

Don Bosco Institute of Technology(DBIT), Mumbai

Department of Information Technology



LAB JOURNAL

On

I.T/T.E/Sem V/ITL503 : DevOps Lab

By

54 Jasmit Rathod

Academic Year : Nov,2022

INDEX

Sr no	Name of the experiment	Pg no	Date
A	Self Study	3	11/07/2022
B	Case Study - Github	9	25/07/2022
1	a. To understand Version Control System /Source Code Management, install git and create a GitHub accounts b. To Perform various GIT operations on local and Remote repositories using GIT Cheat-Sheet	11	01/08/2022
2	a. Installation Of Jenkins and Creation of Different Styles of Project namely Freestyle/ Maven. b. To understand Continuous Integration,install and configure Jenkins with Maven/Freestyle to setup a build Job either in python, java, shell script program	23	12/09/2022
3	a. To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers b. Docker installation & to perform basic docker & container commands c. To learn Docker file instructions, build an image for a sample web application using Docker file	45	26/09/2022
i	Assignment 1 : Docker Volume Creation	65	10/10/2022
ii	Assignment 2 : Installation of Selenium & performing automatic testing	81	17/10/2022

EXPERIMENT A - Self Study

Date of Experiment: 11/07/2022

Prerequisite:

C, Python, Java, Software Engineering, Cloud

Objective:

To make students understand DevOps, for what? Why? and by whom?

Aim:

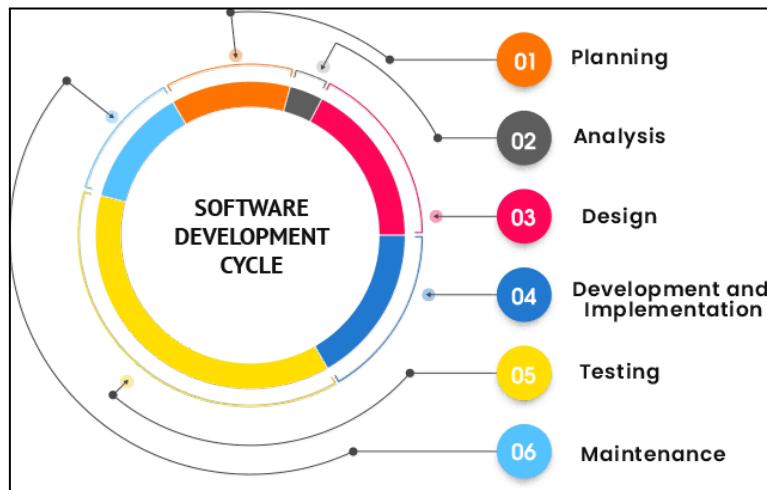
To do self-study on basics understanding of DevOps

Procedure:

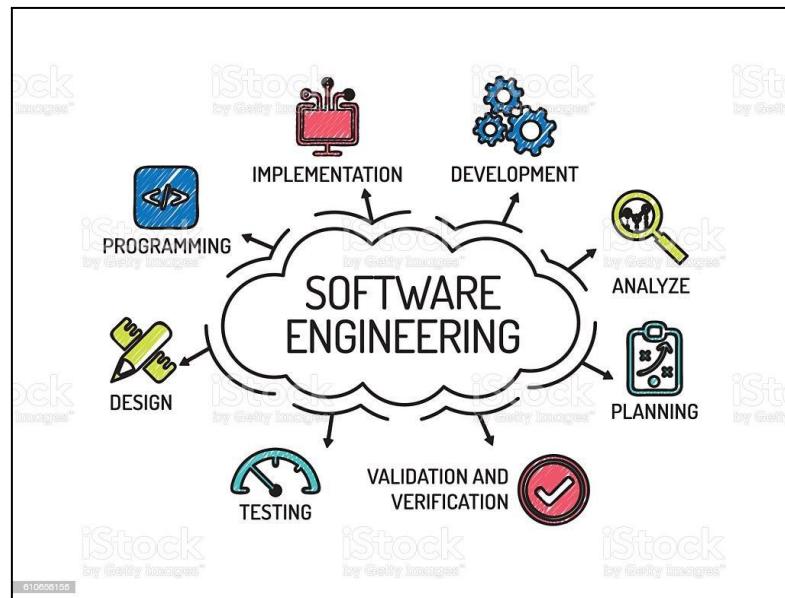
Individual students will perform this self-study over internet resources and will submit a report on their study and give a viva voce on or before 18/07/2022, and they will be graded based on rubrics.

The topics for self-study are :

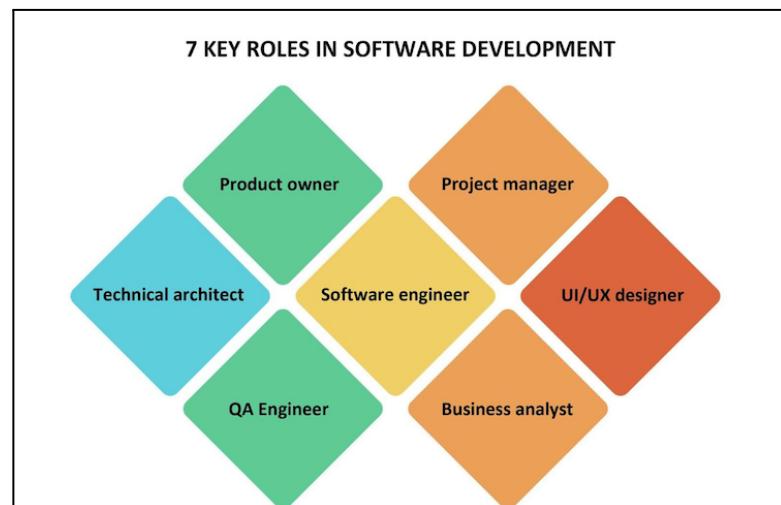
1. Development of Software



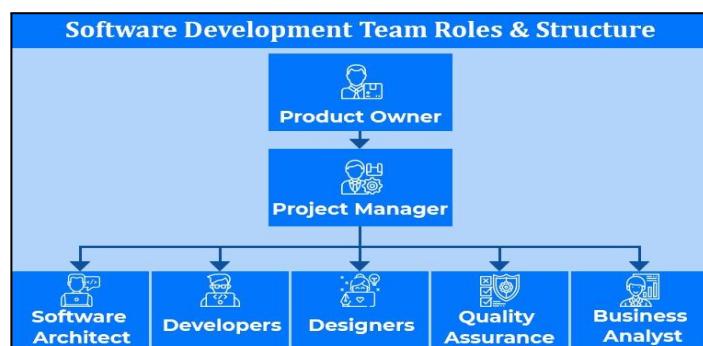
2. Software Engineering



3. Development team



4. Operation team





5. Operations Step



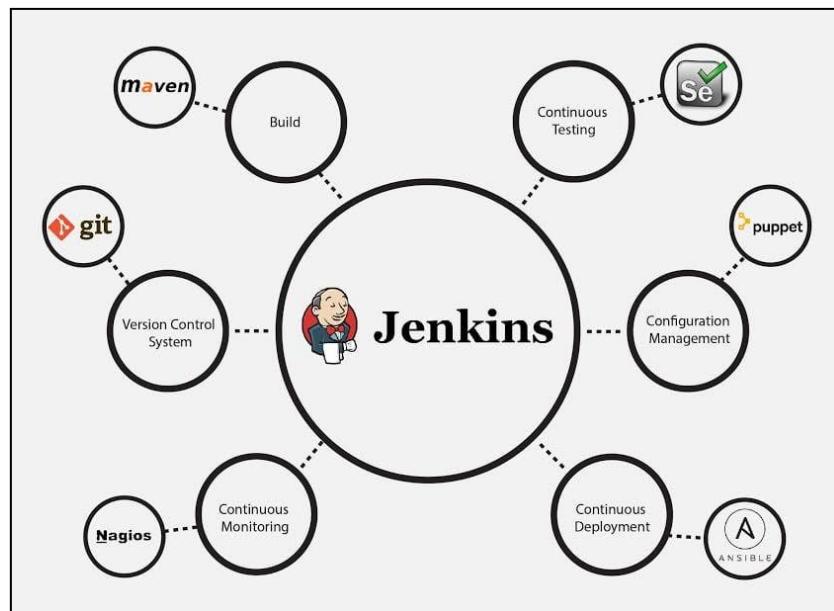
6. GitHub



7. Jenkins

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

It is a server-based system that runs in servlet containers such as Apache Tomcat.



8. Docker

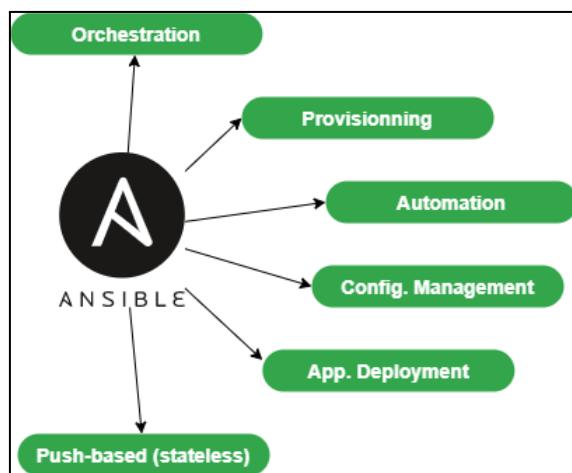
Docker is an open platform for developing, shipping, and running applications.

Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.

Features of Docker:

- Faster and easier configuration.
- Application isolation.
- Increase in productivity.
- Security Management.

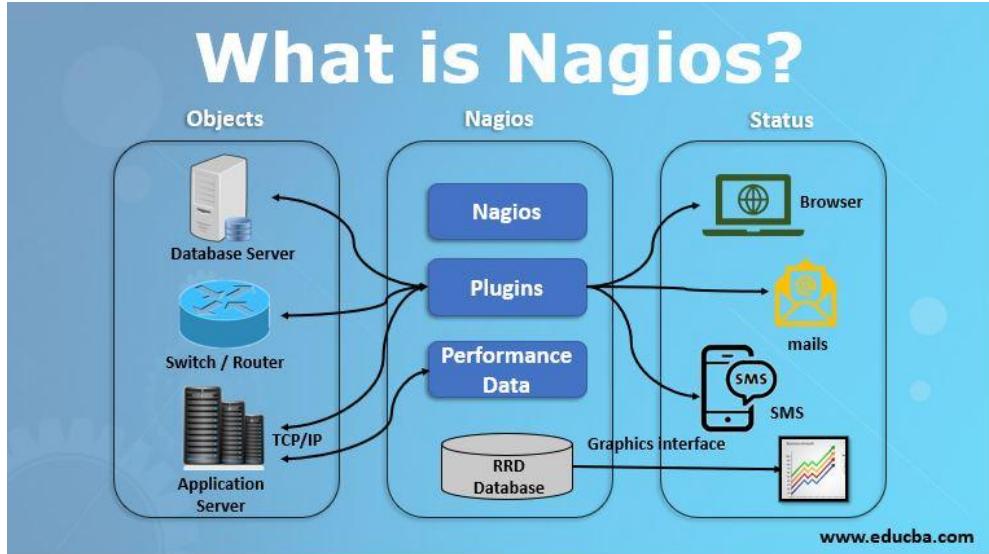
9. Ansible



Features:

- Open source
- Simple
- Versatile
- Powerful
- Agentless

10. NegiOs



Nagios is available as open-source software, and it is used to monitor computer systems. It can be executed on a Linux operating system to screen the devices which are executed on Windows, Unix, and Linux operating systems.

Conclusion: Thus, study was done and to come up with a set of question & answers that can help one to clear or understand basics understanding of DevOps.

References:

[What is Nagios? | Uses,Importance and Architecture of Nagios](#)

[Ansible tutorial | what is ansible? | advantages of ansible - tutorialsinhand.com](#)

[8 steps to developing an Ansible role in Linux | Enable Sysadmin..](#)

[What is Jenkins? | Jenkins For Continuous Integration | Edureka](#)

[What is GitHub - javatpoint](#)

EXPERIMENT B

Case Study - “Github”

Date of Experiment: 25/07/2022

Aim :

To identify and analyze the latest open source DevOps tools in the market

Procedure :

A group of 3 members will perform case study over internet resources with the help of research papers by answering the questions What, Why, Where and How.

Our team members are :

Jasmit Rathod

Rosy

Noel Wilbret

Presentation:



Case Study On GitHub

MEMBERS

1.Jasmit Rathod	54
2.Vijaya Saira Rosy	66
3. Noel Wilbret	53

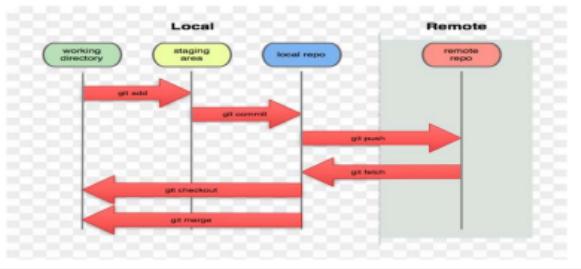
What is Github?

1. GitHub was founded by **Tom Preston-Werner** and was launched in **April 2008**.
2. It is an Internet hosting service for software development and version control using Git.
3. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration for every project.
4. It is commonly used to host open source software development projects.

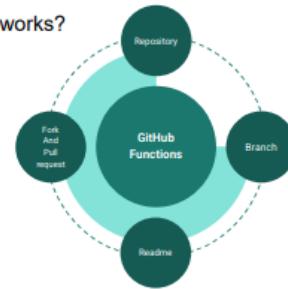
Why Github?

1. GitHub is a code hosting platform for version control and collaboration.
2. A community of over 30 million people helps you on your project and issues.
3. It is helpful for recruitment.
4. It also acts as a Social Network platform , different users can follow other user's profile and keep a track of their activities.

How GitHub works?



How GitHub works?



References

1. <https://ieeexplore.ieee.org/abstract/document/7887704>
2. <https://www.toolsqa.com/git/github/>
3. <https://en.wikipedia.org/wiki/GitHub>

Experiment 1: Version Control using Git

Date of the Experiment: 01/08/2022

Objective:

Is to experience version controlling using GITHUB by answering basic questions like what is Git, GitHub, Clone Repository, Forking and branching

Aim:

Is to use, analyse and experience all version control commands of GIT tool and GitHub Service.

Procedure / Steps to perform the Experiment:

1. Download and install Git tool
2. Create / Use GitHub Account with some directories/Repositories
3. Perform all the below mention commands of Git Local Repository and GitHub Service to reflect upon the version control.
 - a. Create and fork repositories in GitHub
 - b. Apply branching, merging and rebasing concepts.
 - c. Implement different Git workflow strategies in real-time scenarios

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning-fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

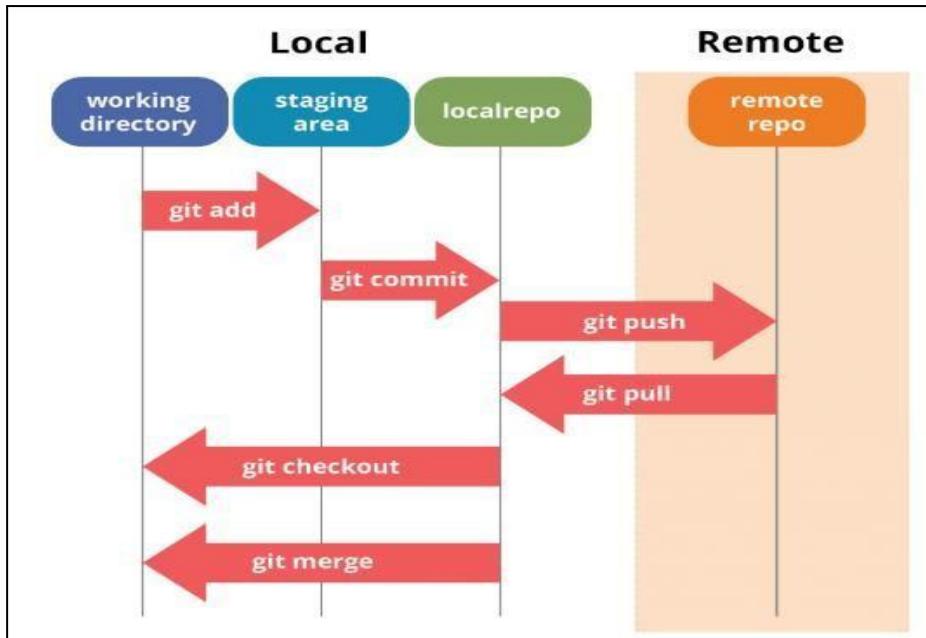
Some of the basic operations in Git are:

1. Initialize
2. Add
3. Commit
4. Pull
5. Push

Some advanced Git operations are:

1. Branching
2. Merging
3. Rebasing

The following diagram depict the all-supported operations in GIT



Commands:

a. Basic Commands:

```
sudo apt-get install git
git -version
git init
```

//configuration details

```
git config --global user.name "sunanthag1998"
git config --global user.email "sunanthag1998@gmail.com"
git config -list
```

b. Staging Commands

```
git add .
git commit -m "v1 of file 1"
```

c. Push and pull the repository from and to GitHub

- Go to GitHub ->User accounts-> setting -> developers portal ->generate token
- And choose the duration for it validity to exchange the repo between Git & GitHub and use the token in the below command to set the authentication for push and pull.

```
git remote add origin https://github.com/sunanthag1998/test4.git //directing to the
repository
git clone "https://github.com/sunanthag1998/test4.git" // cloning remote repository to local
```

```
git remote set-url origin  
http://sunanthag1998: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@github.com  
/sunanthag1998/test4.git
```

(Syntax: git remote set-url origin https://userid:password@github.com/user/repo.git)

```
git add -all  
git commit -m "v1" //Try to commit and then push or pull  
git branch -u origin main // branch a copy from main to reflect changes on the branch  
git push -u origin main // push from git to github  
git pull -u origin main //pull form github to git
```

d: Fork , Branch & Merge Commands

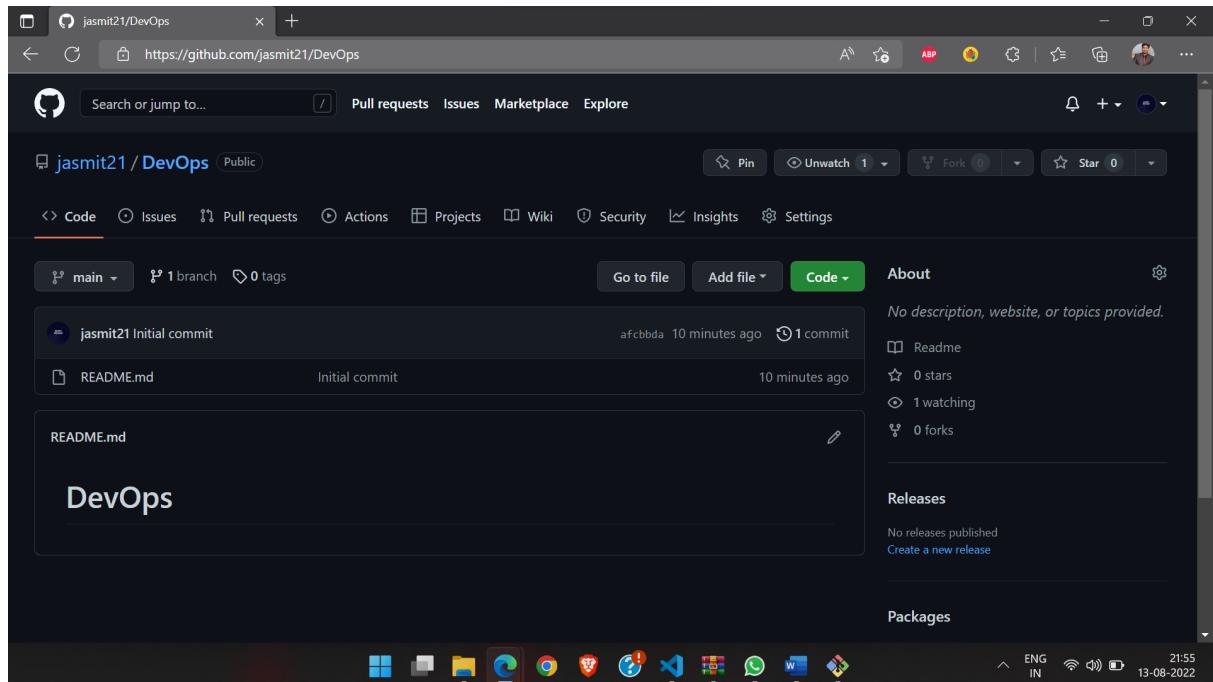
next lab

If you have existing repository, then simply delete .git file and reinitialize it.

```
$ rm -rf .git/
```

Output:

- Create a Github repository:



- **Clone the repository:**

Run the command in terminal: **git clone repo-link**, this will create a copy of the repo locally in our pc

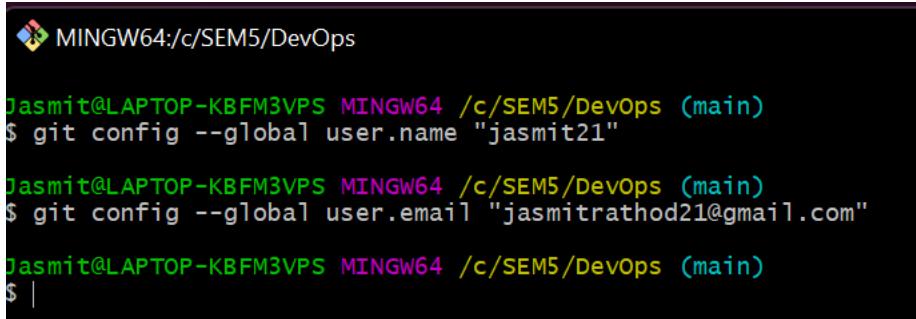
The screenshot shows a Windows PowerShell window and a File Explorer window. The PowerShell window displays the command: PS C:\SEMS> git clone https://github.com/jasmit21/DevOps.git. The File Explorer window shows the directory structure: This PC > Windows (C:) > SEMS. Inside SEMS, there are two folders: CSS-My Site and DevOps. The DevOps folder was just cloned from GitHub.

