a	-	

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

```

aws | Services | Search | [Alt+S]
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1010-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Sun Jul 21 15:31:06 UTC 2024

System load: 0.08          Processes: 120
Usage of /: 55.9% of 6.71GB Users logged in: 1
Memory usage: 50%          IPv4 address for enX0: 172.31.19.111
Swap usage: 0%             Last login: Sun Jul 21 13:32:27 2024 from 152.59.194.52
                           ubuntu@ip-172-31-19-111:~$ 

Expanded Security Maintenance for Applications is not enabled.

15 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sun Jul 21 13:32:27 2024 from 152.59.194.52
ubuntu@ip-172-31-19-111:~$ 

```

i-093fed9383e427973 (DOCKER)

PublicIPs: 3.26.30.161 PrivateIPs: 172.31.19.111

Step 1: Set-up prerequisite packages

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

```

Last login: Sun Jul 21 13:32:27 2024 from 152.59.194.52
ubuntu@ip-172-31-19-111:~$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apt-transport-https is already the newest version (2.7.14build2).
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.1).
software-properties-common is already the newest version (0.99.48).
0 upgraded, 0 newly installed, 0 to remove and 15 not upgraded.
ubuntu@ip-172-31-19-111:~$ 

```

i-093fed9383e427973 (DOCKER)

PublicIPs: 3.26.30.161 PrivateIPs: 172.31.19.111

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"
```

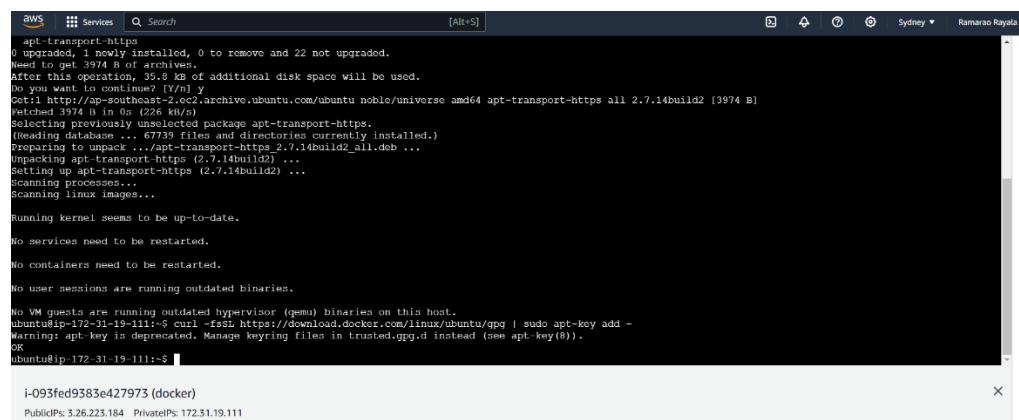
Step 4: Check the installation source priority for docker-ce

```
~$ apt-cache policy docker-ce
```

```
docker@t3st:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.8-3~ubuntu-focal
  Version table:
    5:20.10.8-3~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
```



The screenshot shows a terminal window titled 'AWS' with several tabs open. The current tab displays the output of the 'apt install docker-ce' command. The output includes the following text:

```
apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 22 not upgraded.
Need to get 3974 B of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14build2 [3974 B]
Fetched 3974 B in 0s (226 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database... 67739 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...
Unpacking apt-transport-https (2.7.14build2) ...
Setting up apt-transport-https (2.7.14build2) ...
Scanning processes...
Scanning linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-19-111:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-19-111:~$
```

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
```

```
aws Services Search [Alt+S] Sydney Ramarao Rayala
ubuntu@ip-172-31-19-111:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Sat 2024-07-20 04:18:03 UTC; 1min 27s ago
     TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
      Main PID: 777 (dockerd)
        Tasks: 8
       Memory: 109.9M (peak: 101.1M)
          CPU: 40ms
        Group: /system.slice/docker.service
           └─777 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jul 20 04:18:02 ip-172-31-19-111 dockerd[777]: time="2024-07-20T04:18:02.305740332Z" level=info msg="Starting up"
Jul 20 04:18:02 ip-172-31-19-111 dockerd[777]: time="2024-07-20T04:18:02.321039109Z" level=info msg="detected 127.0.0.53 name server, assuming systemd resolved, c"
Jul 20 04:18:02 ip-172-31-19-111 dockerd[777]: time="2024-07-20T04:18:02.765463560Z" level=info msg="graphdrivervl using prior storage driver: overlay2"
Jul 20 04:18:02 ip-172-31-19-111 dockerd[777]: time="2024-07-20T04:18:02.785039359Z" level=info msg="Loading containers: start."
Jul 20 04:18:02 ip-172-31-19-111 dockerd[777]: time="2024-07-20T04:18:03.415751365Z" level=info msg="Default bridge (docker0) is assigned with an IP address 172.1
Jul 20 04:18:03 ip-172-31-19-111 dockerd[777]: time="2024-07-20T04:18:03.486546613Z" level=info msg="Loading containers: done."
Jul 20 04:18:03 ip-172-31-19-111 dockerd[777]: time="2024-07-20T04:18:03.527357993Z" level=info msg="Docker daemon" commit="662f78c" containerd-snapshotter=false
Jul 20 04:18:03 ip-172-31-19-111 dockerd[777]: time="2024-07-20T04:18:03.587259143Z" level=info msg="API listen on /run/docker.sock"
Jul 20 04:18:03 ip-172-31-19-111 systemd[1]: Started docker.service - Docker Application Container Engine.
ubuntu@ip-172-31-19-111:~$
```

❖ Start using Docker

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

```
root@ip-172-31-19-111:/home/ubuntu
ubuntu@ip-172-31-19-111:~$ sudo su
root@ip-172-31-19-111:/home/ubuntu#
```

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

```
root@ip-172-31-19-111:/home/ubuntu# docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
root@ip-172-31-19-111:/home/ubuntu#
```

- 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
0991256e2f53   jenkins:1.651.3   "/bin/tini -- /usr/l..."   35 hours ago  Up 3 hours   50000/tcp, 0.0.0.0:8081->8080/tcp, :::8081->8080/tcp   amazing_jones
f9fd62620921   hello-world   "hello"
ee76797e8809   myFirst:0.3   "echo echo google.com"
eb45369d506c   myFirst:0.3   "echo echo google.com"
1021457cf4f8   myFirst:0.2   "echo google.com"
a87cd9ce97dc   myFirst:0.1   "echo 'Hello world w..." 
6f7b8a90fcb7   jenkins:1.651.3   "/bin/tini -- /usr/l..."   35 hours ago  Created
7470cb06c3f1   jenkins:1.651.3   "/bin/tini -- /usr/l..."   35 hours ago  Created
1c6dc4d12ae2   jenkins:1.651.3   "/bin/tini -- /usr/l..."   35 hours ago  Exited (143) 33 hours ago
5b3f01dc0c50   hello-world   "hello"
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 logs of 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 download 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              ffffffc90d343   4 weeks ago    188MB
hello-world          latest              d2c94e258dc8   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:fffffc90d343cbbc01a5032edac86db5998c536cd0a366514121a45c6723765c
Deleted: sha256:9f4b5d44149fd139616539e0ef5311a14338a970f25733ba95b4ae6d3becdf0d
Deleted: sha256:8e6e10fbcca1bb180c5cf805a37240945a6ba703cb1f985d4d3b8a1c954fc5b
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**
- Create another directory : ~\$ **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@sha256:2e863c44b718727c860746568e1d54afcd13b2fa71b160f5cd9058fc436217b30
=> 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 Official 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

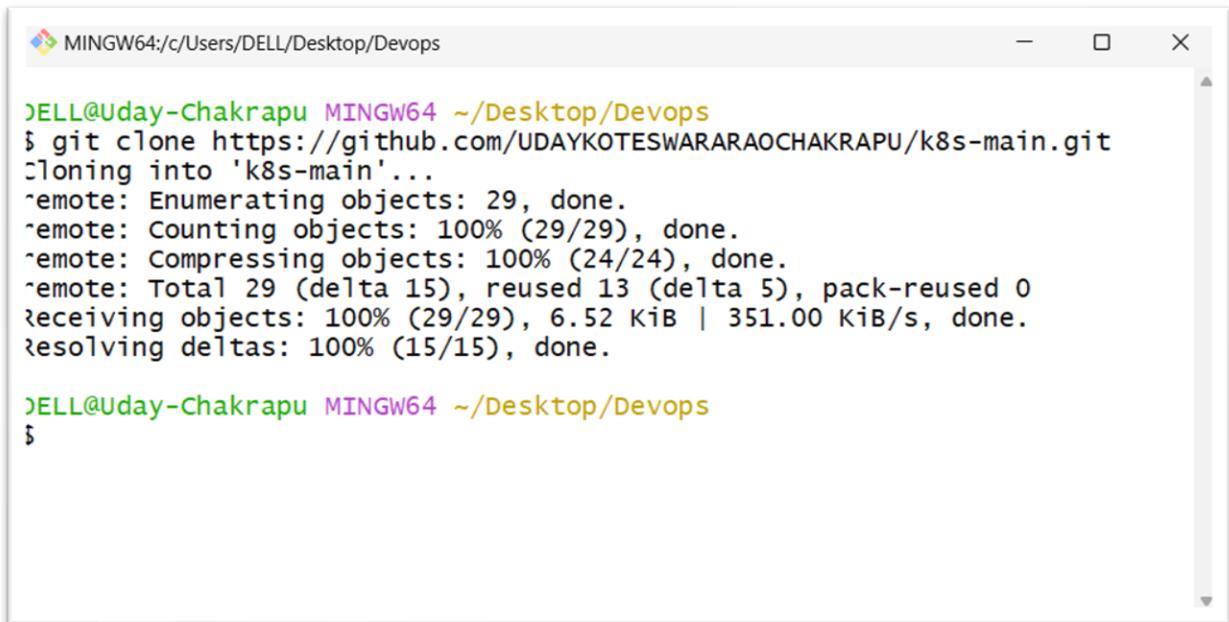
```
~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

- 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.

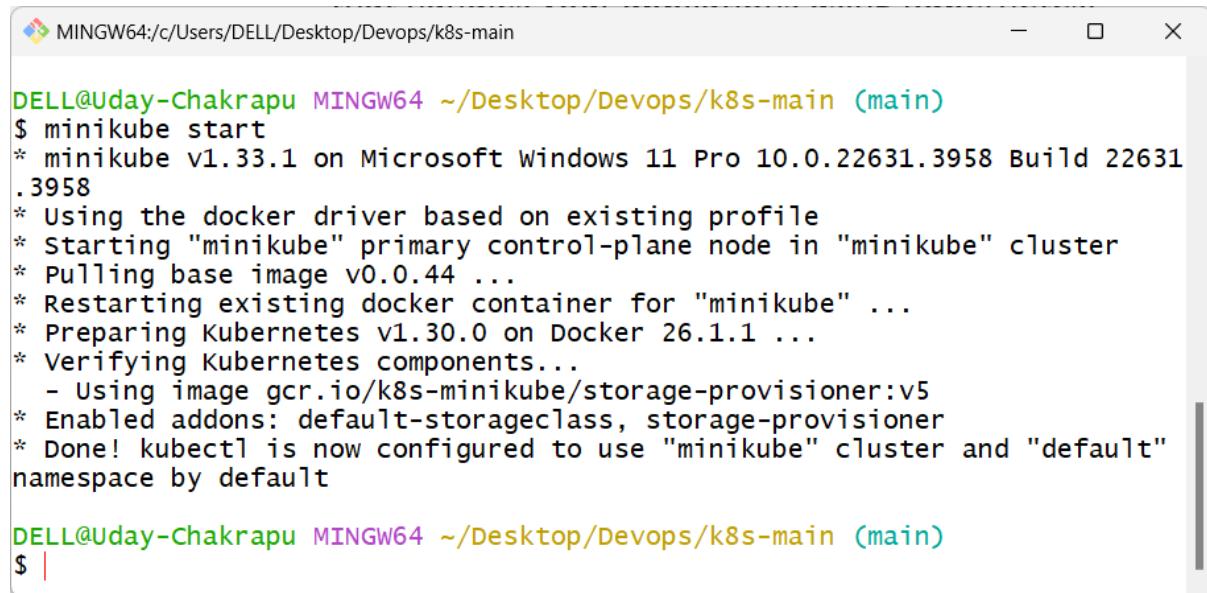
```
~$ git clone https://github.com/UDAYKOTESWARARAOCRAKRAPU/k8s-main.git
```



```
MINGW64:/c/Users/DELL/Desktop/Devops
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops
$ git clone https://github.com/UDAYKOTESWARARAOCRAKRAPU/k8s-main.git
Cloning into 'k8s-main'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 29 (delta 15), reused 13 (delta 5), pack-reused 0
Receiving objects: 100% (29/29), 6.52 KiB | 351.00 KiB/s, done.
Resolving deltas: 100% (15/15), done.

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

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



```
MINGW64:/c/Users/DELL/Desktop/Devops/k8s-main
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ minikube start
* minikube v1.33.1 on Microsoft Windows 11 Pro 10.0.22631.3958 Build 22631.3958
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Restarting existing docker container for "minikube" ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
* Verifying Kubernetes components...
- Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

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

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

~\$ minikube status

```
MINGW64:/c/Users/DELL/Desktop/Devops/k8s-main
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

