

INDEX

SL NO.	PROGRAMS	PAGENO.
01	Explain Git tool in Devops and install git tool and its implementation part.	1 - 14
02	Explain Jenkins tool in Devops and install Jenkins tool and explain the features of Jenkins and explain the features of Jenkins and implementation of Jenkins	15 - 37
03	Explain selenium tools in devops and install selenium tool and explain the features of selenium and implementation of selenium	38 - 50
04	Explain puppet tool in devops and its advantages and features of puppet	51
05	Explain Ansible tool in devops and its advantages and features of Ansible	52
06	Explain Docker tool in devops and its advantages and features of Docker	53
07	Explain saltstack tool in devops and its advantages and features of Saltstack	54
08	Explain Chef tool in devops and its advantages and features of chef	55

1. Explain Git tool in DevOps and install git tool and its implementation part.

Git is an open-source distributed version control system that is freely available for everyone. It is designed to handle minor to major projects with speed and efficiency. It is developed to ordinate the work among programmers.

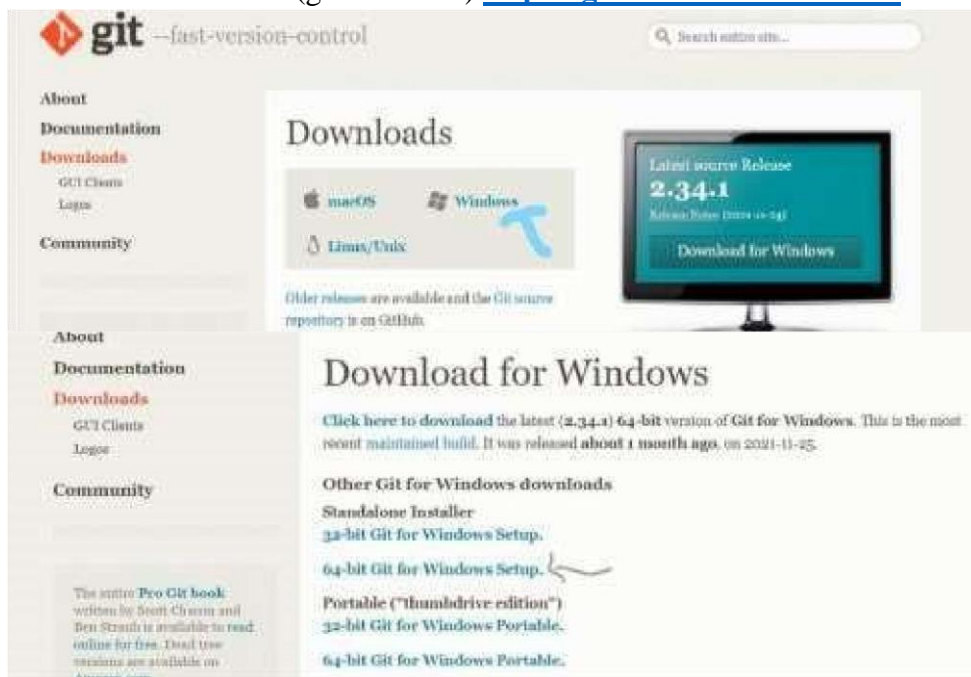
The version control allows you to track and work together with your team members in the same workspace. It is used as a critical distributed version-control for the DevOps tool.

Features

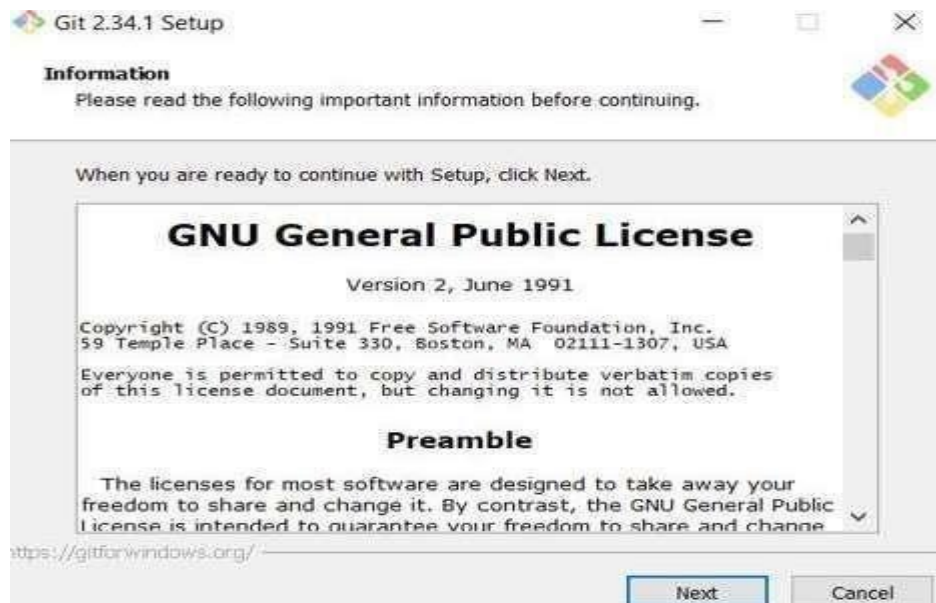
- It is a free open source tool.
- It allows distributed development.
- It supports the pull request.
- It enables a faster release cycle.
- Git is very scalable.
- It is very secure and completes the tasks very fast. **Step 1:**



- Click first link (git-scm.com) <https://git-scm.com/downloads>



- Click Windows option (Depending on your operating system)
- Click 64-bit Git for windows setup
- Download the software exe file



- Open Software exe file
- Click next – next for all the process
- Checking git version in command prompt

```
C:\Users\Shetty>git --version
git version 2.35.1.windows.1
C:\Users\Shetty>
```

Step 2:

- Setting author name and email

```
MINGW64:/c/Users/Shetty/Desktop
Shetty@DESKTOP-3LSVEGU MINGW64 ~/Desktop
$ git config --global user.name yashaswi

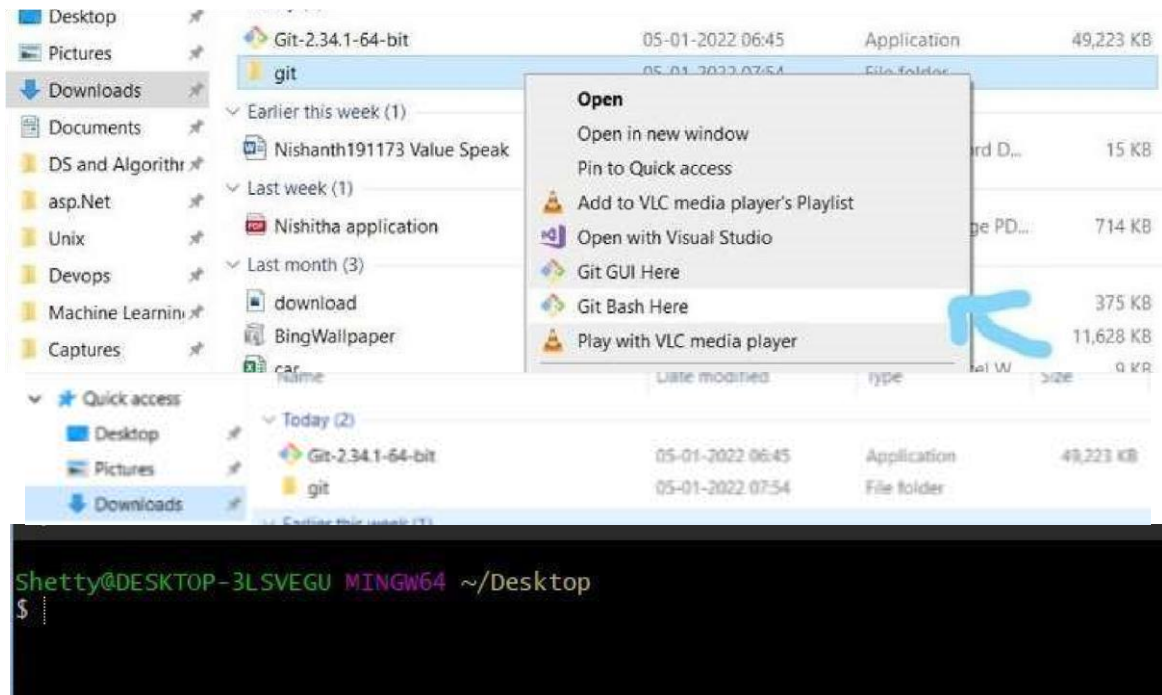
Shetty@DESKTOP-3LSVEGU MINGW64 ~/Desktop
$ git config --global user.email yashaswigowda959@gmail.com

Shetty@DESKTOP-3LSVEGU MINGW64 ~/Desktop
$ git config --global user.name
yashaswi

Shetty@DESKTOP-3LSVEGU MINGW64 ~/Desktop
$ git config --global user.email
yashaswigowda959@gmail.com
```

Step 3:

- Create a file
- Open folder in Git Bash



Step 4:

- Open folder in VS code

```

MINGW64:/c/Users/Shetty/Desktop

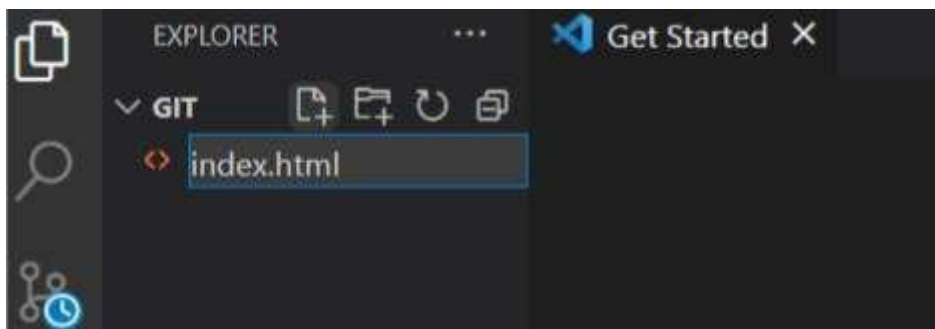
Shetty@DESKTOP-3LSVEGU MINGW64 ~/Desktop
$ code .

Shetty@DESKTOP-3LSVEGU MINGW64 ~/Desktop
$

```

Step 5:

- Create a new file



Step 6:

- Add Data
- Git init
- Initialize empty repository

```

MINGW64/c/Users/Shetty/Desktop

Shetty@DESKTOP-3LSVEGU MINGW64 ~/Desktop
$ code .

Shetty@DESKTOP-3LSVEGU MINGW64 ~/Desktop
$ git init
Initialized empty Git repository in C:/Users/Shetty/Desktop/.git/

```

Step 7:

- Git Status

```

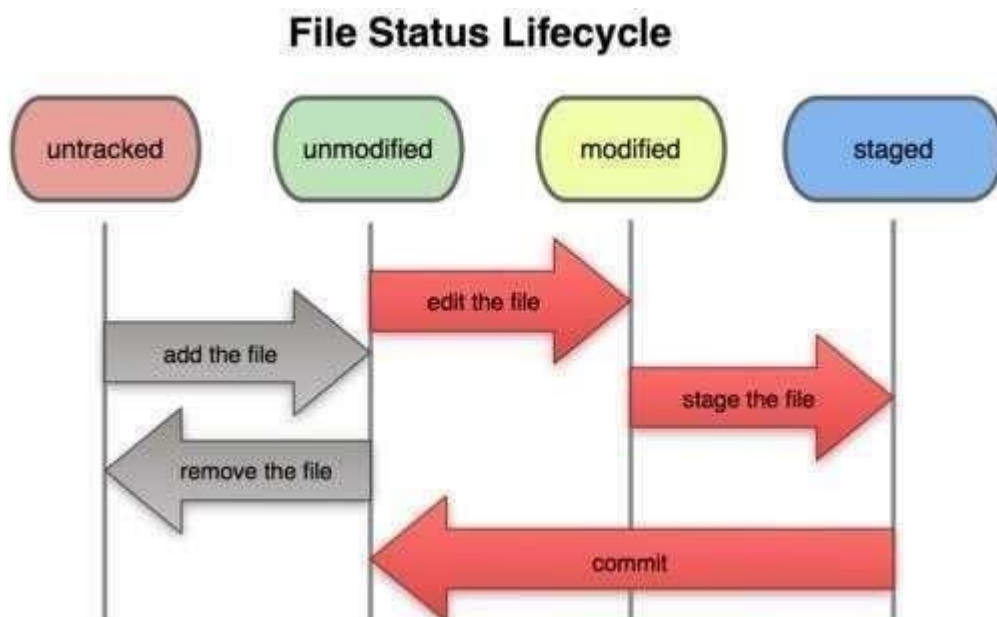
Shetty@DESKTOP-3LSVEGU MINGW64 ~/Desktop (master)
$ git status
On branch master

No commits yet

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

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

```



Step 8:

- Add file to the staged area Git add
- git add -A [to add all files to the staged area]

```

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ git add index.html

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ git status
On branch master

No commits yet

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

```

Step 9:

- Git Commit [Save]

```

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ git commit -m "added first html file"
[master (root-commit) b7e723f] added first html file
1 file changed, 12 insertions(+)
create mode 100644 index.html

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ git status
On branch master
nothing to commit, working tree clean

```

Step 10 :

- Create a file in git
- Touch

```

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ touch contact.html

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ touch about.html

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ touch home.html

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        about.html
        contact.html
        home.html

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

```


Step 11:

- Add all files to staged Area

```
Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ git add -A

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   about.html
    new file:   contact.html
    new file:   home.html
```

Step 12:

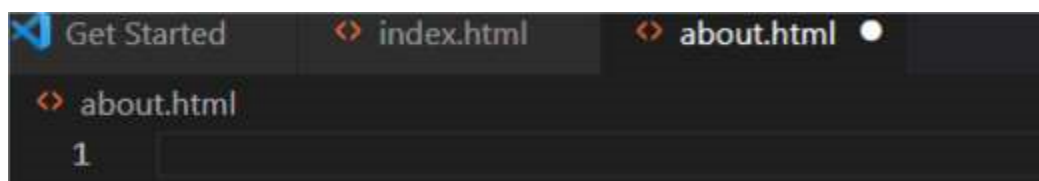
- Save all files [Commit]

```
Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ git commit -m "saved other html files"
[master d8d10c5] saved other html files
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 about.html
create mode 100644 contact.html
create mode 100644 home.html

Nishanth@LAPTOP-E78S7K0S MINGW64 ~/Downloads/git (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Step 13:

- Git notifies



- when data changes in the file


```
Get Started | index.html | about.html X
about.html > html > head > title
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>about</title>
8 </head>
9 <body>
10
11 </body>
12 </html>
```

```
Nishanth@LAPTOP-E7857K05 MINGW64 ~/Downloads/git (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   about.html

no changes added to commit (use "git add" and/or "git commit -a")
```

Step 14:

Git checkout command

- it matches last commit changes
- git checkout -f [to commit all the files]

```
about.html > html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>about</title>
</head>
<body>

</body>
</html>
```

Git diff

- displays the differences between files in two commits or between a commit and your current repository
- git-diff - Show changes between commits, commit, and working tree. Home.html

```
Get Started  index.html  about.html ●
about.html
1  lvfkdfjfkfj|
```

```
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   about.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   about.html
```

```
Nishanth@LAPTOP-E7857K0S MINGW64 ~/Downloads/git (master)
$ git checkout about.html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>about</title>
</head>
<body>

</body>
</html>
```

```
Get Started  index.html  home.html X
home.html > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <title>Document</title>
8  </head>
9  <body>
10
11 </body>
12 </html>
```

Home.html (Edited)

```
Get Started  index.html  home.html X
home.html > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body nishanth>
10
11 </body>
12 </html>
```

```
Nishanth@LAPTOP-E78S7K05 MINGW64 ~/Desktop/git2 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   home.html

no changes added to commit (use "git add" and/or "git commit -a")
```

```
Nishanth@LAPTOP-E78S7K05 MINGW64 ~/Desktop/git2 (master)
$ git diff
diff --git a/home.html b/home.html
index 56efbdb..8c35eac 100644
--- a/home.html
+++ b/home.html
@@ -6,7 +6,7 @@
     <meta name="viewport" content="width=device-width, initial-scale=1.0">
     <title>Document</title>
 </head>
- <body>
+ <body nishanth>

</body>
</html>
\ No newline at end of file
```

Step 15:

□ Git log

```
Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Downloads/git (master)
$ git log
commit d8d10c5245e7963e28dda4c51456d7ce61c688ea (HEAD -> master)
Author: nishanth <nishukiran43@gmail.com>
Date:   Wed Jan 5 08:15:21 2022 +0530

    saved other html files

commit b7e723f13c8e290a5268f91b7154620c334d379c
Author: nishanth <nishukiran43@gmail.com>
Date:   Wed Jan 5 08:09:56 2022 +0530

    added first html file
```

```
Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Downloads/git (master)
$ git log -p -1
commit d8d10c5245e7963e28dda4c51456d7ce61c688ea (HEAD -> master)
Author: nishanth <nishukiran43@gmail.com>
Date:   Wed Jan 5 08:15:21 2022 +0530

    saved other html files

diff --git a/about.html b/about.html
new file mode 100644
index 0000000..e69de29
diff --git a/contact.html b/contact.html
new file mode 100644
index 0000000..e69de29
diff --git a/home.html b/home.html
new file mode 100644
index 0000000..e69de29
```

Step 16: ls

- it shows all the files in the directory

```
Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Desktop/git2 (master)
$ ls
about.html  contact.html  home.html  index.html
```

Step 17:

GitHub



Step 18:

- Click plus symbol to create a new repository





- Click Create repository button to create a gittutorial2 repository

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner ^{*} Repository name ^{*}


 nishanthujire / gittutorial2 

Great repository names are short and memorable. Need inspiration? How about [redesigned-octo-sniffle?](#)

Description (optional)

git seminar

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

[Create repository](#)

Step 19: copy repo

URL

...or create a new repository on the command line

```
echo "# gittutorial" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/nishanthujire/gittutorial.git
git push -u origin main
```

```

Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Desktop/git (master)
$ git remote add origin https://github.com/nishanthujire/gittutorial.git

Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Desktop/git (master)
$ git remote
origin

Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Desktop/git (master)
$ git remote -v
origin https://github.com/nishanthujire/gittutorial.git (fetch)
origin https://github.com/nishanthujire/gittutorial.git (push)

```

Step 20:

- Connecting local repository to remote repository

Step 21:

Git push

- The git push command is used to upload local repository content to a remote repository

```

Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Desktop/git (master)
$ git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (9/9), 883 bytes | 294.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/nishanthujire/gittutorial.git
 * [new branch]      master -> master

```

The screenshot shows the GitHub interface for the repository 'nishanthujire/gittutorial.git'. At the top, there are navigation tabs: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below these, the repository name is displayed with 'master' as the selected branch, '1 branch', and '0 tags'. There are buttons for 'Go to file', 'Add file', and 'Code'. The commit history shows a single commit by 'nishanthujire' titled 'added other html files' from 26 days ago, with 3 commits in total. Below the commit history, a table lists the files added in this commit:

File	Commit Message	Time
about.html	added other html files	26 days ago
contact.html	added other html files	26 days ago
document.html	added other html files	26 days ago
index.html	files add	26 days ago

At the bottom, there is a prompt to 'Help people interested in this repository understand your project by adding a README.' with an 'Add a README' button.

Step 22:

□ Git Clone

```
Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Desktop/github (master)
$ git clone https://github.com/nishanthujire/gitseminar.git sam
Cloning into 'sam'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 8 (delta 0), reused 8 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.

Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Desktop/github (master)
$ ls
sam/

Nishanth@LAPTOP-E78S7KOS MINGW64 ~/Desktop/github (master)
$ ls sam
about.html  contact.html  home.html  index.html
```

2. Explain Jenkins tool in DevOps and install Jenkins tool and explain the features of Jenkins and explain the features of Jenkins and implementation of Jenkins

Jenkins is an open-source automation tool written in Java programming language that allows continuous integration.

Jenkins builds and tests our software projects which continuously makes it easier for developers to integrate changes to the project, and makes it easier for users to obtain a fresh build. It also allows us to continuously deliver our software by integrating with a large number of testing and deployment technologies. **Features**

- It is an open-source tool.
- It is free of cost.
- It does not require additional installations or components. Means it is easy to install.
- Easily configurable.
- It supports 1000 or more plugins to ease your work. If a plugin does not exist, you can write the script for it and share it with the community. • It is built in java and hence it is portable.
- It is platform-independent. It is available for all platforms and different operating systems. Like OS X, Windows or Linux.
- Easy support, since its open-source and widely used.
- Jenkins also supports cloud-based architecture so that we can deploy Jenkins in cloudbased platforms.
-

Step 1:

Install Java Version 8

□ To download the Java **Click here**. Select the file according to your platform.

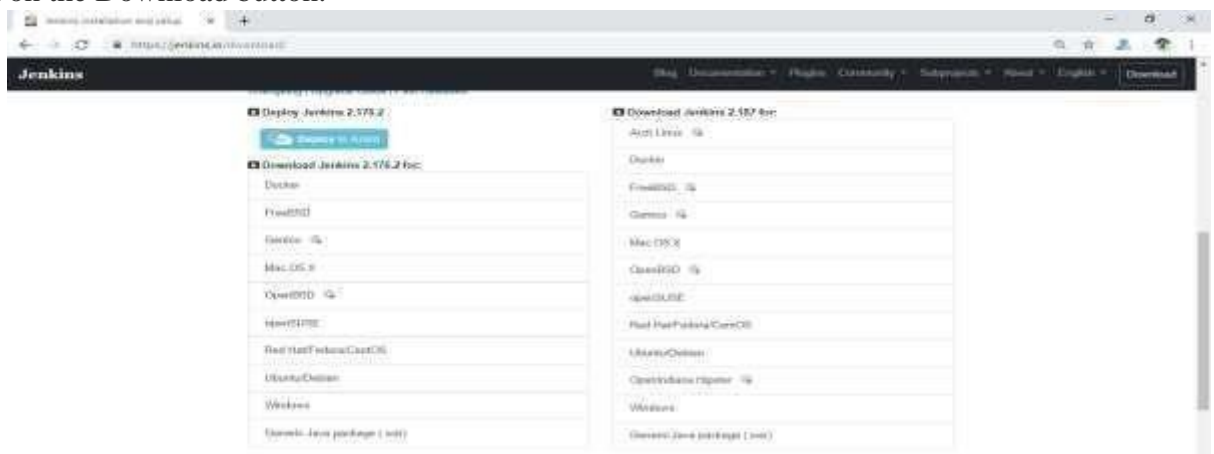
Jenkins requires Java versions [8, 11]

Download Jenkins war File

- This war is required to install Jenkins.
- The official website for Jenkins is <https://jenkins.io/>



Click on the Download button.



- Click on Generic Java Package (.war) to download the Jenkins war file.

Step 2:

- Open the command prompt and go to the directory where the Jenkins.war file is located.
And then run the following command:
- `C:/Java -jar Jenkins.war`
- When you run this command, various tasks will run, one of which is the extraction of the war file which is done by an embedded webserver called winstone.

```
C:\Users\Shetty\Desktop\Jenkin>java -jar jenkins.war
Running from: C:\Users\Shetty\Desktop\Jenkin\jenkins.war
webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
```

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

INFO: Jenkins is fully up and running



Click on **Allow access** button to allow access.

Step 3:

Unlocking Jenkins

- When you first access a new Jenkins instance, you are asked to unlock it using an automatically generated password.
- Browse to localhost:8080 (or whichever port you configured for Jenkins when installing it) and wait until the Unlock Jenkins page appears.
- From the Jenkins console log output, copy the automatically-generated alphanumeric password (between the 2 sets of asterisks).
- On the Unlock Jenkins page, paste this password into the Administrator password field and click Continue.



Then click on continue.

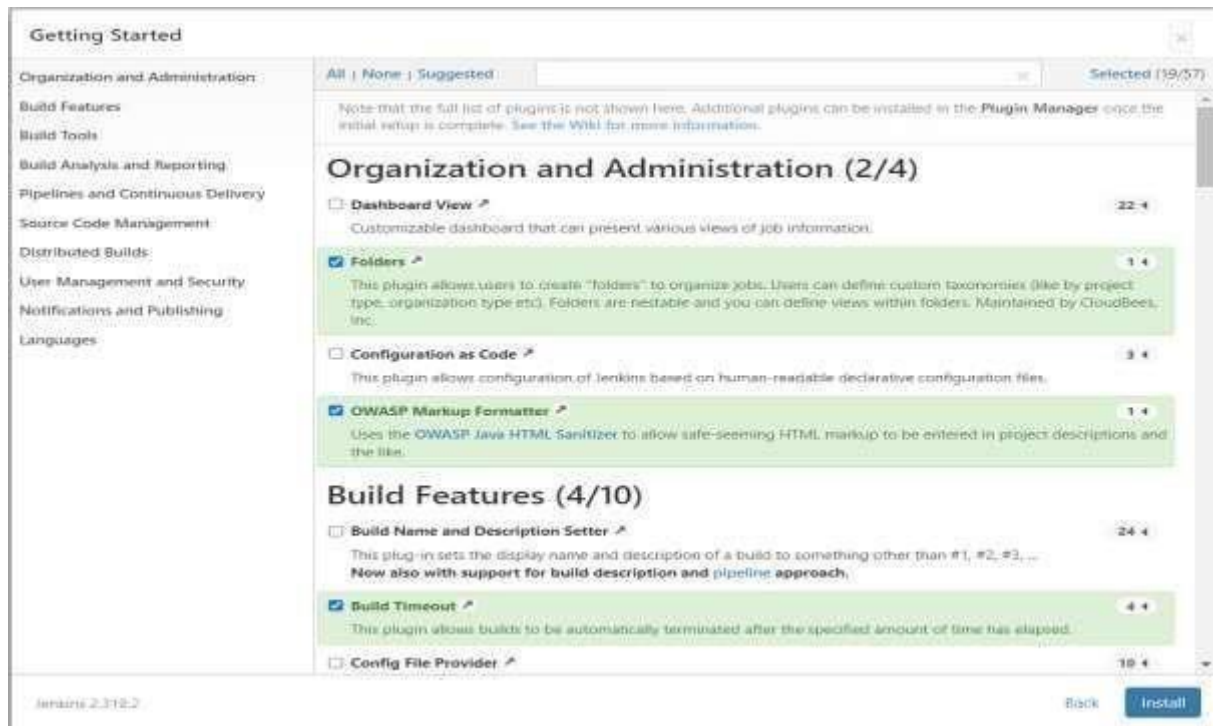
Step 4: Customizing Jenkins with plugins

After unlocking Jenkins, the Customize Jenkins page appears. Here you can install any number of

Click one of the two options shown:

- Install suggested plugins - to install the recommended set of plugins, which are based on the most common use cases.
- Select plugins to install - to choose which set of plugins to initially install. When you first access the plugin selection page, the suggested plugins are selected by default.





Click on install

Step 5:

Creating the first administrator user

Finally, after customizing Jenkins with plugins, Jenkins asks you to create your first administrator user.

1. When the Create First Admin User page appears, specify the details for your administrator user in the respective fields and click Save and Finish.
2. When the Jenkins is ready page appears, click Start using Jenkins. Notes:
 - This page may indicate Jenkins is almost ready! instead and if so, click Restart.
 - If the page does not automatically refresh after a minute, use your web browser to refresh the page manually.
3. If required, login to Jenkins with the credentials of the user you just created and you are ready to start using Jenkins!

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.319.2

[Skip and continue as admin](#) [Save and Continue](#)

- If you create a new user you will be continued as a new user or Skip this and continue as admin.

Getting Started

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.319.2

[Not now](#) [Save and Finish](#)

- Click on save and finish

Jenkins is ready!

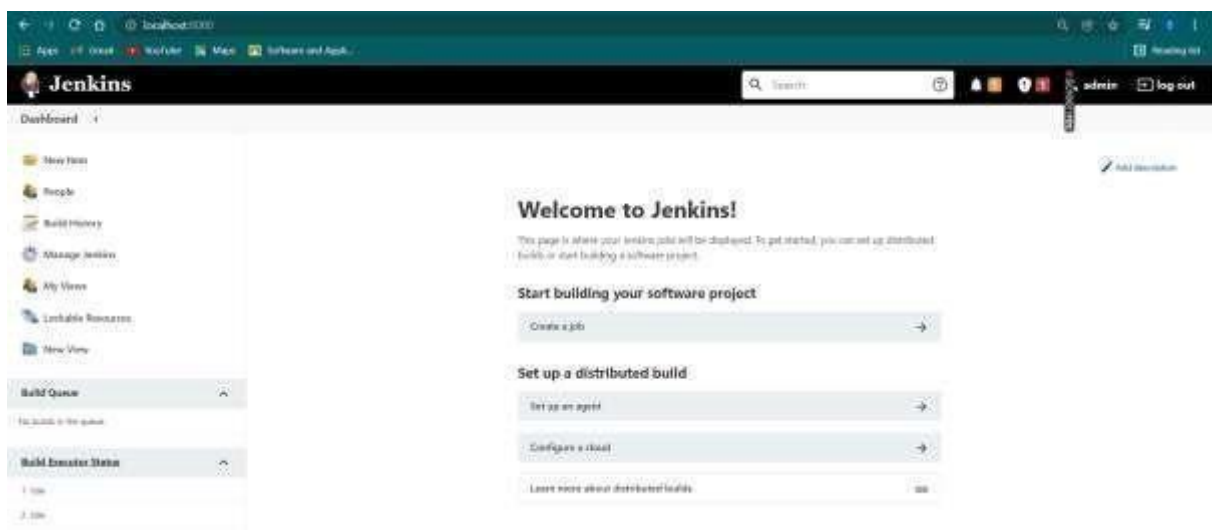
You have skipped the **setup of an admin user**.

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

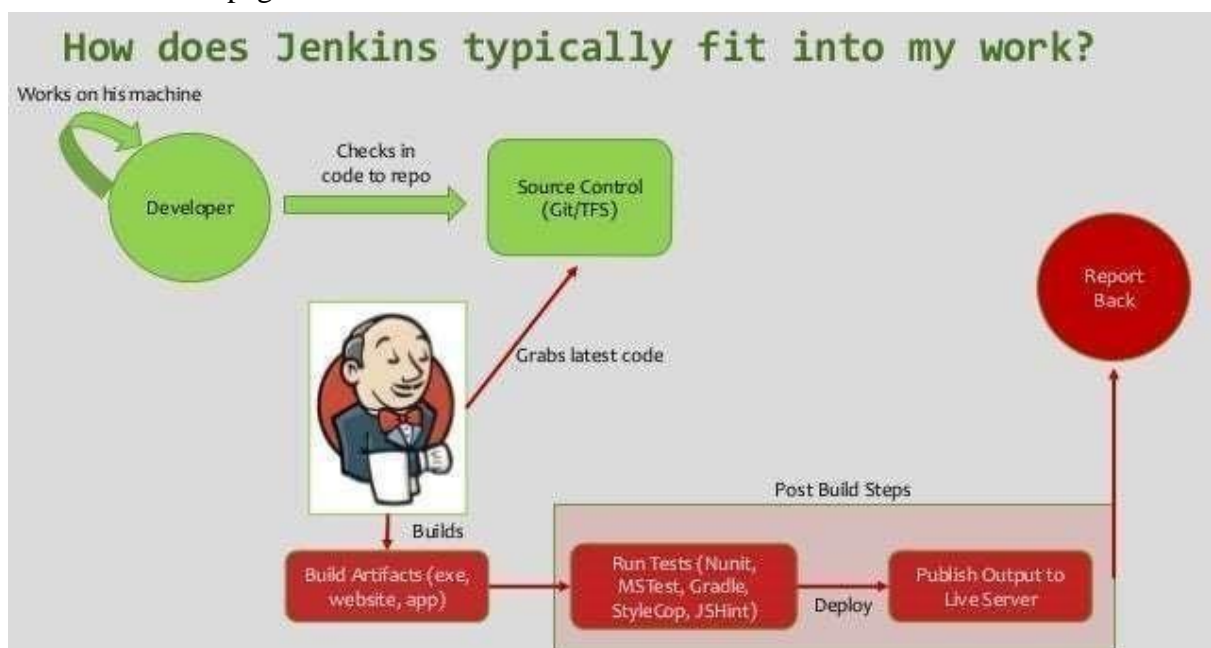
Your Jenkins setup is complete.

Start using Jenkins

- Click on start using Jenkins



- Jenkins home page.



Jenkins home directory: Back up the folder, and you've backed up the entire server. Delete this folder and the next time you run the Jenkins WAR file it will assume it's a brand new installation and configure itself from scratch. The importance and significance of the Jenkins Home folder can not be .jenkins home directory path

C:\Users\Owner\.jenkins

Example : C:\Users\shetty\.jenkins