- **Open gitbash into the project folder (DevOps folder in my case):**

Run command: **git init**, this will create a .git file and then create a file index.html using the command: **touch index.html**

The screenshot shows a File Explorer window and a Git Bash terminal window. The File Explorer window shows the directory structure: This PC > Windows (C:) > SEMS > DevOps. Inside DevOps, there are two files: index.html and README.md. The Git Bash terminal window shows the following commands being run:
Jasmit@LAPTOP-KBFM3VPS MINGW64 ~
\$ cd C:/SEMS/DevOps
bash: cd: C:/SEMS/DevOps: No such file or directory
Jasmit@LAPTOP-KBFM3VPS MINGW64 ~
\$ cd c:/SEMS/DevOps
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEMS/DevOps (main)
\$ git init
Reinitialized existing Git repository in c:/SEMS/DevOps/.git/
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEMS/DevOps (main)
\$ touch index.html
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEMS/DevOps (main)
\$ |

- Configure the git :

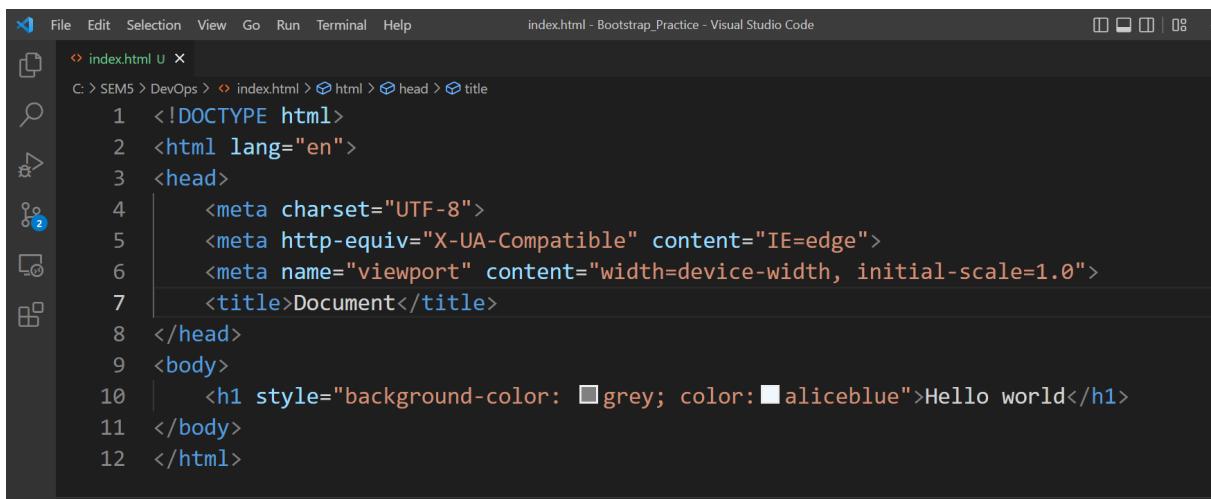


```
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git config --global user.name "jasmit21"

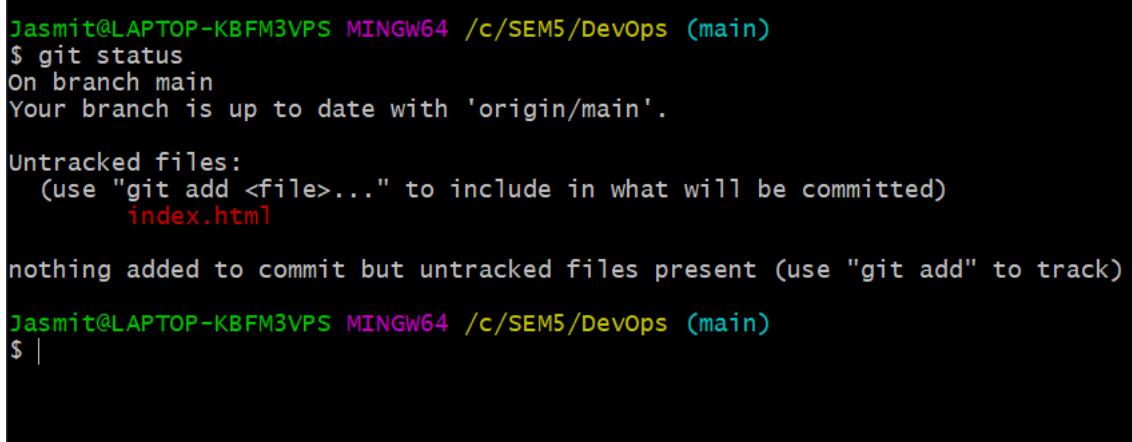
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git config --global user.email "jasmitrathod21@gmail.com"

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ |
```

- Edit the file:



- Check the status of the file:



```
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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)

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ |
```

- Bringing the untracked file to staging area:

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
\$ git add .

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
\$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
\$

- Commit the file in staging area:

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
\$ git commit -m "initial commit"
[main e652ce9] initial commit
 1 file changed, 12 insertions(+)
 create mode 100644 index.html

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
\$

- Check the commit history using “git log”:

```
git add .

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git log
commit e652ce914f2e49f5edc0df538e2c3a13b0361447 (HEAD -> main)
Author: jasmit21 <jasmitrathod21@gmail.com>
Date:   Sat Aug 13 22:12:06 2022 +0530

    initial commit

commit afcbbdad846a529a0bb62ec68cef6f94ac283127 (origin/main,
origin/HEAD)
Author: Jasmit Rathod <89774786+jasmit21@users.noreply.github.
com>
Date:   Sat Aug 13 21:44:48 2022 +0530

    Initial commit

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ |
```

- Connect the github repo with the local folder:

```
MINGW64:/c/SEM5/DevOps

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git remote add ORIGIN https://github.com/jasmit21/DevOps

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ |
```

- Run git remote -v to check the repo link:

```
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git remote -v
ORIGIN  https://github.com/jasmit21/DevOps (fetch)
ORIGIN  https://github.com/jasmit21/DevOps (push)
```

- Push the file into github

```
MINGW64:/c/SEM5/DevOps

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git push origin
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 499 bytes | 249.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/jasmit21/DevOps.git
  afcbbda..e652ce9  main -> main

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ |
```

The file index.html is successfully pushed into github repo

A screenshot of a GitHub repository page for 'jasmit21 / DevOps'. The repository is public. The 'Code' tab is selected. Below it, the 'main' branch is shown with 1 branch and 0 tags. A commit titled 'jasmit21 initial commit' is listed, made by 'jasmit21' at 'e652ce9' 7 minutes ago, containing 2 commits. The commit details show 'index.html' and 'README.md' files with their respective commit messages and times.

File	Commit Message	Time
index.html	initial commit	7 minutes ago
README.md	Initial commit	34 minutes ago

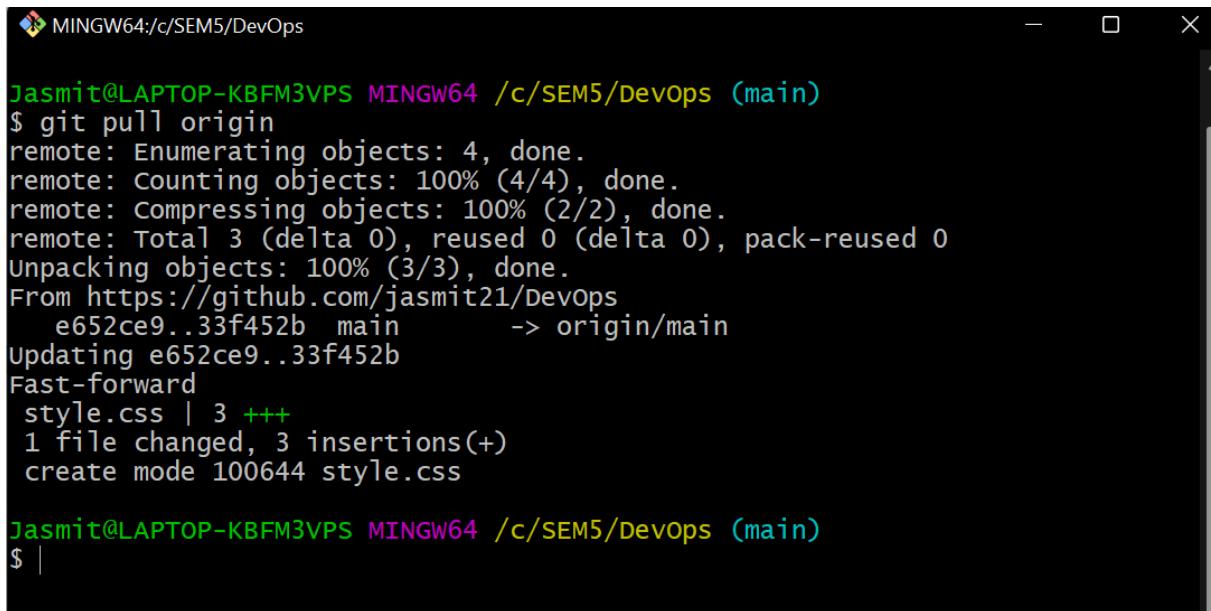
- **Pull the file from github repo:**

To use pull command , firstly create a file in github (styles.css in my case)

This screenshot shows two parts of a GitHub session. The top part is a modal window titled 'Commit new file' where 'style.css' is being committed directly to the 'main' branch. The bottom part is the same GitHub repository page as before, now showing the 'style.css' file in the list of committed files. The commit history shows three commits: 'jasmit21 styles file' (33f452b, now), 'index.html' (initial commit, 39 minutes ago), and 'README.md' (Initial commit, 12 minutes ago).

File	Commit Message	Time
style.css	styles file	now
index.html	initial commit	12 minutes ago
README.md	Initial commit	39 minutes ago

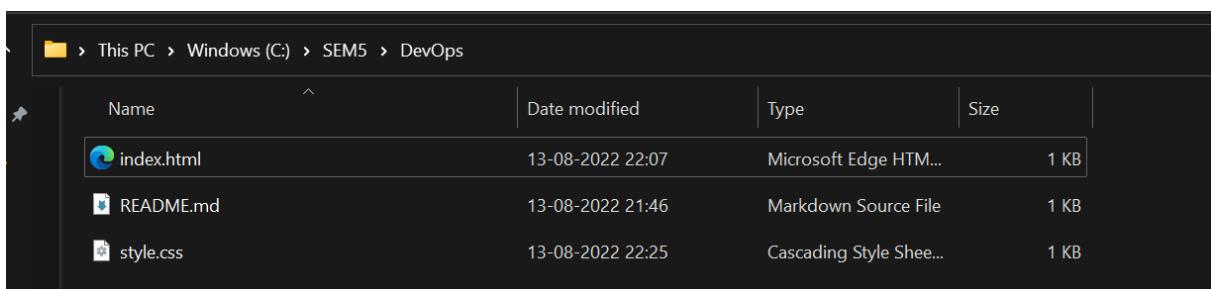
Now run command : **git pull origin**



```
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git pull origin
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/jasmit21/DevOps
  e652ce9..33f452b main      -> origin/main
Updating e652ce9..33f452b
Fast-forward
  style.css | 3 +++
  1 file changed, 3 insertions(+)
  create mode 100644 style.css

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ |
```

Checking for styles.css in local folder ...



Name	Date modified	Type	Size
index.html	13-08-2022 22:07	Microsoft Edge HTM...	1 KB
README.md	13-08-2022 21:46	Markdown Source File	1 KB
style.css	13-08-2022 22:25	Cascading Style Shee...	1 KB

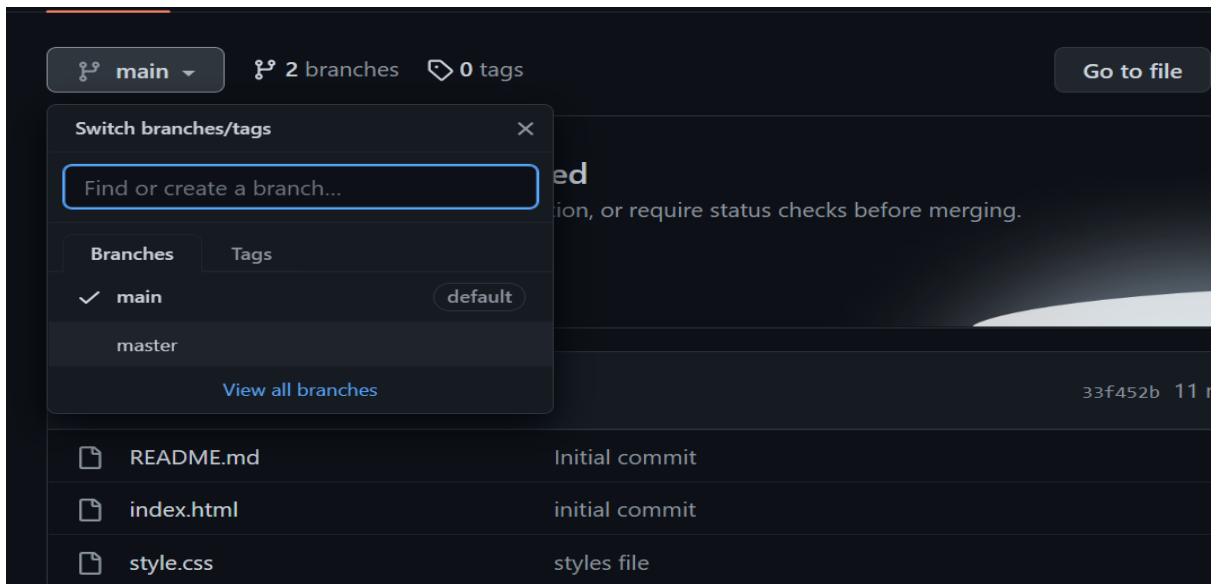
Styles.css pulled successfully.

- **Create a branch**



```
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:     https://github.com/jasmit21/DevOps/pull/new/master
remote:
To https://github.com/jasmit21/DevOps.git
 * [new branch]      master -> master
```

Check in github



- **Make changes in index.html while in master branch:**

Make changes in the file and then perform add , commit and push command in master branch

```
MINGW64:/c/SEM5/DevOps
$ git add .

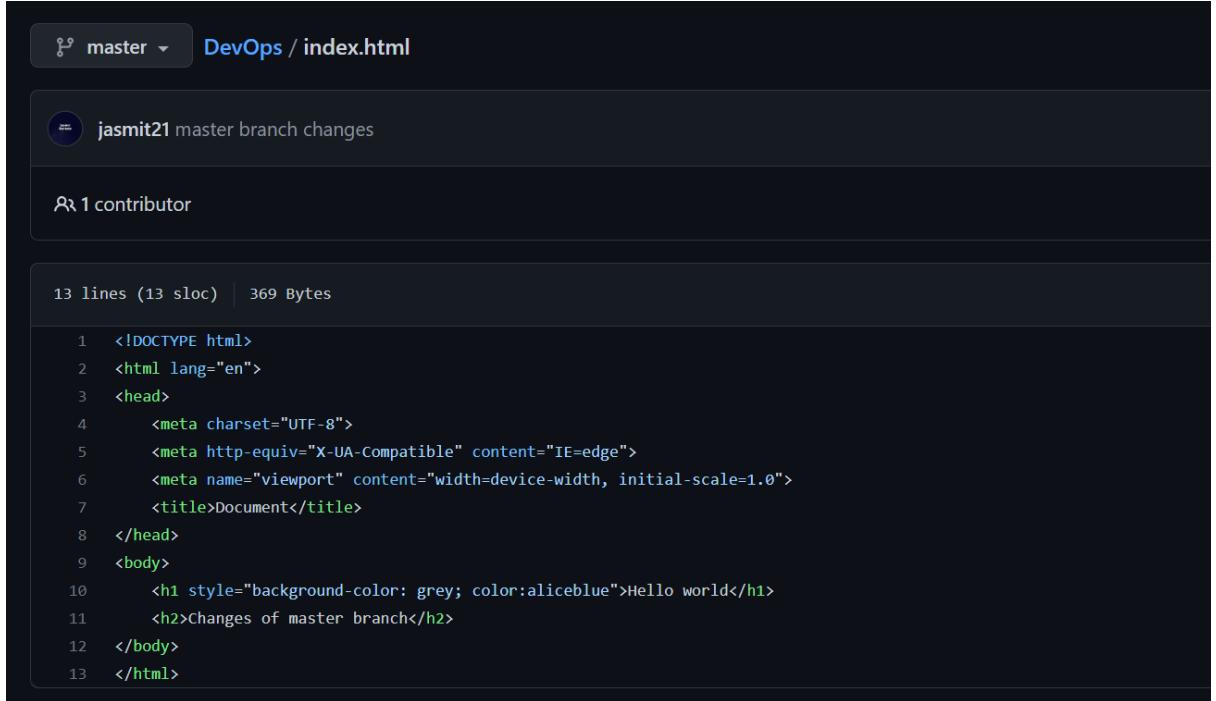
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/Devops (master)
$ git commit -m "master branch changes"
[master 84fee35] master branch changes
 1 file changed, 1 insertion(+)

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/Devops (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 390 bytes | 390.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/jasmit21/DevOps.git
 33f452b..84fee35  master -> master

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/Devops (master)
```

Check master branch in github

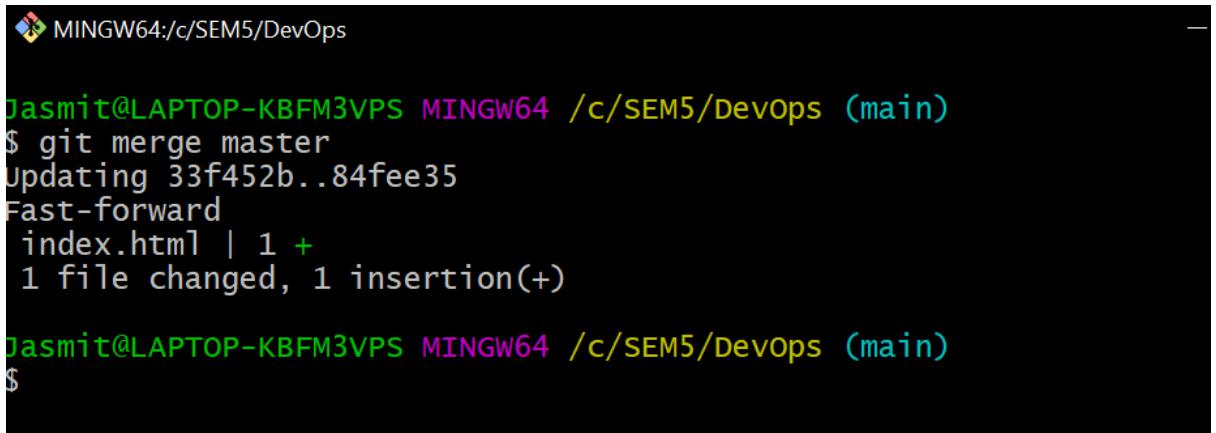
Changes are reflected in master branch



The screenshot shows a GitHub commit page for the file `index.html` in the `master` branch. The commit is titled "jasmit21 master branch changes" and has 1 contributor. The commit message contains 13 lines (13 sloc) and 369 Bytes. The code changes are as follows:

```
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  <h1 style="background-color: grey; color:aliceblue">Hello world</h1>
11  <h2>Changes of master branch</h2>
12 </body>
13 </html>
```

- **Merging branches**



The terminal window shows the command `git merge master` being run in a directory named `/c/SEM5/DevOps`. The output indicates a fast-forward merge where `index.html` was updated, resulting in 1 file changed and 1 insertion(+).

```
Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$ git merge master
Updating 33f452b..84fee35
Fast-forward
 index.html | 1 +
 1 file changed, 1 insertion(+)

Jasmit@LAPTOP-KBFM3VPS MINGW64 /c/SEM5/DevOps (main)
$
```

Checking in GitHub.

Now the changes of the master branch are reflecting in the main branch .Hence merging is successful.



main DevOps / index.html

jasmit21 master branch changes

1 contributor

13 lines (13 sloc) | 369 Bytes

```
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  <h1 style="background-color: grey; color:aliceblue">Hello world</h1>
11  <h2>Changes of master branch</h2>
12 </body>
13 </html>
```

Conclusion:

Thus, the basic commands to access the GitHub version control were performed successfully.