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

- 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 .

```
MINGW64:/c/Users/DELL/Desktop/Devops/k8s-main
apiserver: Running
kubeconfig: Configured

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ eval $(minikube docker-env)

DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ docker build -t flaskimage:latest
ERROR: "docker buildx build" requires exactly 1 argument.
See 'docker buildx build --help'.

Usage: docker buildx build [OPTIONS] PATH | URL | -
      Start a build

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

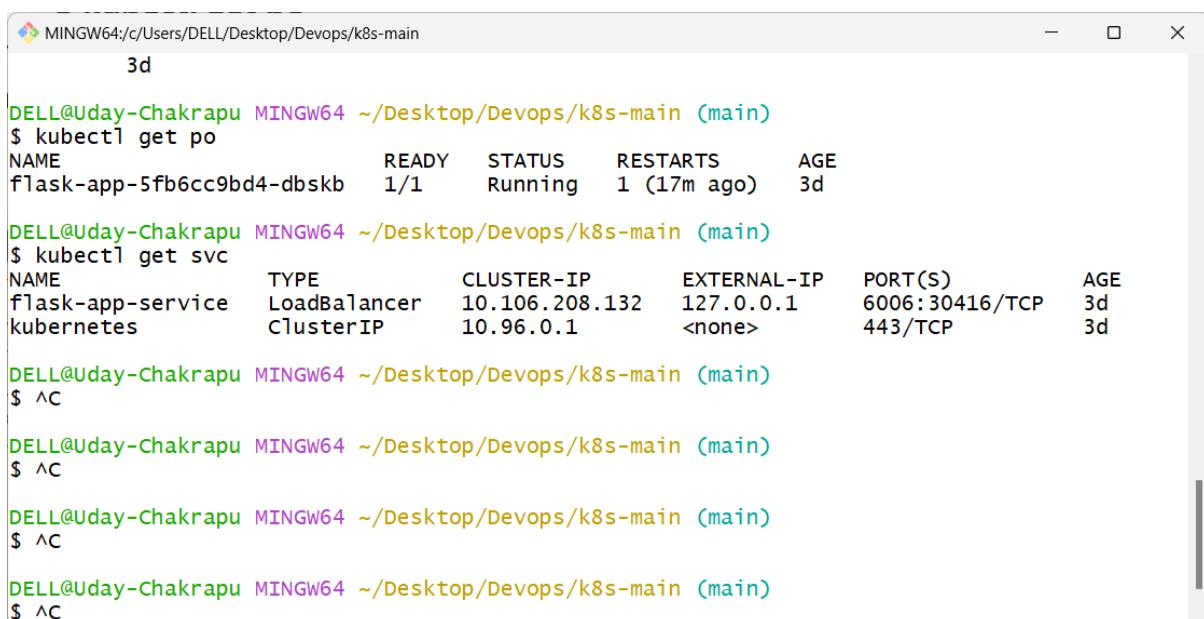
- Now Apply the Kubernetes deployment configuration using the following command.

~\$ 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

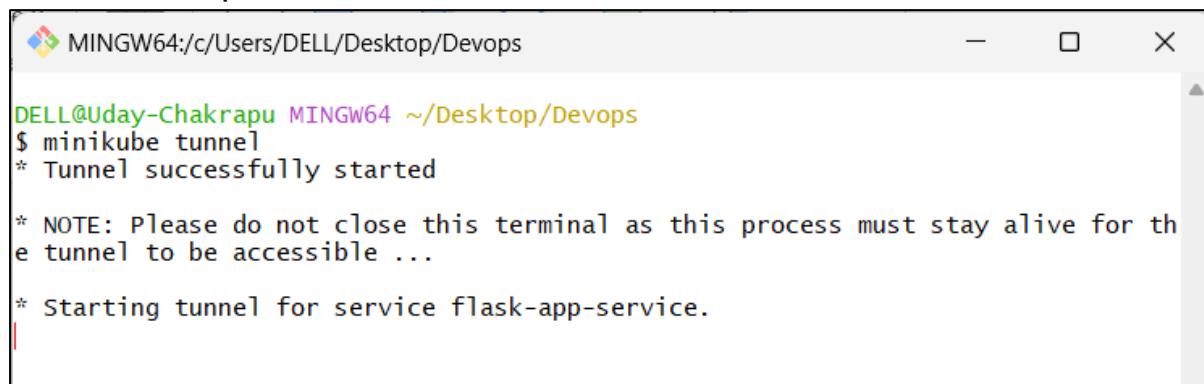
DELL@Uday-Chakrapu MINGW64 ~/Desktop/Devops/k8s-main (main)
$ ^C

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 the tunnel to be accessible ...

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

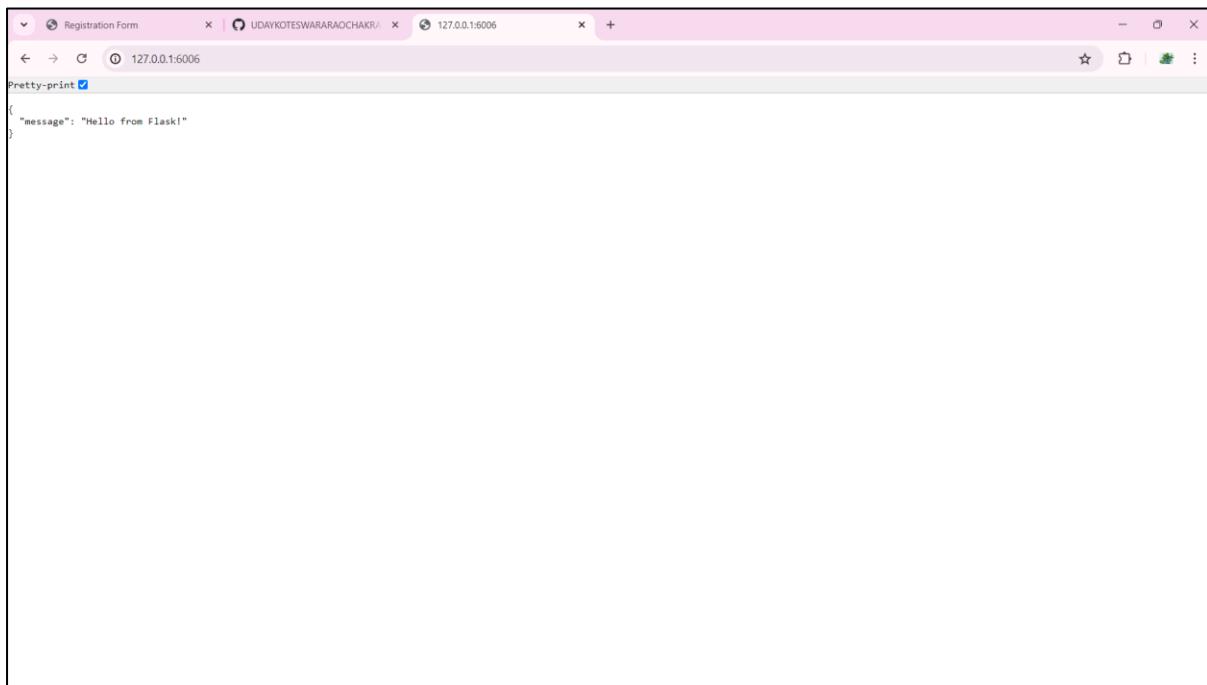
- Now we can access the server at i.e., **127.0.0.1:6006**. The output should like this.
- By following these steps, you can successfully integrate Docker with Kubernetes, deploy a Flask application, and manage it using Minikube on both Windows and Linux environments.

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**

Selenium IDE

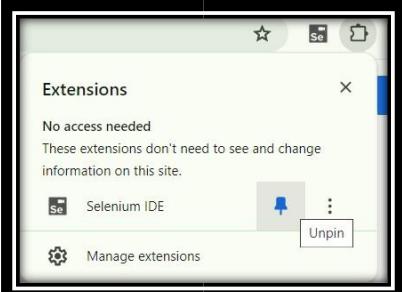
Selenium IDE is a Chrome, Firefox and Edge plugin which records and plays back user interactions, simple scripts or assist in exploratory testing.

Download latest released version for [Chrome](#) or [Firefox](#) or [Edge](#). View the [Release Notes](#).

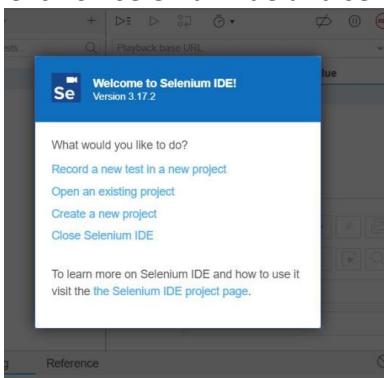
- It opens selenium ide, now click on Add to chrome



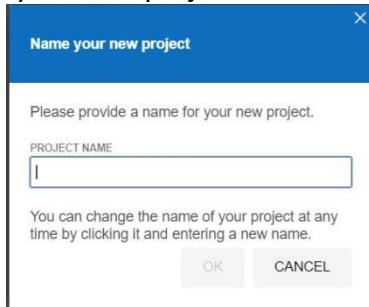
- Click on Extension and select selenium and pin it



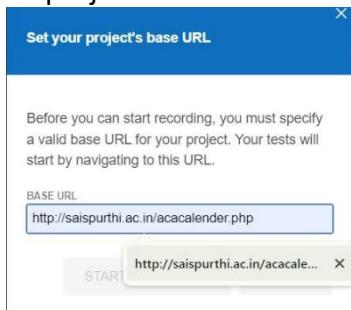
- Click on selenium ide and select record a new test in a new project



- 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**

Project: SSIT-DEMO*			
Tests	+	Command	Target
X ssit*	:	11 click	css=.open li:nth-child(2) > a
		12 click	css=.dropdown:nth-child(3) > .dropdown-toggle
		13 click	css=.open li:nth-child(1) > a
		14 click	css=.dropdown:nth-child(7) .caret
		15 click	css=.dropdown:nth-child(13) > .dropdown-toggle
		16 click	css=.dropdown:nth-child(15) > .dropdown-toggle
		17 click	css=.open > .dropdown-menu a
		18 click	id=name
		19 click	id=name
		20 type	id=name
			ssit

Project: SSIT-DEMO*			
Tests	+	Command	Target
X ssit*	:	1 ✓ open	/acacalender.php
		2 ✓ set window size	1059x816
		3 ✓ click	css=.dropdown:nth-child(3) > .dropdown-toggle
		4 ✓ click	css=.open li:nth-child(2) > a
		5 ✓ click	css=.dropdown:nth-child(5) > .dropdown-toggle
		6 ✓ click	css=.open li:nth-child(1) > a
		7 X click	css=.dropdown:nth-child(14) > .dropdown-toggle
		8 click	css=.open li:nth-child(1) > a
		9 click	css=.dropdown:nth-child(12) > .dropdown-toggle

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.

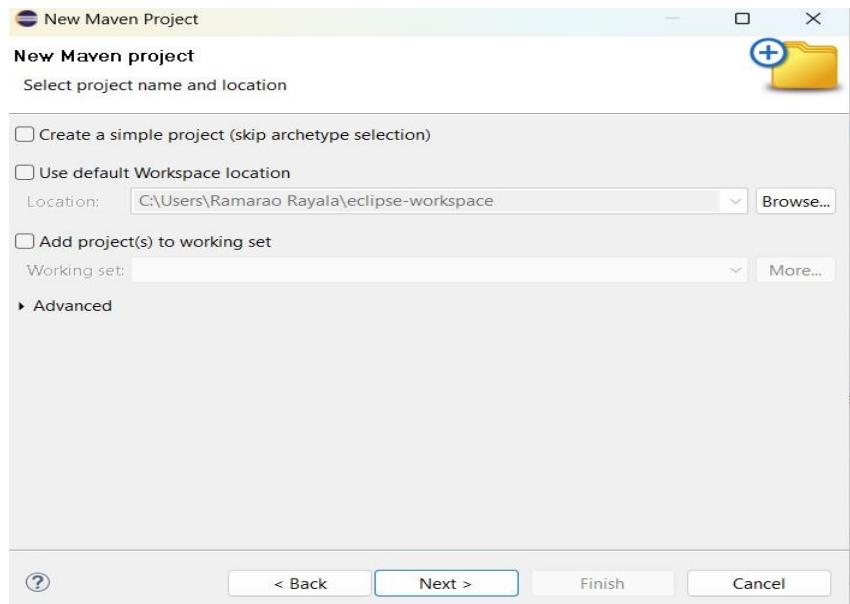
Step 2: Install Eclipse IDE

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

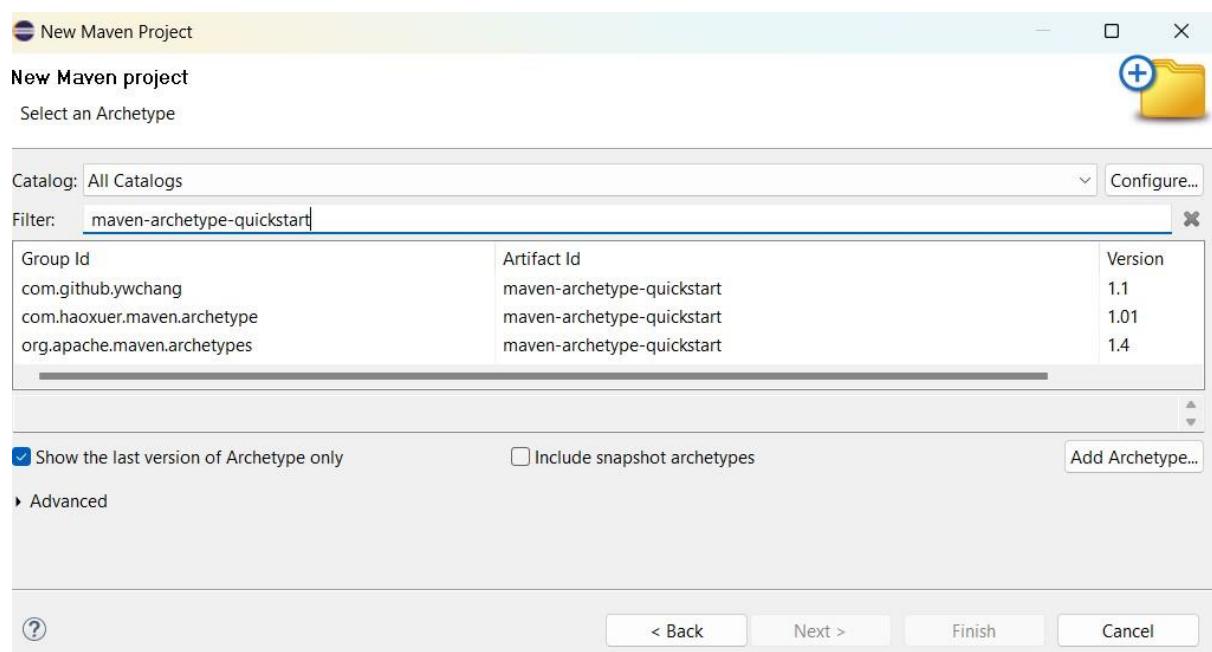
→ After completing the installation of eclipse open eclipse

- **Create a new maven Project**

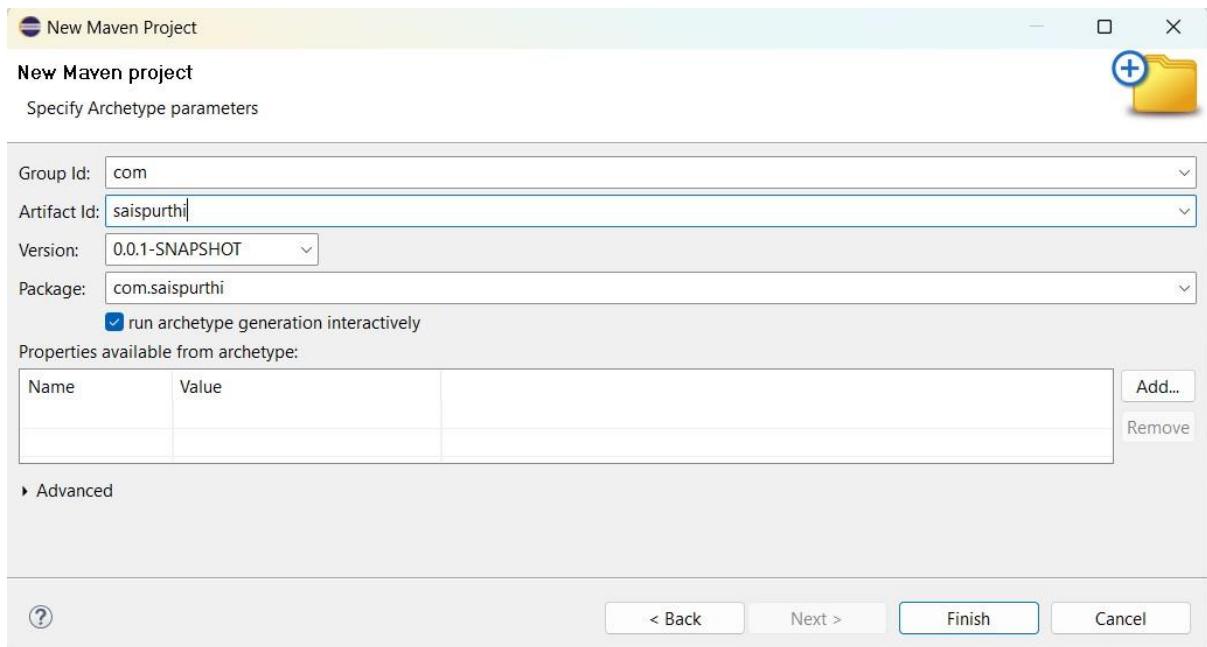
Click on the File menu, then go to New and select maven Project select maven project name and location



- 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**

The screenshot shows the Eclipse IDE interface. At the top, there is a code editor window titled "saipurhi/pom.xml" containing XML code for a Maven project. Below the code editor is a tab bar with "Overview", "Dependencies", "Dependency Hierarchy", and "Effective POM". Underneath the tabs is a "Console" window showing the build output:

```

[terminated] C:\Users\Ramarao Rayala\p2\pool\plugins\org.eclipse.jst\openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\java.exe
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.982 s
[INFO] Finished at: 2024-07-17T20:54:38+05:30
[INFO] -----

```

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

Code :

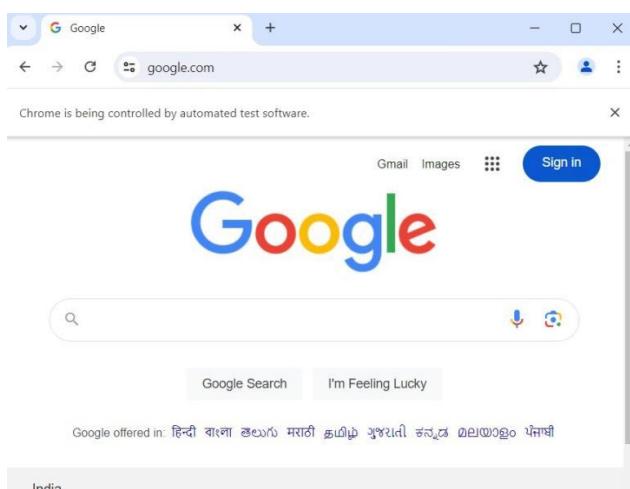
```

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 :

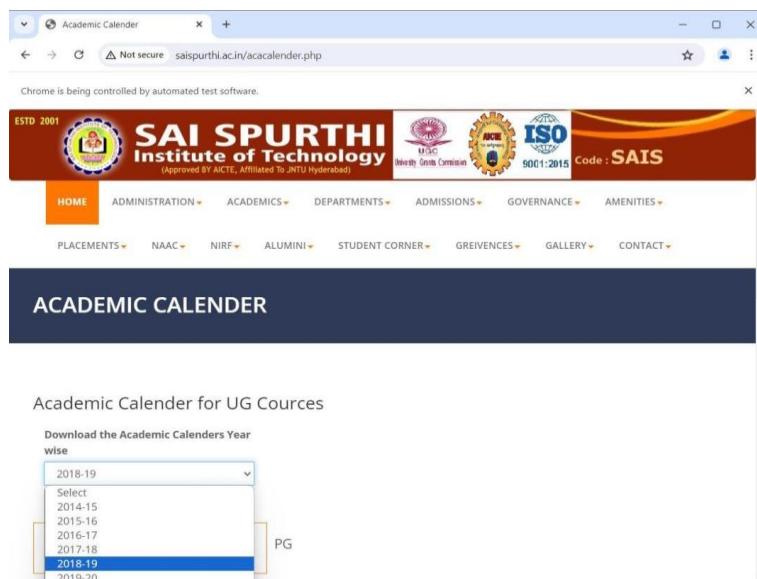


DropDown testing :

Code :

```
package csm.testselenium;
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

A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the following text:

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91863>java -version
java version "22.0.1" 2024-04-16
Java(TM) SE Runtime Environment (build 22.0.1+8-16)
Java HotSpot(TM) 64-Bit Server VM (build 22.0.1+8-16, mixed mode, sharing)

C:\Users\91863>
```

The window has a dark theme with white text and a light gray header bar.

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

The Apache Software Foundation logo is at the top left, followed by the JMeter logo and a "Community CODE" badge.

About

- Overview
- License

Download

- Download Releases
- Release Notes

Documentation

- Get Started
- User Manual
- Best Practices
- Component Reference
- Functions Reference
- Properties Reference
- Change History
- Javadocs
- JMeter Wiki
- FAQ (Wiki)

Tutorials

- Distributed Testing
- Recording Tests
- JUnit Sampler
- Access Log Sampler
- Extending JMeter

Community

Download Apache JMeter

We recommend you use a mirror to download our release builds, but you must [verify the integrity](#) of the downloaded files using signatures downloaded from our main distribution directories. Recent releases (48 hours) may not yet be available from all the mirrors.

You are currently using <https://dlcdn.apache.org/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are [backup](#) mirrors (at the end of the mirrors list) that should be available.

Other mirrors: <https://dlcdn.apache.org/>

The [KEYS](#) link links to the code signing keys used to sign the product. The [PGP](#) link downloads the OpenPGP compatible signature from our main site. The [SHA-512](#) link downloads the sha512 checksum from the main site. Please [verify the integrity](#) of the downloaded file.