```
C:\Users\Shetty\Desktop\Jenkin>java -jar jenkins.war
Running from: C:\Users\Shetty\Desktop\Jenkin\jenkins.war
webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
2022-03-06 02:21:01.445+0000 [id=1] INFO org.eclipse.jetty.util.log.Log#initialized: Logging initialized @3529ms
to org.eclipse.jetty.util.log.JavaUtilLog
2022-03-06 02:21:01.806+0000 [id=1] INFO winstone.Logger#logInternal: Beginning extraction from war file
2022-03-06 02:21:02.013+0000 [id=1] WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2022-03-06 02:21:02.183+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: jetty-9.4.43.v20210629; built:
2021-06-30T11:07:22.254Z; git: 526006ecfa3af7f1a27ef3a288e2bef7ea9dd7e8; jvm 1.8.0_321-b07
2022-03-06 02:21:04.301+0000 [id=1] INFO o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /,
```

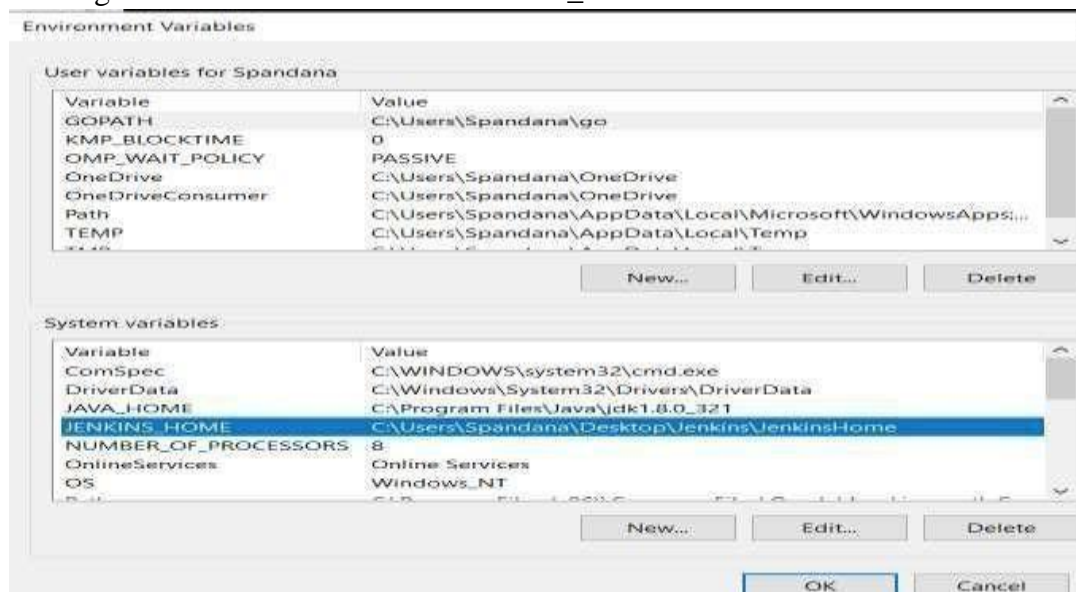
How to change the home directory?

Step 1: Check your current home directory

1. Navigate to your folder where Jenkins.war is present.
2. Java -jar jenkins.war

Step 2: Create a new folder (Which will be a new home directory) Step 3: Copy all data from the old directory to the new directory.

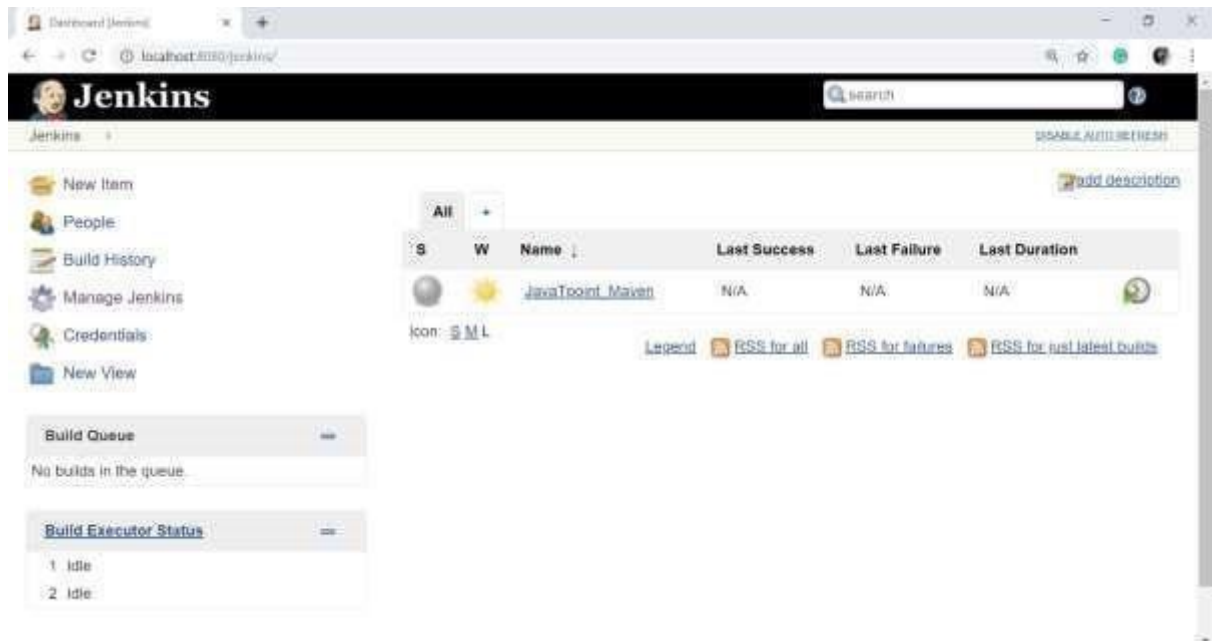
Step 4: Change environment variable – JENKINS_HOME and set to new dir



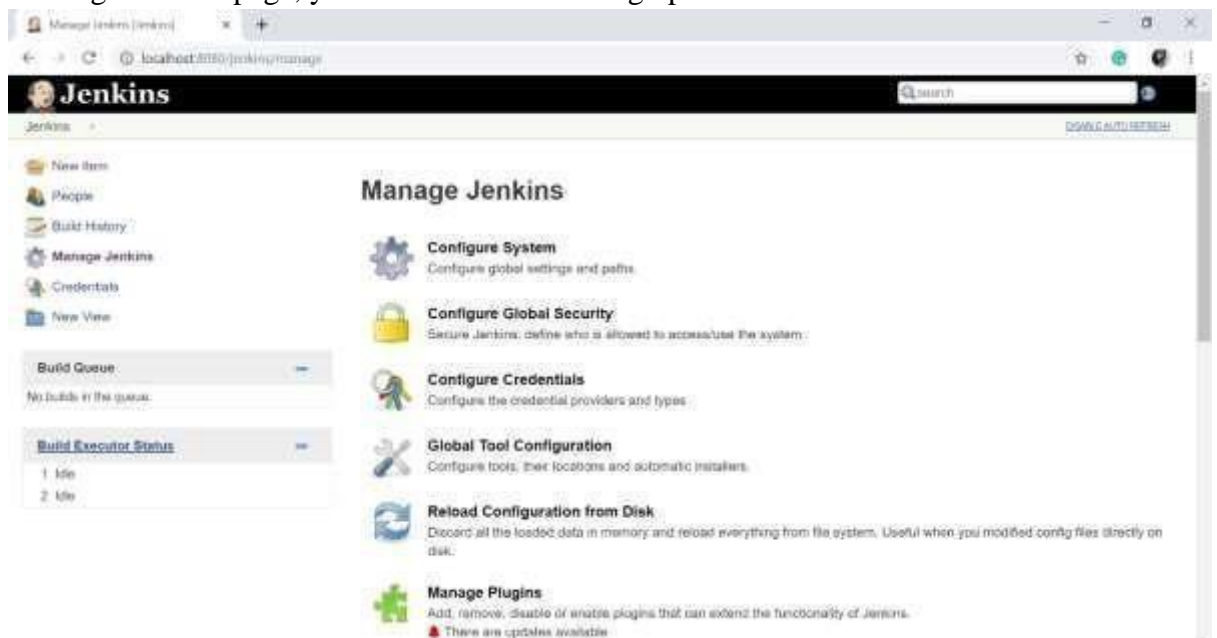
Restart the Jenkins

Jenkins Configuration

To configure the Jenkins, click on the 'Manage Jenkins' menu option from the left-hand side of the Jenkins Dashboard screen.

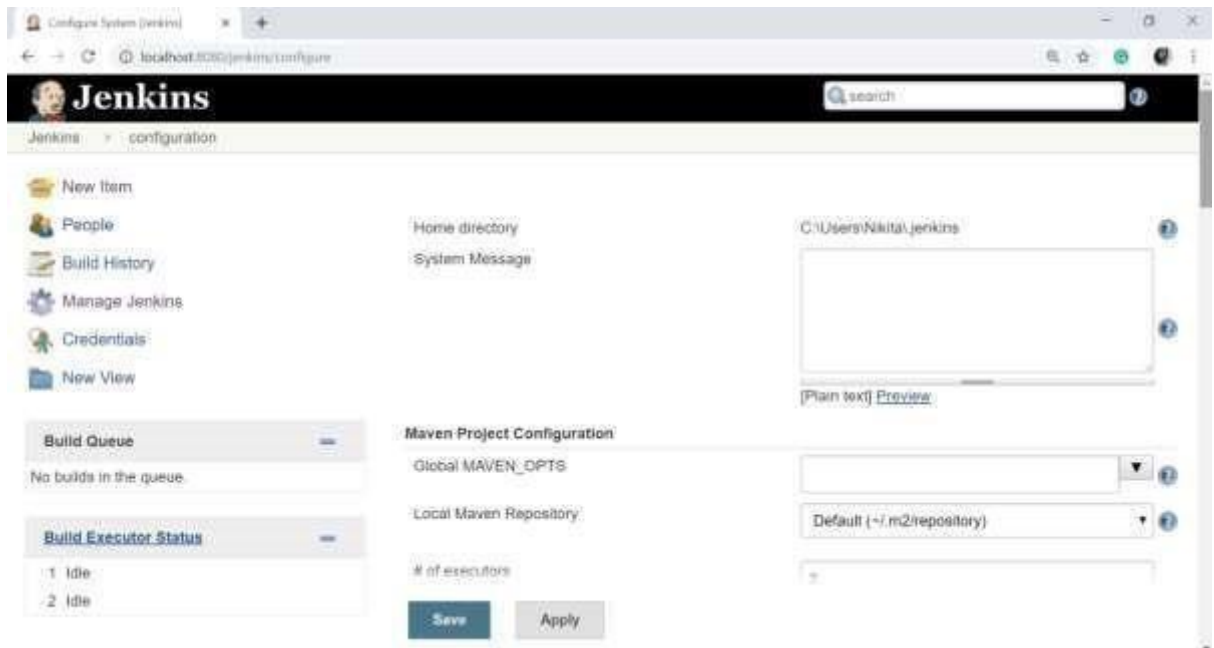


On the Manage Jenkins page, you will see the following options:



Configure System

Click on the 'Configure System'.

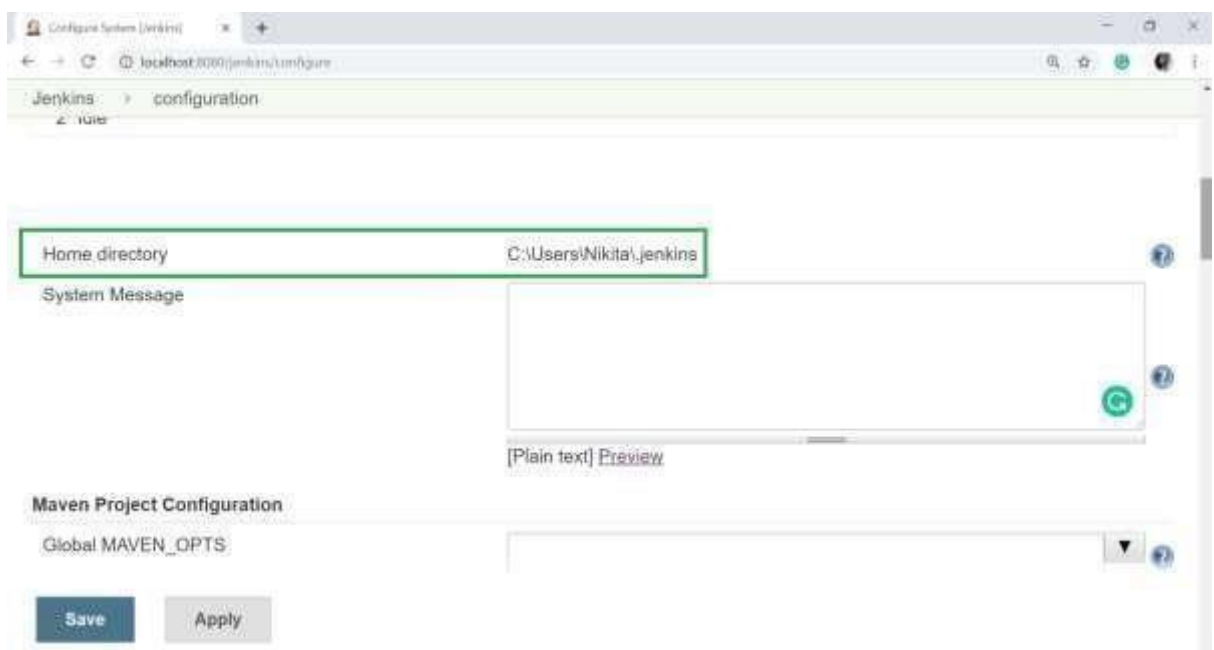


The configure system page is a critical configuration part. This screen represents a variety of sections, each correlating to a different configuration area from generic Jenkins settings, global environment variables, and most installed plugins are configured on this page.

Home directory

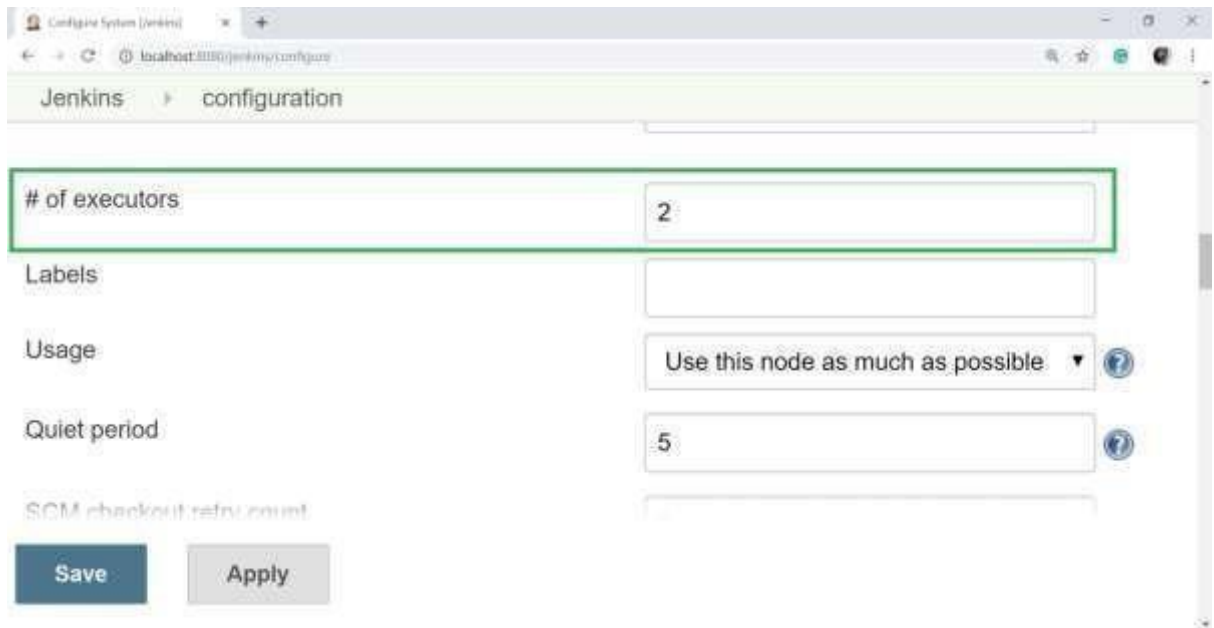
Jenkins requires some disk space to perform builds and keep archives. You can check this location from the configuration screen of Jenkins.

By default, this is set to ~/.jenkins, and this location will be initially stored within your user profile (such as C:\Users\Nikita\.jenkins) location.



of executors

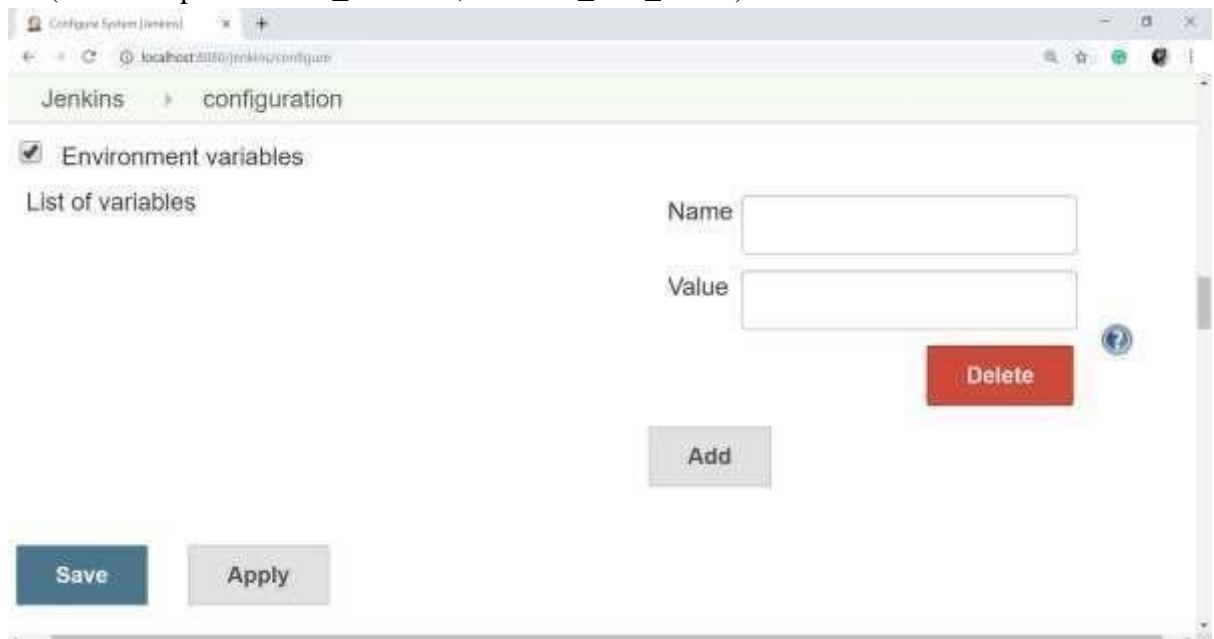
This option refers to the total number of concurrent job executions that can take place on the Jenkins machine. This can be changed based on the requirements. I recommended to you to keep this number the same as the number of CPUs on the systems for better performance.



The screenshot shows the Jenkins configuration page for the system. The '# of executors' field is highlighted with a green box and contains the value '2'. Below it, the 'Labels' field is empty. The 'Usage' dropdown is set to 'Use this node as much as possible'. The 'Quiet period' field contains the value '5'. The 'SCM checkout retry count' field is empty. At the bottom, there are 'Save' and 'Apply' buttons.

Environment Variables

This option is used to add custom environment variables that will apply to all the jobs. Environment variables are key-value pairs and can be accessed and used in Builds wherever required. (For example SLACK_TOKEN, SAUCE_API_KEY).

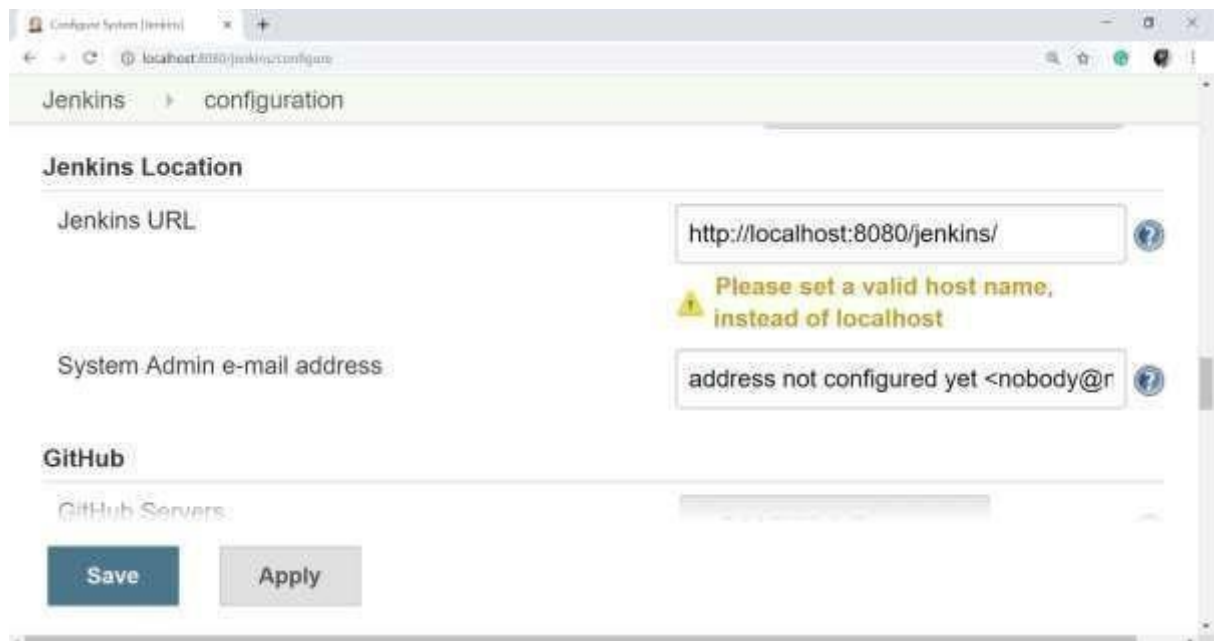


The screenshot shows the 'Environment variables' section of the Jenkins configuration page. The 'Environment variables' checkbox is checked. Below it, there is a 'List of variables' section with two input fields: 'Name' and 'Value'. A 'Delete' button is located to the right of the 'Value' field. An 'Add' button is located below the 'List of variables' section. At the bottom, there are 'Save' and 'Apply' buttons.

Jenkins URL

By default, the Jenkins URL is set to the localhost. If you have a DNS (domain name setup) for your machine, set this to the domain name else overwrite localhost with the IP of the machine. This will help in setting up slaves (nodes) and while sending out links using the email as you

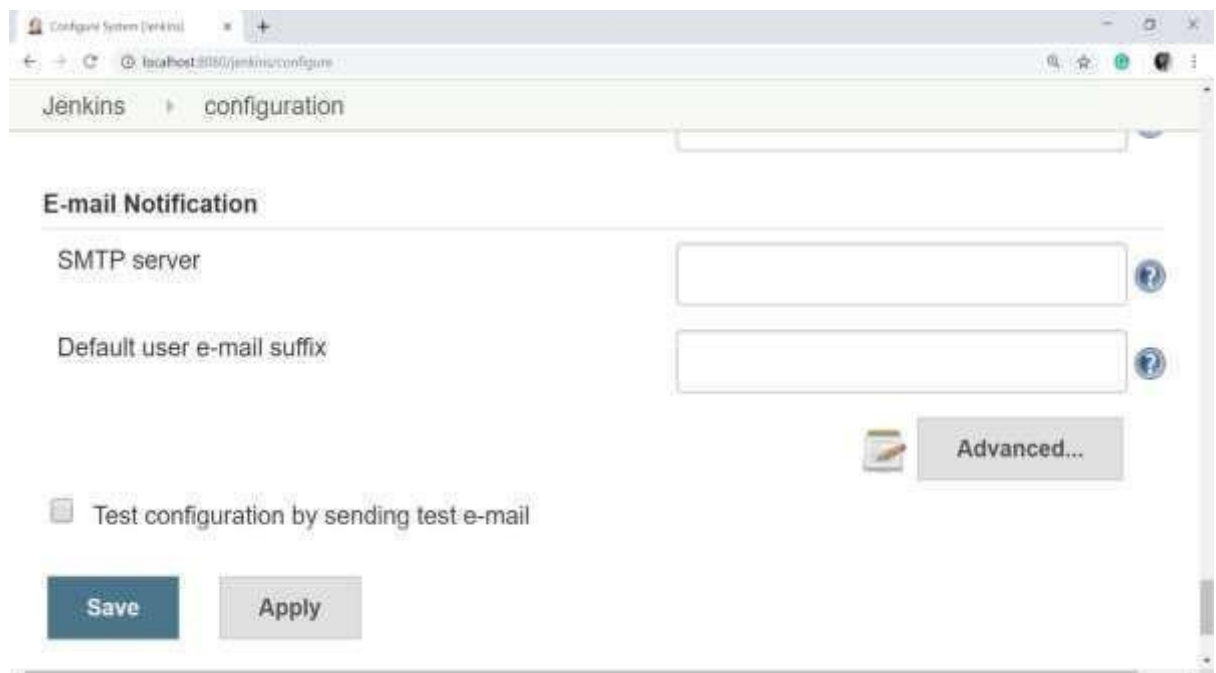
can directly access the Jenkins URL using the environment variable JENKINS_URL which can be accessed as \${JENKINS_URL}.



The screenshot shows the Jenkins configuration page in a web browser. The browser's address bar displays 'localhost:8080/jenkins/configure'. The page title is 'Jenkins configuration'. Under the 'Jenkins Location' section, the 'Jenkins URL' field contains 'http://localhost:8080/jenkins/'. A yellow warning message with a triangle icon states: 'Please set a valid host name, instead of localhost'. The 'System Admin e-mail address' field contains 'address not configured yet <nobody@r'. Below this is the 'GitHub' section with a 'GitHub Servers' field. At the bottom are 'Save' and 'Apply' buttons.

Email Notification

In the Email Notification section, you can configure the SMTP settings for sending out emails. This needs for Jenkins to connect to the SMTP mail server and send out emails to the recipient list.



The screenshot shows the 'E-mail Notification' section of the Jenkins configuration page. The 'SMTP server' field is empty. The 'Default user e-mail suffix' field is also empty. There is an 'Advanced...' button with a pencil icon. A checkbox labeled 'Test configuration by sending test e-mail' is unchecked. At the bottom are 'Save' and 'Apply' buttons.

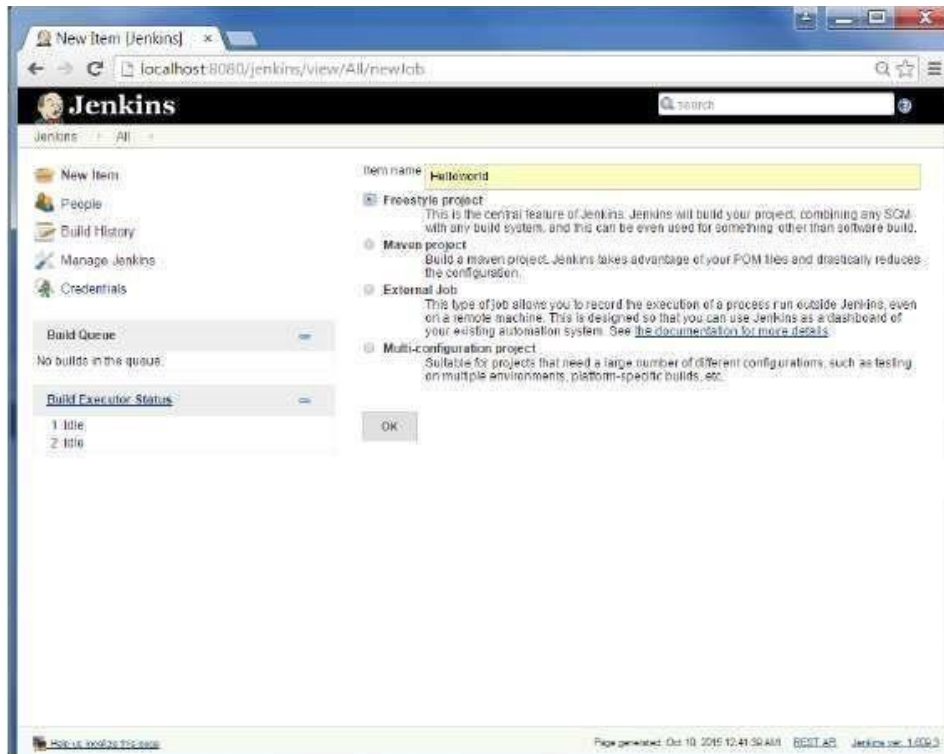
Getting starting with the Jobs

Jobs are the heart of Jenkins's build process. A job can be considered as a particular task to achieve a required objective in Jenkins. Every time you run a job, Jenkins compiles the job configuration inside the project workspace to perform the defined steps. Each run of this job is called a build and each step is called a build step.

Step 1 – Go to the Jenkins dashboard and click on New Item



Step 2 – In the next screen, enter the Item name, in this case, we have named it Test1. Choose the ‘Freestyle project option’.



Step 3 – The following screen will come up in which you can specify the details of the job.

General
Source Code Management
Build Triggers
Build Environment
Build

Description

<h2>This is the test project one.</h2>

[Safe HTML] Preview Hide preview

This is the test project one.

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)
☐ Build after other projects are built
☒ Build periodically

Schedule

* * * * *

- Build takes place every 1 minute.

Build

Execute Windows batch command

Command

echo EVOC (Software and App DeveIopment)

Jenkins

All

+

		Name	Last Success	Last Failure	Last Duration	
		DevProject1	N/A	N/A	N/A	
		Test1	10 sec - #35	3 min 10 sec - #31	0.67 sec	
		TestProject1	N/A	N/A	N/A	

1000

S

M

L

Root legend

Alert feed for all

Alert feed for failures

Alert feed for just latest builds

- Green tick mark indicates that build is successful.

Dashboard
Test1

Back to Dashboard

Status

Changes

Workspace

Build Now

Configure

Delete Project

Rename

Build History
trend ^

#38
Jan 24, 2022 6:20 PM

Project Test1

This is the test project one.

Workspace

Recent Changes

Permalinks

- Last build (#38), 3.4 sec ago
- Last stable build (#38), 3.4 sec ago
- Last successful build (#38), 3.4 sec ago
- Last failed build (#31), 6 min 3 sec ago
- Last unsuccessful build (#31), 6 min 3 sec ago
- Last completed build (#38), 3.4 sec ago

- Click on the build history.

Jenkins

Dashboard
Test1
#38

Back to Project

Status

Changes

Console Output

Edit Build Information

Delete build "#38"

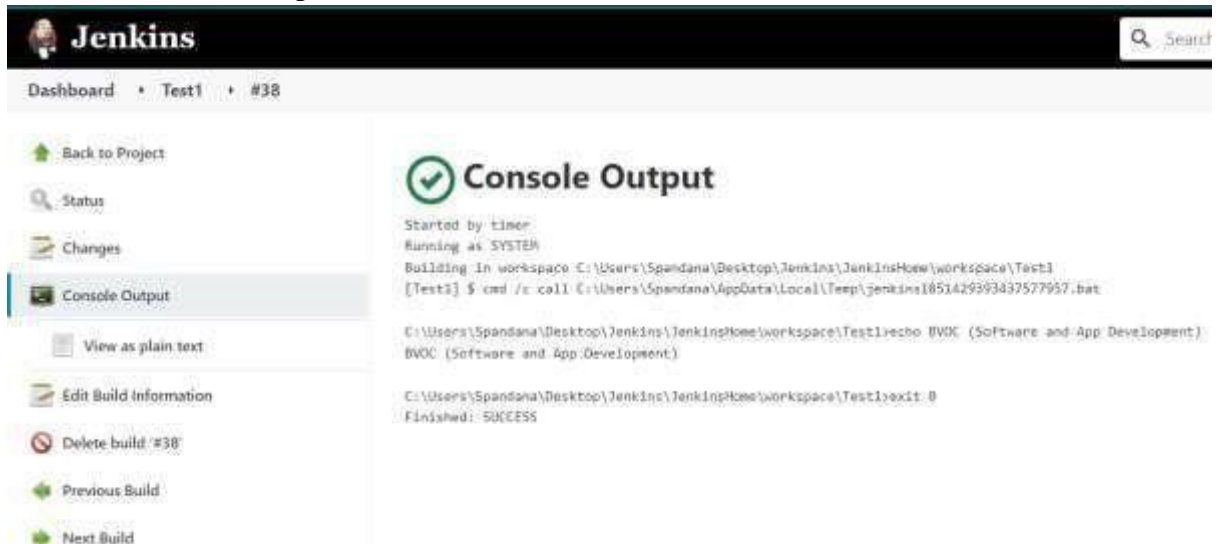
Previous Build

Build #38 (Jan 24, 2022 6:20:00 PM)

No changes

Started by timer

- Click on Console Output.



Jenkins Integration with Git(SCM)

Step 1: Create a java program.

Example