EXPERIMENT 2

Jenkins

Date of Experiment: 12/09/2022

Output:

Add the repository key to the system:

```
javac 11.0.10
jasmit21@jasmit21:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key |sudo gpg --dearmor -o /usr/share/keyrings/jenkins.gpg
-----BEGIN PGP DEARMORED KEY BLOCK-----
```

Next, let's append the Debian package repository address to the server's sources.list

```
jasmit21@jasmit21:~$ sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg] http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
jasmit21@jasmit21:~$ sudo apt update
```

After both commands have been entered, we'll run an update so that apt will use the new repository.

```
jasmit21@jasmit21:~$ sudo apt update
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:3 https://pkg.jenkins.io/debian-stable binary/ Release [2,044 B]

Get:4 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Get:6 https://pkg.jenkins.io/debian-stable binary/ Packages [23.2 kB]
Hit:7 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease

Hit:8 http://security.ubuntu.com/ubuntu jammy-security InRelease
Fetched 26.1 kB in 1s (28.1 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
3 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Finally, we'll install Jenkins and its dependencies.

```
jasmit21@jasmit21:~$ sudo apt install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver intel-media-va-driver libaacs0
  libaom3 libass9 libavcodec58 libavformat58 libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0
  libchromaprint1 libcodec2-1.0 libdav1d5 libflite1 libgme0 libgsm1 libgstreamer-plugins-bad1.0-0
  libigdgmm12 liblilv-0-0 libmfx1 libmysofa1 libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55
  librabbitmq4 librubberband2 libserd-0-0 libshine3 libsnappy1v5 libsrord-0-0 libsratom-0-0
  libsrt1.4-gnutls libssh-gcrypt-4 libswresample3 libswscale5 libudfread0 libva-drm2 libva-wayland2
  libva-x11-2 libva2 libvpau1 libvidstab1.1 libx265-199 libxvidcore4 libzimg2 libzmq5
  libzvbi-common libzvbi0 mesa-va-drivers mesa-vpau-drivers pocketsphinx-en-us va-driver-all
  vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  net-tools
The following NEW packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
```

Step 2 — Starting Jenkins

Let's start Jenkins by using systemctl

```
jasmit21@jasmit21:~$ sudo systemctl start jenkins.service
[sudo] password for jasmit21:
jasmit21@jasmit21:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-10-27 20:50:38 IST; 22min ago
     Main PID: 43709 (java)
        Tasks: 49 (limit: 9175)
       Memory: 1.1G
          CPU: 1min 10.484s
        CGroup: /system.slice/jenkins.service
                 └─43709 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/jenkins_home
```

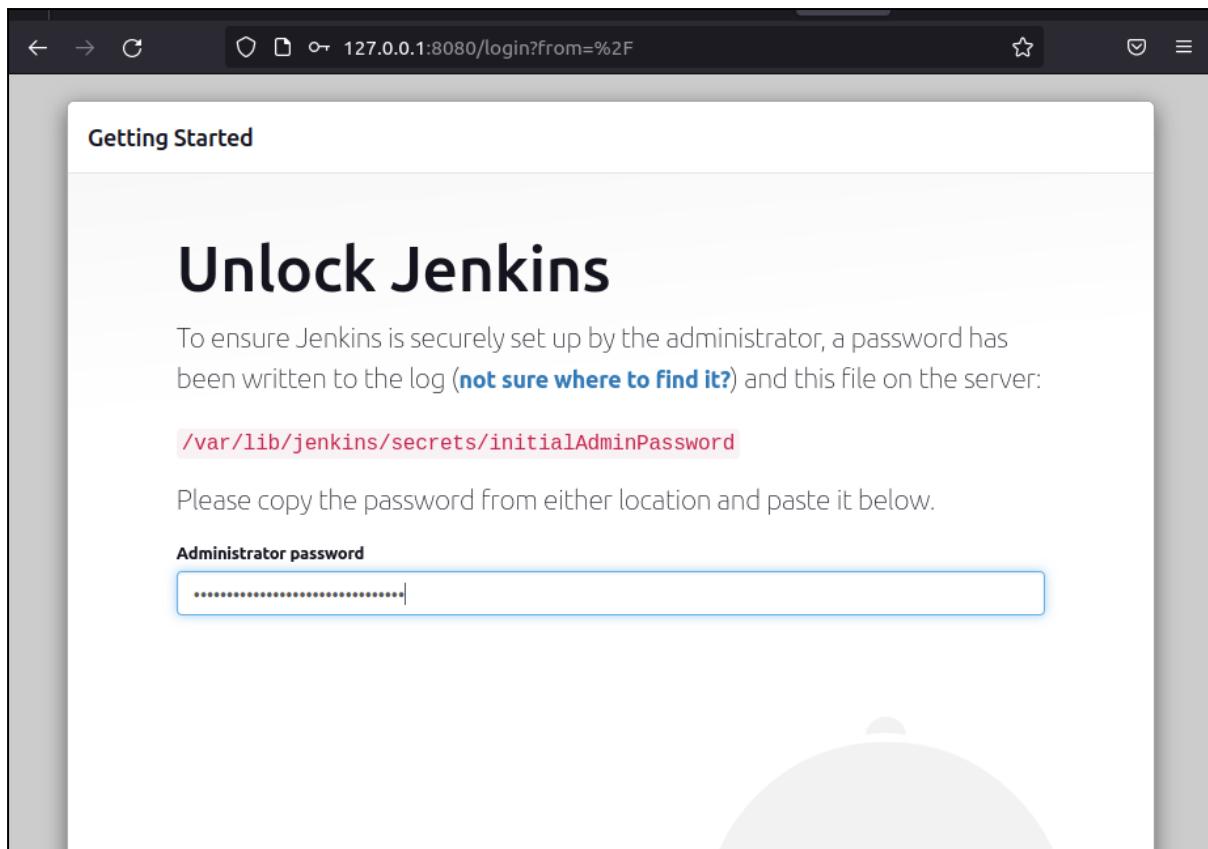


```
Oct 27 20:49:50 jasmit21 jenkins[43709]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Oct 27 20:49:50 jasmit21 jenkins[43709]: ****
Oct 27 20:49:50 jasmit21 jenkins[43709]: ****
Oct 27 20:49:50 jasmit21 jenkins[43709]: ****
Oct 27 20:49:50 jasmit21 jenkins[43709]: 2022-10-27 15:20:36.010+0000 [id=60]      INFO      h.m>
Oct 27 20:50:36 jasmit21 jenkins[43709]: 2022-10-27 15:20:36.012+0000 [id=60]      INFO      hud>
Oct 27 20:50:36 jasmit21 jenkins[43709]: 2022-10-27 15:20:36.014+0000 [id=60]      INFO      hud>
Oct 27 20:50:38 jasmit21 jenkins[43709]: 2022-10-27 15:20:38.377+0000 [id=35]      INFO      jen>
Oct 27 20:50:38 jasmit21 jenkins[43709]: 2022-10-27 15:20:38.394+0000 [id=24]      INFO      hud>
Oct 27 20:50:38 jasmit21 systemd[1]: Started Jenkins Continuous Integration Server.
```

Lines 1-20/20 (END)

Step 3 — Setting Up Jenkins

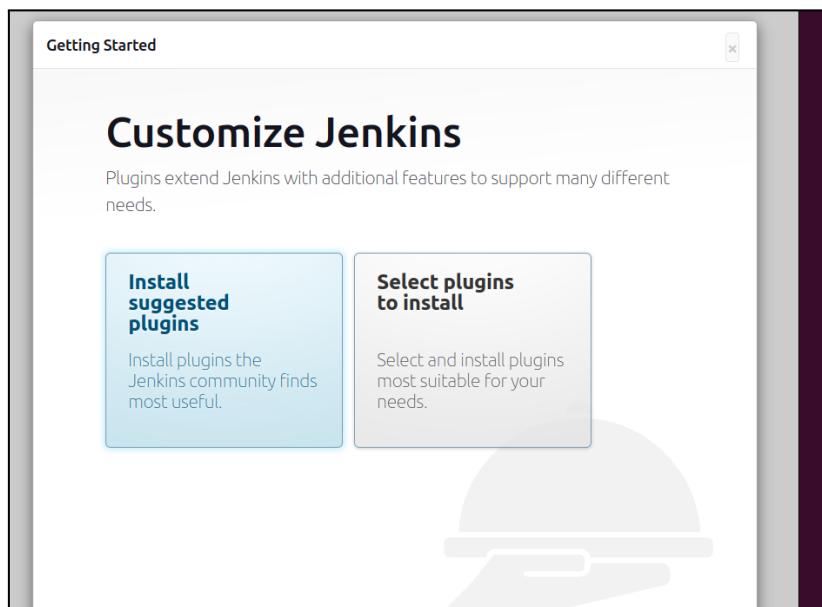
To set up your installation, visit Jenkins on its default port, 8080, using your server domain name or IP address: <http://127.0.0.1:8080/>



In the terminal window, use the cat command to display the password

```
[root@jasmit21 ~]# jasmit21@jasmit21: ~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
3efbfd73078d403fa7aeaa3fce9297b  
jasmit21@jasmit21: ~$
```

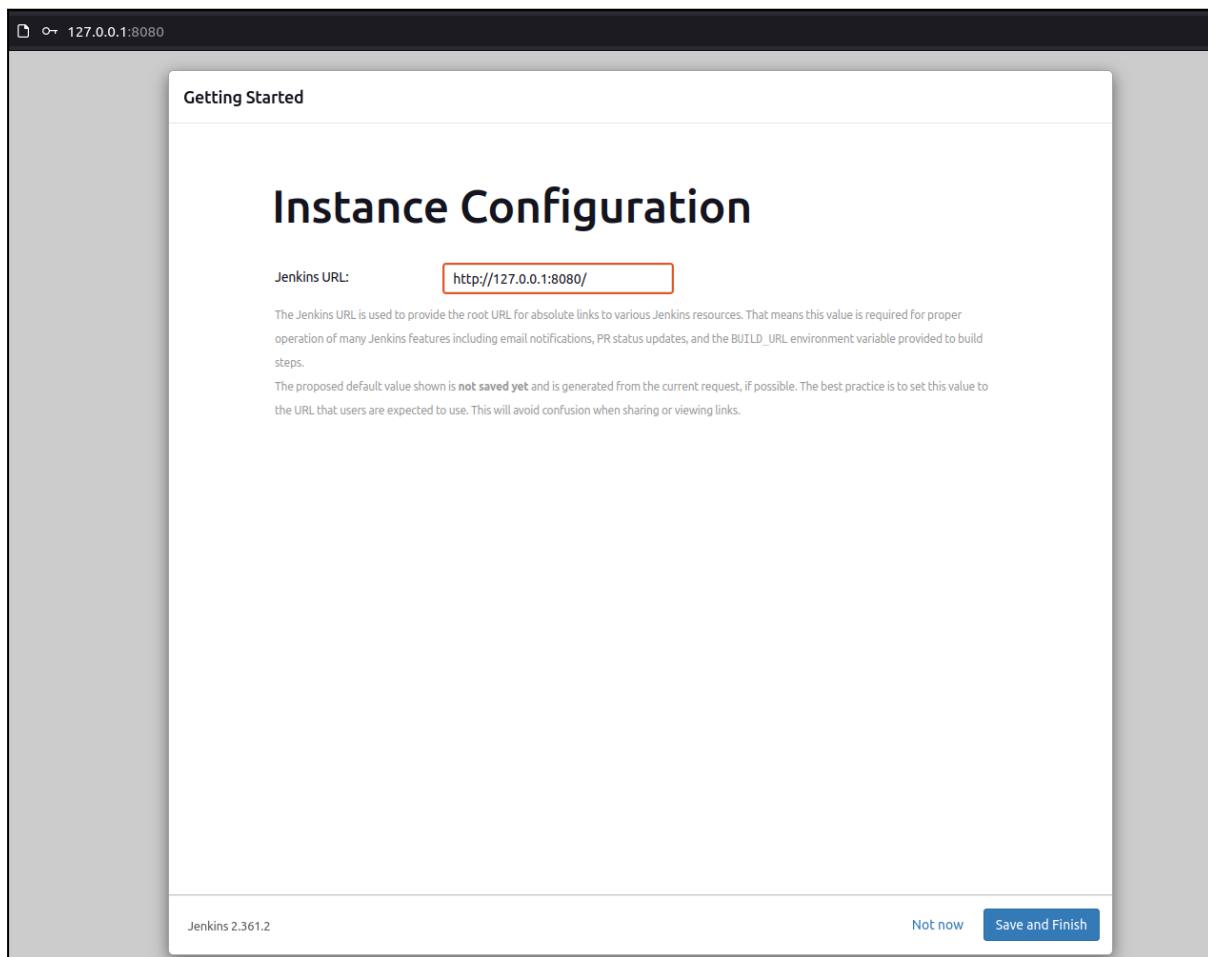
The next screen presents the option of installing suggested plugins



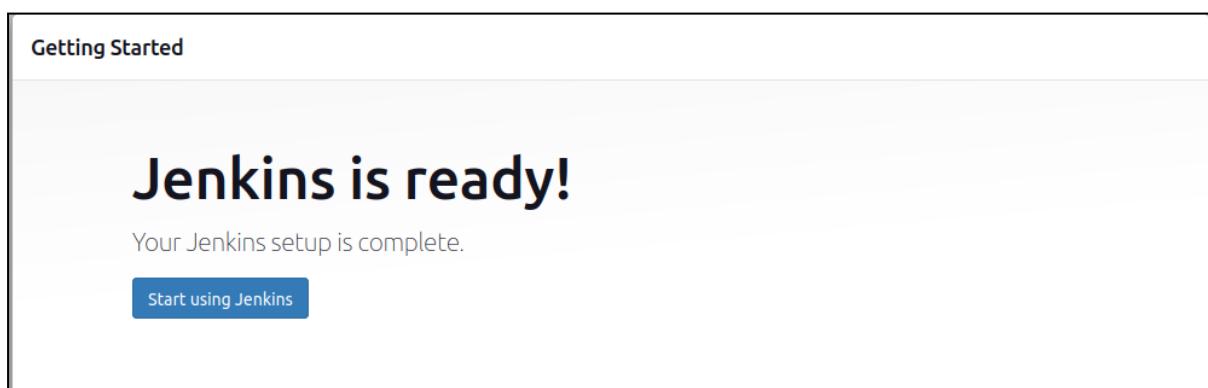
Enter the name and password for your user

A screenshot of the Jenkins 'Getting Started' wizard showing the 'Create First Admin User' step. The page has a form with five fields: 'Username', 'Password', 'Confirm password', 'Full name', and 'E-mail address'. At the bottom, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The URL in the browser bar is '127.0.0.1:8080'.

You'll receive an Instance Configuration page that will ask you to confirm the preferred URL for your Jenkins instance.



After confirming the appropriate information, click Save and Finish. You'll receive a confirmation page confirming that "Jenkins is Ready!"



Click Start using Jenkins to visit the main Jenkins dashboard

The screenshot shows the Jenkins main dashboard. On the left, there's a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are two sections: 'Build Queue' (empty) and 'Build Executor Status' (two idle executors). The main content area is titled 'Welcome to Jenkins!'. It includes a brief introduction: 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' Below this are three buttons: 'Create a job', 'Set up a distributed build', and 'Learn more about distributed builds'. A 'Start building your software project' section is also present.

Click on new item and create a new project and use the freestyle project plugin

The screenshot shows a 'Enter an item name' dialog box. At the top, there's a text input field containing the letter 'H', with a note '(Required field)' below it. Below the input field is a list of project types with their descriptions: 'Freestyle project', 'Pipeline', 'Multi-configuration project', 'Folder', 'Multibranch Pipeline', and 'Organization Folder'. At the bottom of the dialog is an 'OK' button.

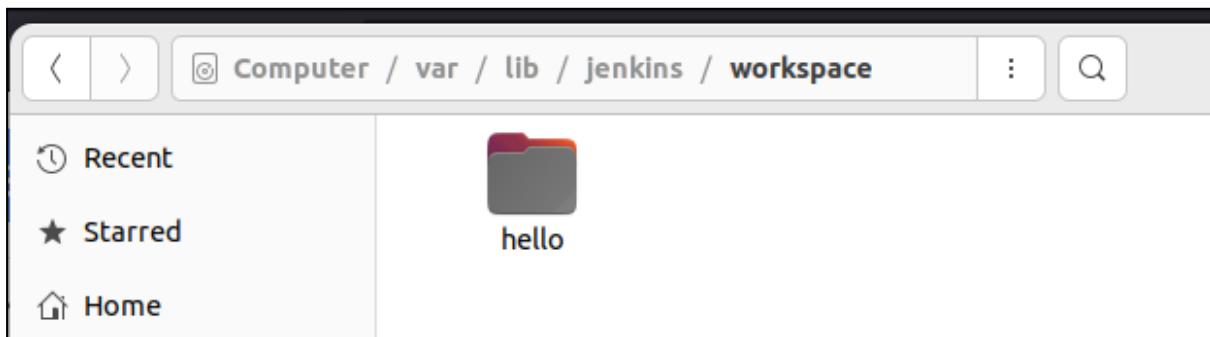
Exercise to create and run a java program using terminal

Create a new freestyle project

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	🕒	hello	1 hr 31 min #6	1 hr 36 min #5	0.47 sec
✗	🕒	HelloWorld	N/A	2 hr 5 min #3	2 ms

Icon: S M L Icon legend Atom feed for all Atom feed for failures Atom feed for just latest builds

Build it, and you will see that it is locally present



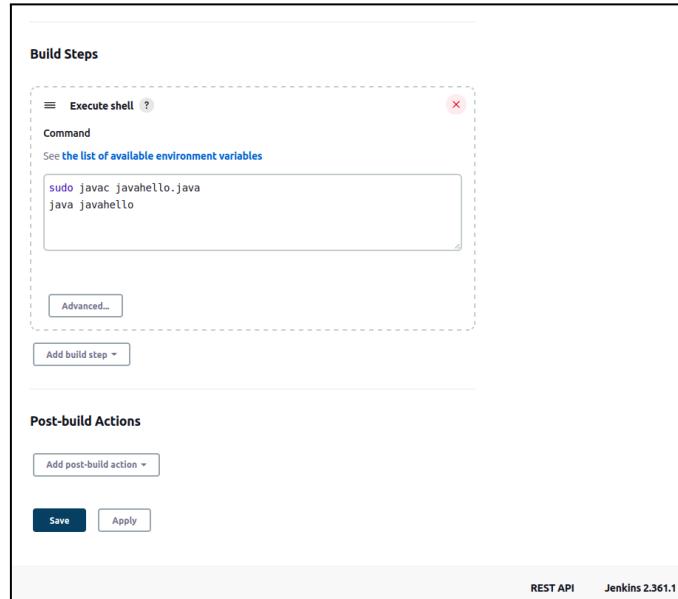
Write a java program

```
purpleven@purpleven:/var/lib/jenkins/workspace/javaprogram$ cat javahello.java
class javahello {
    public static void main(String[] args) {
        System.out.println("Hello!!!");
    }
}
purpleven@purpleven:/var/lib/jenkins/workspace/javaprogram$
```

Compile the java program

```
purpleven@purpleven:/var/lib/jenkins/workspace/javaprogram$ sudo javac javahello.java
purpleven@purpleven:/var/lib/jenkins/workspace/javaprogram$ sudo java javahello
Hello!!!
purpleven@purpleven:/var/lib/jenkins/workspace/javaprogram$
```

Configure the java program and add the two lines in the build step

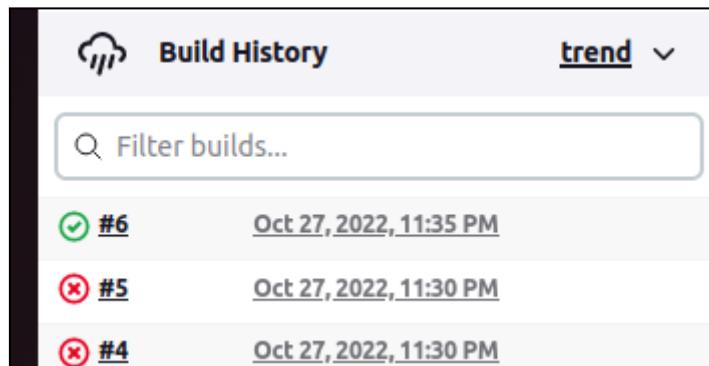


The screenshot shows the 'Build Steps' section of a Jenkins job configuration. It contains one 'Execute shell' step with the command:

```
sudo javac javahello.java  
java javahello
```

Below the build steps is a 'Post-build Actions' section with a 'Save' button.

Click on build now



The screenshot shows the Jenkins 'Build History' page. It lists three builds:

- #6: Oct 27, 2022, 11:35 PM (Success)
- #5: Oct 27, 2022, 11:30 PM (Failure)
- #4: Oct 27, 2022, 11:30 PM (Failure)

Check the output



The screenshot shows the Jenkins 'Console Output' page for build #6. The output log is as follows:

```
Started by user Jasmit Rathod  
Running as SYSTEM  
Building in workspace /var/lib/jenkins/workspace/hello  
[hello] $ /bin/sh -xe /tmp/jenkins16064297320869398180.sh  
+ javac hello.java  
+ java hello  
hello  
Finished: SUCCESS
```

The java program is successfully demonstrated on jenkins

S	W	Name ↓	Last Success	Last Failure	Last Duration	
✓	🕒	hello	9 min 22 sec #6	14 min #5	0.47 sec	▶

Exercise to create a shell script and version it in Jenkins

Enter an item name

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system,

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (former Jenkins Pipeline)

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a list of items, a folder creates a tree structure where items are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy From

```
jasmit21@jasmit21:~$ cat shelljen.sh
#!/bin/bash
echo "Hey! Jasmit Here"
jasmit21@jasmit21:~$
```

```
jasmit21@jasmit21:~$ sudo chmod 777 shelljen.sh
jasmit21@jasmit21:~$ ./shelljen.sh
Hey! Jasmit Here
jasmit21@jasmit21:~$
```

Build Steps

≡ Execute shell ? ×

Command
See [the list of available environment variables](#)