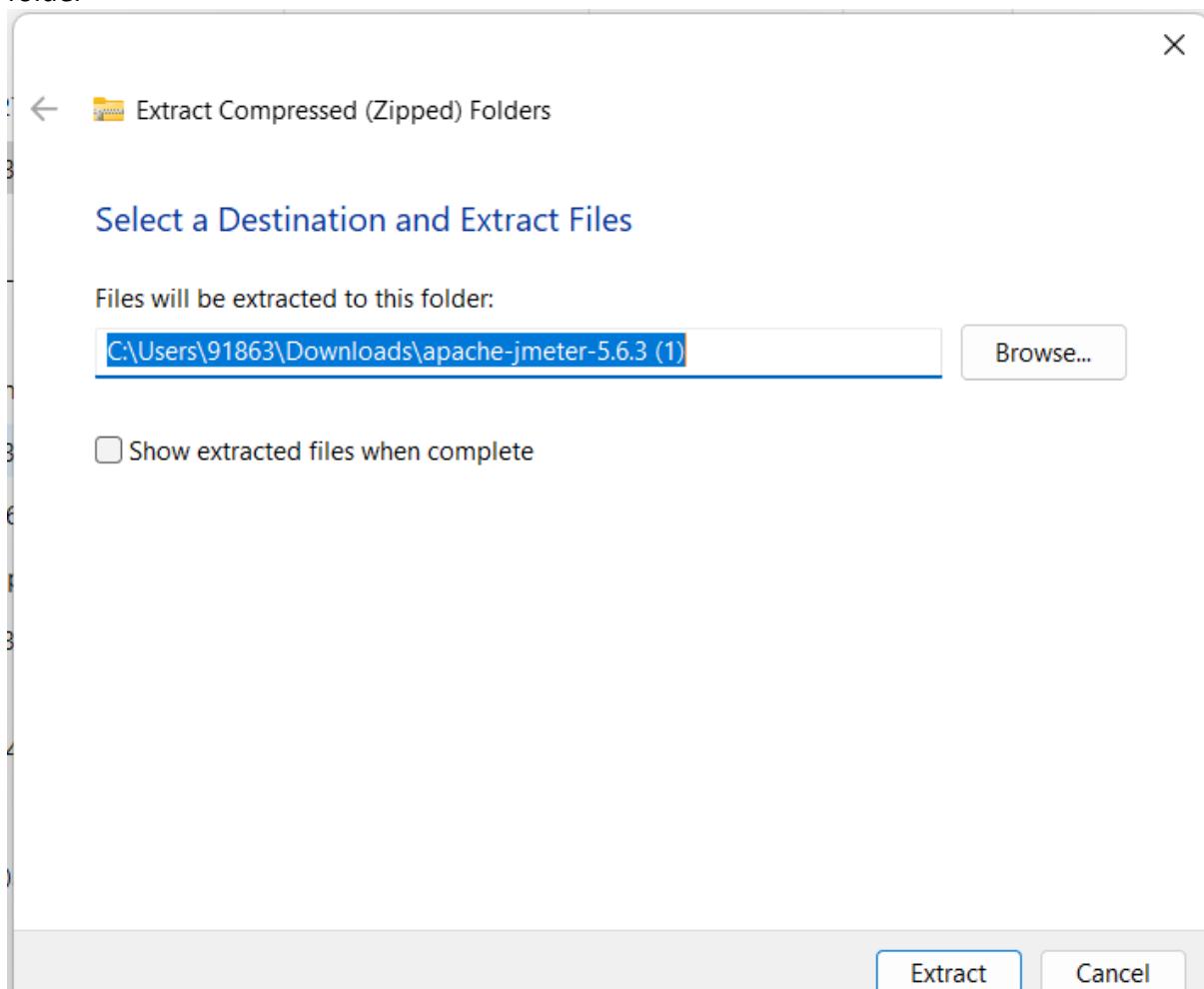
For more information concerning Apache JMeter, see the [Apache JMeter](#) site.

[KEYS](#)

Apache JMeter 5.6.3 (Requires Java 8+)

Binaries

[apache-jmeter-5.6.3.tgz sha512 pgp](#)
[apache-jmeter-5.6.3.zip sha512 pgp](#)



Extract Compressed (Zipped) Folders

Select a Destination and Extract Files

Files will be extracted to this folder:

C:\Users\91863\Downloads\apache-jmeter-5.6.3 (1)

Show extracted files when complete

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**”.

Download Node.js®

Download Node.js the way you want.

[Package Manager](#) [Prebuilt Installer](#) [Prebuilt Binaries](#) [Source Code](#)

I want the [v20.16.0 \(LTS\)](#) version of Node.js for [Windows](#) running [x64](#)

[Download Node.js v20.16.0](#)

Node.js includes [npm \(10.8.1\)](#).

[Read the changelog for this version](#)

[Read the blog post for this version](#)

[Learn how to verify signed SHASUMS](#)

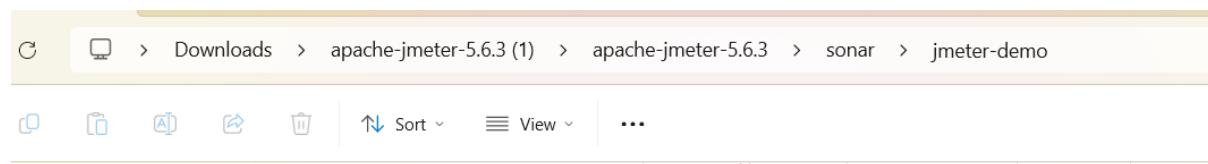
[Check out all available Node.js download options](#)

[Learn about Node.js Releases](#)

- 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/sonar/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"
  }
}
```

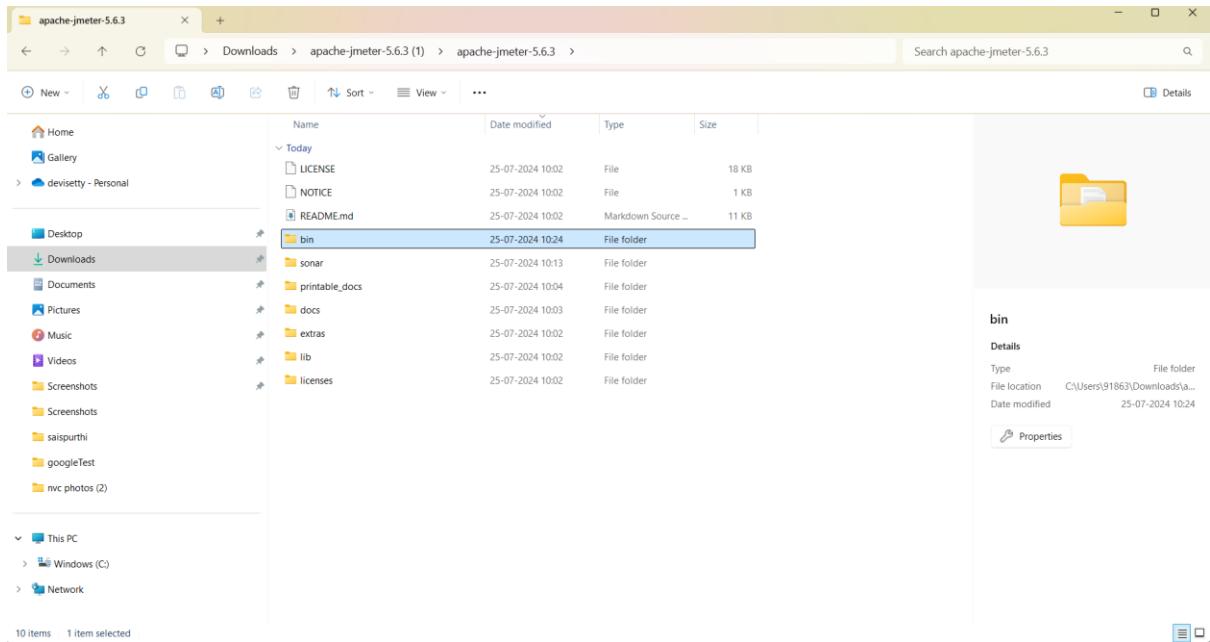
- **Installing gecko driver**

```
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
```

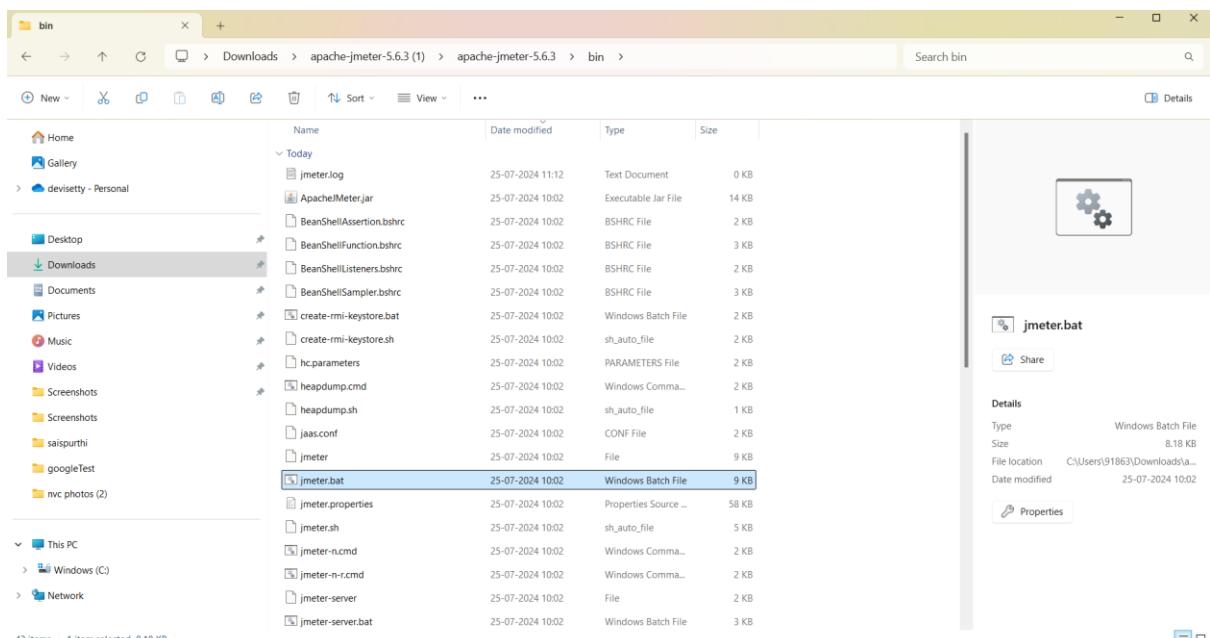
- **Cat command**