```
public class helloworld {

    public static void main(String args[]){

        for(int i=0; i<10;i++){
            System.out.println("Hellow world !");
        }

    }
}
```

- Create the new folder and place the java program file.
- Let us run the java program in the command prompt for testing purposes.

```
Command Prompt
Microsoft Windows [Version 10.0.19042.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Spandana>cd C:\Users\Spandana\Desktop\java

C:\Users\Spandana\Desktop\java>javac helloworld.java

C:\Users\Spandana\Desktop\java>java helloworld
Hellow world !
Hellow world !
Hellow world !
Hellow world !
Hellow world !
Hellow world !
Hellow world !
Hellow world !
Hellow world !
Hellow world !
```

Step 2: Install Plugins and create a Jenkins job to run the program. Plugins to install

Open your dashboard -> Manage Jenkins -> Manage Plugins

- GIT Plugin
- GitHub Branch Source Plugin

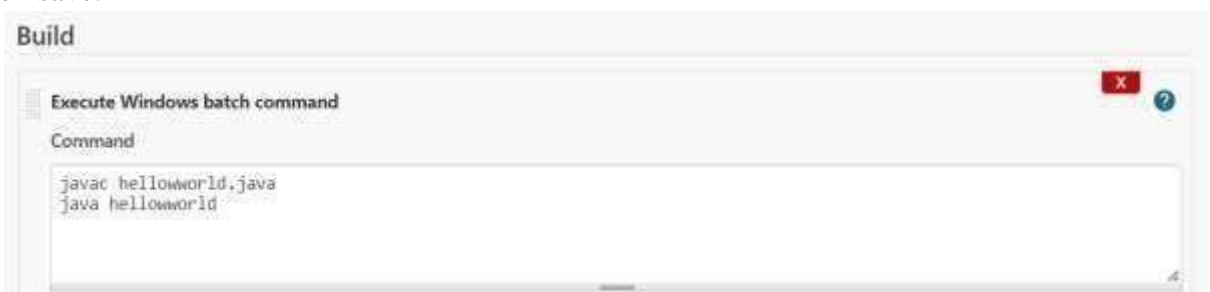
To change the httpport use --httpPort=PORT_NUMBER

```
C:\Users\Spandana\Desktop\Jenkins>java -jar jenkins.war --httpPort=9191
```



The image shows the Jenkins web interface for creating a new job. The title is "Enter an item name". There is a text input field containing "Hellowworld" with a red asterisk icon and the text "Required field" below it. Below the input field is a section titled "Freestyle project" with a small icon of a box and a description: "This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build."

Click on save.



The image shows the "Build" configuration page in Jenkins. The section "Execute Windows batch command" is expanded, showing a text area with the command: `javac helloworld.java` followed by `java helloworld` on a new line. There are "X" and "?" icons in the top right corner of the configuration area.

Step 3: Add project to Git and GitHub.

1. Create a new repository on GitHub.

In Terminal, change the current working directory to your local project.

2. Initialize the local directory as a Git repository.

```
git init
```

Add the files to your new local repository. This stages them for the first commit.

```
git add . or:
```

```
git add --all
```

Commit the files that you've staged in your local repository.

```
git commit -m 'First commit'
```

Copy the remote repository URL field from your GitHub repository, in the right sidebar, copy the remote repository URL.

In Terminal, add the URL for the remote repository where your local repository will be pushed.

```
git remote add origin <remote repository URL> Sets
```

the new remote:

```
git remote -v
```

Push the changes in your local repository to GitHub.

```
git push origin master
```

Pushes the changes in your local repository up to the remote repository you specified as the origin

```

C:\Users\Spandana>cd C:\Users\Spandana\Desktop\java

C:\Users\Spandana\Desktop\java>git init
Initialized empty Git repository in C:/Users/Spandana/Desktop/java/.git/

C:\Users\Spandana\Desktop\java>git status
On branch master

No commits yet

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

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

```

```

C:\Users\Spandana\Desktop\java>git add .