```
/home/purpleven/Documents/shelljen.sh
```

[Advanced...](#)

[Add build step ▾](#)

Post-build Actions

[Add post-build action ▾](#)

[Save](#) [Apply](#)

Jenkins

Dashboard > bash-F1 > #4

Back to Project

✓ Build #4 (Sep 25, 2022, 4:43:01 PM)

[Status] Status

</> Changes

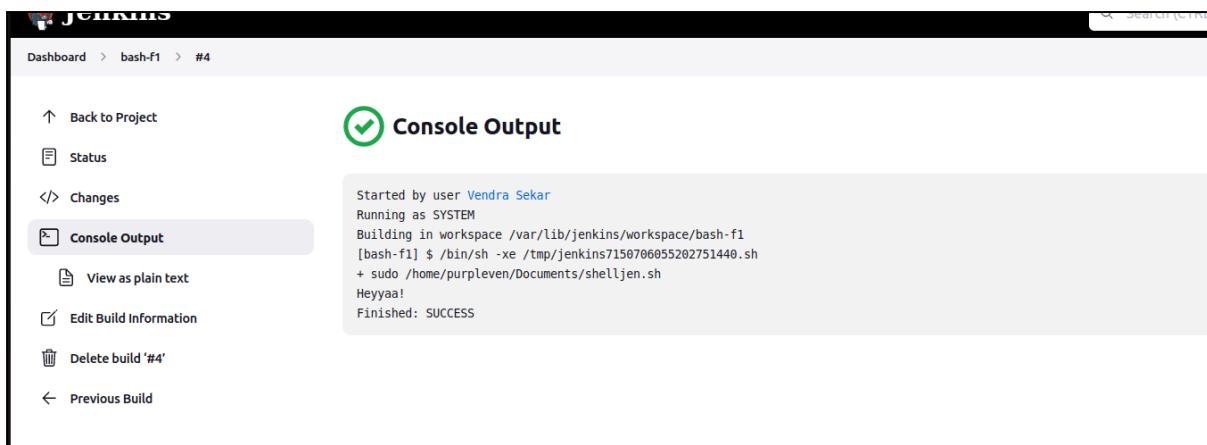
[Console Output] Console Output </> No changes.

[Edit Build Information] Edit Build Information

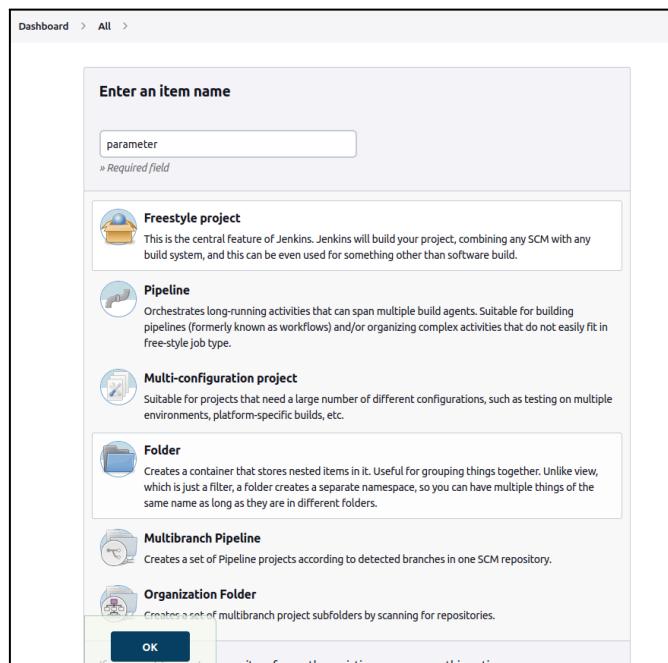
[Delete build '#4'] Delete build '#4'

[← Previous Build] Previous Build

Started by user [Vendra Sekar](#)



Exercise to Create a parameterized project



This project is parameterized ?

String Parameter ?

Name ?
Name

Default Value ?
Jasmit

Description ?
[Plain text] Preview

Trim the string ?

Choice Parameter ?

Name ?
City

Choices ?
Banvel
Pune
Mumbai

Save **Apply**

Build Steps

Execute shell ?

Command
See [the list of available environment variables](#)

```
echo "Hey $Name, you chose $city and clicked ok $ok"
```

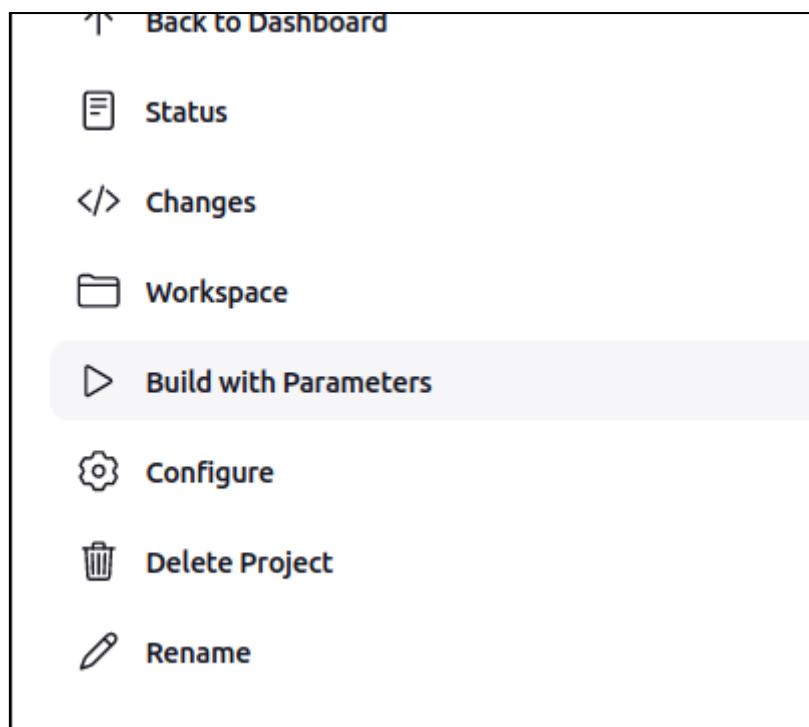
Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾

Save **Apply**



Project parameter

This build requires parameters:

Name: Jasmit

City: Panvel

Build

Build #2 (Sep 25, 2022, 4:49:52 PM)

Keep this build forever

Started 6.8 sec ago
Took 30 ms

Add description

</> No changes.

Started by user [Vendra Sekar](#)



Console Output

```
Started by user Jasmit Rathod
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/parameter
[parameter] $ /bin/sh -xe /tmp/jenkins11816420096904923811.sh
+ echo Hey , I am form Mumbai
Hey , I am form Mumbai
Finished: SUCCESS
```

Exercise: create a maven Project

Install maven

```
jasmit21@jasmit21:~$ sudo apt install maven
[sudo] password for jasmit21:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver
intel-media-va-driver libaacs0 libaom3 libass9 libavcodec58 libavformat58
libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1
libcodec2-1.0 libdav1d5 libflite1 libgme0 libgsm1
libgstreamer-plugins-bad1.0-0 libigdgmm12 liblilv-0-0 libmfx1 libmysofa1
libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4 librubberband2
libserd-0-0 libshine3 libsnappy1v5 libsord-0-0 libsratom-0-0
libsrt1.4-gnutls libssh-gcrypt-4 libswresample3 libswscale5 libudfread0
libva-drm2 libva-wayland2 libva-x11-2 libva2 libvpau1 libvidstab1.1
libx265-199 libxvidcore4 libzimg2 libzmq5 libzvbi-common libzvbi0
mesa-va-drivers mesa-vdpau-drivers pocketsphinx-en-us va-driver-all
```

```

) in auto mode
jasmit21@jasmit21:~$ mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.16, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en_IN, platform encoding: UTF-8
OS name: "linux", version: "5.15.0-52-generic", arch: "amd64", family: "unix"
jasmit21@jasmit21:~$ 

```

Create a maven project

Eg: <https://github.com/devopshint/java-app-with-maven/tree/main/my-app>

Install maven integration plugin from the plugin manager

The screenshot shows the Jenkins Plugin Manager interface. At the top, there are tabs for 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. Below the tabs is a search bar containing the text 'maven'. The main area displays a list of plugins:

- Maven Integration** 3.19 (Build Tools): This plugin provides a deep integration of Jenkins and Maven. It is currently selected, indicated by a checked checkbox. A tooltip explains its function: "This plug-in provides a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTs, automated configuration of various Jenkins publishers (Junit, ...). This plugin is deprecated and it's recommended to avoid using it." It was released 3 months and 25 days ago.
- Config File Provider** 3.11.1 (Groovy-related, External Site/Tool Integrations, Maven): Ability to provide configuration files (e.g. settings.xml for maven, XML, groovy, custom files,...) loaded through the UI which will be copied to the job workspace. It was released 2 months and 6 days ago.
- Jira** 3.8 (External Site/Tool Integrations, Maven, Jira): A plugin for integrating Jenkins with Jira. It was released 1 month and 2 days ago.

At the bottom of the list, there are two buttons: 'Install without restart' and 'Download now and install after restart'. To the right of these buttons, it says 'Update information obtained: 2 hr 52 min ago'.

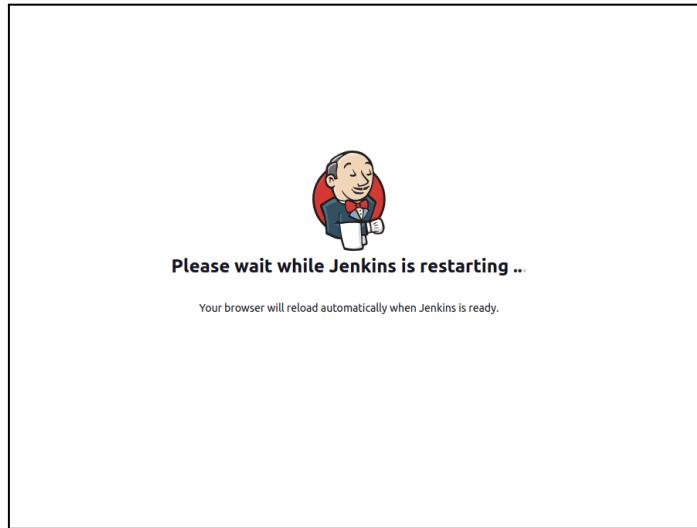
The screenshot shows the 'Installing Plugins/Upgrades' page. At the top, it says 'Preparation' with a list of steps: 'Checking internet connectivity', 'Checking update center connectivity', and 'Success'. Below this, it shows the status of individual plugins:

- Javadoc**: Downloaded Successfully. Will be activated during the next boot.
- Maven Integration**: Installing (progress bar shown).

Below the plugin status, there is a note: "→ [Go back to the top page](#) (you can start using the installed plugins right away)". There is also a checkbox for "Restart Jenkins when installation is complete and no jobs are running".

At the bottom right, it says 'REST API' and 'Jenkins 2.361'.

Restart jenkins



Create a new project

A screenshot of the Jenkins "Create a new project" interface. At the top, there is a field labeled "Enter an item name" containing the text "mavenProject". Below this, there is a note "» Required field". A list of project types is shown with icons next to each:

- Freestyle project**: 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.
- 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.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

At the bottom of the list, there is a note "... or you can always create a new item from other existing, you can use this option:" followed by a small arrow icon pointing right.

General

Description
Maven Project

[Plain text] [Preview](#)

Discard old builds ?

GitHub project

Project url ?
<https://github.com/devopshint/java-app-with-maven>

[Advanced...](#)

This project is parameterized ?

Throttle builds ?

Execute concurrent builds if necessary ?

[Advanced...](#)

Source Code Management

Source Code Management

Username with password

None

Git ?

Repositories

Repository

Please add a repository

Credentials

- none -

+ Add

Advanced

Add Repository

Branches to

Branch Spec

*/*master

Treat username as secret ?

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

Username ?
jasmit21

Password ?

The password.
(from [Credentials Plugin](#))

ID ?

Description ?
My Github

[Add](#) [Cancel](#)

None

Git [?](#)

Repositories [?](#)

Repository URL [?](#) ✖

Credentials [?](#)

[+ Add](#)

[Advanced...](#)

[Add Repository](#)

Build Triggers

Build whenever a SNAPSHOT dependency is built [?](#)
 Schedule build when some upstream has no successful builds [?](#)

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

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

Build periodically [?](#)

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

Poll SCM [?](#)

Build Environment

Delete workspace before build starts

Use secret text(s) or file(s) [?](#)

Add timestamps to the Console Output

Inspect build log for published Gradle build scans

Terminate a build if it's stuck

With Ant [?](#)

Maven Version

! Jenkins needs to know where your Maven is installed.
Please do so from the [tool configuration](#).

Root POM [?](#)

Goals and options [?](#)

[Advanced...](#)

Post Steps

Run only if build succeeds

Run only if build succeeds or is unstable

Run regardless of build result

Should the post-build steps run only for successful builds, etc.

 Build History	
<input type="text"/>	Filter builds...
 #4	Oct 31, 2022, 11:03 PM
 #3	Oct 31, 2022, 11:01 PM
 #2	Oct 31, 2022, 10:54 PM
 #1	Oct 31, 2022, 10:50 PM



Console Output

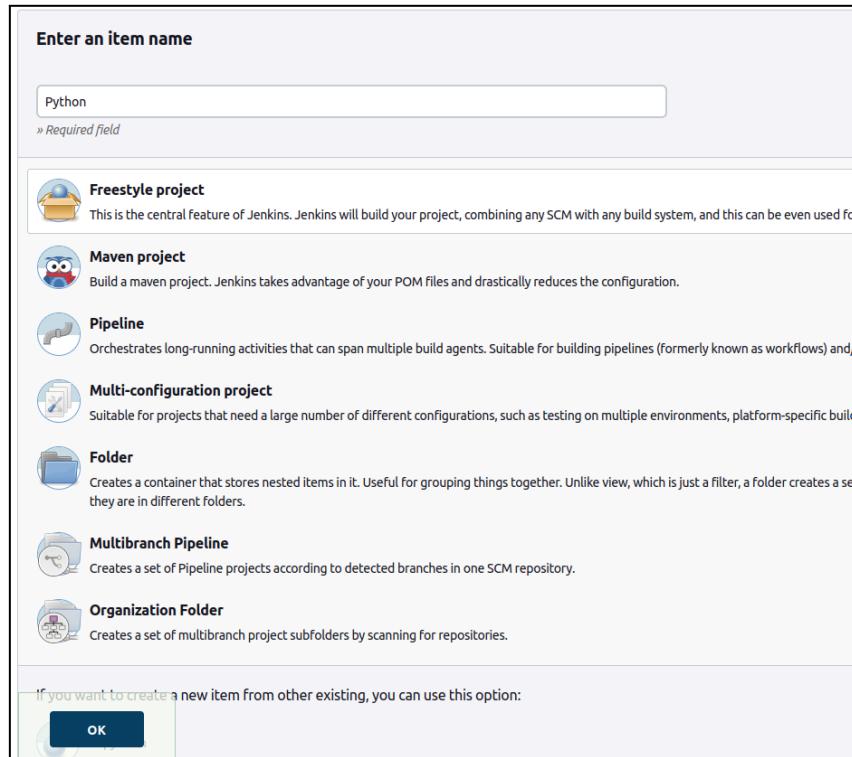
```
Started by user Jasmit Rathod
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/mavenProject
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] Done
The recommended git tool is: NONE
using credential 1bf64179-2a77-44f3-ae58-d56f7292df4e
Cloning the remote Git repository
Cloning repository https://github.com/devopshint/java-app-with-maven.git
> git init /var/lib/jenkins/workspace/mavenProject # timeout=10
Fetching upstream changes from https://github.com/devopshint/java-app-with-maven.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_ASKPASS to set credentials My Github
> git fetch --tags --force --progress -- https://github.com/devopshint/java-app-with-maven.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/devopshint/java-app-with-maven.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 620d5f39aaa3384e66672a6682d94fa62e2ce826 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 620d5f39aaa3384e66672a6682d94fa62e2ce826 # timeout=10
Commit message: "My first commit"
```

```
at 186 kB/s)
[INFO] Building jar: /var/lib/jenkins/workspace/mavenProject/my-app/target/my-app-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:15 min
[INFO] Finished at: 2022-10-31T23:04:51+05:30
[INFO] -----
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/mavenProject/my-app/pom.xml to com.mycompany.app/my-app/1.0-SNAPSHOT/my-app-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/mavenProject/my-app/target/my-app-1.0-SNAPSHOT.jar to com.mycompany.app/my-app/1.0-SNAPSHOT/my-app-1.0-SNAPSHOT.jar
channel stopped
Finished: SUCCESS
```

The maven project is created successfully



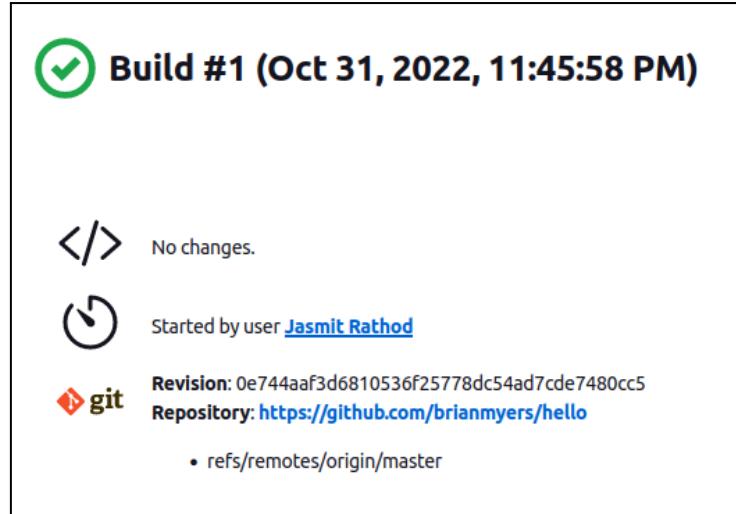
Exercise: to run a python simple program using freestyle



The screenshot shows the Jenkins configuration dialog for a Git repository. The "Git" option is selected. The configuration fields are:

- Repository URL**: https://github.com/brianmyers/hello
- Credentials**: jasmit21/******** (My Github)
- Branches to build**: Branch Specifier (blank for 'any'): */master

At the bottom, there are "Save" and "Apply" buttons.

A Jenkins console output card titled "Console Output" with a green checkmark icon. It displays the command-line log for the build:

```
Started by user Jasmit Rathod
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Python
The recommended git tool is: NONE
using credential 1bf64179-2a77-44f3-ae58-d56f7292df4e
Cloning the remote Git repository
Cloning repository https://github.com/brianmyers/hello
> git init /var/lib/jenkins/workspace/Python # timeout=10
Fetching upstream changes from https://github.com/brianmyers/hello
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_ASKPASS to set credentials My Github
> git fetch --tags --force --progress -- https://github.com/brianmyers/hello +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/brianmyers/hello # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 0e744aaf3d6810536f25778dc54ad7cde7480cc5 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 0e744aaf3d6810536f25778dc54ad7cde7480cc5 # timeout=10
Commit message: "Added FILES.txt"
First time build. Skipping changelog.
Finished: SUCCESS
```

Conclusion: Thus, installation and version controlling for various program was done successfully

References:

- <https://www.youtube.com/watch?v=-5tA3hZTVfA>
- <https://github.com/devopshint/java-app-with-maven>
- <https://www.youtube.com/watch?v=3S4FFwPqxRU&t=215s>

EXPERIMENT 3

Docker

Date of Experiment: 26/09/2022

Installation:

prerequisite:

```
jasmit21@jasmit21:~$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20211016).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.6).
software-properties-common is already the newest version (0.99.22.3).
software-properties-common set to manually installed.
The following packages were automatically installed and are no longer required:
chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver intel-media-va-driver
libaacs0 libaom3 libass9 libavcodec58 libavformat58 libavutil56 libbdplus0 libblas3
libbluray2 libbs2b0 libchromaprint1 libcodec2-1.0 libdav1d5 libflashrom1 libflite1
libfdi1-2 libgme0 libgsm1 libgstreamer-plugins-bad1.0-0 libigdmm12 liblilv-0-0
libmfx1 libmysofa1 libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4
librubberband2 libserd-0-0 libshine3 libsnapy1v5 libsord-0-0 libsratom-0-0
libsrt1.4-gnutls libssh-gcrypt-4 libswresample3 libwscale5 libudfread0 libva-drm2
libva-wayland2 libva-x11-2 libva2 libvdpau1 libvidstab1.1 libx265-199 libxvidcore4
libzimg2 libzmq5 libzvbi-common libzvbi0 mesa-va-drivers mesa-vdpau-drivers
pocketsphinx-en-us va-driver-all vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,506 B of archives.
After this operation, 169 kB of additional disk space will be used.
```

Then add the GPG key for the official Docker repository to your system:

```
jasmit21@jasmit21:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
jasmit21@jasmit21:~$ 
```

Add the Docker repository to APT sources:

```
curl -s https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
[judo] password for jasmit21:
jasmit21@jasmit21:~$ 
```

Update your existing list of packages again for the addition to be recognized:

```
[root@jasmit21 jasmit21]# $ sudo apt update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.9 kB]
Get:3 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [9,481 B]
Hit:4 https://dl.google.com/linux/chrome/deb stable InRelease
Ign:2 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:5 https://pkg.jenkins.io/debian-stable binary/ Release
Get:6 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:7 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Hit:10 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 283 kB in 1s (266 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
jasmit21@jasmit21:~$
```

Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:

You'll see output like this,