```
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"
  }
}
```

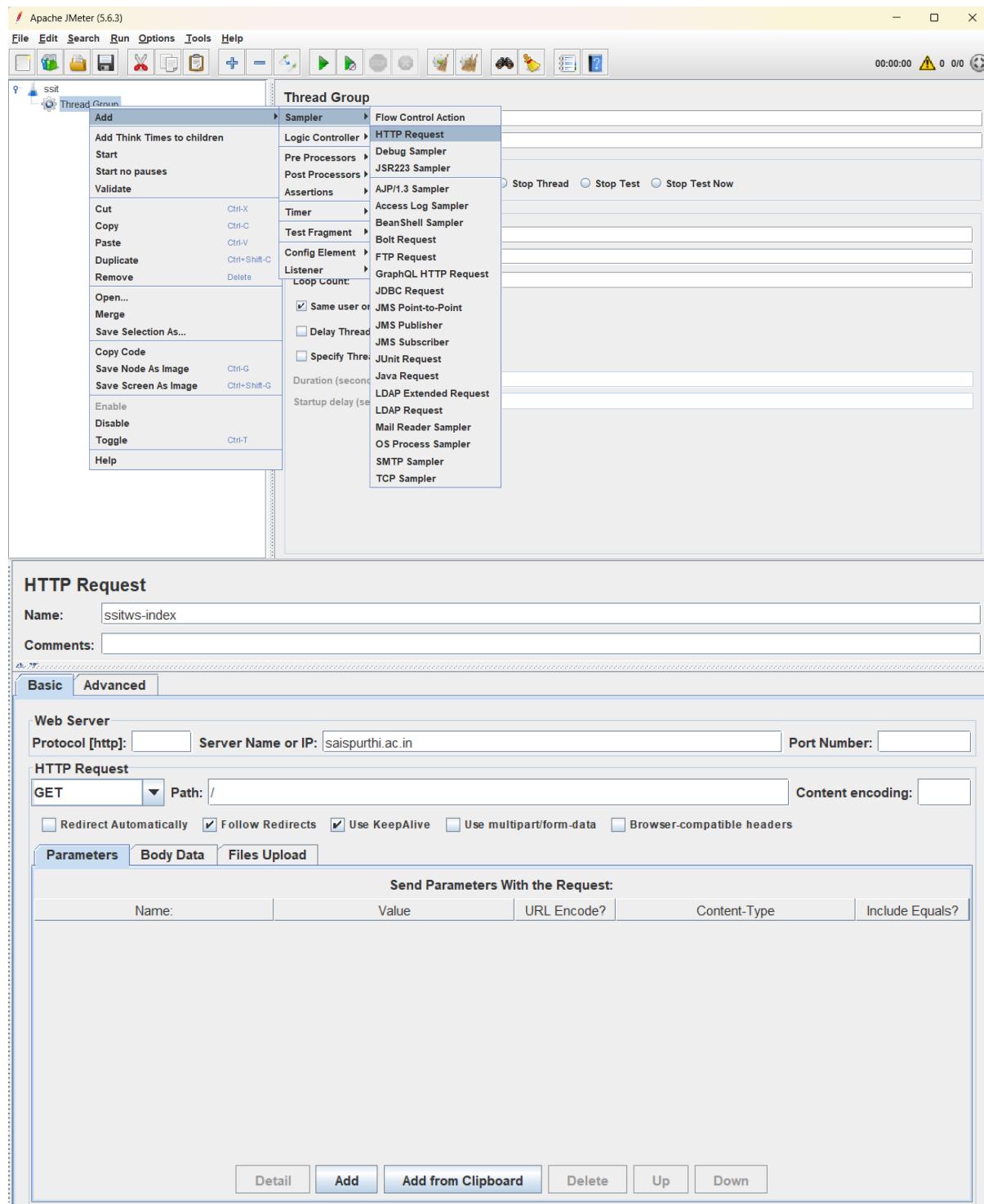
- **Open bin file in JMeter-folder**



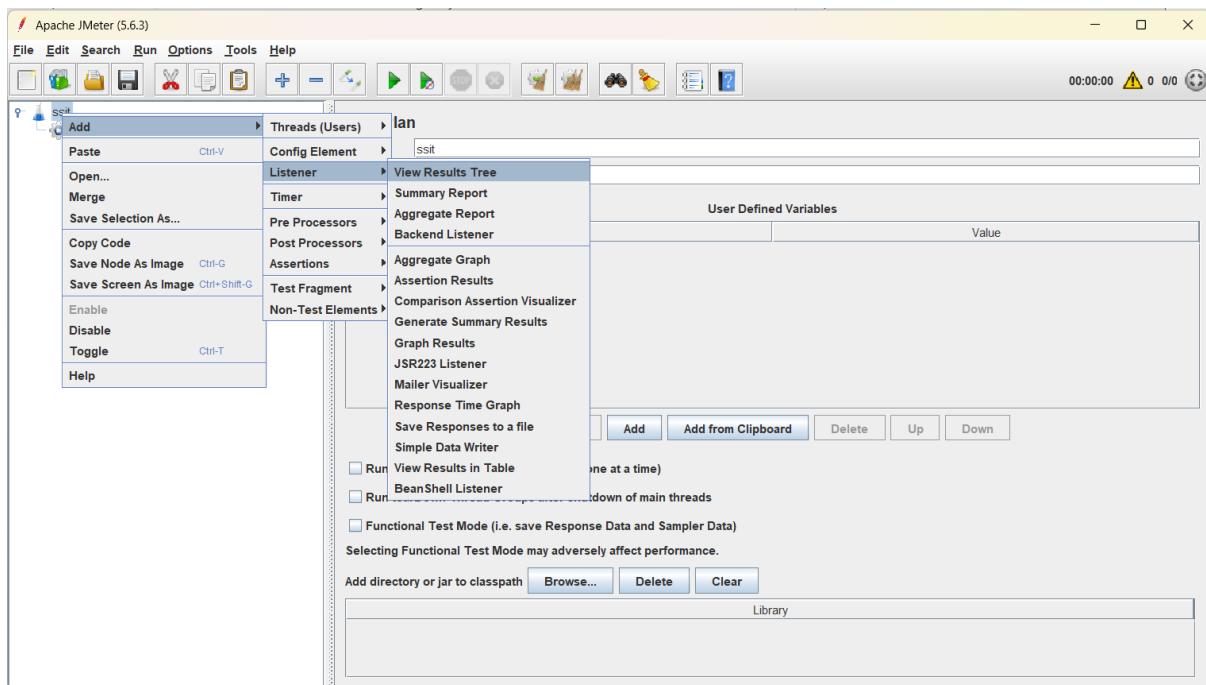
- Open jmeter.bat in JMeter-folder after bin



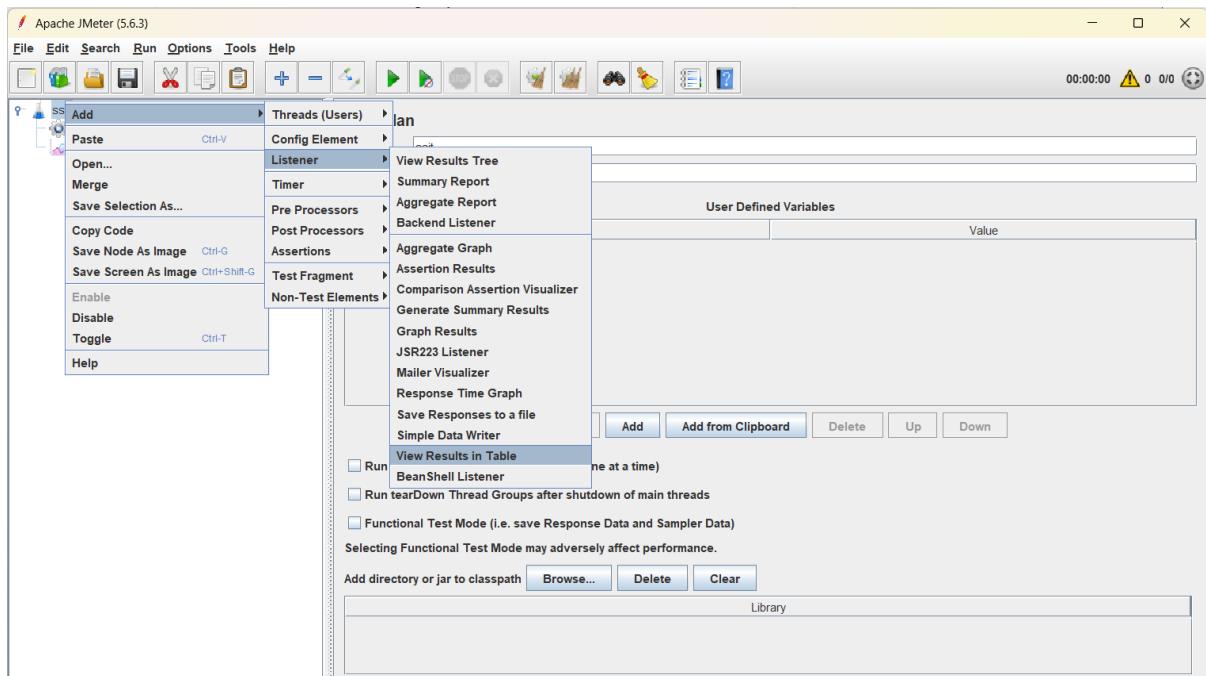
- From that bin folder open Apache-JMeter and select all the options displaying in the image



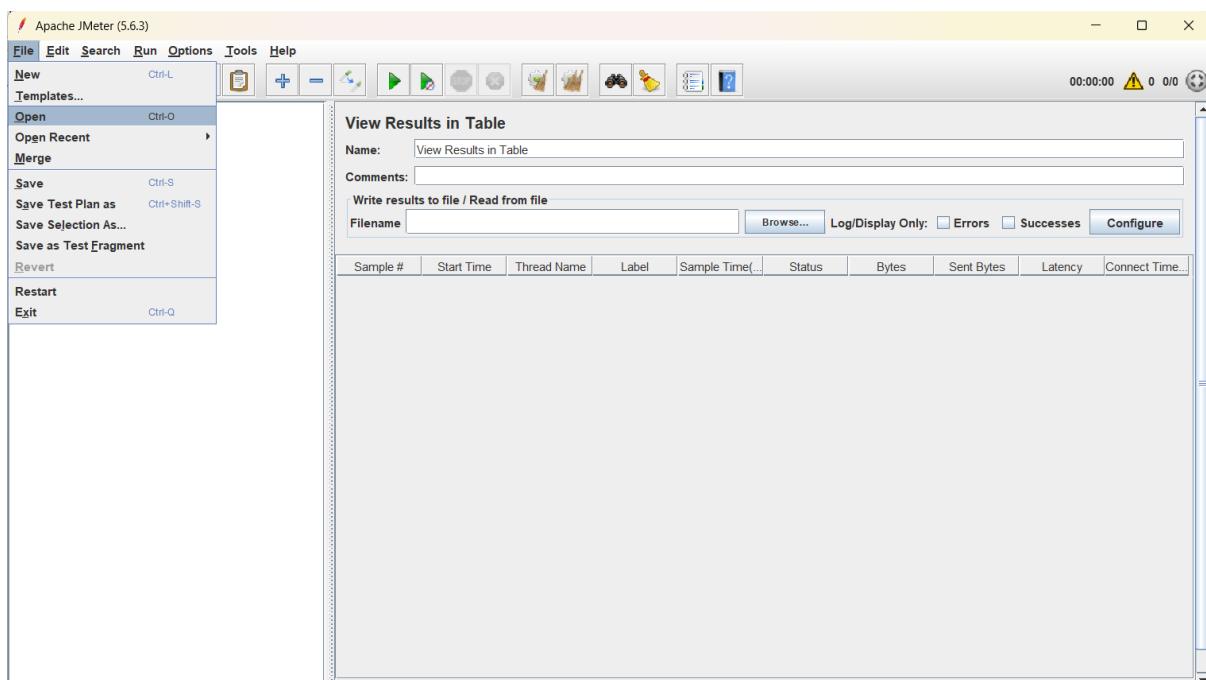
- Viewing the results in the tree



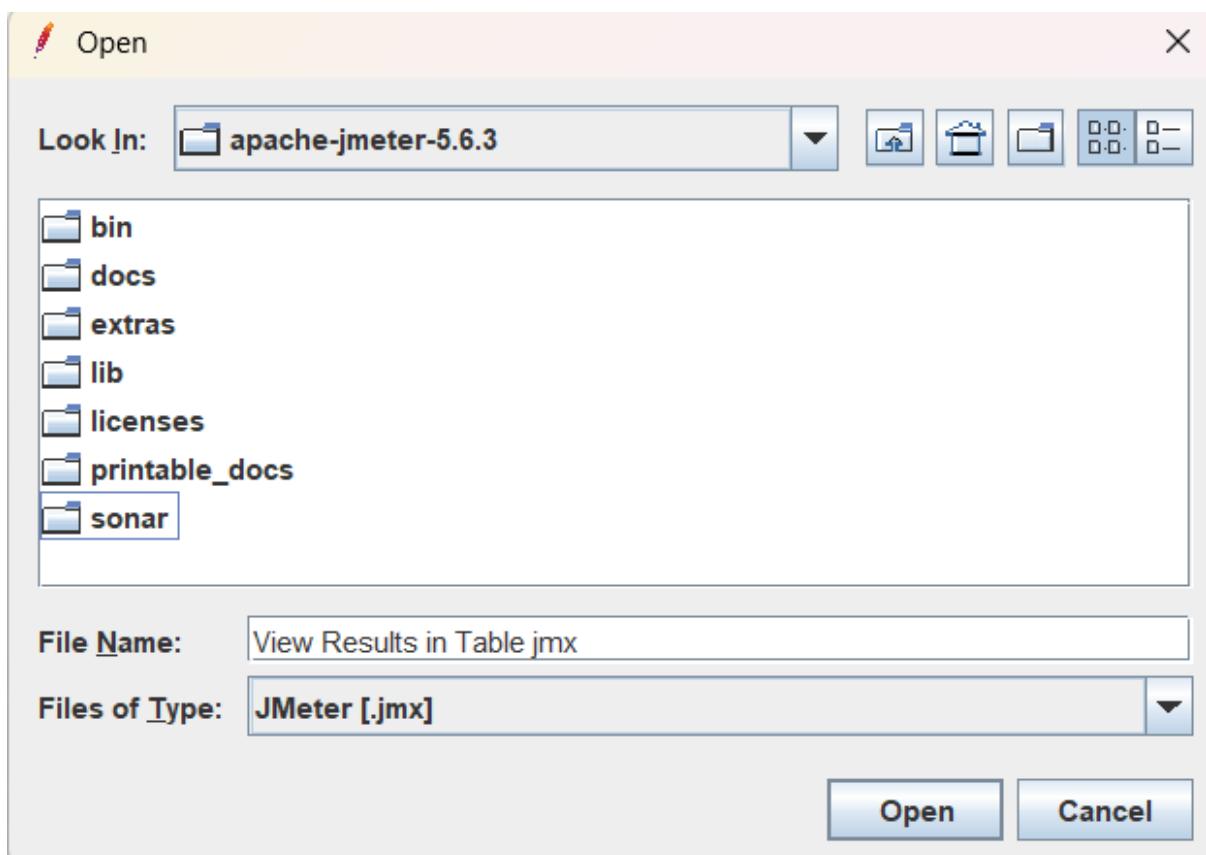
- Viewing the results in the table



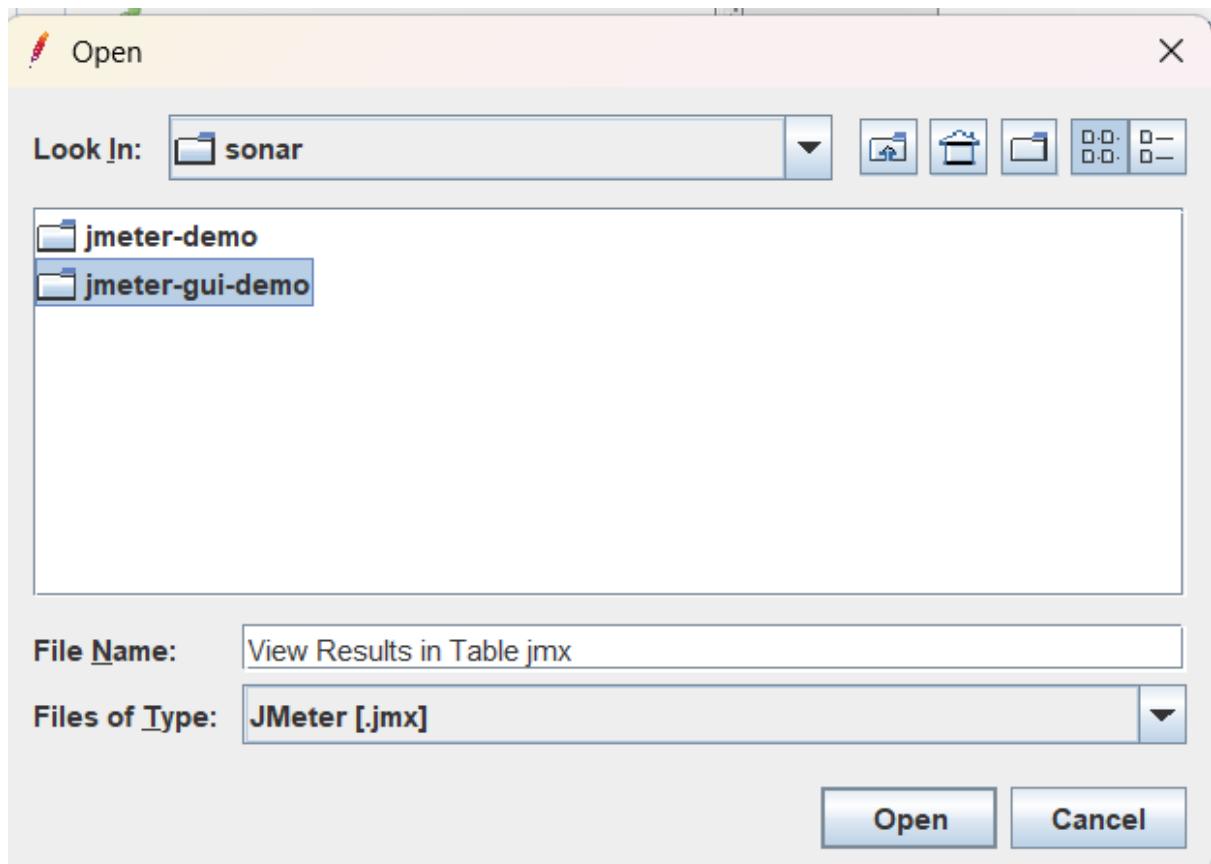
- Click on file and in that click on open



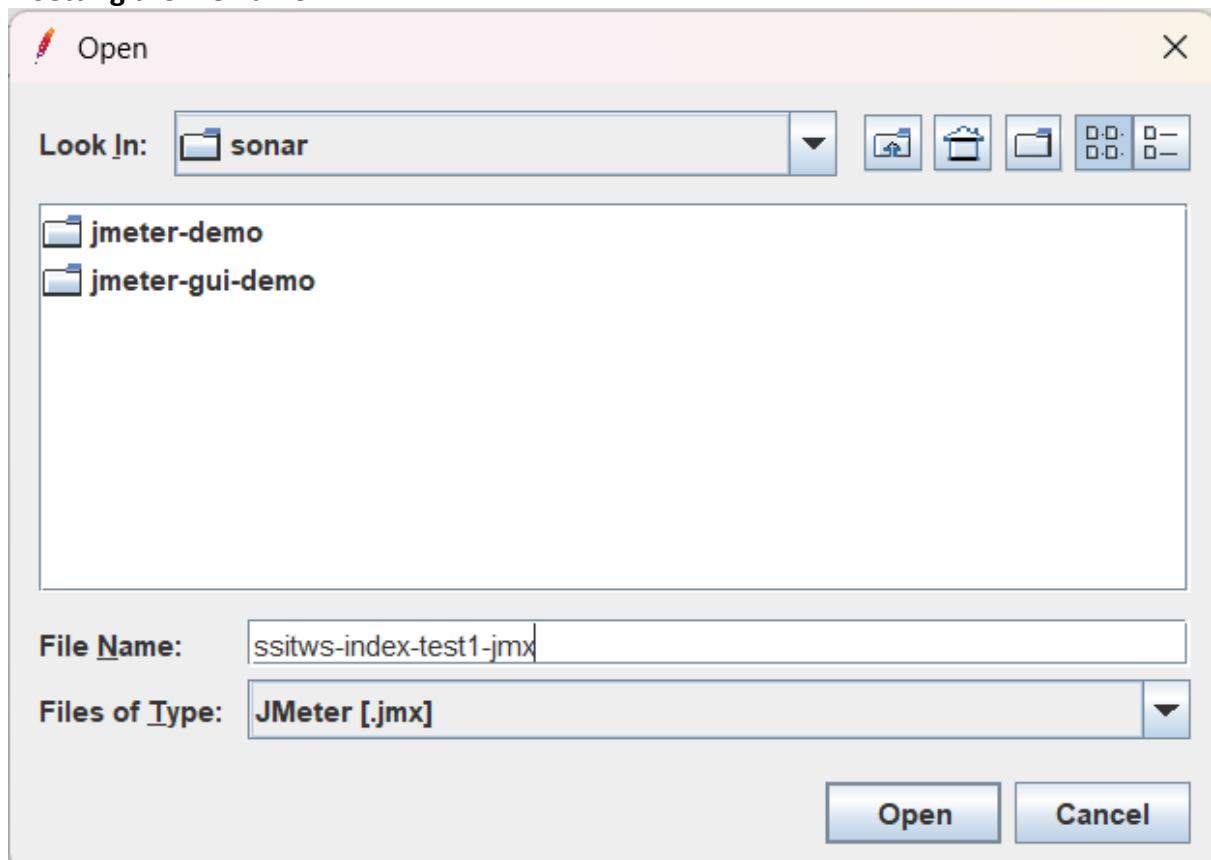
- Click on sonar



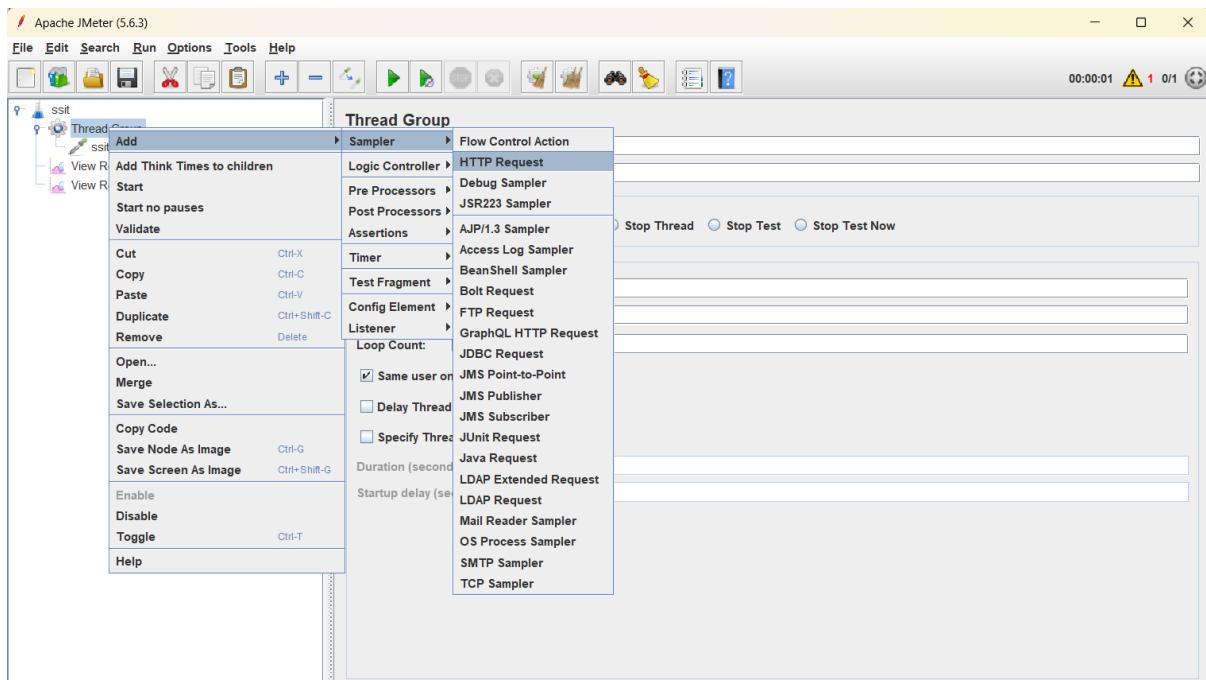
- Selecting JMeter-demo and JMeter-Gui-demo



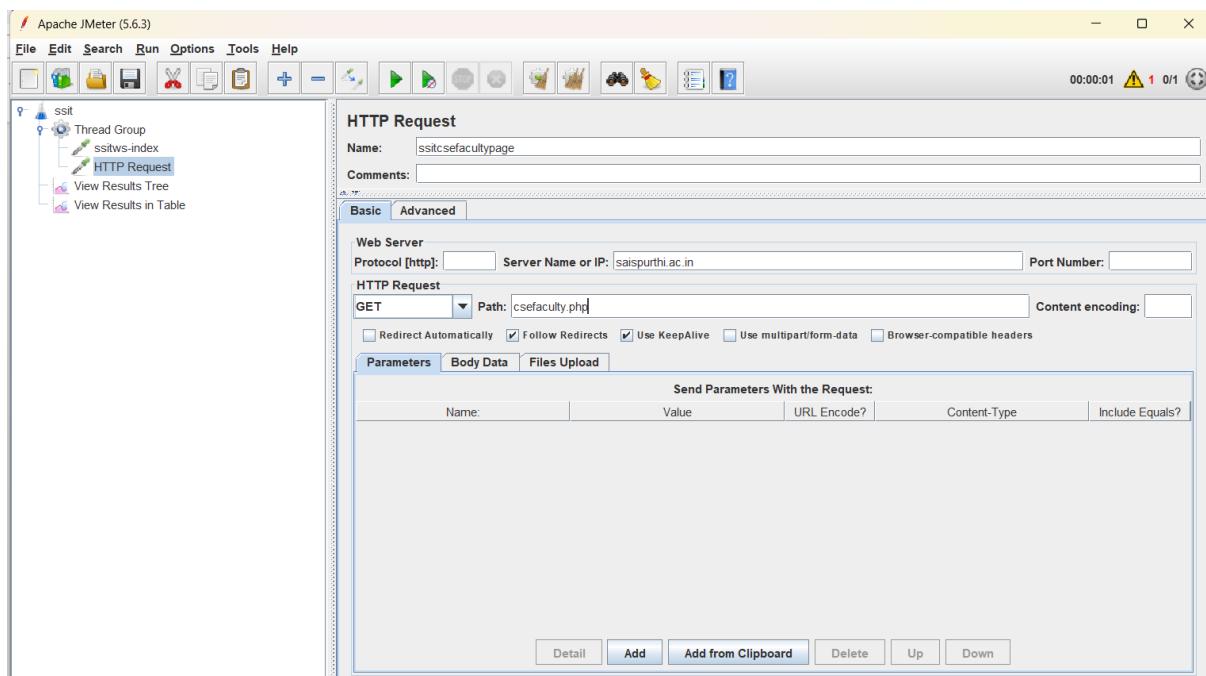
- Setting the file name



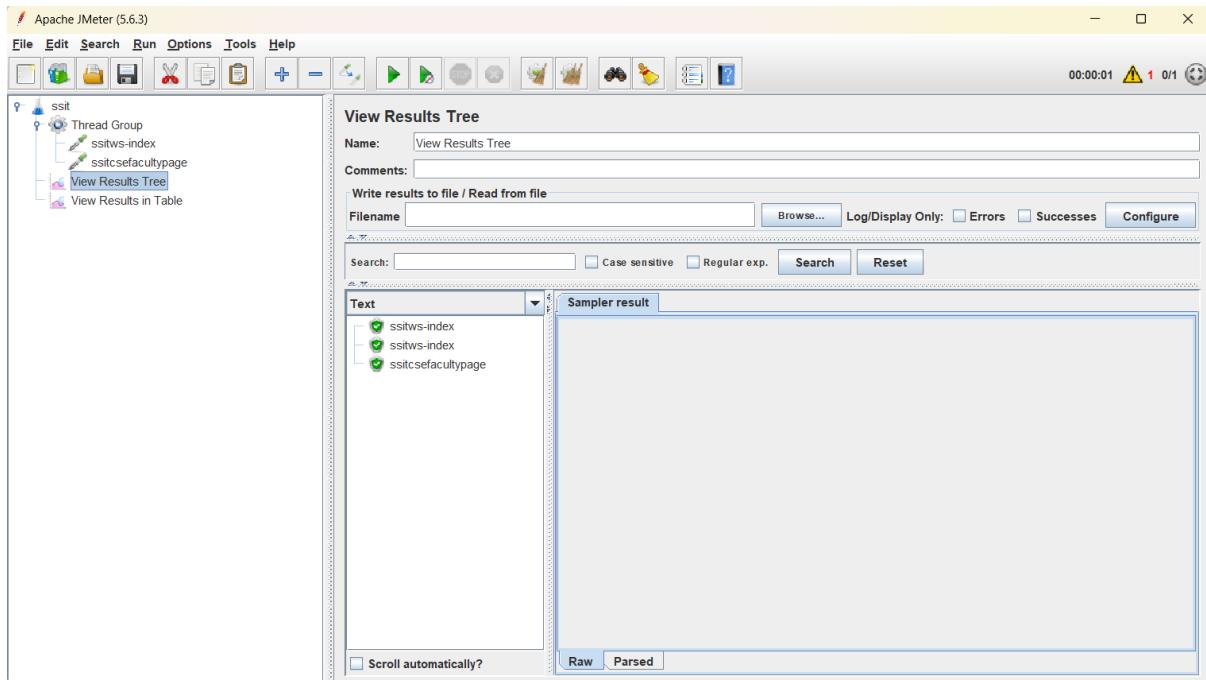
- Then select the http request



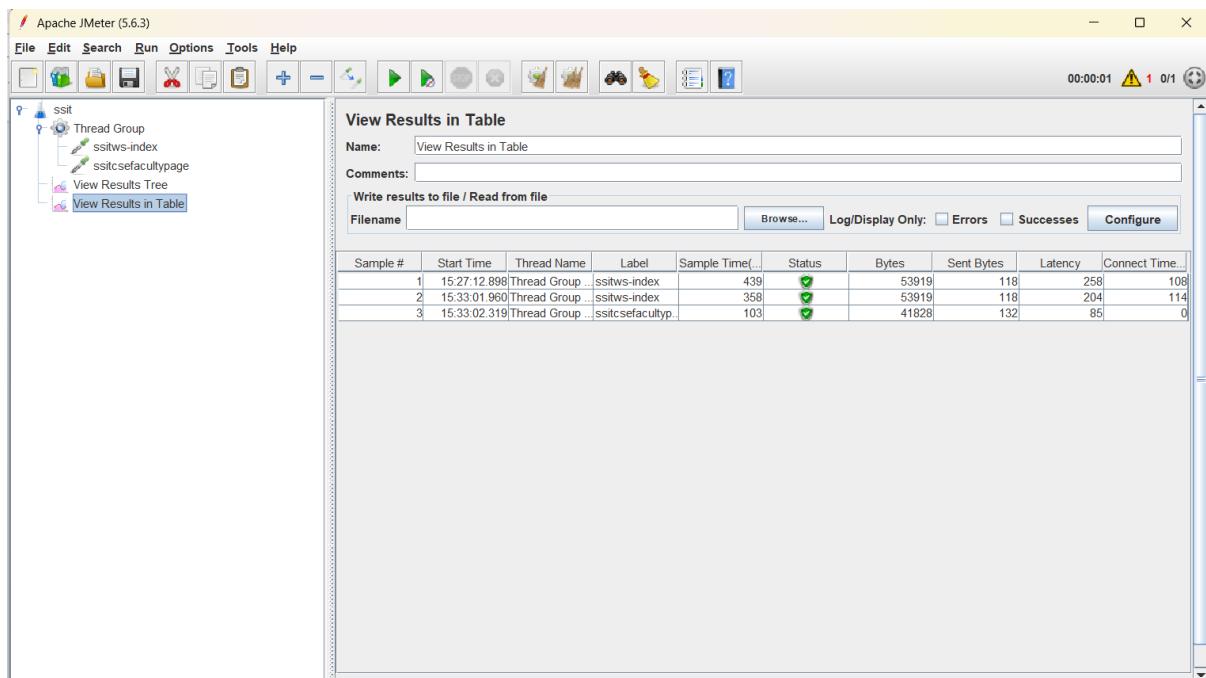
- In http request place the server's name or Ipas saispurthi.ac.in and path as csefaculty.php



- Viewing the results in tree

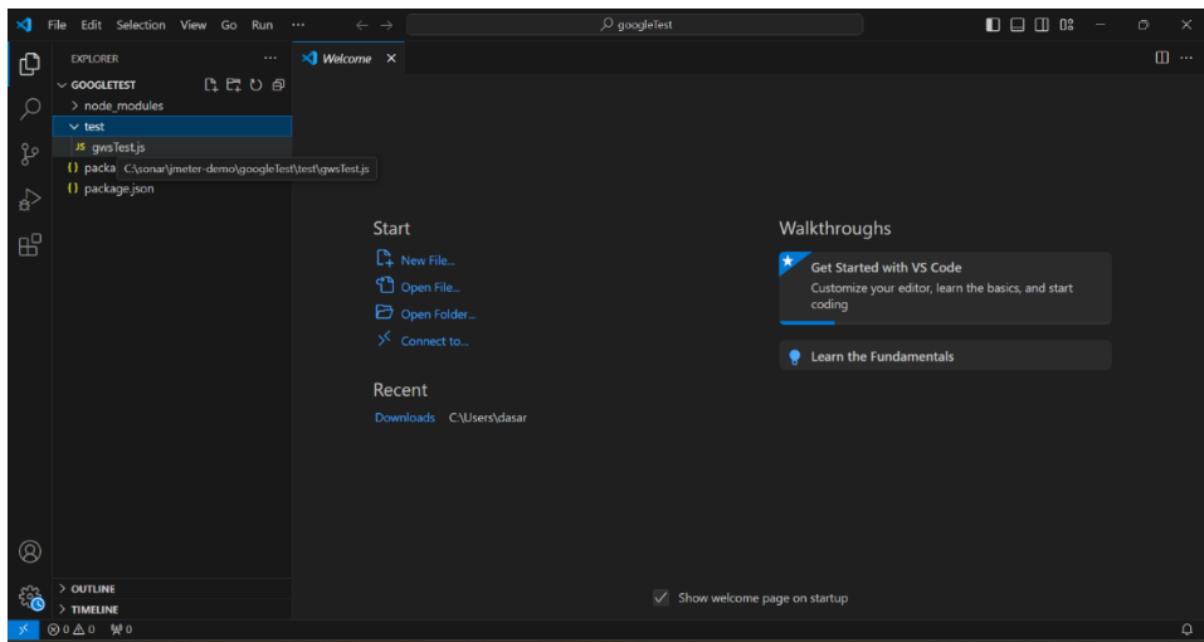


- Viewing the results in table



- 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().forBrowser("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:

The screenshot shows a Google search results page with the query "Automation testing with Selenium". The top result is from BrowserStack, titled "Selenium Testing: Detailed Guide | BrowserStack". A featured snippet is displayed on the right, containing text about Selenium automation and a small 3D illustration of a person working on a laptop with code and charts. Below the main search results, there's a "People also ask" section and a "Selenium" related section.

Automation testing with Selenium

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.

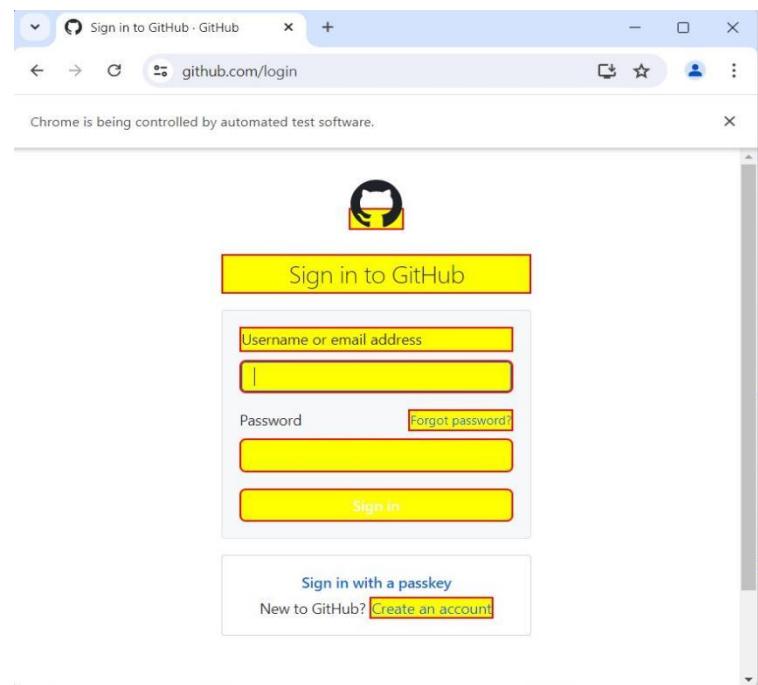
- Testing the code for highlighting the specific words :

Code:

```
package csm.testselenium;
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 :



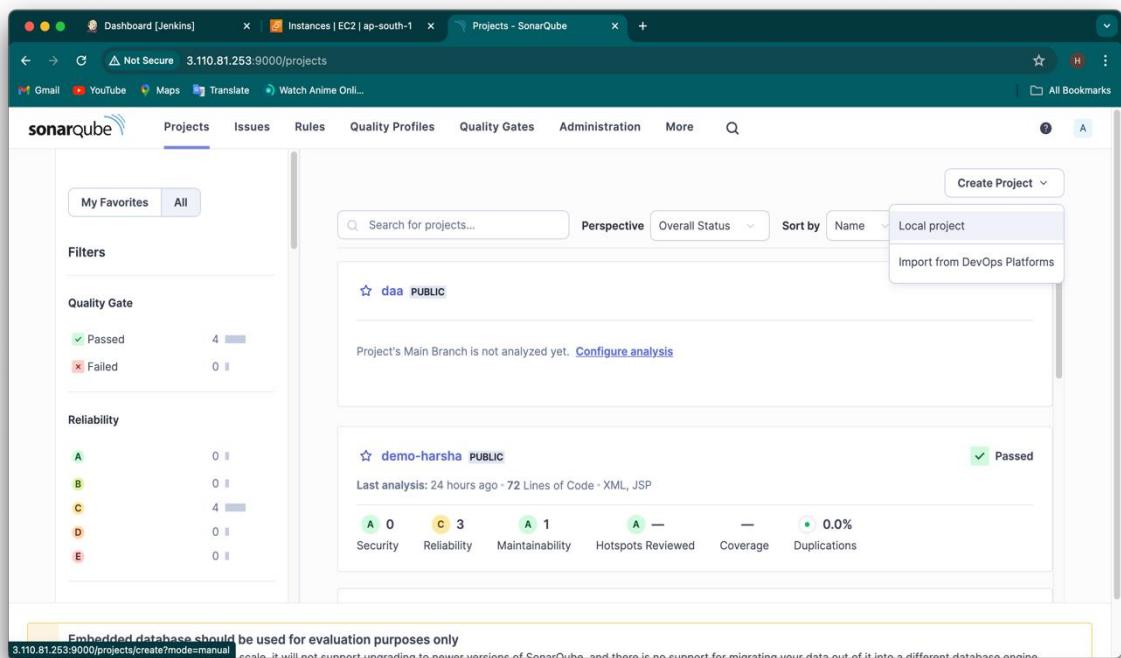
12. Demonstrate the Integration of SonarQube with Jenkins for code analysis.

Step 1: Install SonarQube and SonarQube Scanner

Download and Install SonarQube:

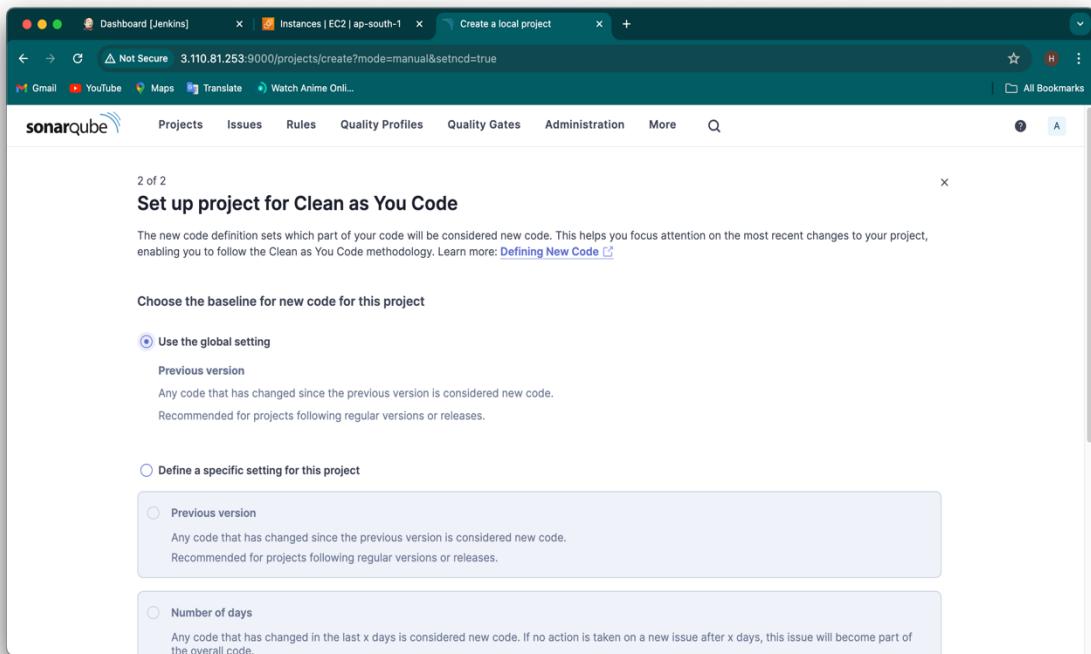
1. **Download SonarQube from the official website:** Visit the SonarQube download page and download the appropriate version for your operating system.
2. **Unzip the file and follow the installation instructions for your operating system:** Unzip the downloaded file. For Windows, unzip it and run StartSonar.bat 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

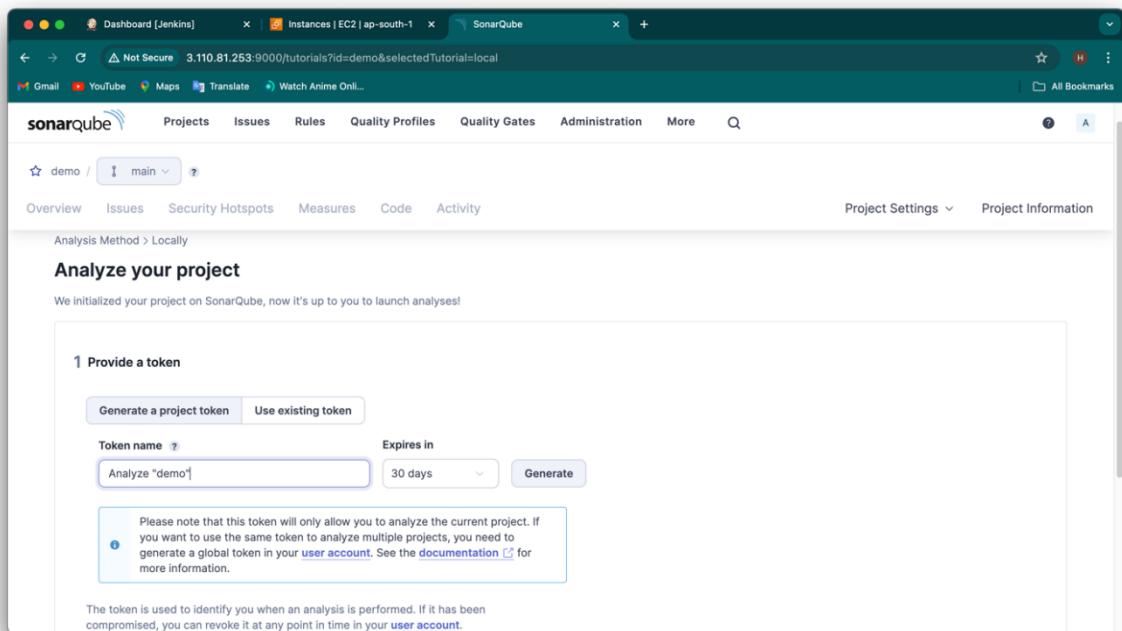


The screenshot shows the SonarQube web interface. At the top, there's a navigation bar with links for Dashboard [Jenkins], Instances | EC2 | ap-south-1, and Projects - SonarQube. Below the navigation is a toolbar with icons for Gmail, YouTube, Maps, Translate, and Watch Anime Onli... A search bar and a 'Create Project' button are also present. The main content area is titled 'sonarqube' and shows two projects: 'daa PUBLIC' and 'demo-harsha PUBLIC'. The 'daa' project has 4 Passed and 0 Failed issues. The 'demo-harsha' project has 0 Security, 3 Reliability, 1 Maintainability, 0 Hotspots Reviewed, 0 Coverage, and 0.0% Duplications. A message at the bottom states: 'Embedded database should be used for evaluation purposes only'.