C:\Users\Spandana\Desktop\java>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   helloworld.class
        new file:   helloworld.java

```

```

C:\Users\Spandana\Desktop\java>git commit -m "added hellow world program"
[master (root-commit) a78c052] added hellow world program
 2 files changed, 9 insertions(+)
 create mode 100644 helloworld.class
 create mode 100644 helloworld.java

```

```

C:\Users\Spandana\Desktop\java>git remote add origin https://github.com/spandanasalian/Hellowworld.git

```

```

C:\Users\Spandana\Desktop\java>git push -u origin master
Everything up-to-date
branch 'master' set up to track 'origin/master'.

```

Step 4: Configure the project.

- Install the git plugin.
- Go to Manage Jenkins -> Manage Plugins -> install
- Run where git command on command prompt


```
C:\Users\Spandana>where git
C:\Program Files\Git\cmd\git.exe
```

- Go to Manage Jenkins -> Global Tool Configuration -> Git -> Path to Git executable > set the git.exe path.

Git

Git installations



Git

Name

Default

Path to Git executable ?

C:\Program Files\Git\cmd\git.exe

☐ Install automatically ?

- Configure the Hellowworld project.

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

https://github.com/spandanasaian/Hellowworld.git

Credentials

- none -

Add

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☐ Build periodically

☐ GitHub hook trigger for GITScm polling

☒ Poll SCM

Schedule

Step 23:

For testing

- Create a file in the Java folder in which the helloworld java program is present.

```
C:\Users\Spandana\Desktop\java>dir > read.txt
```

```
C:\Users\Spandana\Desktop\java>dir > text.txt
```

```
C:\Users\Spandana\Desktop\java>git status
On branch master
Your branch is up to date with 'origin/master'.

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

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

C:\Users\Spandana\Desktop\java>git add .

C:\Users\Spandana\Desktop\java>git add .

C:\Users\Spandana\Desktop\java>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   read.txt
    new file:   text.txt

C:\Users\Spandana\Desktop\java>git commit -m "two files were added"
[master 011386f] two files were added
 2 files changed, 25 insertions(+)
 create mode 100644 read.txt
 create mode 100644 text.txt

C:\Users\Spandana\Desktop\java>git push -u origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 675 bytes | 337.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/spandanasalian/Helloworld.git
   a78c052..011386f  master -> master
branch 'master' set up to track 'origin/master'.
```

Build #2 (Jan 25, 2022 4:25:10 PM)



No changes.



[Started by an SCM change](#)



Revision: 011386f3ae207549bb90df19ffaa09521050d553

Repository: <https://github.com/spandanasanian/Hellowworld.git>

- `refs/remotes/origin/master`

- Build took place due to the changes made to the source code.

Console Output

```
Started by an SCM change
Running as SYSTEM
Building in workspace C:\Users\Spandana\Desktop\Jenkins\JenkinsHome\workspace\Hellowworld
the recommended git tool is: none
no credentials specified
Cloning the remote git repository
Cloning repository https://github.com/spandanasanian/Hellowworld.git
> C:\Program Files\Git\cmd\git.exe init C:\Users\Spandana\Desktop\Jenkins\JenkinsHome\workspace\Hellowworld * timeout=10
Fetching upstream changes from https://github.com/spandanasanian/Hellowworld.git
> C:\Program Files\Git\cmd\git.exe --version * timeout=10
> git --version * 'git version 2.35.0.windows.1'
> C:\Program Files\Git\cmd\git.exe fetch --tags --force --progress -- https://github.com/spandanasanian/Hellowworld.git +refs/heads/*:refs/remotes/origin/* * timeout=10
> C:\Program Files\Git\cmd\git.exe config remote.origin.url https://github.com/spandanasanian/Hellowworld.git * timeout=10
> C:\Program Files\Git\cmd\git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* * timeout=10
Avoid second fetch
> C:\Program Files\Git\cmd\git.exe rev-parse --verify "refs/remotes/origin/master:{commit}" * timeout=10
Checking out Revision 011386f3ae207549bb90df19ffaa09521050d553 (refs/remotes/origin/master)
> C:\Program Files\Git\cmd\git.exe config core.sparsecheckout * timeout=10
> C:\Program Files\Git\cmd\git.exe checkout -f 011386f3ae207549bb90df19ffaa09521050d553 * timeout=10
Commit message: 'two files were added'
First time build. Skipping changelog.
[Hellowworld] $ cmd /c call C:\Users\Spandana\AppData\Local\Temp\Jenkins968813446326416579.bat

C:\Users\Spandana\Desktop\Jenkins\JenkinsHome\workspace\Hellowworld> C:\Users\Spandana\Desktop\java

C:\Users\Spandana\Desktop\java> javac Hellowworld.java

C:\Users\Spandana\Desktop\java> java Hellowworld
Hellow world
Hellow world
Hellow world
Hellow world
Hellow world
Hellow world
Hellow world
Hellow world
Hellow world
Hellow world