```
jasmit21@jasmit21:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.21~3-0~ubuntu-jammy
  Version table:
    5:20.10.21~3-0~ubuntu-jammy 500
      500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
    5:20.10.20~3-0~ubuntu-jammy 500
      500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
    5:20.10.19~3-0~ubuntu-jammy 500
      500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
    5:20.10.18~3-0~ubuntu-jammy 500
      500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
    5:20.10.17~3-0~ubuntu-jammy 500
      500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
    5:20.10.16~3-0~ubuntu-jammy 500
      500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
    5:20.10.15~3-0~ubuntu-jammy 500
      500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
    5:20.10.14~3-0~ubuntu-jammy 500
      500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
    5:20.10.13~3-0~ubuntu-jammy 500
      500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
jasmit21@jasmit21:~$ █
```

Finally, install Docker:

```
jasmit21@jasmit21:~$ sudo apt install docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver intel-media-va-driver
  libaaacs0 libaoam3 libass9 libavcodec58 libavformat58 libavutil56 libbdplus0 libblas3
  libbluray2 libbs2b0 libchromaprint1 libcodec2-1.0 libdav1d5 libflashrom1 libflite1
  libftdi1-2 libgme0 libgsm1 libgstreamer-plugins-bad1.0-0 libigdgmm12 liblilv-0-0
  libmfx1 libmysofa1 libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4
  librubberband2 libserd-0-0 libshine3 libsnapy1v5 libsord-0-0 libsratom-0-0
  libsrt1.4-gnutls libssh-gcrypt-4 libswresample3 libswscale5 libudfread0 libva-drm2
  libva-wayland2 libva-x11-2 libva2 libvdpa1 libvidstab1.1 libx265-199 libxvidcore4
  libzimg2 libzmq5 libzvbi-common libzvbi0 mesa-va-drivers mesa-vdpau-drivers
  pocketsphinx-en-us va-driver-all vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  containerd.io docker-ce-cli docker-ce-rootless-extras docker-scan-plugin libslirp0 pigz
  slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras docker-scan-plugin
  libslirp0 pigz slirp4netns
```

Check that it's running:

```
jasmit21@jasmit21:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
    Active: active (running) since Fri 2022-10-28 22:08:23 IST; 12s ago
  TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 13660 (dockerd)
     Tasks: 10
    Memory: 26.2M
      CPU: 152ms
     CGroup: /system.slice/docker.service
             └─13660 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Oct 28 22:08:23 jasmit21 dockerd[13660]: time="2022-10-28T22:08:23.194414735+05:30" level=>
Oct 28 22:08:23 jasmit21 dockerd[13660]: time="2022-10-28T22:08:23.194423988+05:30" level=>
Oct 28 22:08:23 jasmit21 dockerd[13660]: time="2022-10-28T22:08:23.194428951+05:30" level=>
Oct 28 22:08:23 jasmit21 dockerd[13660]: time="2022-10-28T22:08:23.398635555+05:30" level=>
Oct 28 22:08:23 jasmit21 dockerd[13660]: time="2022-10-28T22:08:23.564534086+05:30" level=>
Oct 28 22:08:23 jasmit21 dockerd[13660]: time="2022-10-28T22:08:23.631962673+05:30" level=>
Oct 28 22:08:23 jasmit21 dockerd[13660]: time="2022-10-28T22:08:23.727007526+05:30" level=>
Oct 28 22:08:23 jasmit21 dockerd[13660]: time="2022-10-28T22:08:23.727118867+05:30" level=>
Oct 28 22:08:23 jasmit21 systemd[1]: Started Docker Application Container Engine.
Oct 28 22:08:23 jasmit21 dockerd[13660]: time="2022-10-28T22:08:23.748951826+05:30" level=>
lines 1-22/22 (END)
```

Pull an image from docker

```
jasmit21@jasmit21:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Already exists
Digest: sha256:e18f0a777aefabe047a671ab3ec3eed05414477c951ab1a6f352a06974245fe7
Status: Downloaded newer image for hello-world:latest

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/
jasmit21@jasmit21:~$
```

The actual Hello World command of docker is

```
$ docker run docker/whalesay cowsay boo
```

```
jasmit21@jasmit21:~$ sudo docker run docker/whalesay
Unable to find image 'docker/whalesay:latest' locally
latest: Pulling from docker/whalesay
Image docker.io/docker/whalesay:latest uses outdated schema1 manifest format. Please upgrade to a schema2 image for better future compatibility. More information at https://docs.docker.com/registry/spec/deprecated-schema-v1/
e190868d63f8: Pull complete
909cd34c6fd7: Pull complete
0b9bfabab7c1: Pull complete
a3ed95caeb02: Pull complete
00bf65475aba: Pull complete
c57b6bcc83e3: Pull complete
8978f6879e2f: Pull complete
8eed3712d2cf: Pull complete
Digest: sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest
```

The default image of docker appears with the message boo.

```
jasmit21@jasmit21:~$ sudo docker run docker/whalesay cowsay Jasmit
< Jasmit >
-----
      \
      \
      \
          ##      .
          ##  ##  ##      ==
          ##  ##  ##  ##      ===
          /"*****" \_/_\ ====
~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~~~ /  ===- ~~~
      \____ o      _/
      \_\_\_ \_\_\_ /
```

Check if the docker image has been pulled and is present in your system using the following command:

```
jasmit21@jasmit21:~$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-world    latest    feb5d9fea6a5  13 months ago  13.3kB
docker/whalesay  latest    6b362a9f73eb  7 years ago  247MB
```

To display all the containers pulled, use the following command:

```
$ sudo docker ps -a
```

```
jasmit21@jasmit21:~$ sudo docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
b3fbfbcc26c88 hello-world "/hello" 3 minutes ago Exited (0) 3 minutes ago dazzling_allen
a7991b5c984c docker/whalesay "cowsay Jasmit" 9 minutes ago Exited (0) 9 minutes ago peaceful_banach
d4876933d81a docker/whalesay "/bin/bash" 10 minutes ago Exited (0) 10 minutes ago admiring_lumiere
b2f0f047491f hello-world "/hello" 3 days ago Exited (0) 3 days ago objective_mcnulty
jasmit21@jasmit21:~$
```

To check for containers in a running state, use the following command:

```
$ sudo docker ps
```

```
jasmit21@jasmit21:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND       CREATED          STATUS          PORTS     NAMES
jasmit21@jasmit21:~$
```

You've just successfully installed Docker on Ubuntu!

Part B:

1. docker search

Use the command `docker search` to search for public images on the Docker hub. It will return information about the image name, description, stars, official and automated.

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
mysql	MySQL is a widely used, open-source relation...	13418	[OK]	
mariadb	MariaDB Server is a high performing open sou...	5117	[OK]	
phpmyadmin	phpMyAdmin - A web interface for MySQL and M...	672	[OK]	
percona	Percona Server is a fork of the MySQL relati...	592	[OK]	
bitnami/mysql	Bitnami MySQL Docker Image	78		[OK]
databack/mysql-backup	Back up mysql databases to... anywhere!	74		
linuxserver/mysql-workbench		45		
ubuntu/mysql	MySQL open source fast, stable, multi-thread...	38		
linuxserver/mysql	A MySql container, brought to you by LinuxSe...	37		
circlegci/mysql	MySQL is a widely used, open-source relation...	28		
google/mysql	MySQL server for Google Compute Engine	21		[OK]
rapidfort/mysql	RapidFort optimized, hardened image for MySQL	13		
bitnami/mysqld-exporter		4		
ibmcom/mysql-s390x	Docker image for mysql-s390x	2		
newrelic/mysql-plugin	New Relic Plugin for monitoring MySQL databa...	1		[OK]
vitess/mysqlctld	vitess/mysqlctld	1		[OK]
hashicorp/mysql-portworx-demo		0		
docksal/mysql	MySQL service images for Docksal - https://d...	0		
mirantis/mysql		0		
rapidfort/mysql8-ib	RapidFort optimized, hardened image for MySQL...	0		
cimg/mysql		0		
drud/mysql		0		
silintl/mysql-backup-restore	Simple docker image to perform mysql backups...	0		[OK]
corpusops/mysql	https://github.com/corpusops/docker-images/	0		
drud/mysql-local-57	ddev mysql local container	0		

2. docker pull

Now that we know the name of the image, we can pull that from the Docker hub using the command `docker pull`. Here, we are setting the platform option as well.

```
docker pull --platform linux/x86_64 mysql
```

```
jasmit21@jasmit21:~$ docker pull --platform linux/x86_64 mysql
Using default tag: latest
latest: Pulling from library/mysql
d67a603b911a: Pull complete
0cf69c8f1492: Pull complete
a5ee239a0d3a: Pull complete
0f166cb3e327: Pull complete
882d294bf188: Pull complete
2649fc7eb806: Pull complete
bddb3394e2e3: Downloading [=====] 7.019MB/56.04MB
93c83d9a2206: Download complete
bddb3394e2e3: Pull complete
93c83d9a2206: Pull complete
99d7f45787c0: Pull complete
234663a2e3ee: Pull complete
74531487bb7b: Pull complete
Digest: sha256:d4055451e7f42869e64089a60d1abc9e66eccde2910629f0dd666b53a5f230d8
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
jasmit21@jasmit21:~$
```

3. docker images

By this time, we should have some images in our local machine, and to confirm, let's run the following command to list all the local images.

```
docker images
```

```
jasmit21@jasmit21:~$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
mysql           latest   c2c2eba5ae85  4 days ago   535MB
hello-world     latest   feb5d9fea6a5  13 months ago 13.3kB
docker/whalesay latest   6b362a9f73eb  7 years ago  247MB
jasmit21@jasmit21:~$
```

4. docker run

Alright, now that we have some images, we can try to create a container. Here we used the --env option to set a mandatory environment variable and --detach option to run the container in the background.

```
docker run --env MYSQL_ROOT_PASSWORD=my-secret-pw --detach mysql
```

```
jasmit21@jasmit21:~$ sudo docker run --env MYSQL_ROOT_PASSWORD=jasmit --detach mysql
8dd6f84bdff684dec87db77b4687fa96f73e4354050fd6e6f10d7f2152084dc7
jasmit21@jasmit21:~$
```

5. docker ps

We can list all the running containers by using the following command.

```
docker ps
```

```
jasmit21@jasmit21:~$ docker ps
CONTAINER ID  IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
8dd6f84bdff6  mysql      "docker-entrypoint.s..."  About a minute ago  Up About a minute  3306/tcp, 33060/tcp  sweet_rosalind
jasmit21@jasmit21:~$
```

To list all containers user docker ps --all

```
jasmit21@jasmit21:~$ docker ps --all
CONTAINER ID  IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
8dd6f84bdff6  mysql      "docker-entrypoint.s..."  2 minutes ago  Up 2 minutes  3306/tcp, 33060/tcp  sweet_rosalind
b3fbfb26c88  hello-world  "/hello"    18 minutes ago  Exited (0) 18 minutes ago  dazzling_allen
a7991b5c984c  docker/whalesay  "cowsay Jasmit"  24 minutes ago  Exited (0) 24 minutes ago  peaceful_banach
d4876933d81a  docker/whalesay  "/bin/bash"   25 minutes ago  Exited (0) 25 minutes ago  admiring_lumiere
b2f0f047491f  hello-world  "/hello"    3 days ago    Exited (0) 3 days ago   objective_mcnelly
jasmit21@jasmit21:~$
```

6. docker stop

To stop a container, use the docker stop command with either the container id or container name. We may stop a container if we want to change our docker run command.

```
docker stop 8dd6f84bdff6
```

```
jasmit21@jasmit21:~$ docker stop 8dd6f84bdff6
8dd6f84bdff6
jasmit21@jasmit21:~$
```

7. docker restart

Let's restart our stops by using the following command. We may want to use this after we reboot our machine.

```
docker restart 8dd6f84bdff6
```

```
jasmit21@jasmit21:~$ docker restart 8dd6f84bdff6
8dd6f84bdff6
jasmit21@jasmit21:~$
```

8. docker rename

Now, let's change the container name from `compassionate_fermi` to `test_db`. We may want to change the name to keep track of our containers more easily.

```
docker rename sweet_rosalind db_54
```

```
jasmit21@jasmit21:~$ docker rename sweet_rosalind db_54
jasmit21@jasmit21:~$ docker ps --all
CONTAINER ID        IMAGE       COMMAND       CREATED      STATUS      PORTS          NAMES
8dd6f84bdff6        mysql       "docker-entrypoint.s..."   47 minutes ago   Up 18 minutes   3306/tcp, 33060/tcp   db_54
b3fbfb26c88        hello-world    "/hello"      About an hour ago   Exited (0) About an hour ago   dazzling_allen
a7991b5c984c        docker/whalesay  "cowsay Jasmit"  About an hour ago   Exited (0) About an hour ago   peaceful_banach
d4876933d81a        docker/whalesay  "/bin/bash"    About an hour ago   Exited (0) About an hour ago   admiring_lumiere
b2f0f047491f        hello-world    "/hello"      3 days ago     Exited (0) 3 days ago     objective_mcnulty
jasmit21@jasmit21:~$
```

9. docker exec

Access the running container test_db by running the following command. It's helpful if we want to access the MySQL command line and execute MySQL queries.

```
docker exec -it db_54 bash  
mysql -uroot -pmy-secret-pw  
SHOW DATABASES;
```

```
bash-4.4# mysql -uroot -pjasmitt  
mysql: [Warning] Using a password on the command line interface can be insecure.  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 8  
Server version: 8.0.31 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2022, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> SHOW DATABASES  
    -> ;  
+-----+  
| Database      |  
+-----+  
| information_schema |  
| mysql          |  
| performance_schema |  
| sys            |  
+-----+  
4 rows in set (0.01 sec)  
  
mysql>
```

10. docker logs

```
jasmit21@jasmit21:~$ docker logs db_54
2022-11-01 08:28:01+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.31-1.el8 started.
2022-11-01 08:28:01+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2022-11-01 08:28:01+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.31-1.el8 started.
2022-11-01 08:28:01+00:00 [Note] [Entrypoint]: Initializing database files
2022-11-01T08:28:01.703548Z 0 [Warning] [MY-01068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
2022-11-01T08:28:01.703616Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.31) initializing of server in progress as process 79
2022-11-01T08:28:01.710599Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2022-11-01T08:28:02.447825Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2022-11-01T08:28:03.813672Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
2022-11-01 08:28:07+00:00 [Note] [Entrypoint]: Database files initialized
2022-11-01 08:28:07+00:00 [Note] [Entrypoint]: Starting temporary server
2022-11-01T08:28:07.898842Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
2022-11-01T08:28:07.899959Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.31) starting as process 130
2022-11-01T08:28:07.912412Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2022-11-01T08:28:08.219219Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2022-11-01T08:28:08.219246Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
2022-11-01T08:28:08.222638Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
2022-11-01T08:28:08.236990Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.31' socket: '/var/run/mysqld/mysqld.sock' port: 0 MySQL Community Server - GPL.
2022-11-01 08:28:08+00:00 [Note] [Entrypoint]: X Plugin ready for connections. Socket: /var/run/mysqld/mysqlx.sock
2022-11-01 08:28:08+00:00 [Note] [Entrypoint]: Temporary server started.
'/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/tzdata.zl' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.

2022-11-01 08:28:09+00:00 [Note] [Entrypoint]: Stopping temporary server
2022-11-01T08:28:09.719353Z 10 [System] [MY-013172] [Server] Received SHUTDOWN from user root. Shutting down mysqld (Version: 8.0.31).
2022-11-01T08:28:11.251740Z 0 [System] [MY-0100101] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.31) MySQL Community Server - GPL.
```

11. docker rm

To remove a container, we can use the following command.

```
docker rm db_54
```

```
jasmit21@jasmit21:~$ docker rm -f db_54
db_54
jasmit21@jasmit21:~$
```

12. docker rmi

To free some disk space, we can use the docker rmi command with the image id to remove an image.

```
docker rmi eb0e825dc3cf
```

```
jasmit21@jasmit21:~$ docker rmi c2c2eba5ae85
Untagged: mysql:latest
Untagged: mysql@sha256:d4055451e7f42869e64089a60d1abc9e66eccde2910629f0dd666b53a5f230d8
Deleted: sha256:c2c2eba5ae857a8ab9bffd11c5f15ed693dc65ac035948696f370f2895ae3062
Deleted: sha256:210b4dce38af03de3f57240d06ca8ca60b426a30b84ea74a35a61bd42caff054
Deleted: sha256:ba25a3896ed49216e74d63280aa2797490babecb29779ef128994a3ed84241ce
Deleted: sha256:d3860b757f15dd46d3e1dd6098611a1478a57630ca92f5a9fa8f7f0dc08559f1
Deleted: sha256:f1a04d895a8bd362e477e5654cd24c5f4f3cefdf99a1dc5073408d21ea4ac60
Deleted: sha256:b648b662b97c6ddb59fac71ce7860297060624ffff5102f021a69f38c09df58fd
Deleted: sha256:c3a7ba2a7418cdfd9e3d58af2488f73921499254c0761644caf5dbb057951323
Deleted: sha256:86349bd8052cbee20aab4ab699c5b553e57de91a7038850c881bc57be6c78862
Deleted: sha256:25698a118589bb2e601dcbe9f6d6ab72678a6b033ecc90ae83e767138c892dce
Deleted: sha256:d9da521068fe6f1de86d0894c7234069abdae3f9db44e923c66614bb3e2e999b
Deleted: sha256:1c63c8b11ddd9ac84858ee47dfab5464bd2581fb48fefef75b3c16ba32176e75
Deleted: sha256:bb4173a55532f72ee01e6aa78ee0208d520b2825596ca90ac73a5be99b38012f
jasmit21@jasmit21:~$
```

Part C: Docker Container Commands:

Containers

Use docker container my_command

create — Create a container from an image.

start — Start an existing container.

run — Create a new container and start it.

ls — List running containers.

inspect — See lots of info about a container.

logs — Print logs.

stop — Gracefully stop running container.

`kill` — Stop main process in container abruptly.

`rm` — Delete a stopped container.

Images

Use `docker image my_command`

`build` — Build an image.

`push` — Push an image to a remote registry.

`ls` — List images.

`history` — See intermediate image info.

`inspect` — See lots of info about an image, including the layers.

`rm` — Delete an image.

Misc

`docker version` — List info about your Docker Client and Server versions.

`docker login` — Log in to a Docker registry.

`docker system prune` — Delete all unused containers, unused networks, and dangling images.

Containers

Container Beginnings

The terms create, start, and run all have similar semantics in everyday life. But each is a separate Docker command that creates and/or starts a container. Let's look at creating a container first.

docker container create hello-world — Create a container from an image.

```
jasmit21@jasmit21:~$ docker container create hello-world  
b4e15a9b7ee49399dbe49a9017da2d745d881876eb9d975f68bf2fdcd7951e25  
jasmit21@jasmit21:~$
```

I'll shorten my_repo/my_image:my_tag to my_image for the rest of the article.

There are [a lot of possible flags](#) you could pass to create.

docker container create -a STDIN my_image - docker container create -a STDIN hello-world

```
jasmit21@jasmit21:~$ docker container create -a STDIN hello-world  
41b0722ba3b227ba75daf0dd88c2eadacefff242d98220b0aaf71c034afd8aab  
jasmit21@jasmit21:~$
```

-a is short for --attach. Attach the container to STDIN, STDOUT or STDERR.

Now that we've created a container let's start it.

docker container start hello-world — Start an existing container.

```
jasmit21@jasmit21:~$ docker container start hello-world  
Error response from daemon: No such container: hello-world  
Error: failed to start containers: hello-world  
jasmit21@jasmit21:~$
```

Note that the container can be referred to by either the container's ID or the container's name.

```
docker container start distracted_heisenberg
```

```
jasmit21@jasmit21:~$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
41b072ba3b2 hello-world "/hello" About a minute ago Created
b4e15a9b7ee4 hello-world "/hello" 2 minutes ago Created
b3fbfb2c26c88 hello-world "/hello" 2 hours ago Exited (0) 2 hours ago
a7991b5c984c docker/whalesay "cowsay Jasmit" 2 hours ago Exited (0) 2 hours ago
d4876933d81a docker/whalesay "/bin/bash" 2 hours ago Exited (0) 2 hours ago
b2f0f047491f hello-world "/hello" 3 days ago Exited (0) 3 days ago
jasmit21@jasmit21:~$ docker container start distracted_heisenberg
distracted_heisenberg
jasmit21@jasmit21:~$
```

Now that you know how to create and start a container, let's turn to what's probably the most common Docker command. It combines both `create` and `start` into one command: `run`.