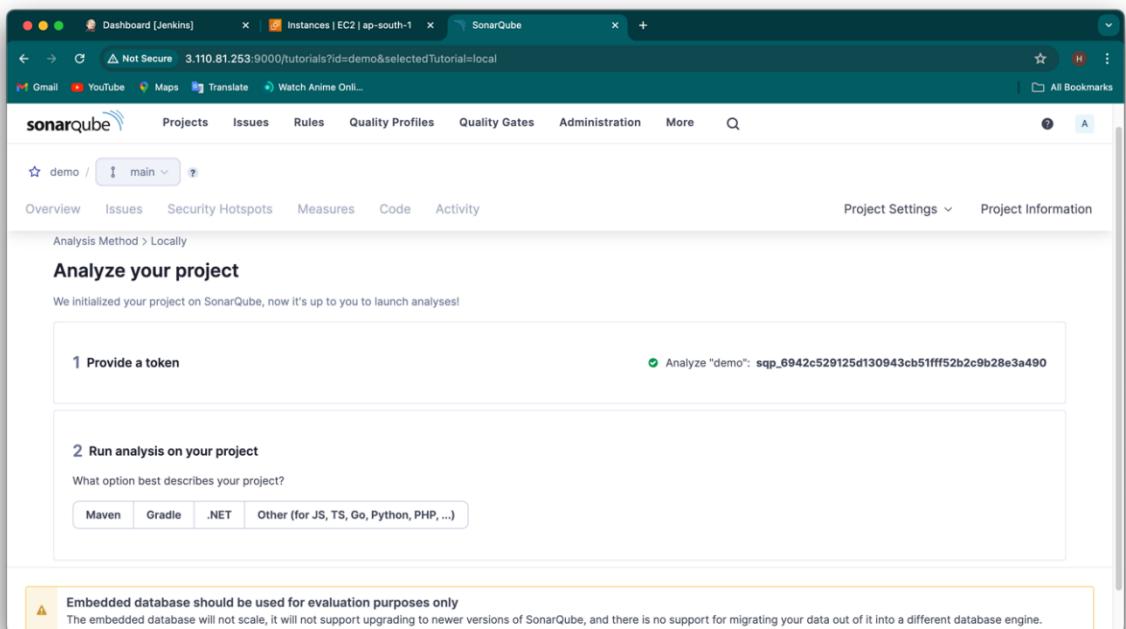
→ Select (use the global setting) and create project



→ **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



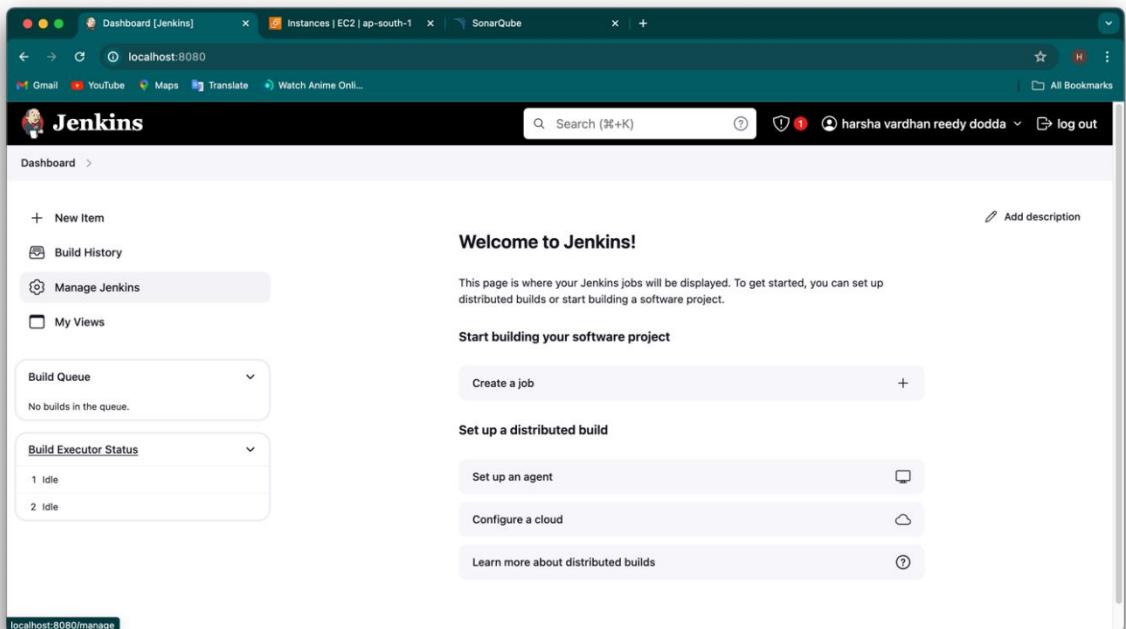
→ **Copy this token:** This token will be used for Jenkins integration.



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.

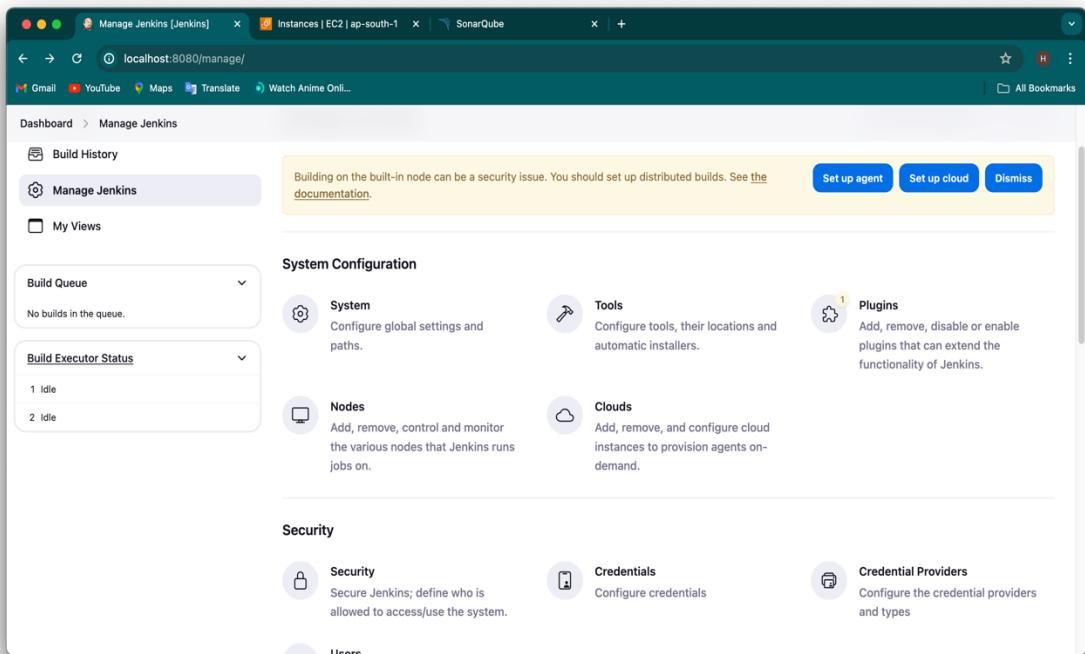
→ Go to the "Available" tab: Search for Sonarqube scanner plugin.

Install	Name	Released
<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.2 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	5 mo 6 days ago
<input type="checkbox"/>	Sonar Gerrit 388.v9b_ffcb_e42306 External Site/Tool Integrations This plugin allows to submit issues from SonarQube to Gerrit as comments directly.	1 mo 20 days ago
<input type="checkbox"/>	SonarQube Generic Coverage 1.0 TODO	4 yr 12 mo ago

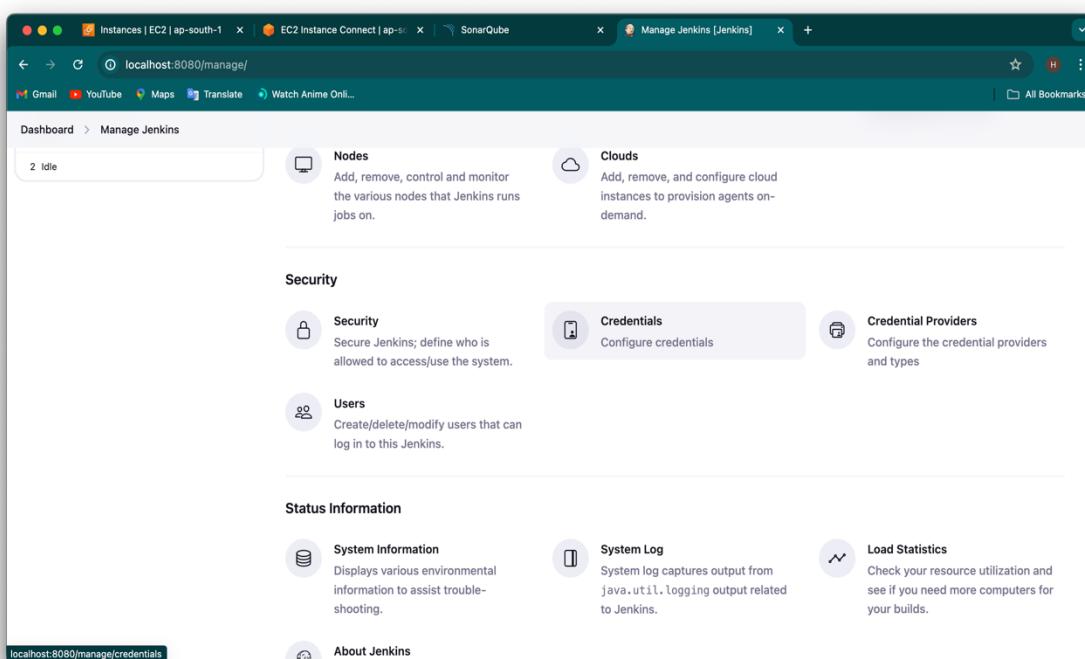
→ Restart Jenkins if required: Some plugin installations require a Jenkins restart to take effect.

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 .

The screenshot shows the Jenkins Global credentials (unrestricted) page. At the top, there is a search bar and a 'log out' button. Below the header, the breadcrumb navigation shows: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted). A blue button labeled '+ Add Credentials' is located in the top right corner. The main content area is titled 'Global credentials (unrestricted)' and contains a message: 'Credentials that should be available irrespective of domain specification to requirements matching.' Below this, there is a table with columns: ID, Name, Kind, and Description. A note in the table says: 'This credential domain is empty. How about adding some credentials?' At the bottom left, there are icon size options: S, M, L. The bottom right corner shows 'REST API' and 'Jenkins 2.452.3'. The address bar at the bottom is 'localhost:8080/manage/credentials/store/system/domain/_newCredentials'.

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

The screenshot shows the 'New credentials' form. The title is 'New credentials'. The 'Kind' dropdown is set to 'Username with password'. Other fields include 'Scope' (set to 'Global (Jenkins, nodes, items, all child items, etc)'), 'Username' (empty), 'Treat username as secret' (unchecked), 'Password' (empty), and a 'Create' button at the bottom.

→ Select the Secret Text

- ✓ Username with password
- SSH Username with private key
- Secret file
- Secret text**
- Certificate

New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: (redacted)

ID: sonarcube

Description:

Create

→ Click On Tools

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

Tools Configure tools, their locations and automatic installers.

Plugins Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

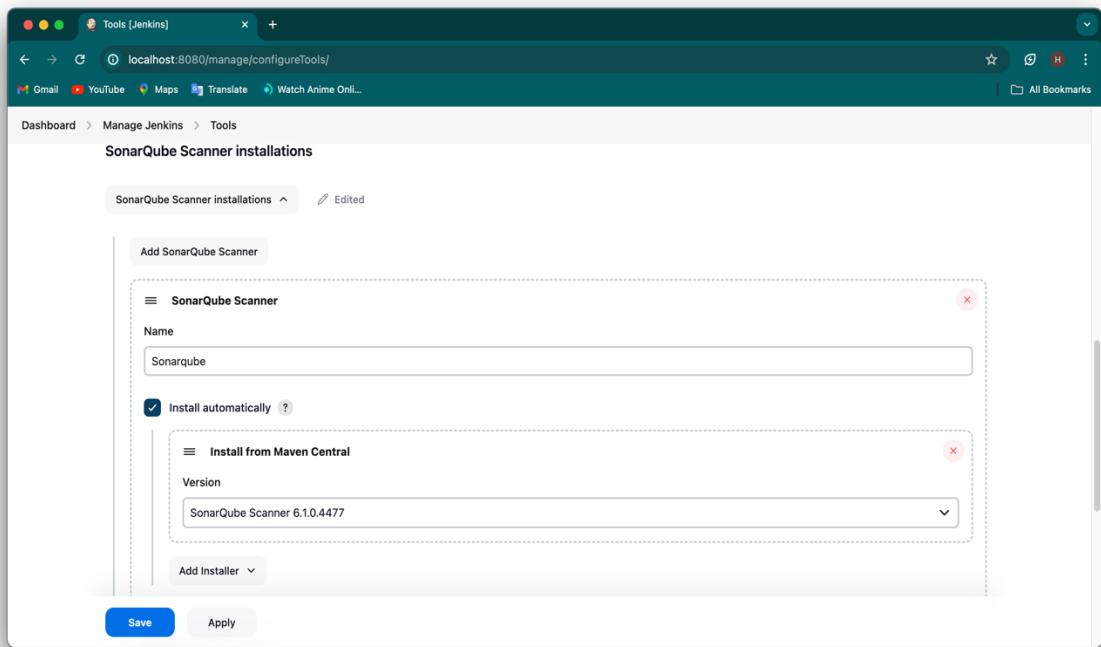
System Configuration

- Build Queue: No builds in the queue.
- Build Executor Status: 1 Idle, 2 Idle
- System: Configure global settings and paths.
- Nodes: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds: Add, remove, and configure cloud instances to provision agents on-demand.
- Security
- Credentials
- Credential Providers

→ Scroll down to the "Sonarqube scanner installations"

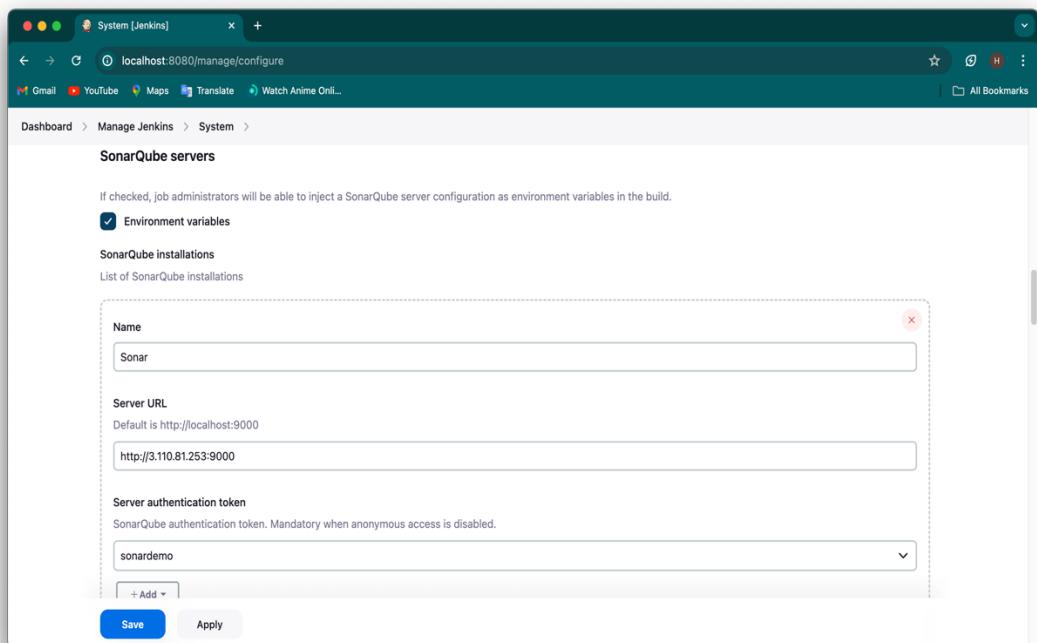
→ Now, Fill the name with installed plugin name i.e, Sonarqube and select the specific version of the plugin then click on 'Save'.