C:\Users\Spandana\Desktop\java> exit #
Finished: SUCCESS
```

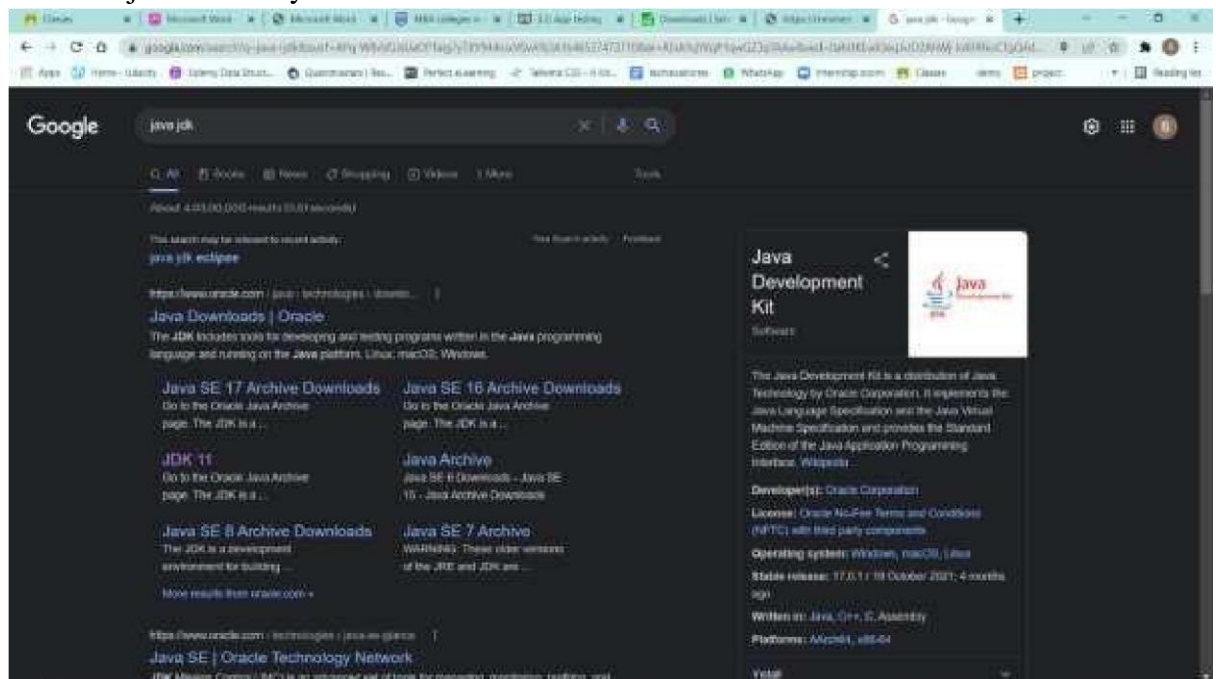
3. Explain selenium tools in DevOps and install selenium tool and explain the features of selenium and implementation of selenium

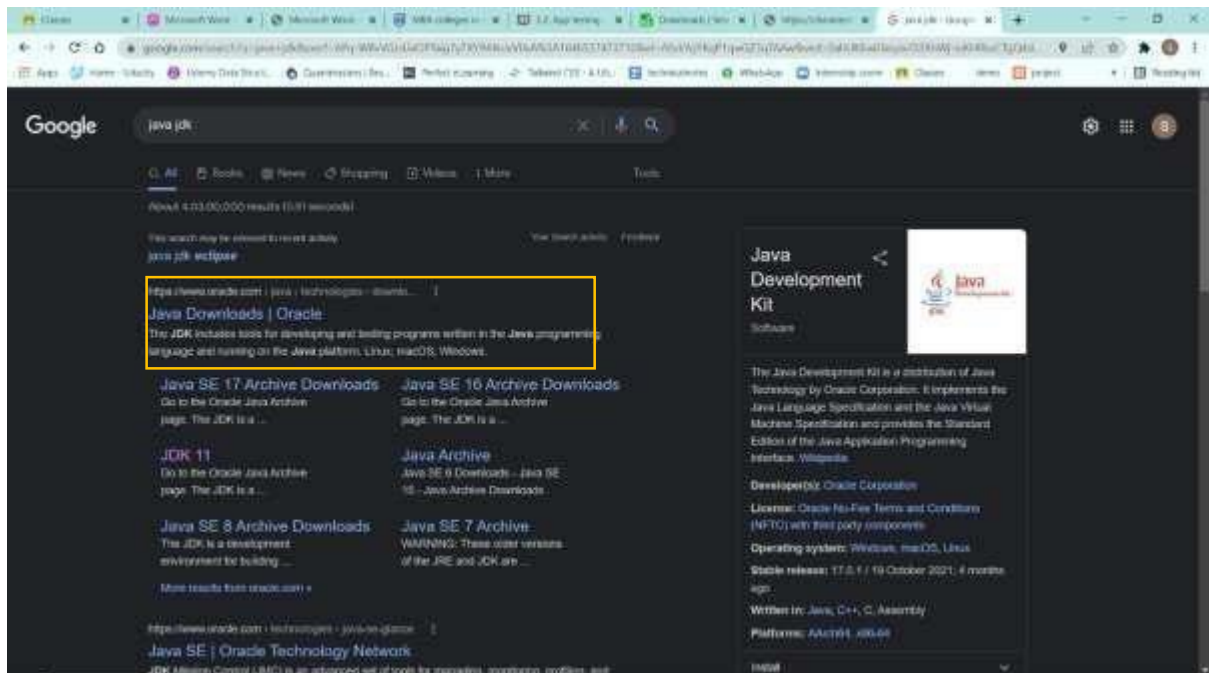
Selenium is a portable software testing framework for web applications. It provides an easy interface for developing automated tests. **Features**

- It is a free open source tool.
- It supports multiplatform for testing, such as Android and ios.
- It is easy to build a keyword-driven framework for a WebDriver.
- It creates robust browser-based regression automation suites and tests.

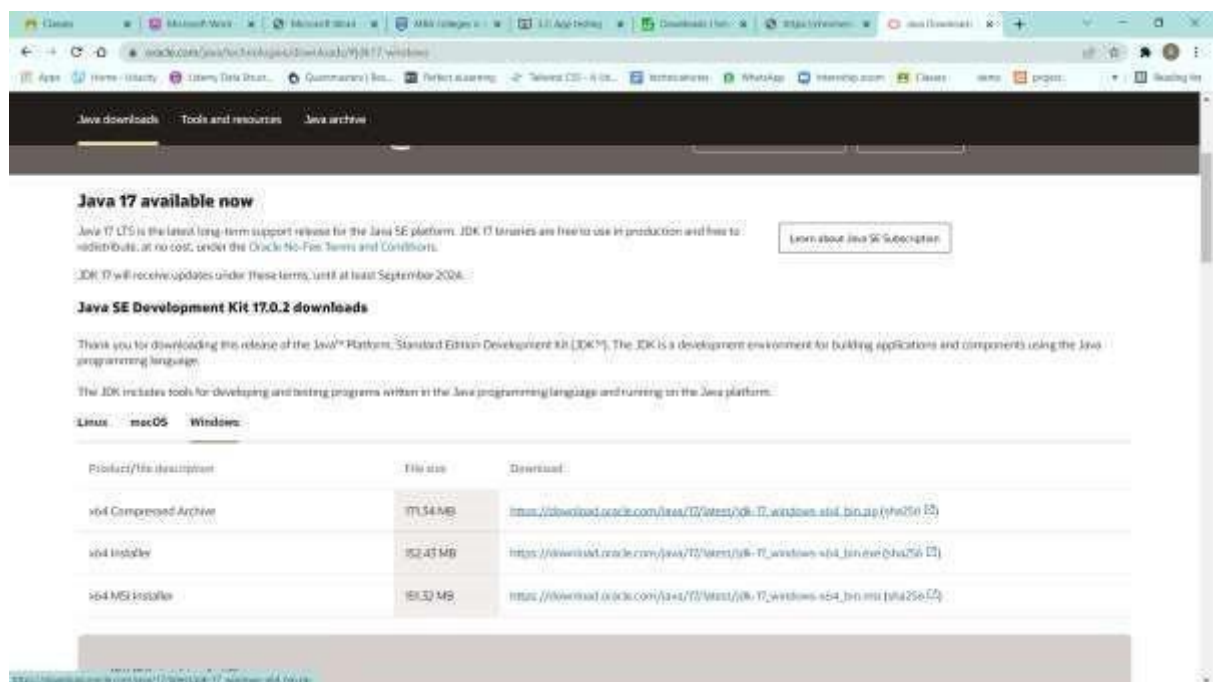
Step 1.

- Install java in our system
- Search as java JDK in your favorite browser

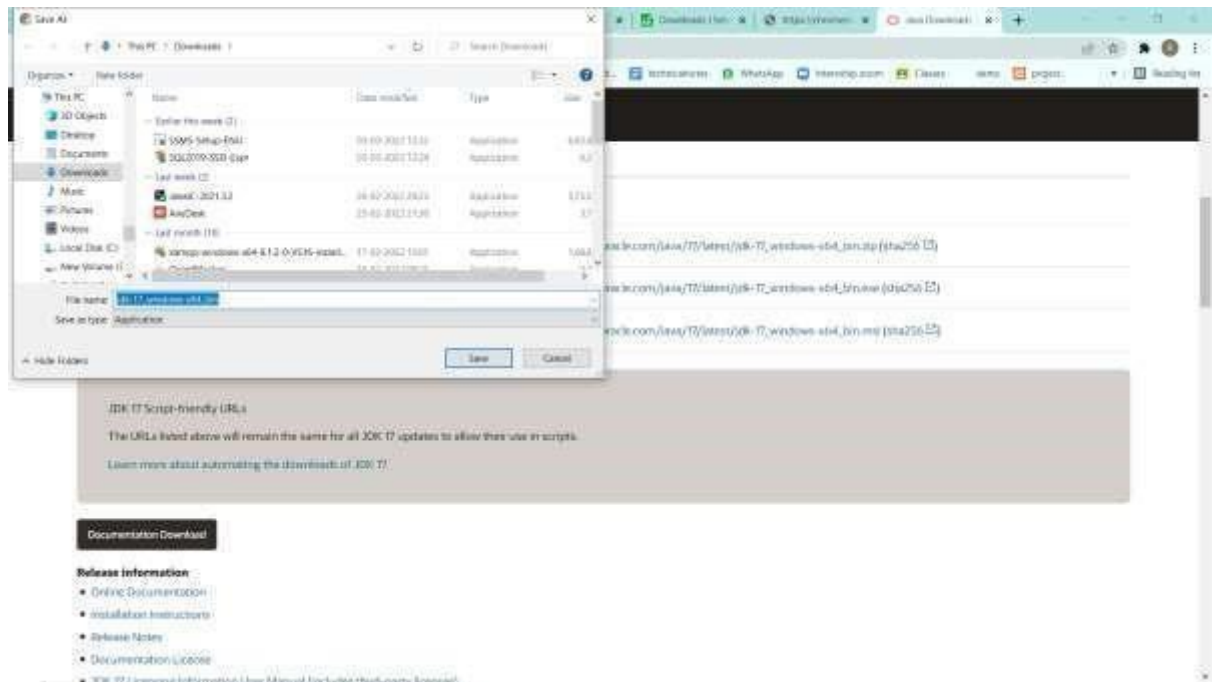




- Click the link mentioned above and you will be redirected to the below page.
- In the windows, section downloads the X64 windows downloader.

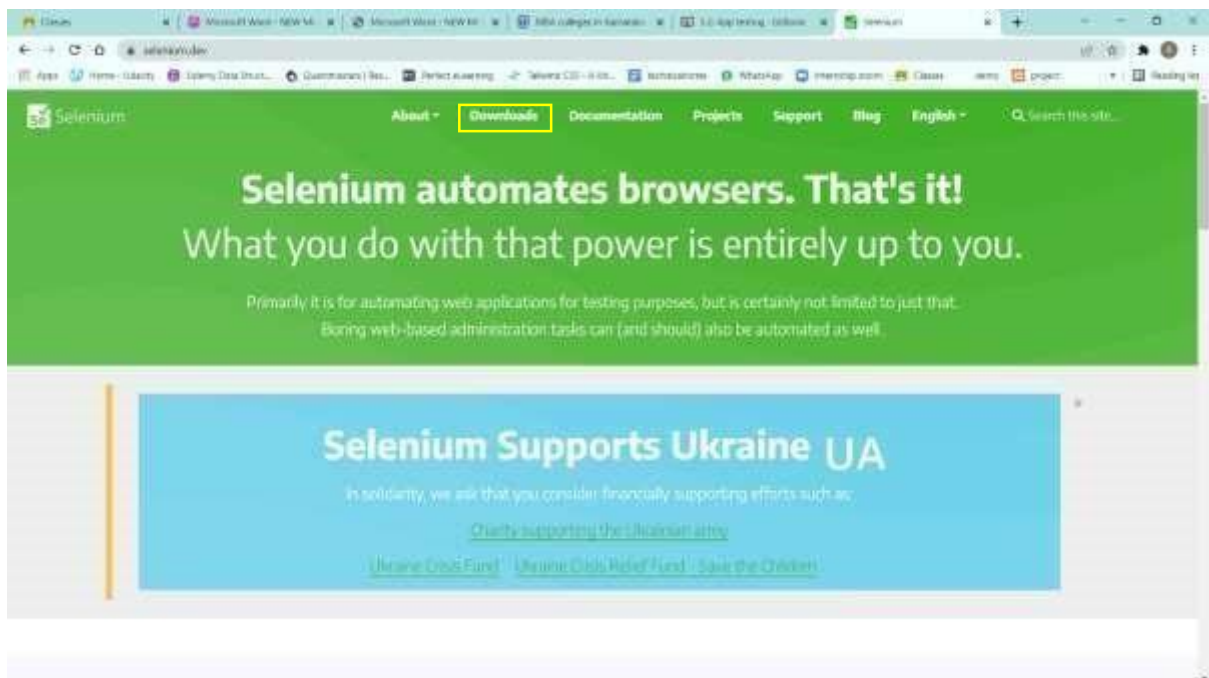


- Save the file and install the JDK.

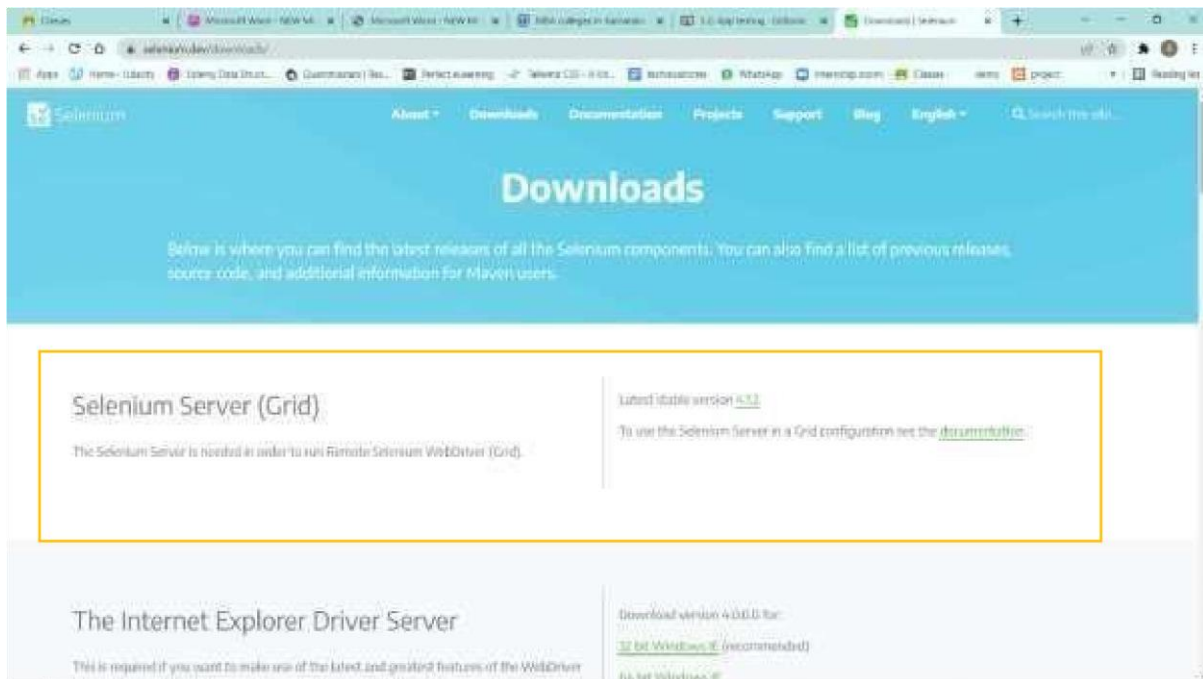


Step 2:

- Install the Selenium Grid :
- Open the selenium dev in your favorite browser and click on downloads.

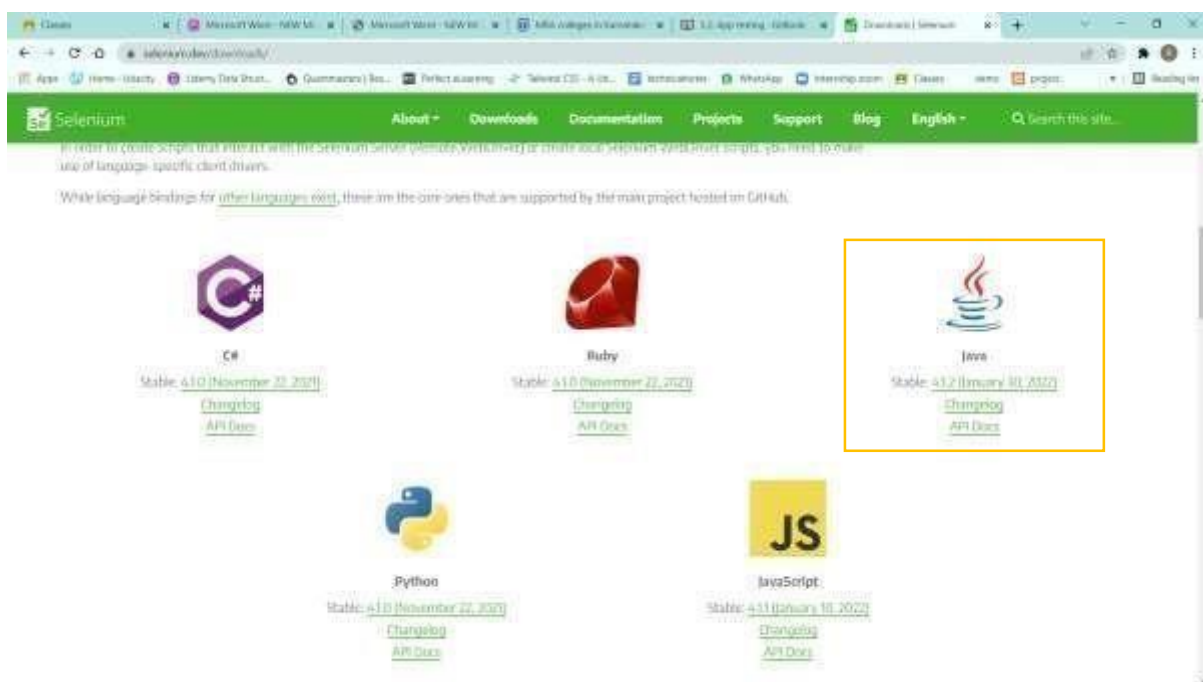


- In downloads click the below-mentioned link and download the file and place it in the c folder.

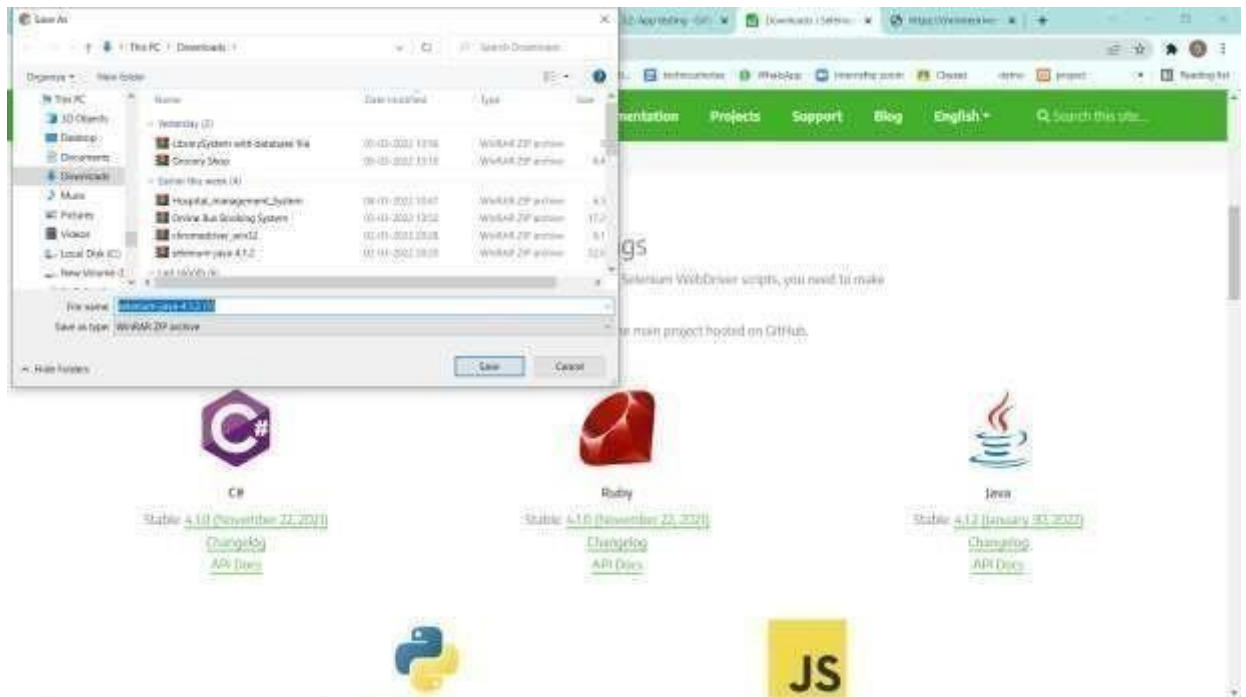


Step 3:-

- Download the java client programming language for the selenium.
- On the same page scroll down and click the below-mentioned link.

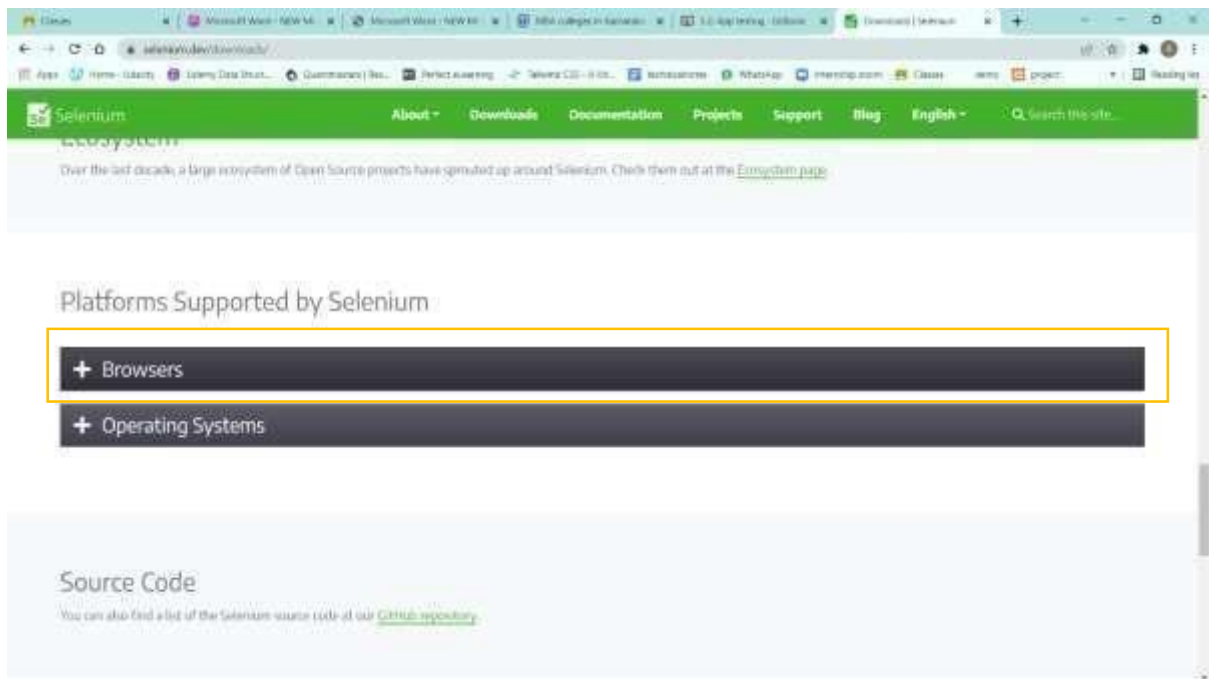


- Download the file and place it in the c folder same as of selenium grid as extracted.

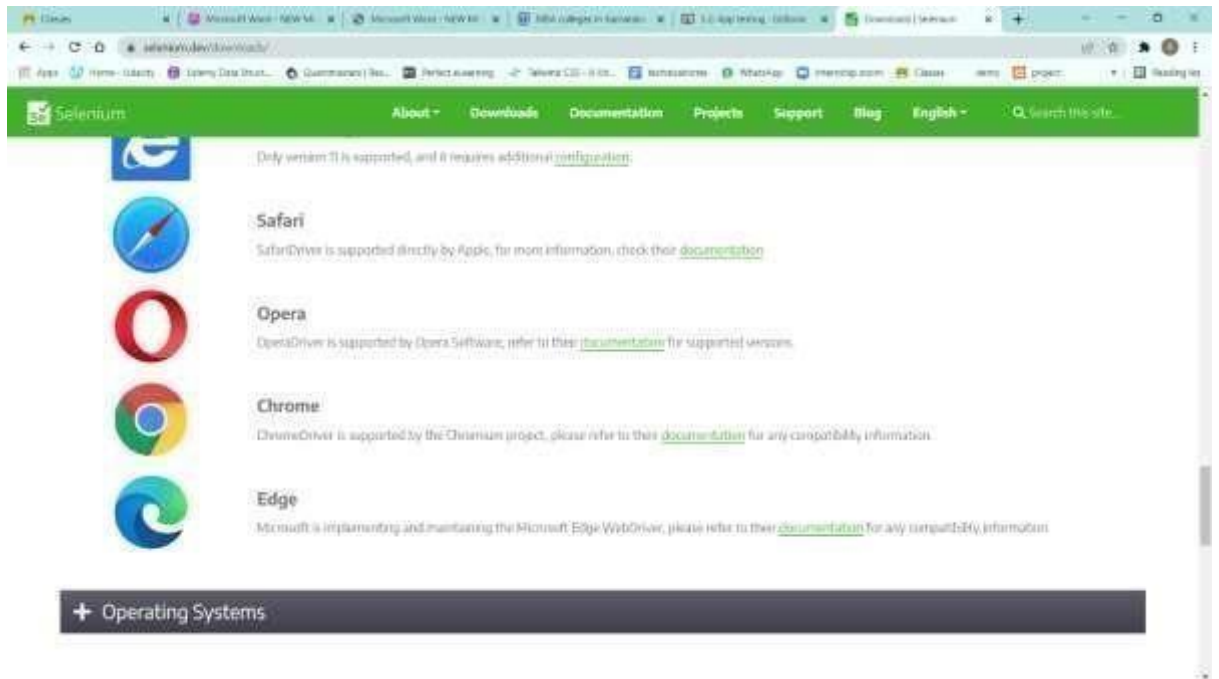


Step 4 :

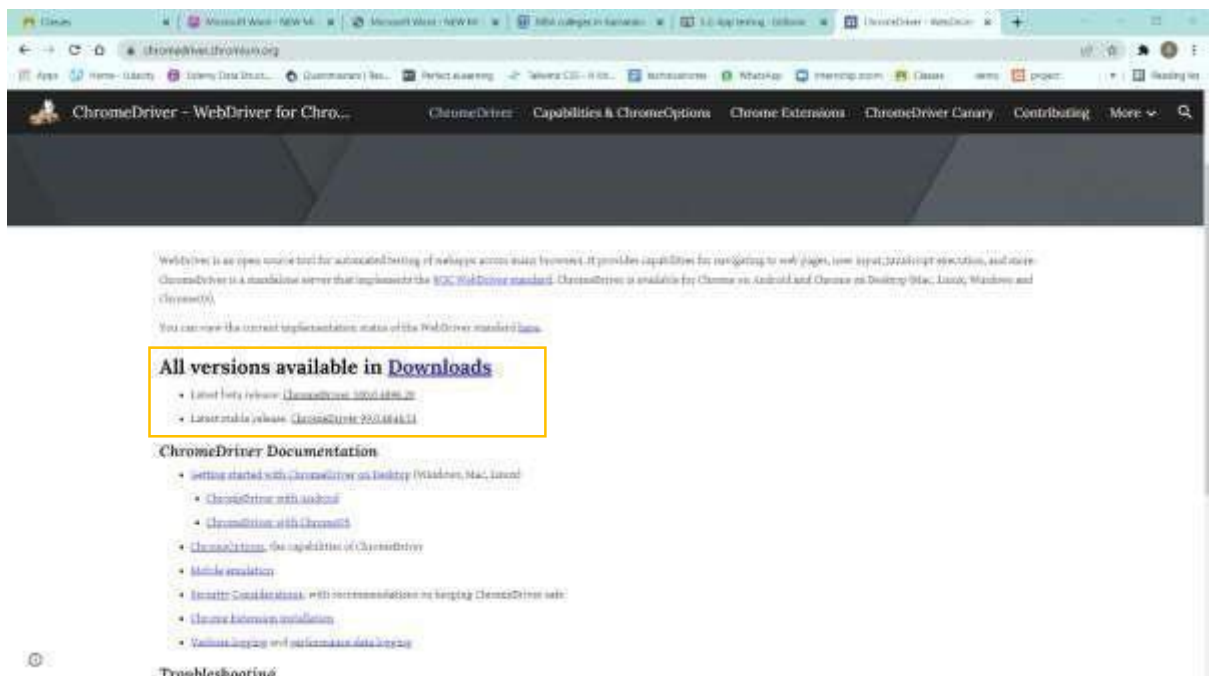
- Download the web driver for the selenium.
- Scroll down a bit on the same page and click the link below the mentioned figure.



- Download the driver of your choice and place it in the same folder as of selenium grid.



- Check your chrome version and download the respective web driver from the mentioned list

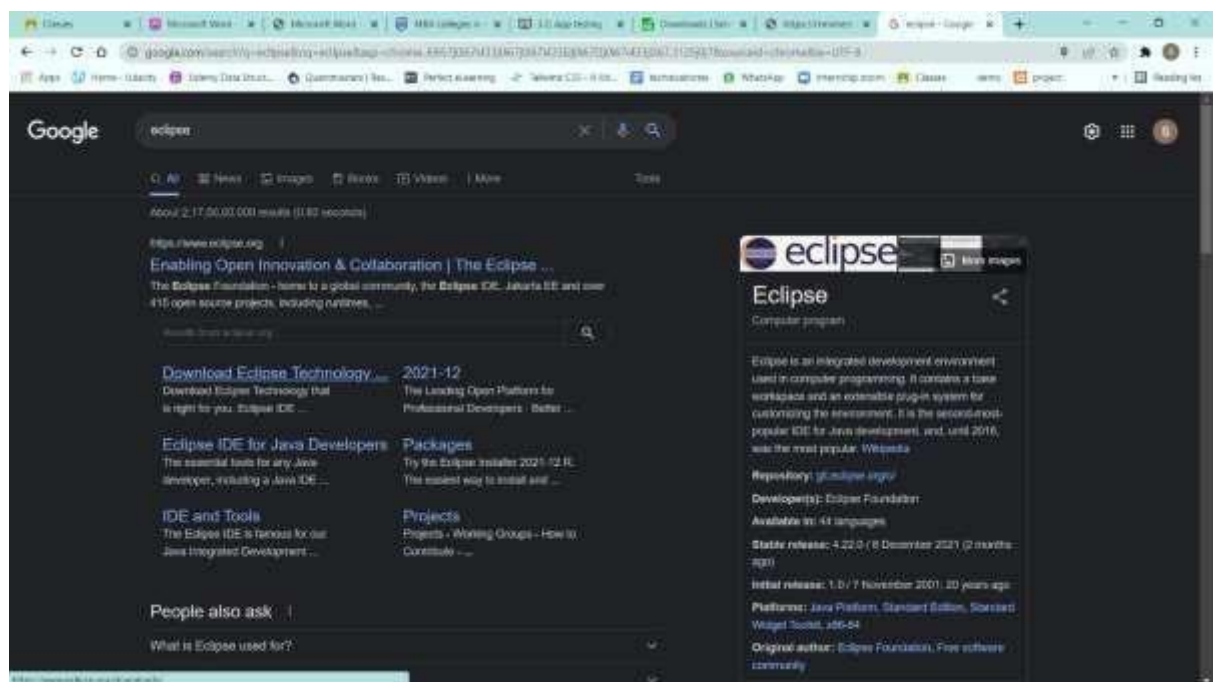


Download the zip file and extract to the selenium folder.

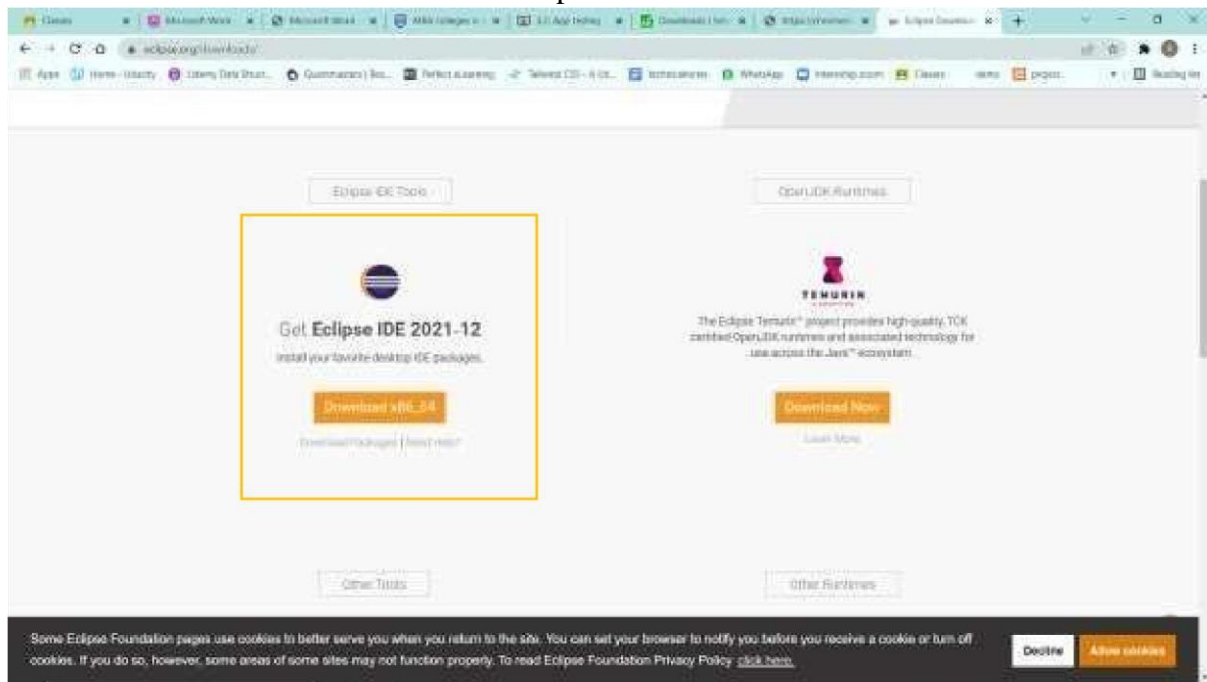


Step-5:

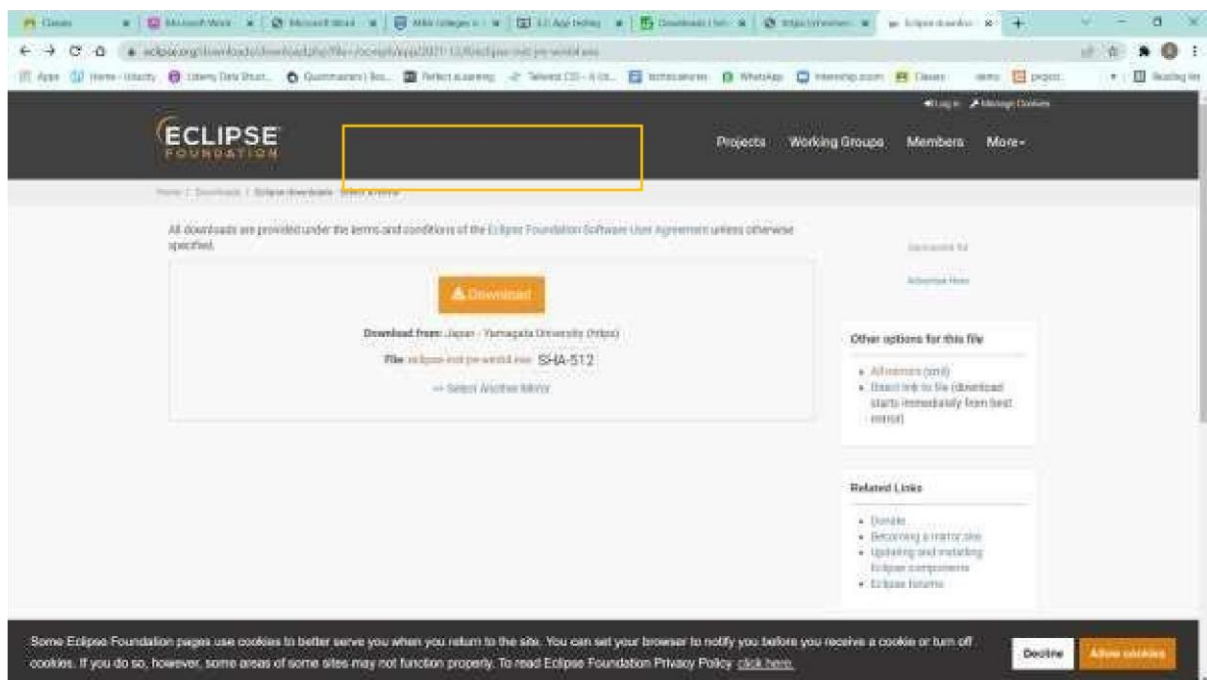
- Installation of the eclipse.
- Search the eclipse in your favorite browser.