```
docker container run my_image — Create a new container and start it. It also has a lot of options. Let's look at a few.
```

```
jasmit21@jasmit21:~$ docker container 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/
jasmit21@jasmit21:~$
```

```
docker container run -i -t -p 1000:8000 --rm my_image
```

```
jasmit21@jasmit21:~$ docker container run -i -t -p 1000:8000 --rm 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/
```

```
jasmit21@jasmit21:~$
```

-i is short for --interactive. Keep STDIN open even if unattached.

-t is short for --tty. Allocates a pseudo [terminal](#) that connects your terminal with the container's STDIN and STDOUT.

You need to specify both -i and -t to then interact with the container through your terminal shell.

-p is short for --port. The port is the interface with the outside world. 1000:8000 maps the Docker port 8000 to port 1000 on your machine. If you had an app that output something to the browser you could then navigate your browser to localhost:1000 and see it.

--rm Automatically delete the container when it stops running.

Let's look at some more examples of run.

docker container run -it my_image my_command

```
jasmit21@jasmit21:~$ docker container run -it hello-world jasmit
docker: Error response from daemon: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: "jasmit": executable file not found in $PATH: unknown.
ERRO[0000] error waiting for container: context canceled
jasmit21@jasmit21:~$
```

```
jasmit21@jasmit21:~$ docker container ls -a -s
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES SIZE
21a209c026a3 hello-world "/jasmit" About a minute ago Created focused_meitner 0B (virtual 13.3kB)
73da64055d14 hello-world "/hello" 3 minutes ago Exited (0) 3 minutes ago jovial_kirch 0B (virtual 13.3kB)
41b0722ba3b2 hello-world "/hello" 7 minutes ago Created reverent_meninsky 0B (virtual 13.3kB)
b4e15a9b7ee4 hello-world "/hello" 8 minutes ago Exited (0) 5 minutes ago distracted_heisenberg 0B (virtual 13.3kB)
b3fbfb26c88 hello-world "/hello" 2 hours ago Exited (0) 2 hours ago dazzling_allen 0B (virtual 13.3kB)
a7991b5c984c docker/whalesay "cowsay Jasmit" 2 hours ago Exited (0) 2 hours ago peaceful_banach 0B (virtual 247MB)
d4876933d81a docker/whalesay "/bin/bash" 2 hours ago Exited (0) 2 hours ago admiring_lumiere 0B (virtual 247MB)
b2f0f047491f hello-world "/hello" 3 days ago Exited (0) 3 days ago objective_mcnuity 0B (virtual 13.3kB)
jasmit21@jasmit21:~$
```

sh is a command you could specify at run time. sh will start a shell session inside your container that you can interact with through your terminal. sh is preferable to bash for Alpine images because Alpine images don't come with bash installed. Type exit to end the interactive shell session.

Notice that we combined -i and -t into -it.

docker container run -d my_image

```
jasmit21@jasmit21:~$ docker container run -d hello-world
1a998bed21ef4d2f34f936be1c619c6dea591cfa061a4b139a3990ff872de155
jasmit21@jasmit21:~$
```

-d is short for --detach. Run the container in the background. Allows you to use the terminal for other commands while your container runs.

Checking Container Status

If you have running Docker containers and want to find out which one to interact with, then you need to list them.

`docker container ls` — List running containers. Also provides useful information about the containers.

`docker container ls -a -s`

```
jasmit21@jasmit21:~$ docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
jasmit21@jasmit21:~$ docker container ls -a -s
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES SIZE
1a998bed21ef hello-world "/hello" About a minute ago Exited (0) About a minute ago 0B (virtual 13
.jkB)
21a209c026a3 hello-world "jasmit" 3 minutes ago Created focused_meitner 0B (virtual 13
.jkB)
73da64055d14 hello-world "/hello" 5 minutes ago Exited (0) 5 minutes ago jovial_kirch 0B (virtual 13
.jkB)
41b0722ba3b2 hello-world "/hello" 9 minutes ago Created reverent_meninsky 0B (virtual 13
.jkB)
b4e15a9b7ee4 hello-world "/hello" 10 minutes ago Exited (0) 7 minutes ago distracted_heisenberg 0B (virtual 13
.jkB)
b3fbfbcc26c88 hello-world "/hello" 2 hours ago Exited (0) 2 hours ago dazzling_allen 0B (virtual 13
.jkB)
a7991b5c984c docker/whalesay "cowsay Jasmit" 2 hours ago Exited (0) 2 hours ago peaceful_banach 0B (virtual 24
7MB)
d4876933d81a docker/whalesay "/bin/bash" 2 hours ago Exited (0) 2 hours ago admiring_lumiere 0B (virtual 24
7MB)
b2f0f047491f hello-world "/hello" 3 days ago Exited (0) 3 days ago objective_mcnullty 0B (virtual 13
.jkB)
jasmit21@jasmit21:~$
```

`-a` is short for `-all`. List all containers (not just running ones).

`-s` is short for `--size`. List the size for each container.

`docker container inspect nifty_brattain` — See lots of info about a container.

```
jasmit21@jasmit21:~$ docker container inspect nifty_brattain
[{"Id": "1a998bed21ef4d2f34f936be1c619c6dea591cfa061a4b139a3990ff872de155",
 "Created": "2022-11-01T09:54:38.650222129Z",
 "Path": "/hello",
 "Args": [],
 "State": {
     "Status": "exited",
     "Running": false,
     "Paused": false,
     "Restarting": false,
     "OOMKilled": false,
     "Dead": false,
     "Pid": 0,
     "ExitCode": 0,
     "Error": "",
     "StartedAt": "2022-11-01T09:54:38.650222129Z",
     "FinishedAt": "2022-11-01T09:54:38.649918926Z"
 },
 "Image": "sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412",
 "ResolvConfPath": "/var/lib/docker/containers/1a998bed21ef4d2f34f936be1c619c6dea591cfa061a4b139a3990ff872de155/resolv.conf",
 "HostnamePath": "/var/lib/docker/containers/1a998bed21ef4d2f34f936be1c619c6dea591cfa061a4b139a3990ff872de155/hostname",
 "HostsPath": "/var/lib/docker/containers/1a998bed21ef4d2f34f936be1c619c6dea591cfa061a4b139a3990ff872de155/hosts",
 "LogPath": "/var/lib/docker/containers/1a998bed21ef4d2f34f936be1c619c6dea591cfa061a4b139a3990ff872de155/1a998bed21ef4d2f34f936be1c619c6dea591cfa061a4b139a3990ff872de155-1a998bed21ef4d2f34f936be1c619c6dea591cfa061a4b139a3990ff872de155.json.log",
 "Name": "nifty_brattain",
 "RestartCount": 0,
 "Driver": "overlay2",
 "Platform": "linux",
 "MountLabel": "",
 "ProcessLabel": "",
 "AppArmorProfile": "docker-default",
 "ExecIDs": null,
 "HostConfig": {
     "Binds": null,
     "ContainerIDFile": "",
     "LogConfig": {
         "Type": "json-file"
     }
 }
```

```
docker container logs nifty_brattain — Print a container's logs.
```

```
jasmit21@jasmit21:~$ docker container logs nifty_brattain

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/

jasmit21@jasmit21:~$
```

Container Endings

Sometimes you need to stop a running container.

```
docker container stop nifty_brattain — Stop one or more running containers gracefully.
Gives a default of 10 seconds before container shutdown to finish any processes.
```

```
jasmit21@jasmit21:~$ docker container stop nifty_brattain
nifty_brattain
jasmit21@jasmit21:~$ █
```

```
jasmit21@jasmit21:~$ docker container stop nifty_brattain
nifty_brattain
jasmit21@jasmit21:~$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1a998bed21ef hello-world "/hello" 5 minutes ago Exited (0) 5 minutes ago
21a209c026a3 hello-world "jasmit" 7 minutes ago Created
73da64055d14 hello-world "/hello" 10 minutes ago Exited (0) 10 minutes ago
41b0722ba3b2 hello-world "/hello" 14 minutes ago Created
b4e15a9b7ee4 hello-world "/hello" 15 minutes ago Exited (0) 12 minutes ago
b3fbfb26c88 hello-world "/hello" 2 hours ago Exited (0) 2 hours ago
a7991b5c984c docker/whalesay "cowsay Jasmit" 2 hours ago Exited (0) 2 hours ago
d4876933d81a docker/whalesay "/bin/bash" 2 hours ago Exited (0) 2 hours ago
b2f0f047491f hello-world "/hello" 3 days ago Exited (0) 3 days ago
jasmit21@jasmit21:~$
```

Or if you are impatient:

`docker container kill exciting_fermat` — Stop one or more running containers abruptly. It's like pulling the plug on the TV. Prefer `stop` in most situations.

`docker container kill $(docker ps -q)` — Kill all running containers.

Then you delete the container with:

`docker container rm nifty_brattain` — Delete one or more containers.

```
jasmit21@jasmit21:~$ docker container rm nifty_brattain
nifty_brattain
jasmit21@jasmit21:~$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
21a209c026a3 hello-world "jasmit" 9 minutes ago Created
73da64055d14 hello-world "/hello" 12 minutes ago Exited (0) 12 minutes ago
41b0722ba3b2 hello-world "/hello" 16 minutes ago Created
b4e15a9b7ee4 hello-world "/hello" 17 minutes ago Exited (0) 14 minutes ago
b3fbfb26c88 hello-world "/hello" 2 hours ago Exited (0) 2 hours ago
a7991b5c984c docker/whalesay "cowsay Jasmit" 2 hours ago Exited (0) 2 hours ago
d4876933d81a docker/whalesay "/bin/bash" 2 hours ago Exited (0) 2 hours ago
b2f0f047491f hello-world "/hello" 3 days ago Exited (0) 3 days ago
jasmit21@jasmit21:~$
```

`docker container rm $(docker ps -a -q)` — Delete all containers that are not running.

Those are the eight essential commands for Docker containers.

To recap, you first create a container. Then, you start the container. Or combine those steps with `docker run my_container`. Then, your app runs. Yippee!

Then, you stop a container with `docker stop my_container`. Eventually you delete the container with `docker rm my_container`.

Conclusion: Thus, the docker installation on ubuntu was done successfully and the basic commands of docker has been

Reference:

1. <https://www.simplilearn.com/tutorials/docker-tutorial/how-to-install-docker-on-ubuntu>
2. <https://towardsdatascience.com/12-essential-docker-commands-you-should-know-c2d5a7751bb5>
3. <https://docs.docker.com/engine/reference/commandline/container/>
4. <https://towardsdatascience.com/15-docker-commands-you-should-know-970ea5203421>

ASSIGNMENT 1 : DOCKER VOLUME

Docker volumes

How To Share Data Between the Docker Container and the Host

In general, Docker containers are ephemeral, running just as long as it takes for the command issued in the container to complete. By default, any data created inside the container is only available from within the container and only while the container is running.

Docker volumes can be used to share files between a host system and the Docker container. For example, let's say you wanted to use the official Docker Nginx image and keep a permanent copy of Nginx's log files to analyze later. By default, the nginx Docker image will log to the /var/log/nginx directory inside the Docker Nginx container. Normally it's not reachable from the host filesystem.

In this tutorial, we'll explore how to make data from inside the container accessible on the host machine.

Step 1 — Bind Mounting a Volume

The following command will create a directory called nginx logs in your current user's home directory and bind mount it to /var/log/nginx in the container:

```
$ docker run --name=nginx -d -v ~/nginxlogs:/var/log/nginx -p 5000:80 nginx
```

```
jasmit21@jasmit21: ~$ sudo su
[sudo] password for jasmit21:
root@jasmit21:/home/jasmit21# cd
root@jasmit21:~# cd /home/jasmit21
root@jasmit21:/home/jasmit21# docker run --name=nginx -d -v ~/nginxlogs:/var/log/nginx -p 5000:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
e9995326b091: Pull complete
71689475aec2: Pull complete
f88a23025338: Pull complete
0df440342e26: Pull complete
eef26ceb3309: Pull complete
8e3ed6a9e43a: Pull complete
Digest: sha256:943c25b4b66b332184d5ba6bb18234273551593016c0e0ae906bab111548239f
Status: Downloaded newer image for nginx:latest
4e45f8cce6b5b641c2cac1a48cc0ac0f4098892e1630cbfefc7fdadc0434e752
root@jasmit21:/home/jasmit21#
```

Let's take a moment to examine this command in detail:

- --name=nginx names the container so we can refer to it more easily.

- `-d` detaches the process and runs it in the background. Otherwise, we would just be watching an empty Nginx prompt and wouldn't be able to use this terminal until we killed Nginx.
- `-v ~/nginxlogs:/var/log/nginx` sets up a bind mount volume that links the `/var/log/nginx` directory from inside the Nginx container to the `~/nginxlogs` directory on the host machine. Docker uses a `:` to split the host's path from the container path, and the host path always comes first.
- `-p 5000:80` sets up a port forward. The Nginx container is listening on port 80 by default. This flag maps the container's port 80 to port 5000 on the host system.
- `nginx` specifies that the container should be built from the Nginx image, which issues the command [`nginx -g "daemon off"`](#) to start Nginx.

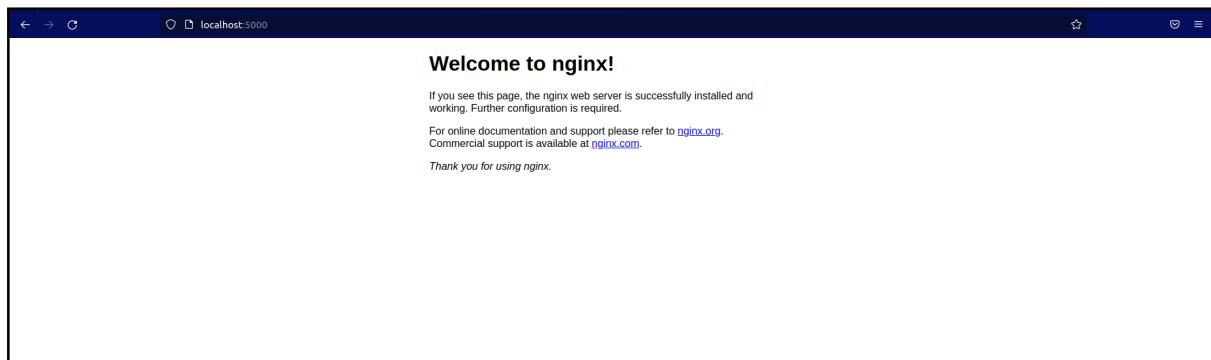
Note: The `-v` flag is very flexible . It can bindmount or name a volume with just a slight adjustment in syntax. If the first argument begins with a `/` or `~/`, you're creating a bindmount. Remove that, and you're naming the volume.

- `-v /path:/path/in/container` mounts the host directory, `/path` at the `/path/in` `/container`
- `-v path:/path/in/container` creates a volume named `path` with no relationship to the host.

Step 2 — Accessing Data on the Host

We now have a copy of Nginx running inside a Docker container on our machine, and our host machine's port 5000 maps directly to that copy of Nginx's port 80.

Load the address in a web browser, using the IP address or hostname of your server and the port number: `http://your_server_ip:5000`. You should see:



More interestingly, if we look in the `~/nginxlogs` directory on the host, we'll see the `access.log` created by the container's nginx which will show our request:

```
$cat ~/nginxlogs/access.log
```

```
root@jasmit21:/home/jasmit21# cat ~/nginxlogs/access.log
172.17.0.1 - - [01/Nov/2022:10:23:46 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:106.0) Gecko/20100101 Firefox/106.0
" "
root@jasmit21:/home/jasmit21#
```

This should display something like:

Output

```
203.0.113.0 - - [11/Jul/2018:00:5 9:11 +0000] "GET / HTTP/1.1" 200 612 "-"
```

```
"Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
```

```
(KHTML, like Gecko) Chrome/54 .0.2840.99 Safari/537.36" "-"
```

If you make any changes to the `~/nginxlogs` folder, you'll be able to see them from inside the Docker container in real time as well.

How To Share Data between Docker Containers

Step 1 — Creating an Independent Volume

Introduced in Docker's 1.9 release, the `docker volume create` command allows you to create a volume without relating it to any particular container. We'll use this command to add a volume named `DataVolume1`:

```
#docker volume create --name DataVolume1
```

The name is displayed, indicating that the command was successful:

```
root@jasmit21:/home/jasmit21# docker volume create --name DataVolume1
DataVolume1
root@jasmit21:/home/jasmit21#
```

Output

```
DataVolume1
```

To make use of the volume, we'll create a new container from the Ubuntu image, using the `--rm` flag to automatically delete it when we exit. We'll also use `-v` to mount the new volume. `-v` requires the name of the volume, a colon, then the absolute path to where the volume should appear inside the container. If the directories in the path don't exist as part of

the image, they'll be created when the command runs. If they do exist, the mounted volume will hide the existing content:

```
#docker run -ti --rm -v DataVolume1:/datavolume1 ubuntu
```

```
root@jasmit21:/home/jasmit21# docker run -ti --rm -v DataVolume1:/datavolume1 ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
301a8b74f71f: Pull complete
Digest: sha256:7cfe75438fc77c9d7235ae502bf229b15ca86647ac01c844b272b56326d56184
Status: Downloaded newer image for ubuntu:latest
root@fc27537e05b0:/#
```

While in the container, let's write some data to the volume:

```
$echo "Example1" > /datavolume1/Example1.txt
```

Because we used the --rm flag, our container will be automatically deleted when we exit. Our volume, however, will still be accessible.

```
$exit
```

```
root@fc27537e05b0:/# echo "Example1" > datavolume1/Example1.txt
root@fc27537e05b0:/# exit
exit
root@jasmit21:/home/jasmit21#
```

We can verify that the volume is present on our system with docker volume inspect:

```
#docker volume inspect DataVolume1
```

```
root@jasmit21:/home/jasmit21# docker volume inspect DataVolume1
[
  {
    "CreatedAt": "2022-11-01T15:59:18+05:30",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/DataVolume1/_data",
    "Name": "DataVolume1",
    "Options": {},
    "Scope": "local"
  }
]
root@jasmit21:/home/jasmit21#
```

Output

```
[{
  {
    "CreatedAt": "2018-07-11T16:57:54Z",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/DataVolume1/_data",
    "Name": "DataVolume1",
    "Options": {},
    "Scope": "local"
  }
]
```

Note: We can even look at the data on the host at the path listed as the Mountpoint. We should avoid altering it, however, as it can cause data corruption if applications or containers are unaware of changes.

Next, let's start a new container and attach DataVolume1:

```
#docker run --rm -ti -v DataVolume1:/datavolume1 ubuntu
```

Verify the contents:

```
$cat /datavolume1/Example1.txt
```

```
root@jasmit21:/home/jasmit21# docker run --rm -ti -v DataVolume1:/datavolume1 ubuntu
root@6976dbe88b1e:/# cat /datavolume1/Example1.txt
Example1
root@6976dbe88b1e:/#
```

Output

Example1

Exit the container:

```
$exit
```

```
root@jasmit21:/home/jasmit21# docker run --rm -ti -v DataVolume1:/datavolume1 ubuntu
root@6976dbe88b1e:/# cat /datavolume1/Example1.txt
Example1
root@6976dbe88b1e:/# exit
exit
root@jasmit21:/home/jasmit21#
```

In this example, we created a volume, attached it to a container, and verified its persistence.

Step 2 — Creating a Volume that Persists when the Container is Removed

In our next example, we'll create a volume at the same time as the container, delete the container, then attach the volume to a new container.

We'll use the docker run command to create a new container using the base Ubuntu image. -t will give us a terminal, and -i will allow us to interact with it. For clarity, we'll use --name to identify the container.

The -v flag will allow us to create a new volume, which we'll call DataVolume2. We'll use a colon to separate this name from the path where the volume should be mounted in the container. Finally, we will specify the base Ubuntu image and rely on the default command in the [Ubuntu base image's Docker file](#), bash, to drop us into a shell:

```
$ docker run -ti --name=Container2 -v DataVolume2:/datavolume2 ubuntu
```

Note: The -v flag is very flexible. It can bindmount or name a volume with just a slight adjustment in syntax. If the first argument begins with a / or ~/ you're creating a bindmount. Remove that, and you're naming the volume. For example:

- -v /path:/path/in/container mounts the host directory, /path at the /path/in/container
- -v path:/path/in/container creates a volume named path with no relationship to the host.

While in the container, we'll write some data to the volume:

```
$echo "Example2" > /datavolume2/Example2.txt
```

```
$cat /datavolume2/Example2.txt
```

Output

```
Example2
```

```
root@jasmit21:/home/jasmit21# docker run -ti --name=Container2 -v DataVolume2:/datavolume2 ubuntu
root@5273b1ec5179:/# echo "Example2" > /datavolume2/Example2.txt
root@5273b1ec5179:/# cat /datavolume2/Example2.txt
Example2
root@5273b1ec5179:/# exit
exit
root@jasmit21:/home/jasmit21#
```

Let's exit the container:

```
$exit
```

When we restart the container, the volume will mount automatically:

```
#docker start -ai Container2
```

Let's verify that the volume has indeed mounted and our data is still in place:

```
$cat /datavolume2/Example2.txt
```

Output

Example2

```
root@jasmit21:/home/jasmit21# docker start -ai Container2
root@5273b1ec5179:/# cat /datavolume2/Example2.txt
Example2
root@5273b1ec5179:/# exit
exit
root@jasmit21:/home/jasmit21#
```

Finally, let's exit and clean up:

```
$exit
```

Docker won't let us remove a volume if it's referenced by a container. Let's see what happens when we try:

```
#docker volume rm DataVolume2
```

The message tells us that the volume is still in use and supplies the long version of the container ID:

```
root@jasmit21:/home/jasmit21# docker volume rm DataVolume2
Error response from daemon: remove DataVolume2: volume is in use - [5273b1ec5179a8b91cbb015e0cc9226780b332baccefd736426ceec7196e6091]
root@jasmit21:/home/jasmit21#
```

Output

Error response from daemon: unable to remove volume: remove DataVolume2: volume is in use - [5273b1ec5179a8b91cbb015e0cc9226780b332baccefd736426ceec7196e6091]

We can use this ID to remove the container:

```
#docker rm 5273b1ec5179a8b91cbb015e0cc9226780b332baccefd736426ceec7196e6091
```

Output