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



Configure SonarQube Scanner:

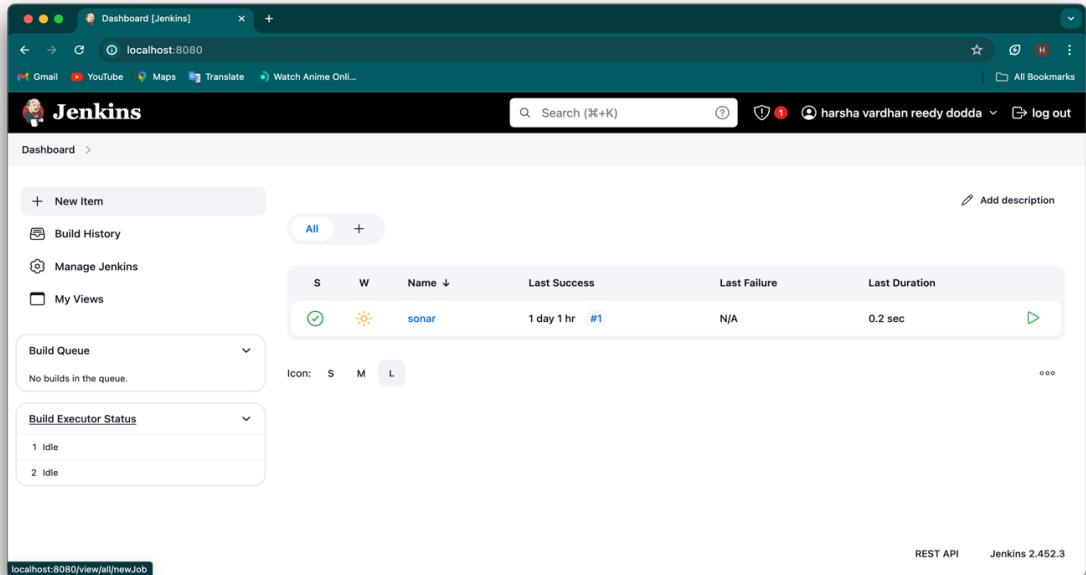
1. **Go to the Jenkins Dashboard:** Open your Jenkins instance.
2. **Click on "Manage Jenkins":** This will take you to the Jenkins management page.
3. **Click on "System":**
4. **Scroll down to the "SonarQube Server" section:** Fill the details and save



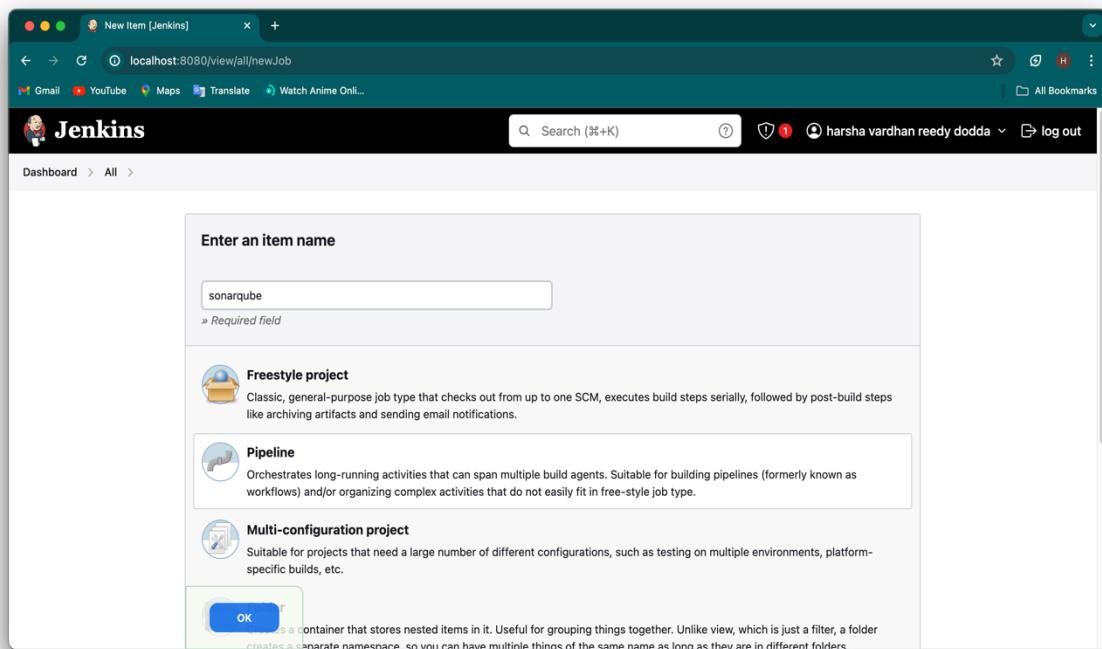
Step 4: Create a Jenkins Pipeline for SonarQube Analysis

Create a New Pipeline Job:

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



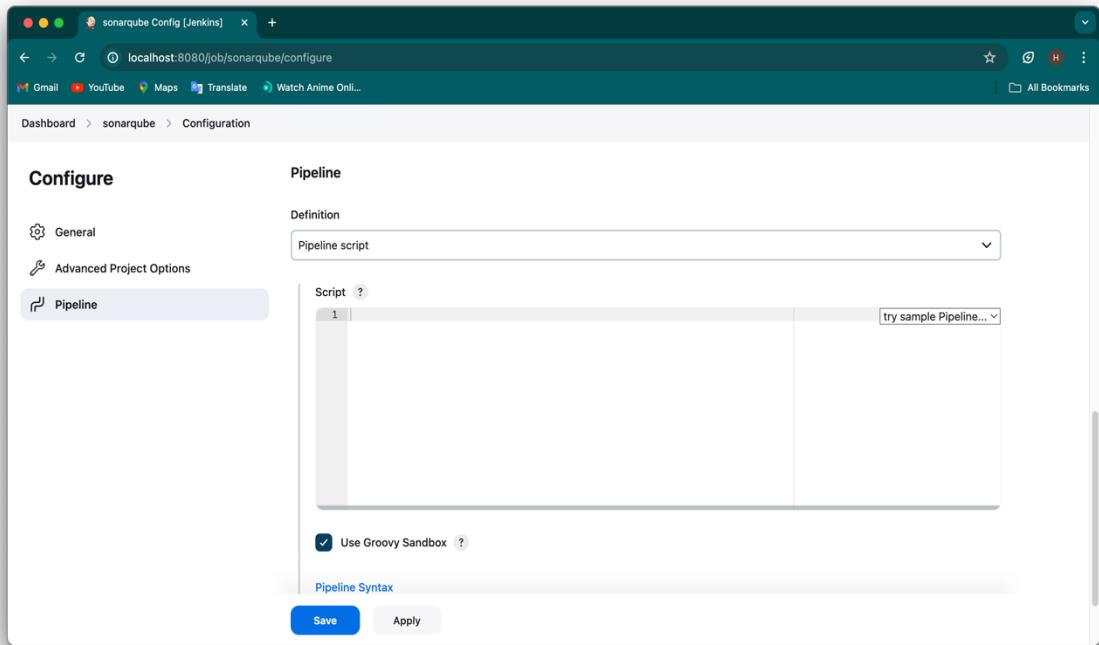
→ Enter a name for your job and select "Pipeline": Provide a name for your pipeline job and select the pipeline option.



→ 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:

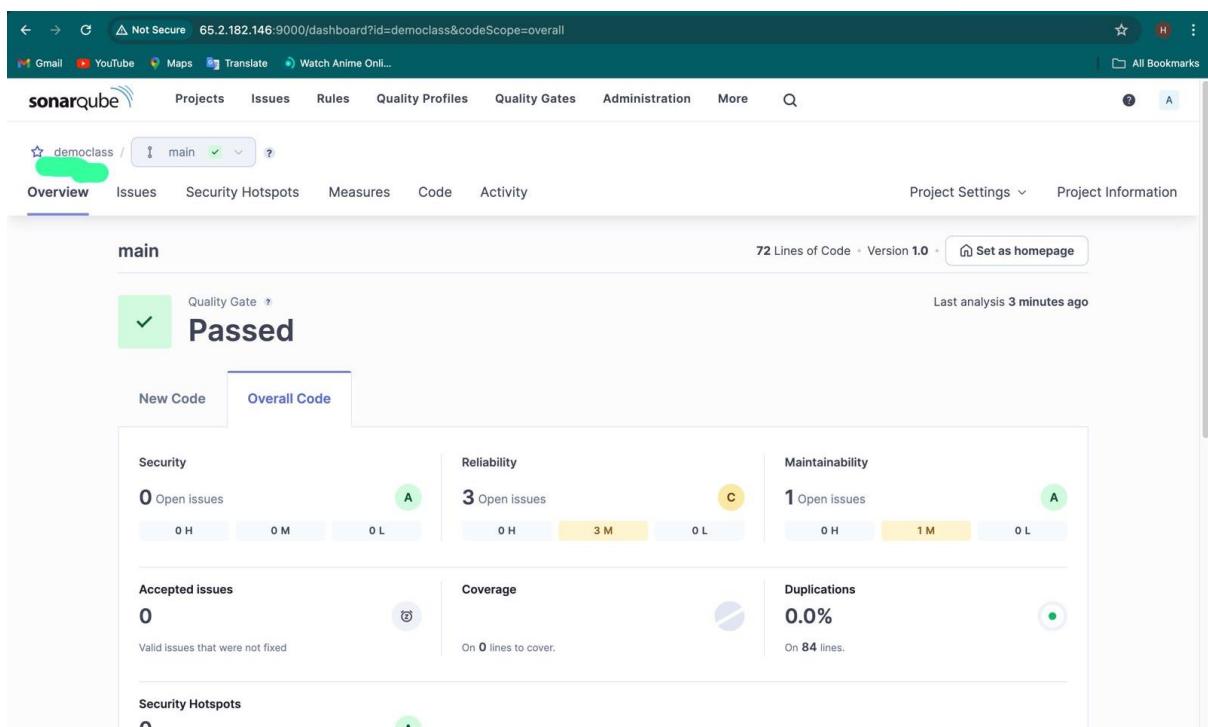
```
pipeline {  
    agent any  
    stages {  
        stage('Fetch Code') {  
            steps {  
                git "https://github.com/harsha5969/web-app"  
            }  
        }  
        stage('Code Analysis') {  
            environment {  
                scannerHome = tool name: 'Sonarqube', 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 Swarm 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.

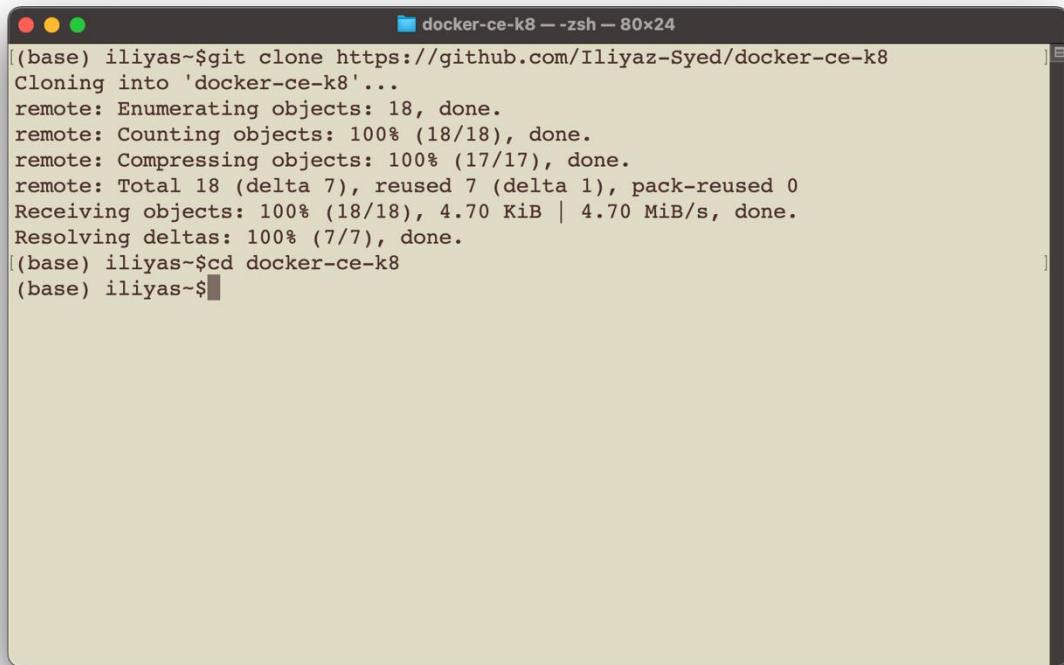
Prerequisites:

- 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
```

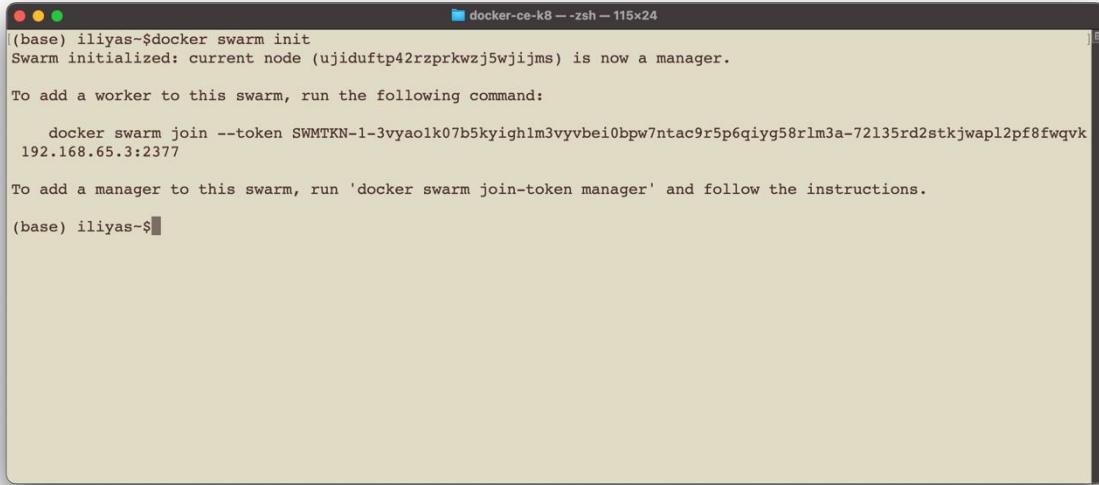


The screenshot shows a terminal window titled "docker-ce-k8 -- zsh - 80x24". The command "git clone https://github.com/Iliyaz-Syed/docker-ce-k8" was run, followed by "cd docker-ce-k8". The output shows the progress of cloning, including object enumeration, counting, compressing, and receiving objects, and finally resolving deltas. The process completed successfully.

```
(base) iliyas-$git clone https://github.com/Iliyaz-Syed/docker-ce-k8  
Cloning into 'docker-ce-k8'...  
remote: Enumerating objects: 18, done.  
remote: Counting objects: 100% (18/18), done.  
remote: Compressing objects: 100% (17/17), done.  
remote: Total 18 (delta 7), reused 7 (delta 1), pack-reused 0  
Receiving objects: 100% (18/18), 4.70 KiB | 4.70 MiB/s, done.  
Resolving deltas: 100% (7/7), done.  
(base) iliyas-$cd docker-ce-k8  
(base) iliyas-$
```

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

```
~$ docker swarm init
```



```
(base) iliyas-$docker swarm init
Swarm initialized: current node (ujiduftp42rzprkwzj5wjijms) is now a manager.

To add a worker to this swarm, run the following command:

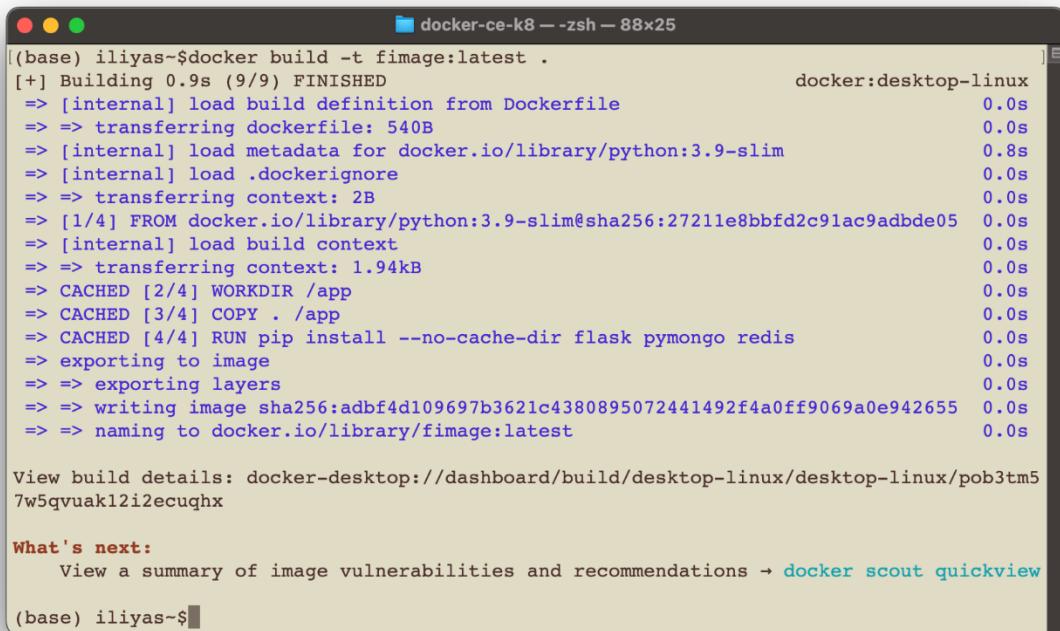
    docker swarm join --token SWMTKN-1-3vyao1k07b5kyigh1m3vyvbei0bpw7ntac9r5p6qiyg58rlm3a-72135rd2stkjwapl2pf8fwqv
192.168.65.3:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

(base) iliyas-$
```

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

```
~$ docker build -t fimage:latest .
```



```
(base) iliyas-$docker build -t fimage:latest .
[+] Building 0.9s (9/9) FINISHED
      docker:desktop-linux
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 540B
=> [internal] load metadata for docker.io/library/python:3.9-slim
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/python:3.9-slim@sha256:27211e8bbfd2c91ac9adbde05
=> [internal] load build context
=> => transferring context: 1.94kB
=> CACHED [2/4] WORKDIR /app
=> CACHED [3/4] COPY . /app
=> CACHED [4/4] RUN pip install --no-cache-dir flask pymongo redis
=> exporting to image
=> => exporting layers
=> => writing image sha256:adbf4d109697b3621c4380895072441492f4a0ff9069a0e942655
=> => naming to docker.io/library/fimage:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/pob3tm5
7w5qvuak12i2ecuqhx

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
(base) iliyas-$
```

→ 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
```

```
(base) iliya$ docker stack deploy -c docker-compose.yml myswarm
Ignoring unsupported options: build

Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network myswarm_default
Creating service myswarm_web
Creating service myswarm_mongo
Creating service myswarm_redis
(base) iliya$
```

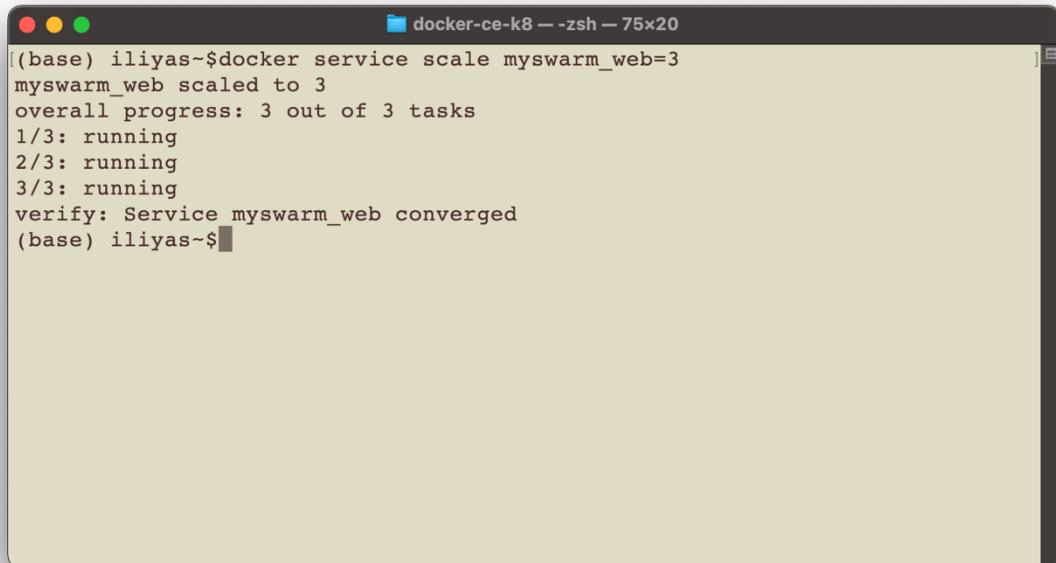
→ Now to monitor the running swarm nodes you can use the following command.
~\$ **docker service ps myswarm_web**

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR	PORT
kf2sb96aychr	myswarm_web.1	fimage:latest	docker-desktop	Running	Running 2 minutes ago		

```
(base) iliya$ docker service ps myswarm_web
ID          NAME      IMAGE      NODE      DESIRED STATE  CURRENT STATE      ERROR      PORT
TS
kf2sb96aychr  myswarm_web.1  fimage:latest  docker-desktop  Running        Running 2 minutes ago
(base) iliya$
```

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

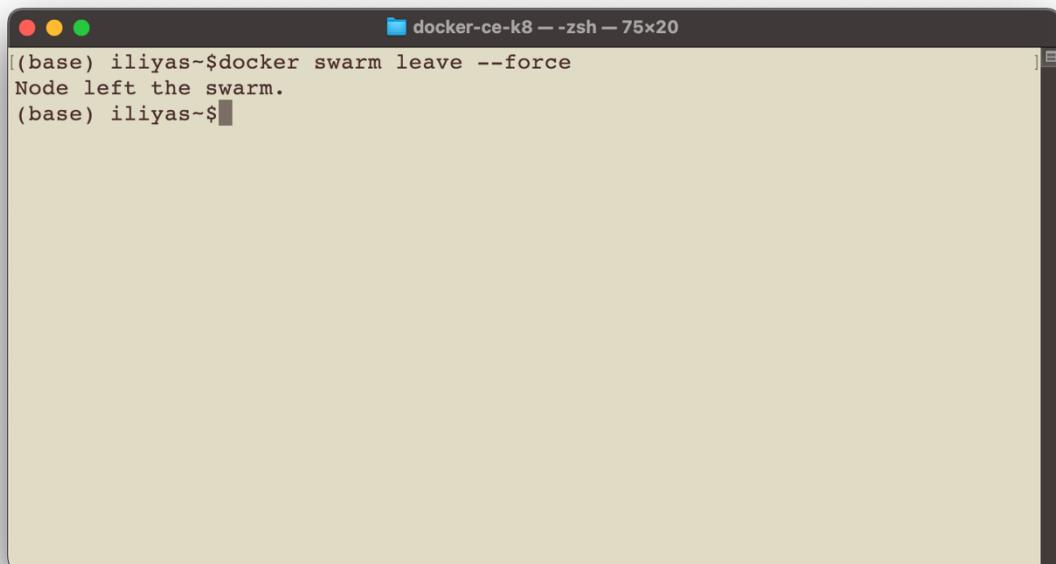
~\$ **docker service scale myswarm_web=3**



```
(base) iliyas-$docker service scale myswarm_web=3
myswarm_web scaled to 3
overall progress: 3 out of 3 tasks
1/3: running
2/3: running
3/3: running
verify: Service myswarm_web converged
(base) iliyas-$
```

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

~\$ docker swarm leave –force



```
(base) iliyas-$docker swarm leave --force
Node left the swarm.
(base) iliyas-$
```

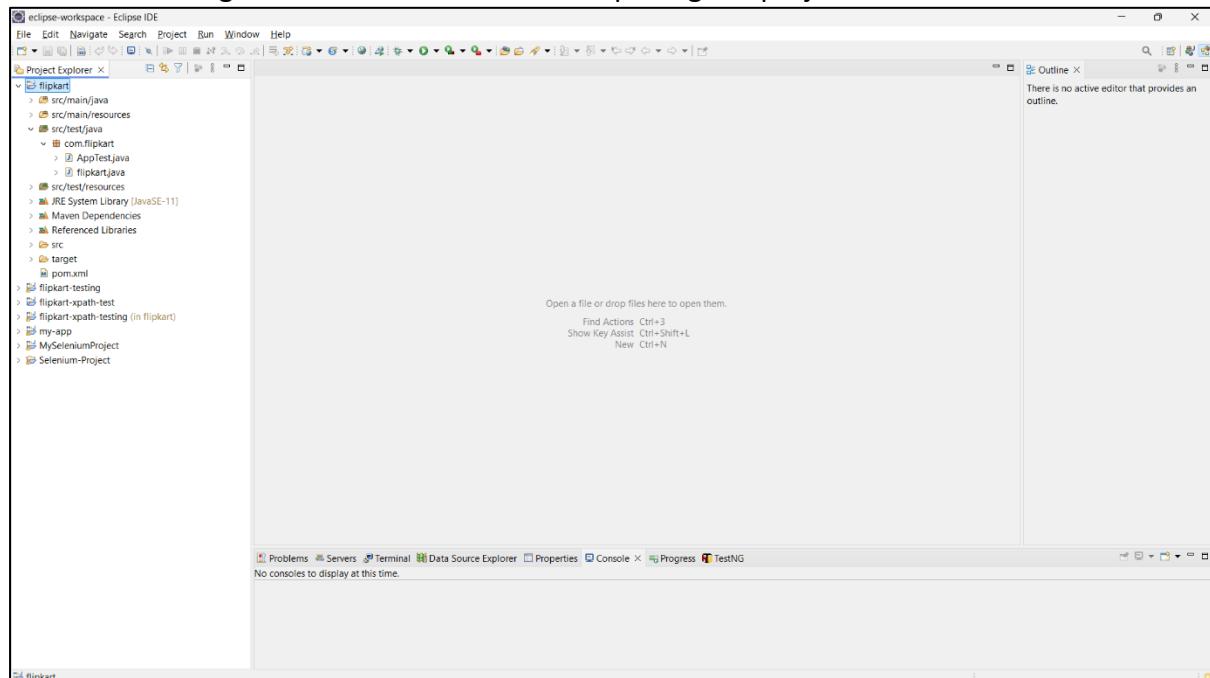
14. Demonstration of XPath using Java Maven Project using Flipkart website.

→ Initially clone the given repo using the following command.

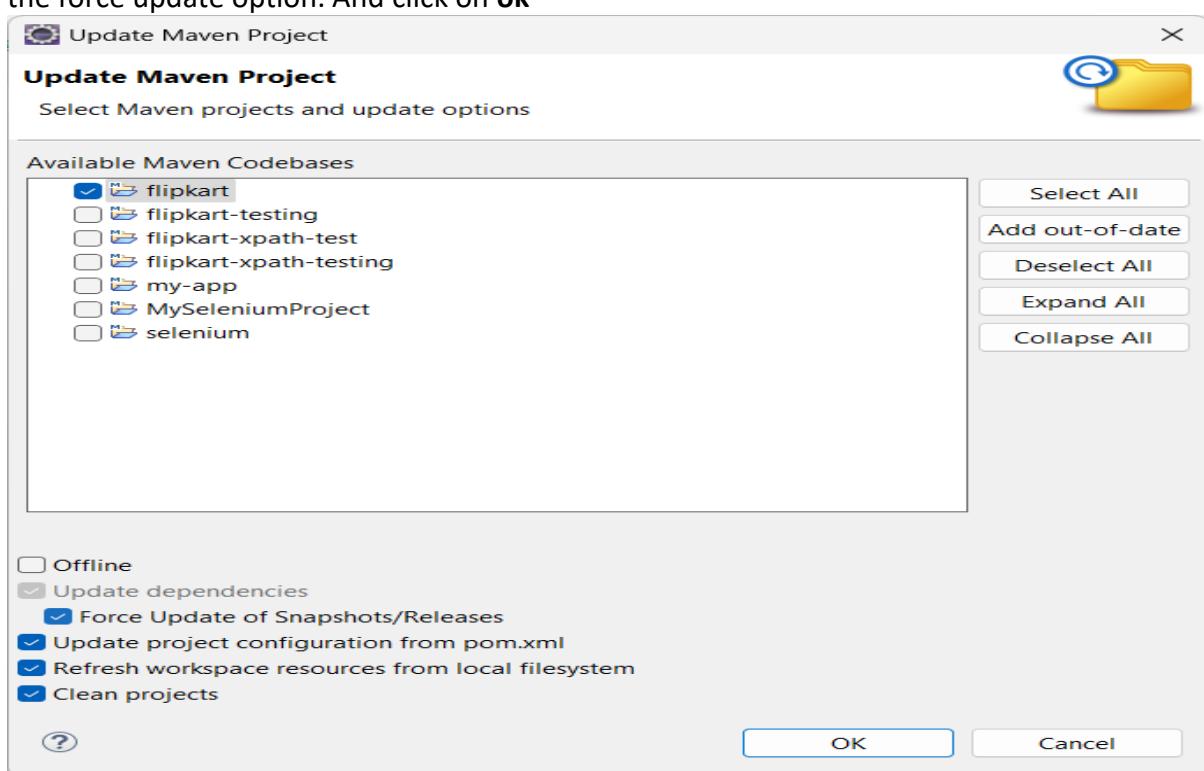
```
~$ git clone https://github.com/UDAYKOTESWARARAOCRAKRAPU/flipkart-xpath.git
```

→ Now open the repo using Eclipse IDE Java Enterprise Edition.

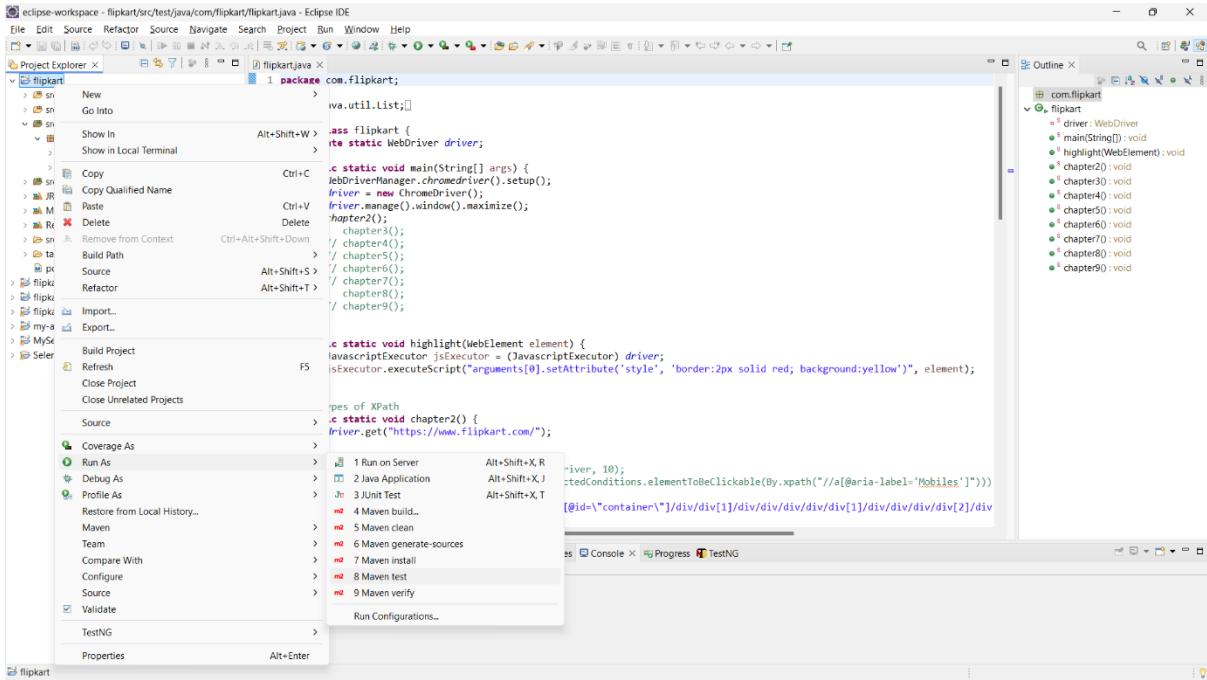
→ The Home Page should Look like this after opening the project.



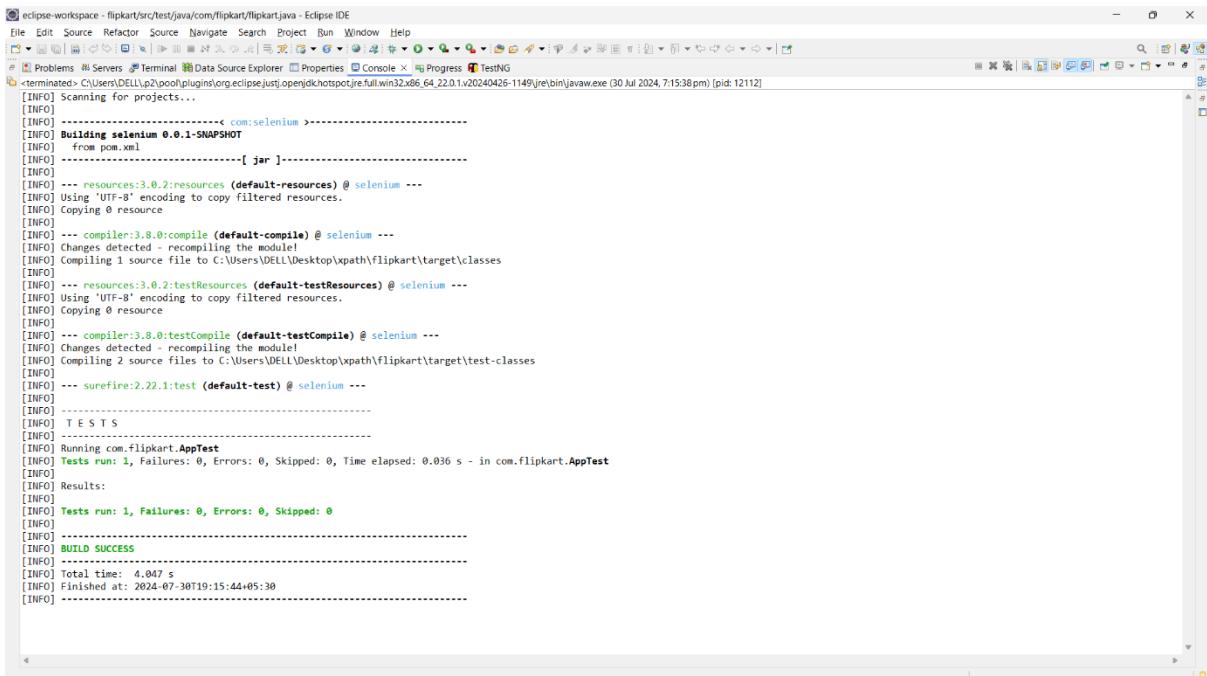
→ Now click on Project in Top bar and click on **Update Maven Project**. Make sure to check the force update option. And click on **ok**



→ 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**

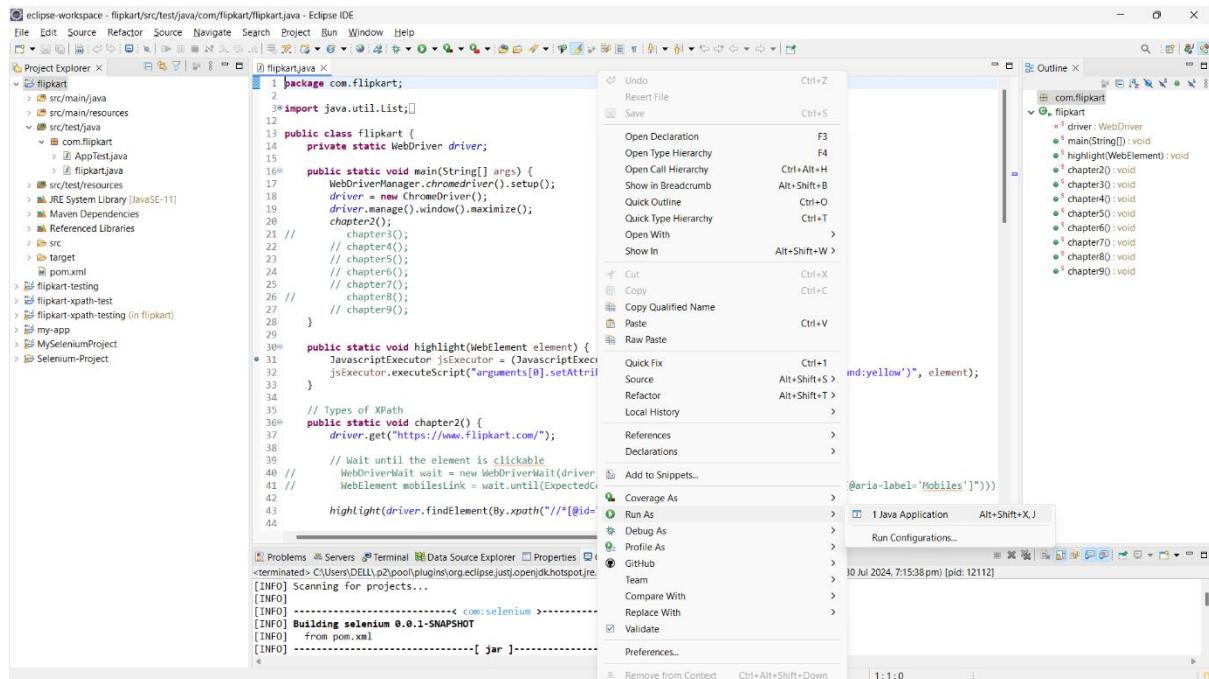


→ Here the build is successful and now we can proceed with running the application.



→ 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.