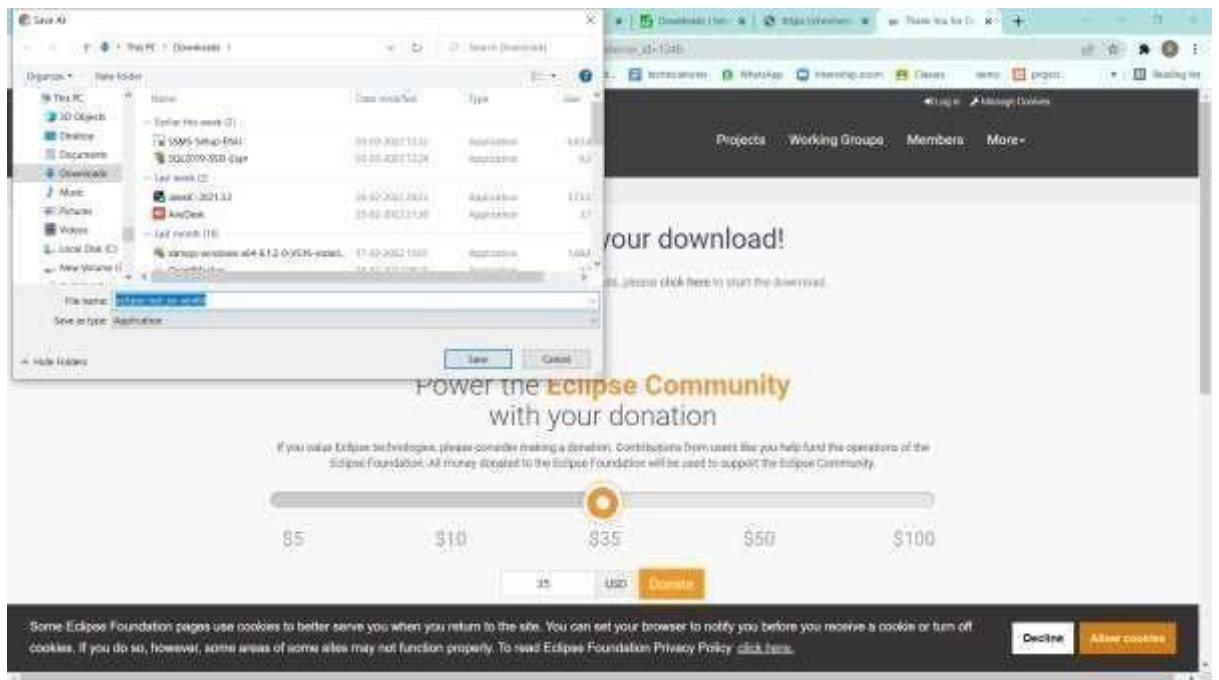
Click the official website of the eclipse and click on the download X64 for windows.



- Click on download

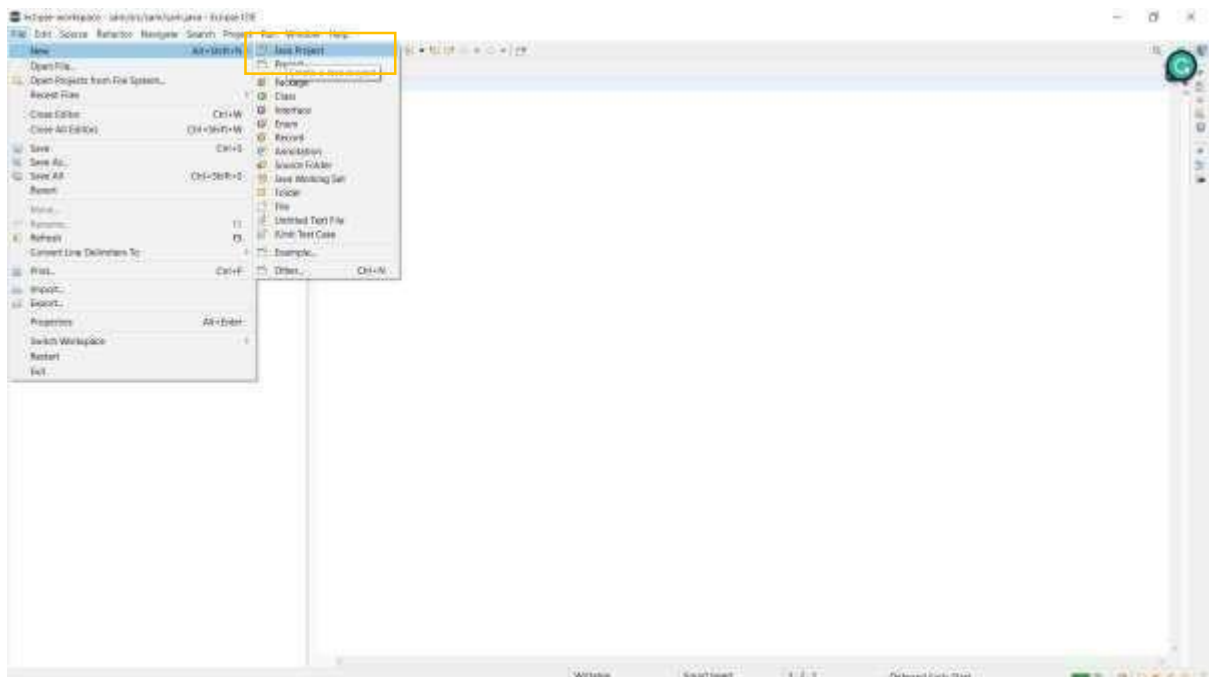


Specify the relevant path and install the ide in your system.

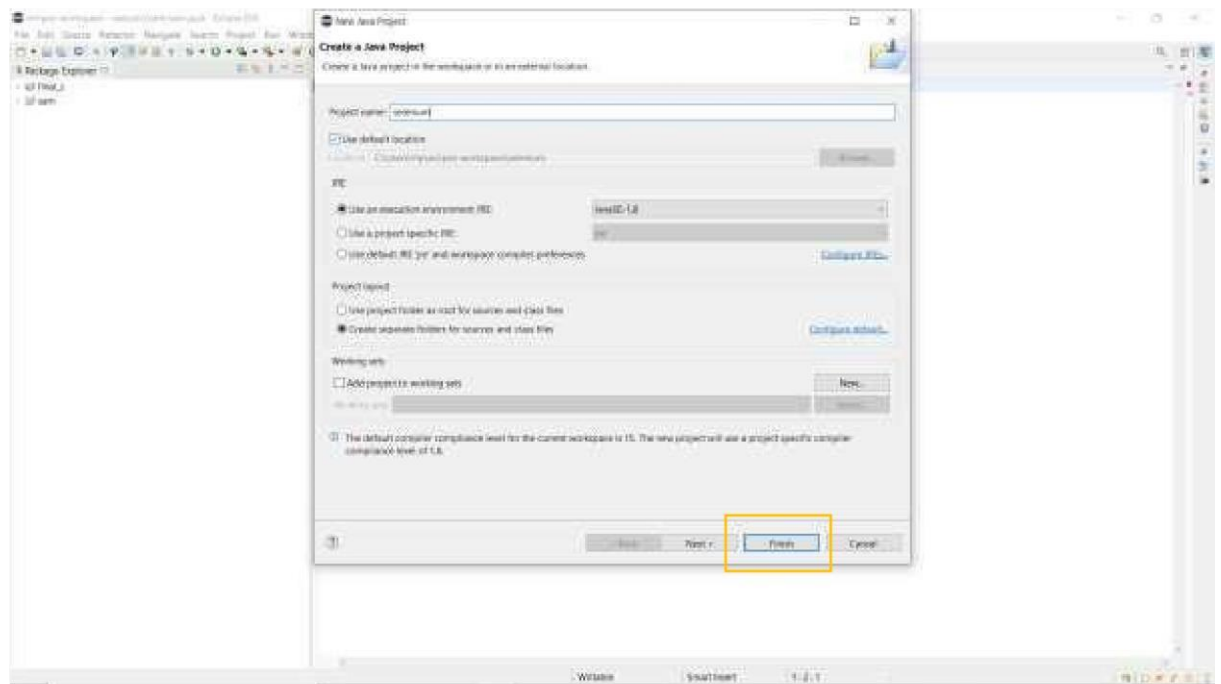


Step 6 :

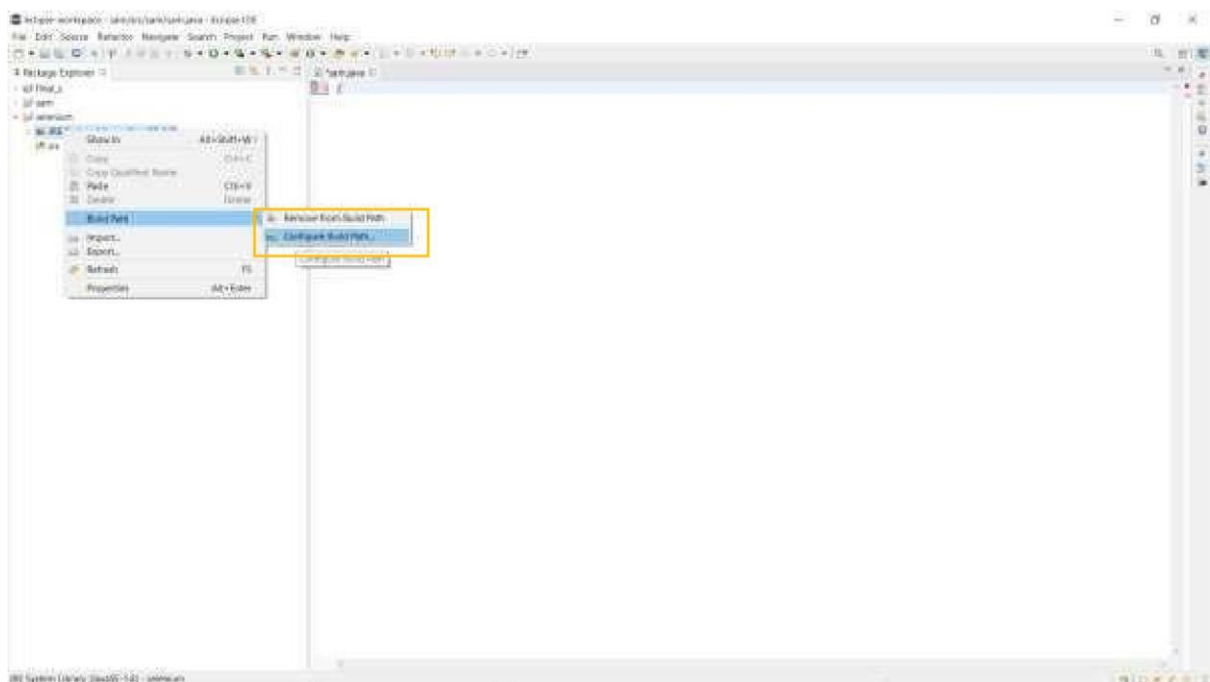
- Running the project.
- Open the ide and create the new project with installed java plugins and create a new project.



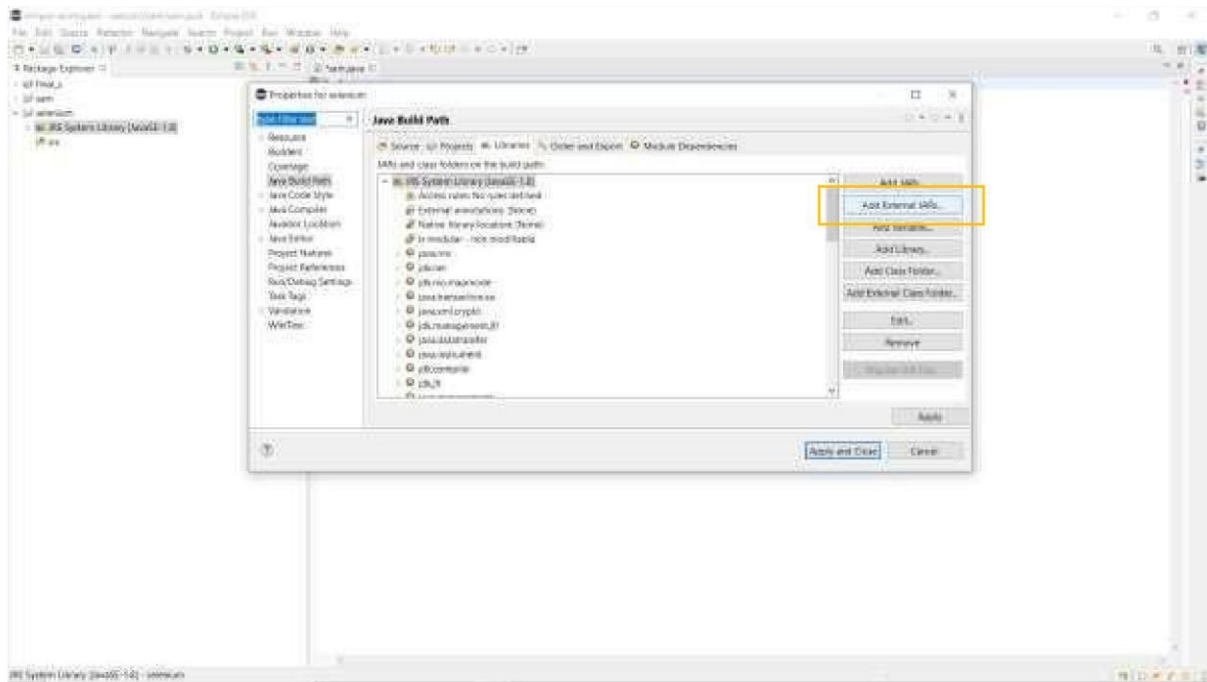
- After selecting the java project give the name and click on finish.



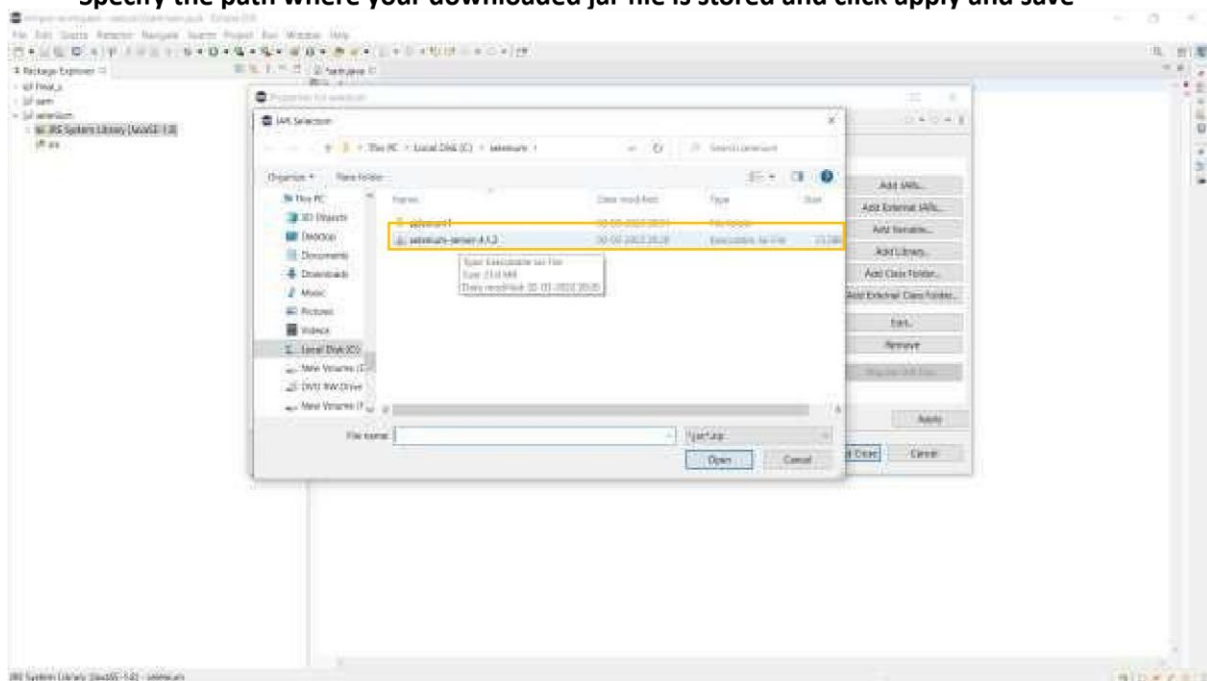
- In the project select Build -> Build path -> Configure Build path.



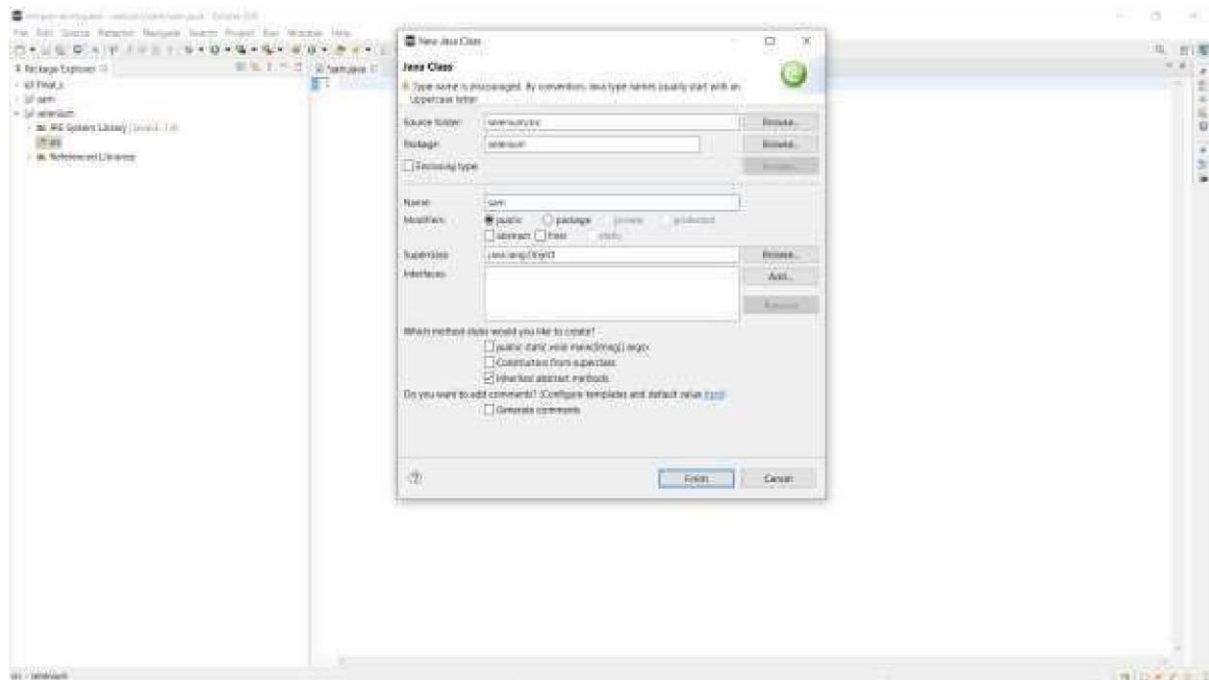
- After clicking on Add external jar file.



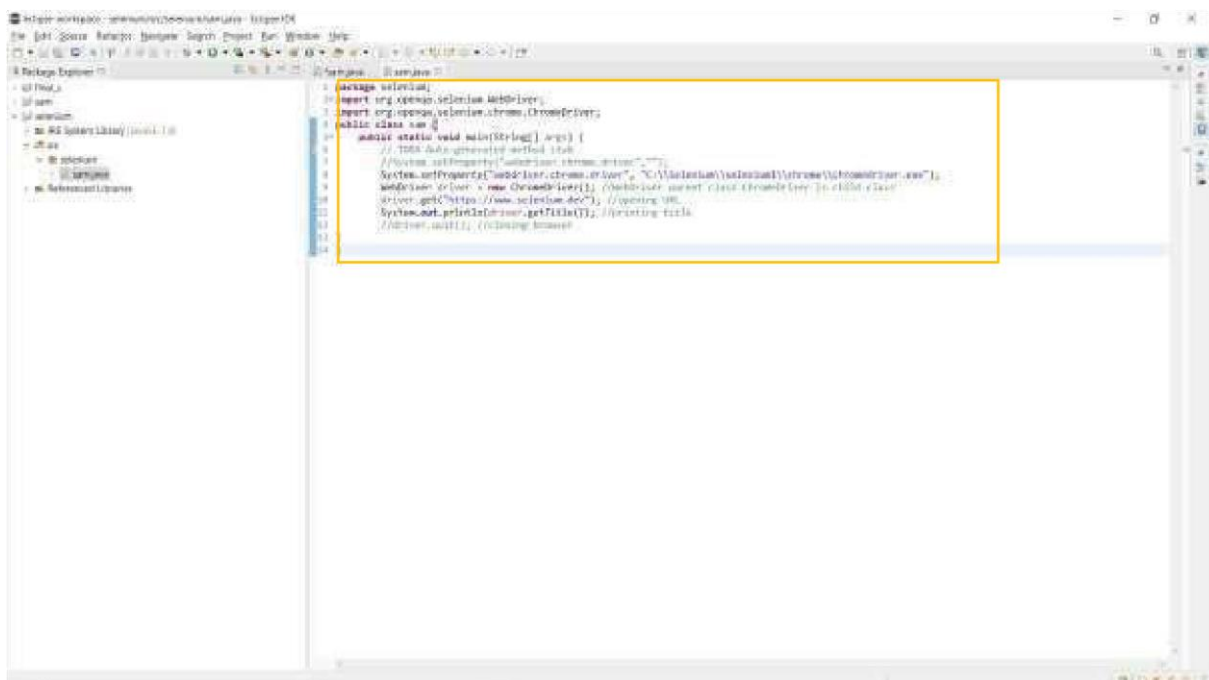
- Specify the path where your downloaded jar file is stored and click apply and save



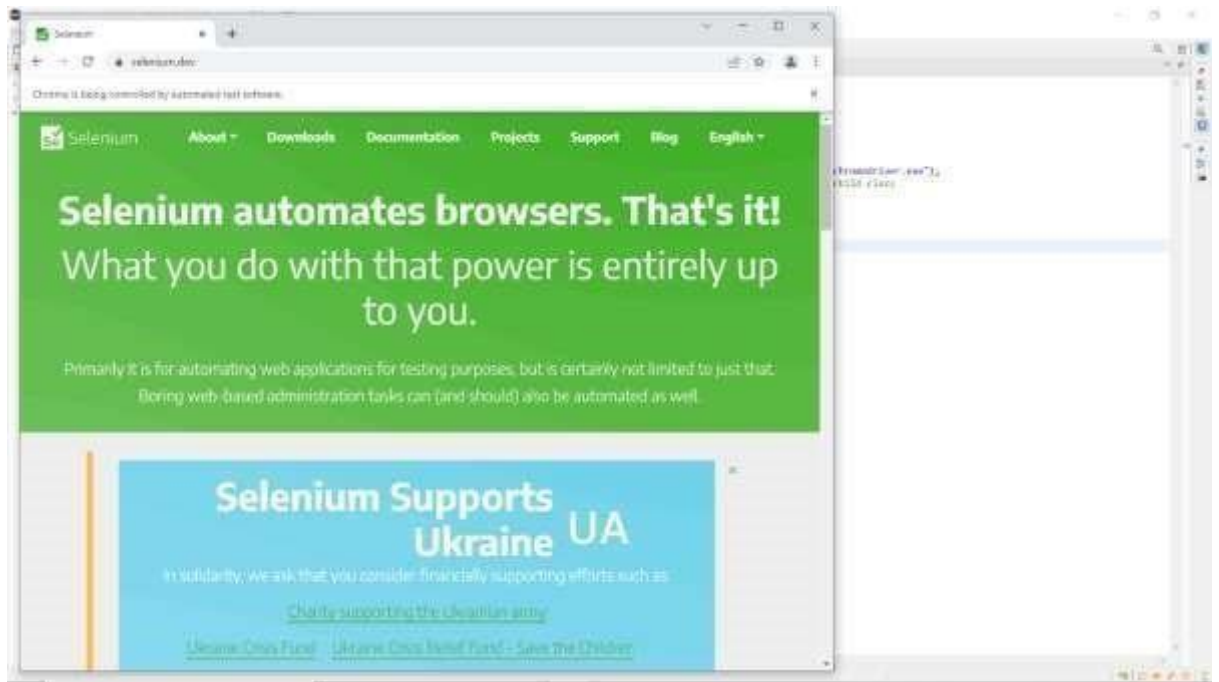
- Create a new class.



- Apply the below code in the class.



- Run the project to automate the browser.



4 . Explain puppet tool in DevOps and its advantages and features of puppet

- Puppet is the most widely used DevOps tool.
- It allows the delivery and release of the technology to change quickly and frequently. It has features of versioning, automated
- Testing, and continuous delivery. It enables to manage entire infrastructure as code without expanding the size of the team.

Features

- Real-time context-aware reporting.
- Model and manage the entire environment.
- Defined and continually enforce infrastructure.
- Desired state conflict detection and remediation.
- It inspects and reports on packages running across the infrastructure.
- It eliminates manual work for the software delivery process.
- It helps the developer to deliver great software quickly.

5 . Explain Ansible tool in DevOps and its advantages and features of Ansible

- Ansible is a leading DevOps tool.
- Ansible is an open-source IT engine that automates application deployment, cloud provisioning, and other IT tools. It makes it easier for DevOps teams to scale automation and speed up productivity.
- Ansible is easy to deploy because it does not use any **agents** or **custom security** infrastructure on the client-side, and by pushing modules to the clients. These modules are executed locally on the client-side, and the output is pushed back to the Ansible server.