```
root@jasmit21:/home/jasmit21# docker rm 5273b1ec5179a8b91cbb015e0cc9226780b332baccefd736426ceec7196e6091
5273b1ec5179a8b91cbb015e0cc9226780b332baccefd736426ceec7196e6091
root@jasmit21:/home/jasmit21#
```

Removing the container won't affect the volume. We can see it's still present on the system by listing the volumes with docker volume ls:

```
#docker volume ls
```

Output	DRIVER	VOLUME NAME
	local	DataVolume2

```
root@jasmit21:/home/jasmit21# docker rm 5273b1ec5179a8b91cbb015e0cc9226780b332baccefd736426ceec7196e6091
5273b1ec5179a8b91cbb015e0cc9226780b332baccefd736426ceec7196e6091
root@jasmit21:/home/jasmit21#
root@jasmit21:/home/jasmit21# docker volume ls
DRIVER      VOLUME NAME
local      7af2bb204bae47ed0ae86c99f120af21b6f30e56d7e8f4d80f366fe9fb92cda9
local      DataVolume1
local      DataVolume2
root@jasmit21:/home/jasmit21#
```

And we can use docker volume rm to remove it:

```
#docker volume rm DataVolume2
```

```
root@jasmit21:/home/jasmit21# docker volume rm DataVolume2
DataVolume2
root@jasmit21:/home/jasmit21#
```

In this example, we created an empty data volume at the same time that we created a container. In our next example, we'll explore what happens when we create a volume with a container directory that already contains data.

Step 3 — Creating a Volume from an Existing Directory with Data

Generally, creating a volume independently with docker volume create and creating one while creating a container are equivalent, with one exception. If we create a volume at the same time that we create a container and we provide the path to a directory that contains data in the base image, that data will be copied into the volume.

As an example, we'll create a container and add the data volume at /var, a directory which contains data in the base image:

```
#docker run -ti --rm -v DataVolume3:/var ubuntu
```

```
root@jasmit21:/home/jasmit21# docker run -ti --rm -v DataVolume3:/var ubuntu
root@7b322bf52cdb:# exit
exit
root@jasmit21:/home/jasmit21#
```

All the content from the base image's /var directory is copied into the volume, and we can mount that volume in a new container.

Exit the current container:

```
$exit
```

This time, rather than relying on the base image's default bash command, we'll issue our own ls command, which will show the contents of the volume without entering the shell:

```
#docker run --rm -v DataVolume3:/datavolume3 ubuntu ls datavolume3
```

The directory datavolume3 now has a copy of the contents of the base image's /var directory:

Output
backups
cache
lib
local
lock
log
mail
opt
run
spool
tmp

```
root@jasmit21:/home/jasmit21# docker run --rm -v DataVolume3:/datavolume3 ubuntu ls datavolume3
backups
cache
lib
local
lock
log
mail
opt
run
spool
tmp
root@jasmit21:/home/jasmit21#
```

It's unlikely that we would want to mount /var/ in this way, but this can be helpful if we've crafted our own image and want an easy way to preserve data. In our next example, we'll demonstrate how a volume can be shared between multiple containers.

Step 4 — Sharing Data Between Multiple Docker Containers

So far, we've attached a volume to one container at a time. Often, we'll want multiple containers to attach to the same data volume. This is relatively straightforward to accomplish, but there's one critical caveat: at this time, Docker doesn't handle file locking. If you need multiple containers writing to the volume, the applications running in those containers must be designed to write to shared data stores in order to prevent data corruption.

Create Container4 and DataVolume4

Use docker run to create a new container named Container4 with a data volume attached:

```
#docker run -ti --name=Container4 -v DataVolume4:/datavolume4 ubuntu
```

Next we'll create a file and add some text:

```
$echo "This file is shared between containers" > /datavolume4/Example4.txt
```

Then, we'll exit the container:

```
$exit
```

```
root@jasmit21:/home/jasmit21# docker run -ti --name=Container4 -v DataVolume4:/datavolume4 ubuntu
root@d605be7b9d07:/# echo "This file is shared between containers" > /datavolume4/Example4.txt
root@d605be7b9d07:/# exit
root@jasmit21:/home/jasmit21#
```

This returns us to the host command prompt, where we'll make a new container that mounts the data volume from Container4.

Create Container5 and Mount Volumes from Container4

We're going to create Container5, and mount the volumes from

```
Container4: #docker run -ti --name=Container5 --volumes-from
```

Container4 ubuntu Let's check the data persistence:

```
$cat /datavolume4/Example4.txt
```

Output

```
root@jasmit21:/home/jasmit21# docker run -ti --name=Container5 --volumes-from Container4 ubuntu
root@e69ab967d662:/# cat /datavolume4/Example4.txt
This file is shared between containers
root@e69ab967d662:/#
```

This file is shared between containers

Now let's append some text from Container5:

```
$echo "Both containers can write to DataVolume4" >> /datavolume4/Example4.txt
```

```
...This file is shared between containers...
root@e69ab967d662:/# echo "Both containers can write to DataVolume4" >> /datavolume4/Example4.txt
root@e69ab967d662:/# exit
exit
root@jasmit21:/home/jasmit21#
```

Finally, we'll exit the container:

```
$exit
```

Next, we'll check that our data is still present to Container4.

View Changes Made in Container5

Let's check for the changes that were written to the data volume by Container5 by restarting Container4:

```
#docker start -ai Container4
```

Check for the changes:

```
$cat /datavolume4/Example4.txt
```

Output

This file is shared between containers
Both containers can write to DataVolume4

```
root@d605be7b9d07:/# cat /datavolume4/Example4.txt
This file is shared between containers
Both containers can write to DataVolume4
root@d605be7b9d07:/# exit
exit
root@jasmit21:/home/jasmit21#
```

Now that we've verified that both containers were able to read and write from the data volume, we'll exit the container:

```
$exit
```

Again, Docker doesn't handle any file locking, so applications must account for the file locking themselves. It is possible to mount a Docker volume as read-only to ensure that data corruption won't happen by accident when a container requires read-only access by adding :ro. Let's look at how this works.

Start Container 6 and Mount the Volume Read-Only

Once a volume has been mounted in a container, rather than unmounting it like we would with a typical Linux file system, we can instead create a new container mounted the way we want and,

if needed, remove the previous container. To make the volume read-only, we append :ro to the end of the container name:

```
#docker run -ti --name=Container6 --volumes-from Container4:ro ubuntu
```

We'll check the read-only status by trying to remove our example file:

```
$rm /datavolume4/Example4.txt
```

Output

```
rm: cannot remove '/datavolume4/Example4.txt': Read-only file system
root@jasmit21:/home/jasmit21# docker run -ti --name=Container6 --volumes-from Container4:ro ubuntu
root@07f674a6343e:/# rm /datavolume4/Example4.txt
rm: cannot remove '/datavolume4/Example4.txt': Read-only file system
root@07f674a6343e:/# exit
exit
root@jasmit21:/home/jasmit21#
```

Finally, we'll exit the container and clean up our test containers and volumes:

```
$exit
```

Now that we're done, let's clean up our containers and volume:

```
#docker rm Container4 Container5 Container6
```

```
#docker volume rm DataVolume4
```

In this example, we've shown how to share data between two containers using a data volume and how to mount a data volume as read-only.

```
root@jasmit21:/home/jasmit21# docker rm Container4 Container5 Container6
Container4
Container5
Container6
root@jasmit21:/home/jasmit21# docker rm DataVolume4
Error: No such container: DataVolume4
root@jasmit21:/home/jasmit21# docker volume rm DataVolume4
DataVolume4
root@jasmit21:/home/jasmit21#
```

Delete all volumes at once

Using docker rm command, we can remove one volume at a time. If we have multiple volumes and want to delete all volumes then we have to use prune command.

Let us create a few volumes:

```
root@jasmit21:/home/jasmit21# docker volume create volume1
volume1
root@jasmit21:/home/jasmit21# docker volume create volume2
volume2
root@jasmit21:/home/jasmit21# docker volume create volume3
volume3
root@jasmit21:/home/jasmit21# docker volume ls
DRIVER      VOLUME NAME
local      7af2bb204bae47ed0ae86c99f120af21b6f30e56d7e8f4d80f366fe9fb92cda9
local      DataVolume1
local      DataVolume3
local      volume1
local      volume2
local      volume3
```

Now delete all docker volumes at once using command:

```
# docker volume prune
```

```
root@jasmit21:/home/jasmit21# docker volume prune
WARNING! This will remove all local volumes not used by at least one container
.
Are you sure you want to continue? [y/N] y
Deleted Volumes:
DataVolume1
DataVolume3
volume1
volume2
volume3
7af2bb204bae47ed0ae86c99f120af21b6f30e56d7e8f4d80f366fe9fb92cda9