Features

- It is easy to use to open source deploy applications.
- It helps in avoiding complexity in the software development process.
- It eliminates repetitive tasks.
- It manages complex deployments and speeds up the development process.

6. Explain Docker tool in DevOps and its advantages and features of Docker

Docker is a high-end DevOps tool that allows building, ship, and run distributed applications on multiple systems. It also helps to assemble the apps quickly from the components, and it is typically suitable for container management.

Features

- It configures the system more comfortable and faster.
- It increases productivity.
- It provides containers that are used to run the application in an isolated environment.
- It routes the incoming request for published ports on available nodes to an active container. This feature enables the connection even if there is no task running on the node.
- It allows saving secrets into the swarm itself.

7. Explain saltstack tool in DevOps and its advantages and features of Saltstack

Stackify is a lightweight DevOps tool. It shows real-time error queries, logs, and more directly into the workstation. SALTSTACK is an ideal solution for intelligent orchestration for the software-defined data center.

Features

- It eliminates messy configuration or data changes.
- It can trace detail of all the types of web requests.
- • It allows us to find and fix the bugs before production.

8. Explain Chef tool in DevOps and its advantages and features of chef

A chef is a useful tool for achieving scale, speed, and consistency. The chef is a cloud-based system and open-source technology. This technology uses Ruby encoding to develop essential building blocks such as recipes and cookbooks. The chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.

- Chef has got its convention for different building blocks, which are required to manage and automate infrastructure.

Features

- It maintains high availability.
- It can manage multiple cloud environments.
- It uses popular Ruby language to create a domain-specific language.
- The chef does not make any assumptions about the current status of the node. It uses its mechanism to get the current state of the machine.