Total reclaimed space: 212.7MB
root@jasmit21:/home/jasmit21# docker volume ls
DRIVER      VOLUME NAME
root@jasmit21:/home/jasmit21# 
```

Conclusion

In this tutorial, we created a data volume which allowed data to persist through the deletion of a container. We shared data volumes between containers, with the caveat that applications will need to be designed to handle file locking to prevent data corruption. Finally, we showed how to mount a shared volume in read-only mode. If you're interested in learning about sharing data between containers and the host system,

REFERENCES :

<https://www.digitalocean.com/community/tutorials/how-to-share-data-between-docker-containers>

<https://www.digitalocean.com/community/tutorials/how-to-share-data-between-docker-containers>

Assignment 02

To Download & Install Selenium WebDriver on Ubuntu

Selenium installation is a 3 step process:

1. Install Java SDK - <https://www.oracle.com/java/technologies/javase-downloads.html>
2. Install Eclipse - <http://www.eclipse.org/downloads/>
3. Install Selenium Webdriver Files - <https://www.selenium.dev/downloads/>

In this tutorial, we will learn how to install Selenium Webdriver . Below is the detailed process

NOTE: The versions of Java, Eclipse, Selenium will keep updating with time. But the installation steps will remain the same. Please select the latest version and continue the installation steps below-

Step 1 – Install Java on your computer

Download and install the **Java Software Development Kit (JDK)** [here](#).

[Java 19](#) [Java 17](#)

Java SE Development Kit 19.0.1 downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

[Linux](#) [macOS](#) [Windows](#)

Product/file description	File size	Download
x64 Compressed Archive	179.13 MB	https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.zip (sha256)
x64 Installer	158.91 MB	https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.exe (sha256)
x64 MSI Installer	157.76 MB	https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.msi (sha256)

Next –

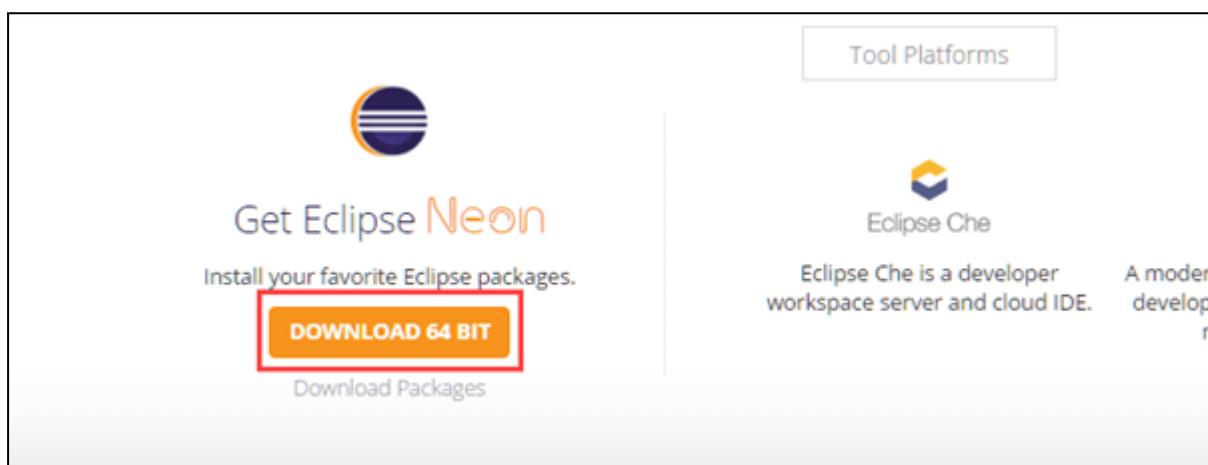
This JDK version comes bundled with Java Runtime Environment (JRE), so you do not need to download and install the JRE separately.

Once installation is complete, open command prompt and type “java”. If you see the following screen you are good to move to the next step

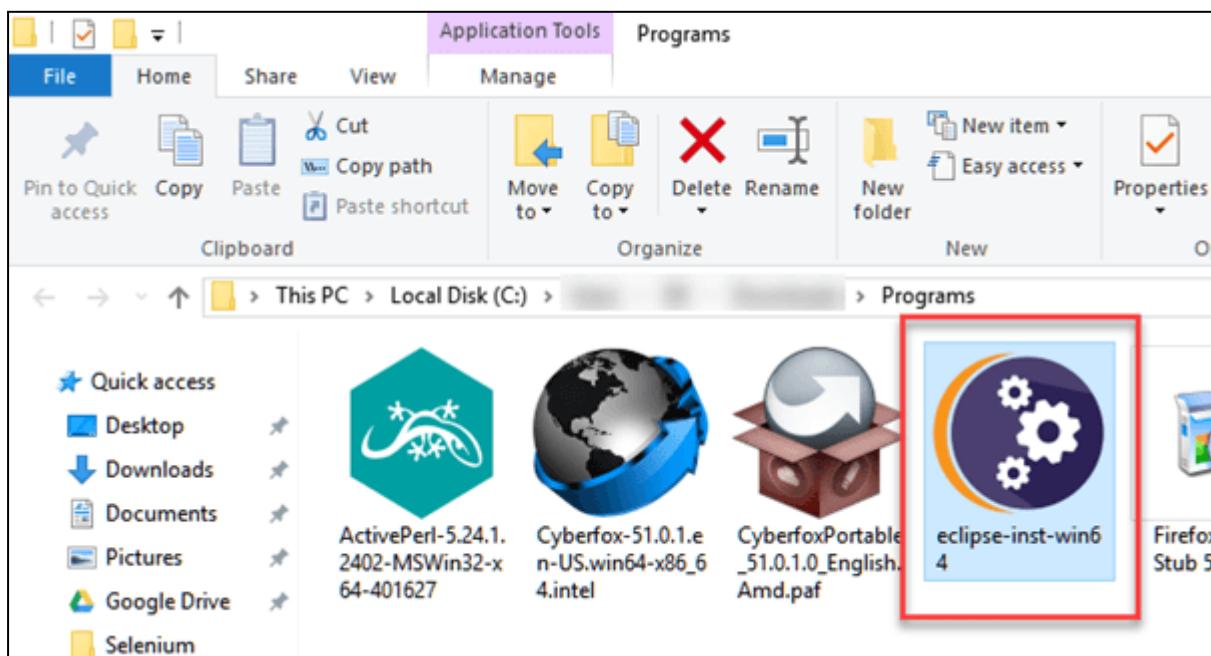
```
jasmit21@jasmit21:~$ java --version
openjdk 11.0.16 2022-07-19
OpenJDK Runtime Environment (build 11.0.16+8-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.16+8-post-Ubuntu-0ubuntu122.04, mixed mode,
sharing)
jasmit21@jasmit21:~$
```

Step 2 – Install Eclipse IDE

Download latest version of “Eclipse IDE for Java Developers” [here](#). Be sure to choose correctly between Windows 32 Bit and 64 Bit versions.



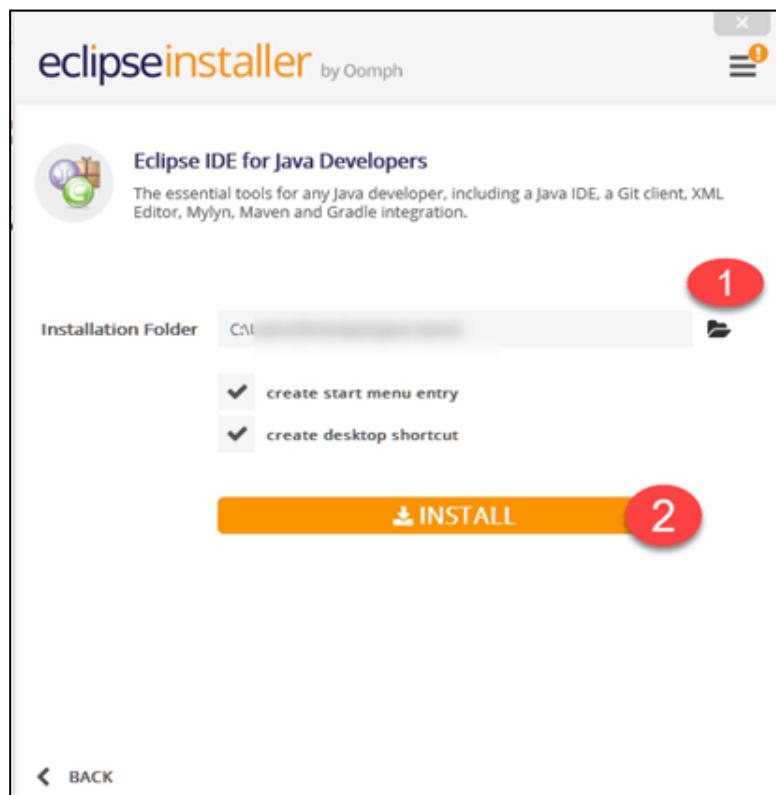
You should be able to download an exe file named “eclipse-inst-win64” for Setup.



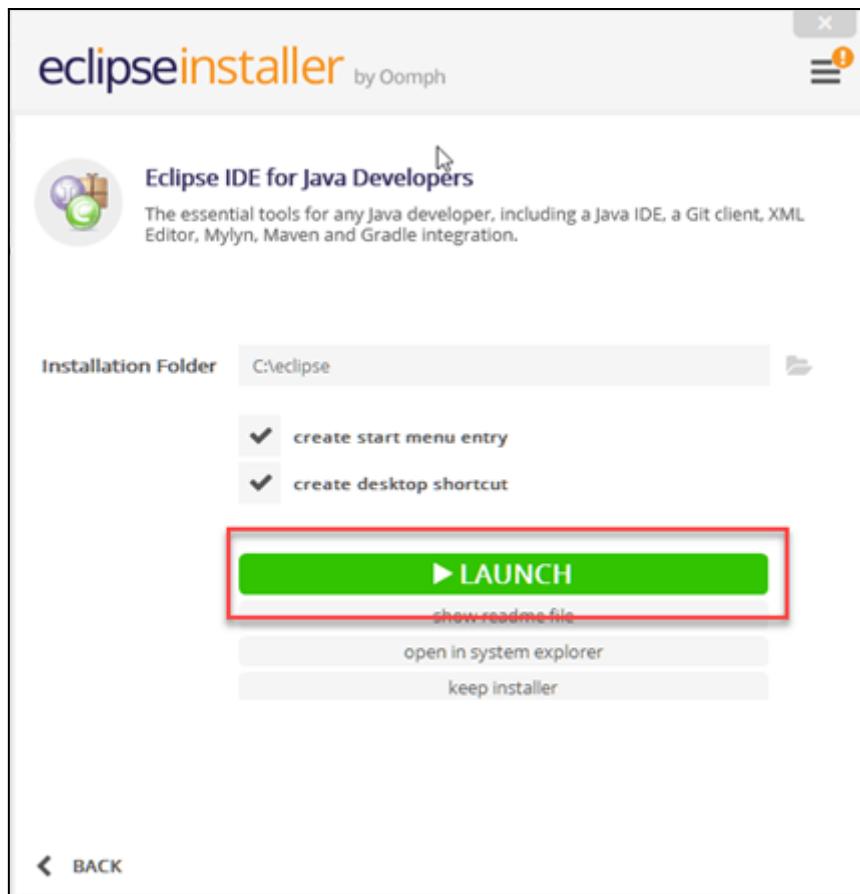
Double-click on file to Install the Eclipse. A new window will open. Click Eclipse IDE for Java Developers.



After that, a new window will open which click button marked 1 and change path to "C:\eclipse". Post that Click on Install button marked 2



After successful completion of the installation procedure, a window will appear. On that window click on Launch



This will start eclipse neon IDE for you.

Step 3 – Download the Selenium Java Client Driver

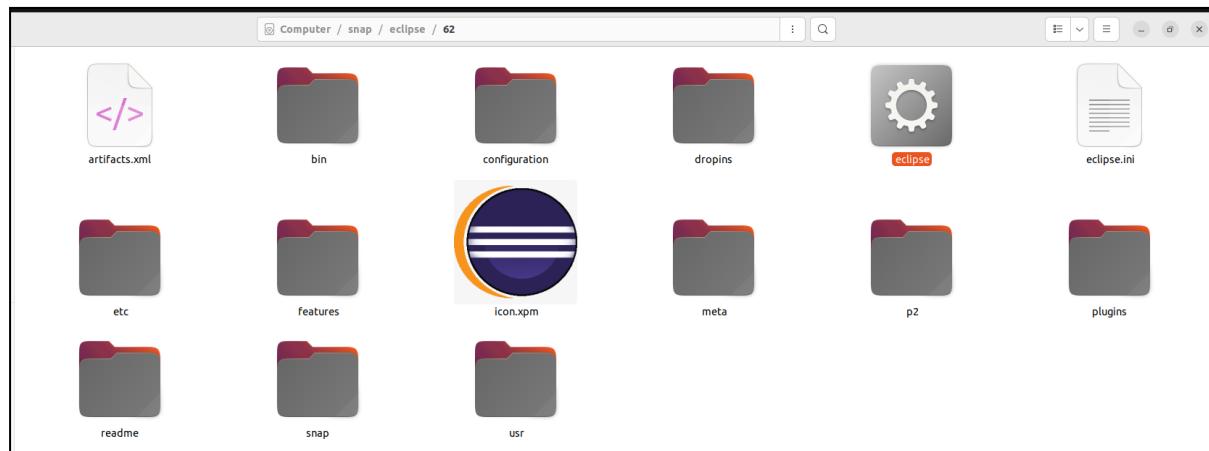
You can download **Selenium Webdriver for Java Client Driver** [here](#). You will find client drivers for other languages there, but only choose the one for Java.

Selenium Client & WebDriver Language Bindings		
LANGUAGE	VERSION	RELEASE DATE
Ruby	3.142.6	October 04, 2019
JavaScript	4.0.0-alpha.5	September 08, 2019
Java	3.141.59	November 14, 2018
Python	3.141.0	November 01, 2018
C#	3.14.0	August 02, 2018

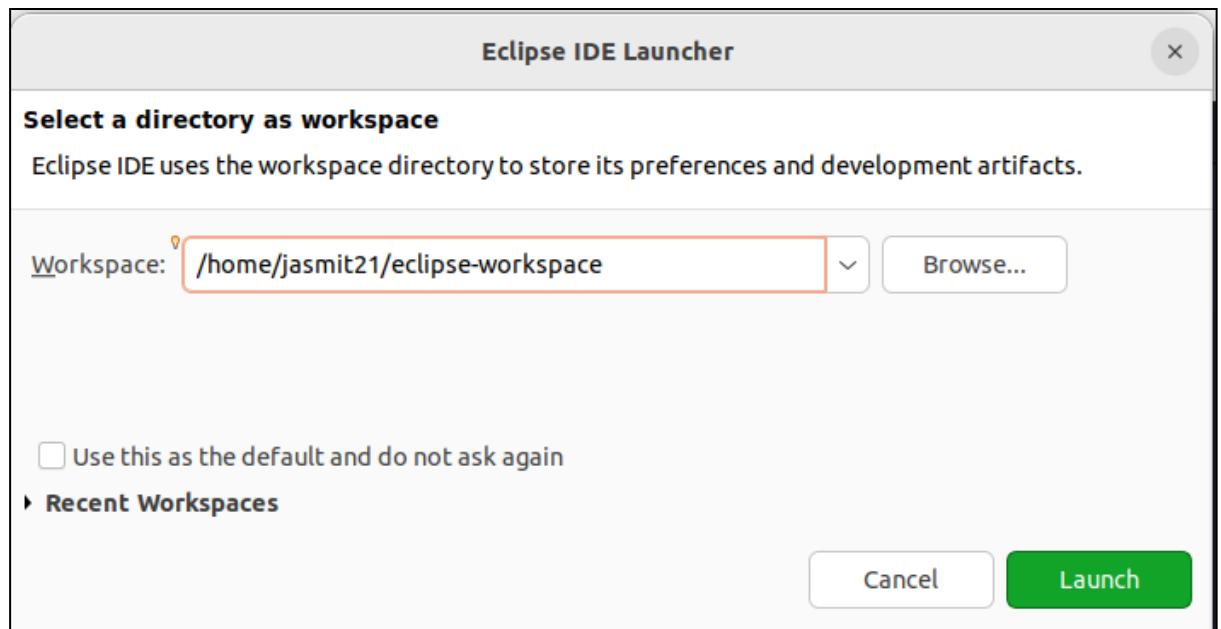
This download comes as a ZIP file named “selenium-3.14.0.zip”. For simplicity of Selenium installation on Windows 10, extract the contents of this ZIP file on your C drive so that you would have the directory “C:\selenium-3.14.0\”. This directory contains all the JAR files that we would later import on Eclipse for Selenium setup.

Step 4 – Configure Eclipse IDE with WebDriver

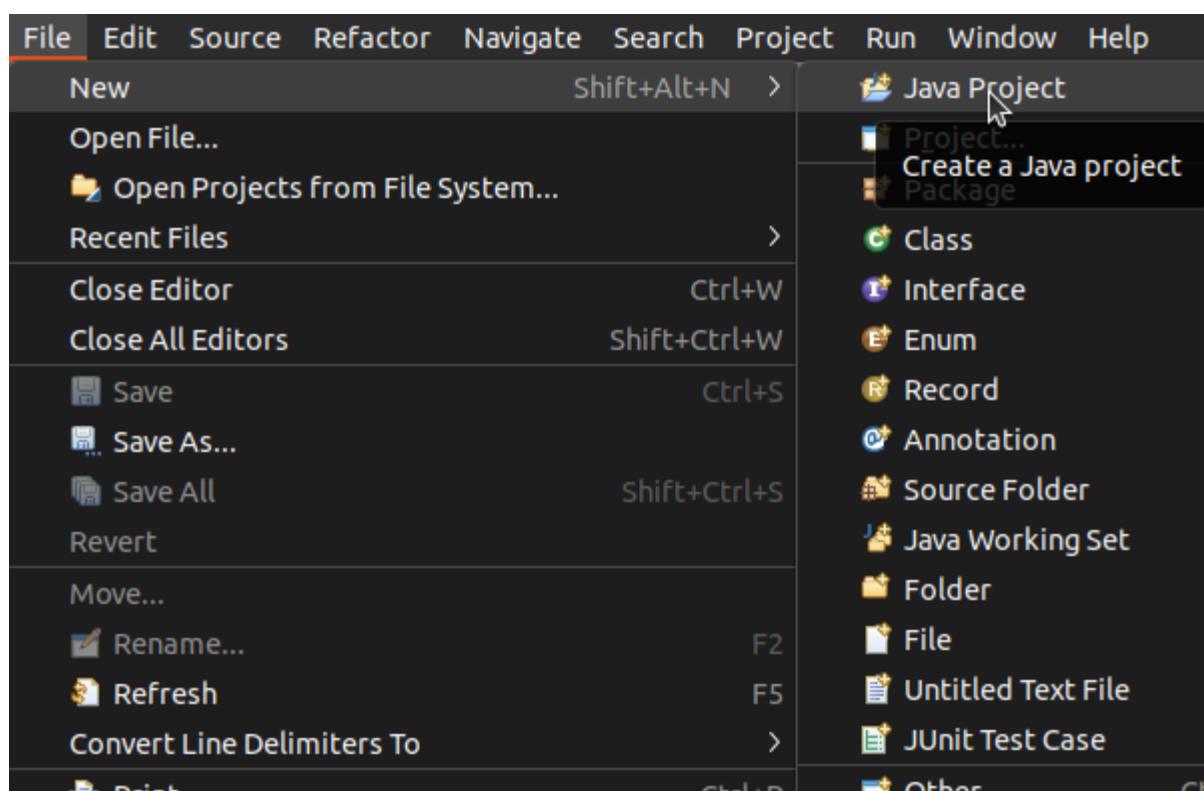
1. Locate the eclipse folder and search for executable file , then launch the eclipse



- When asked to select a workspace, just accept the default location.

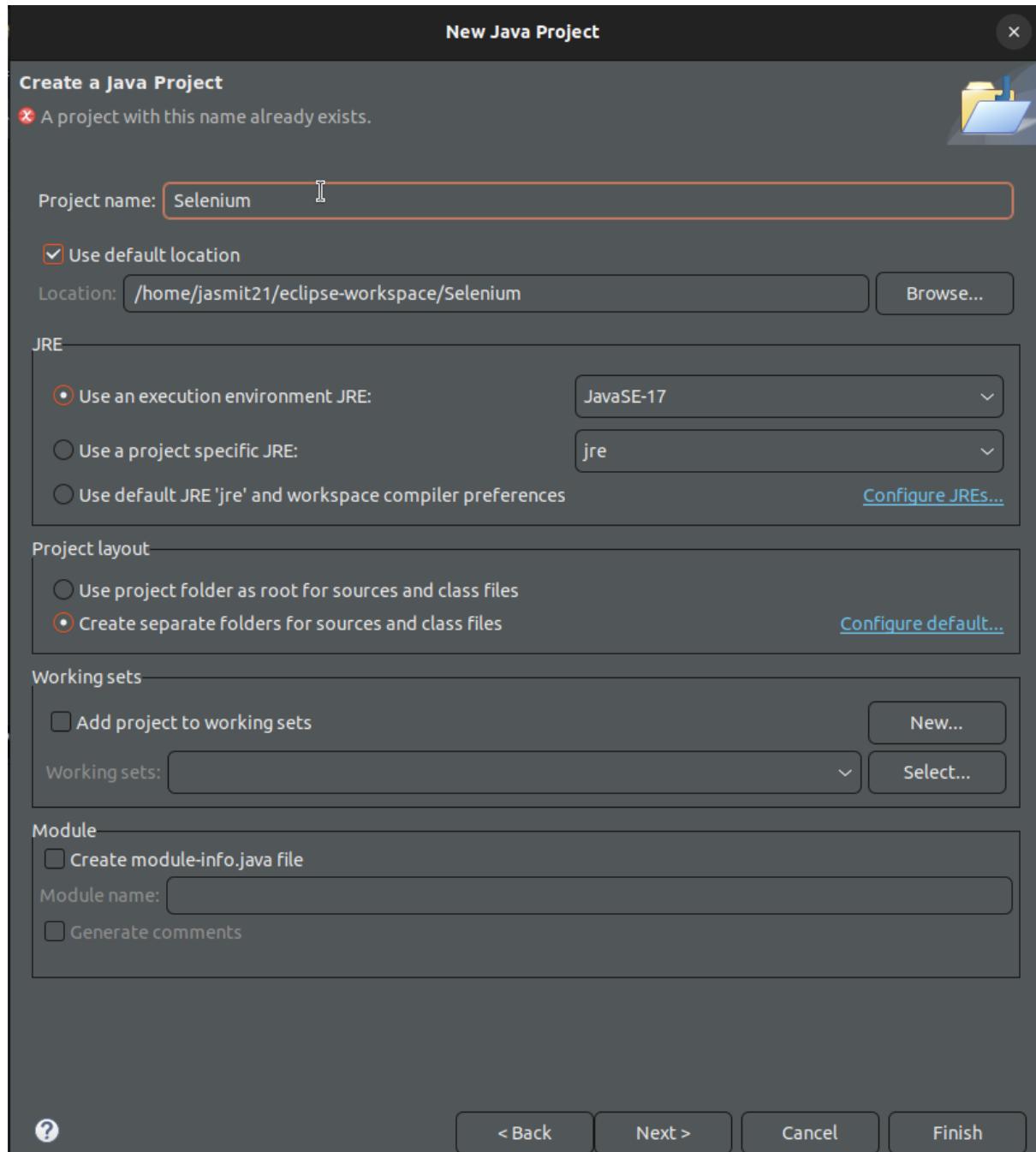


- Create a new project through File > New > Java Project. Name the project as "Selenium_Demo".



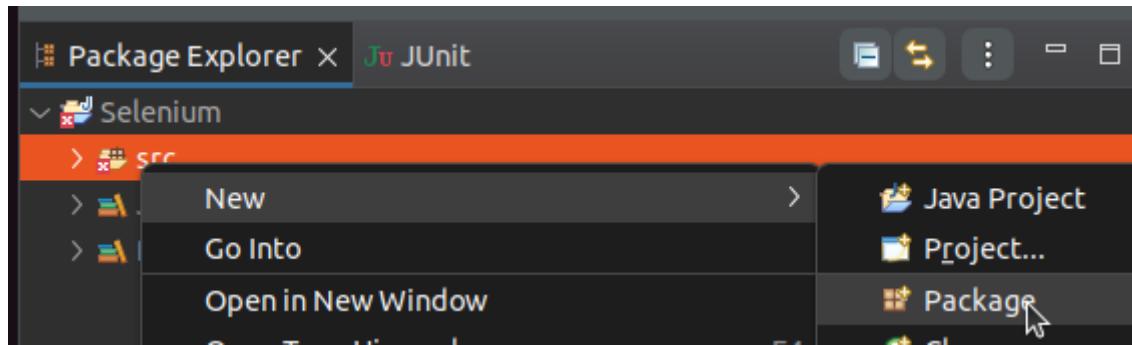
A new pop-up window will open enter details as follow

1. Project Name
2. Location to save project
3. Select an execution JRE
4. Select layout project option
5. Click on Finish button



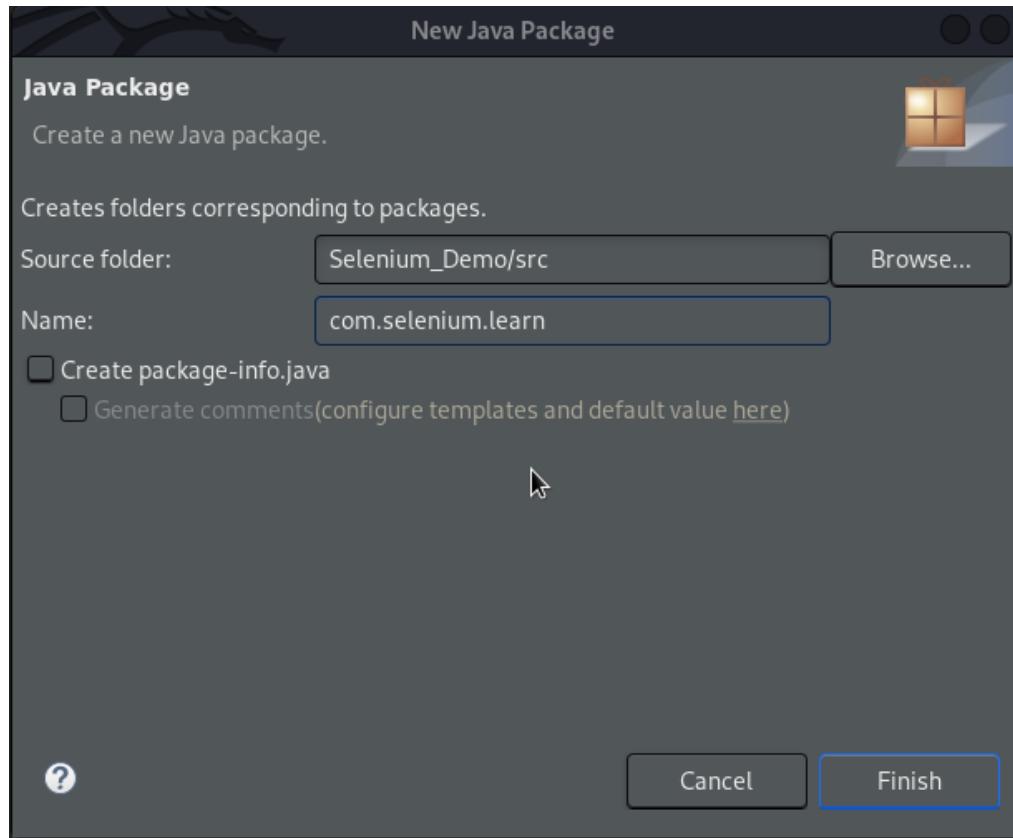
4. In this step,

1. Right-click on the newly created project and
2. Select New > Package, and name that package as “com.selenium.learn”.

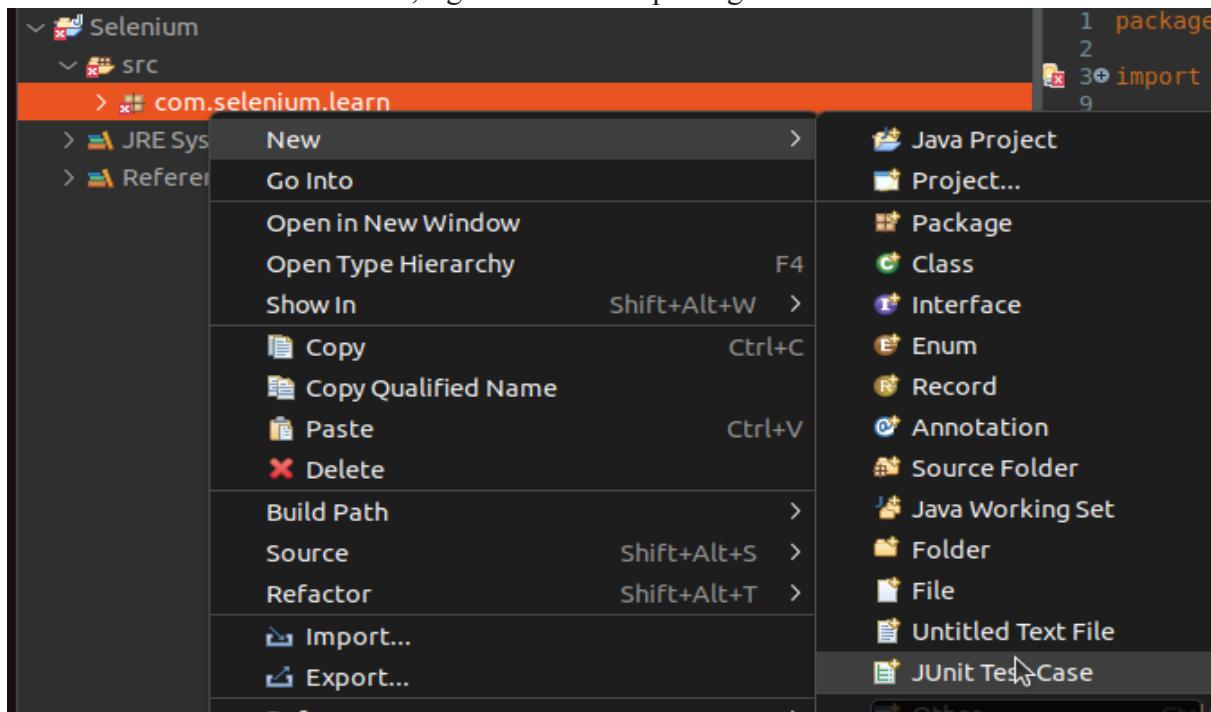


A pop-up window will open to name the package,

1. Enter the name of the package
2. Click on Finish button

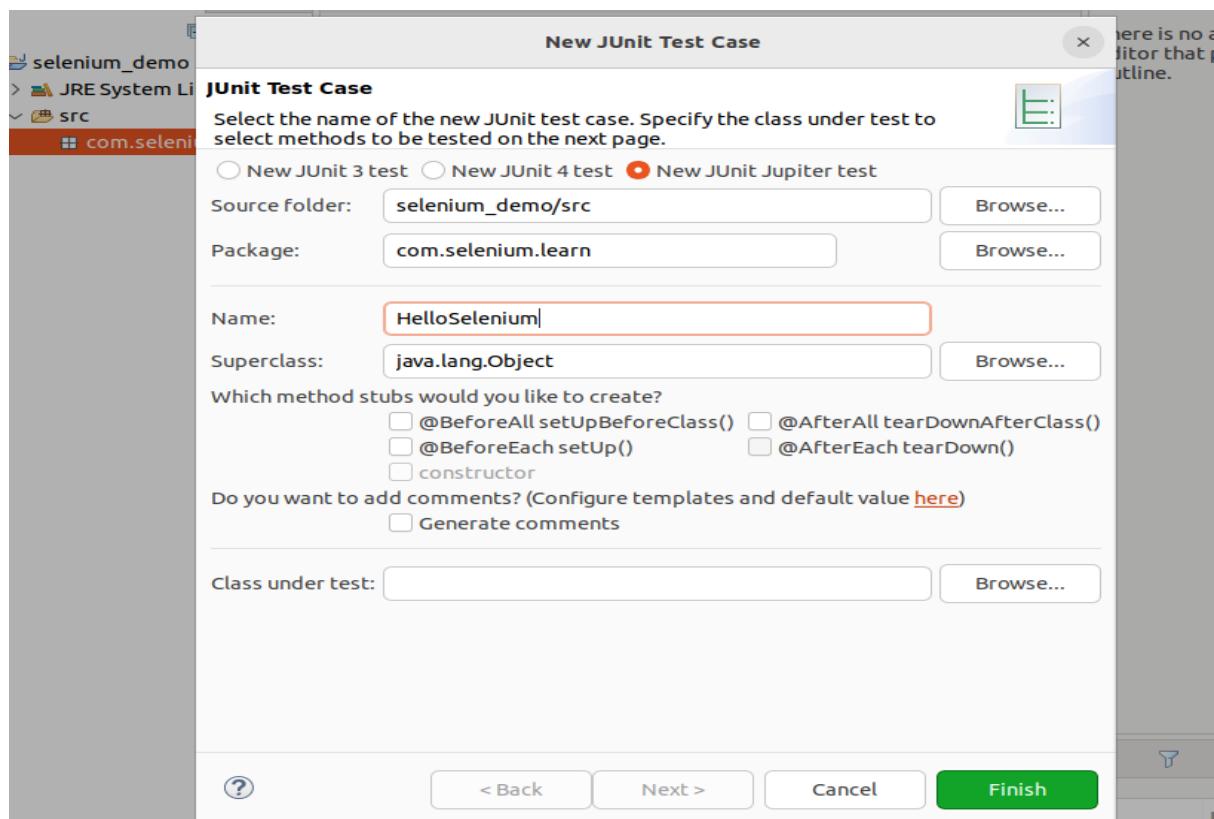


5. Create a new Junit Test Case , right click on the package → new → Junit test case



When you click on Junit Test Case, a pop-up window will open, enter details as

1. Name (Put name as HelloSelenium)
2. Click on Finish button



This is how it looks like after creating Junit

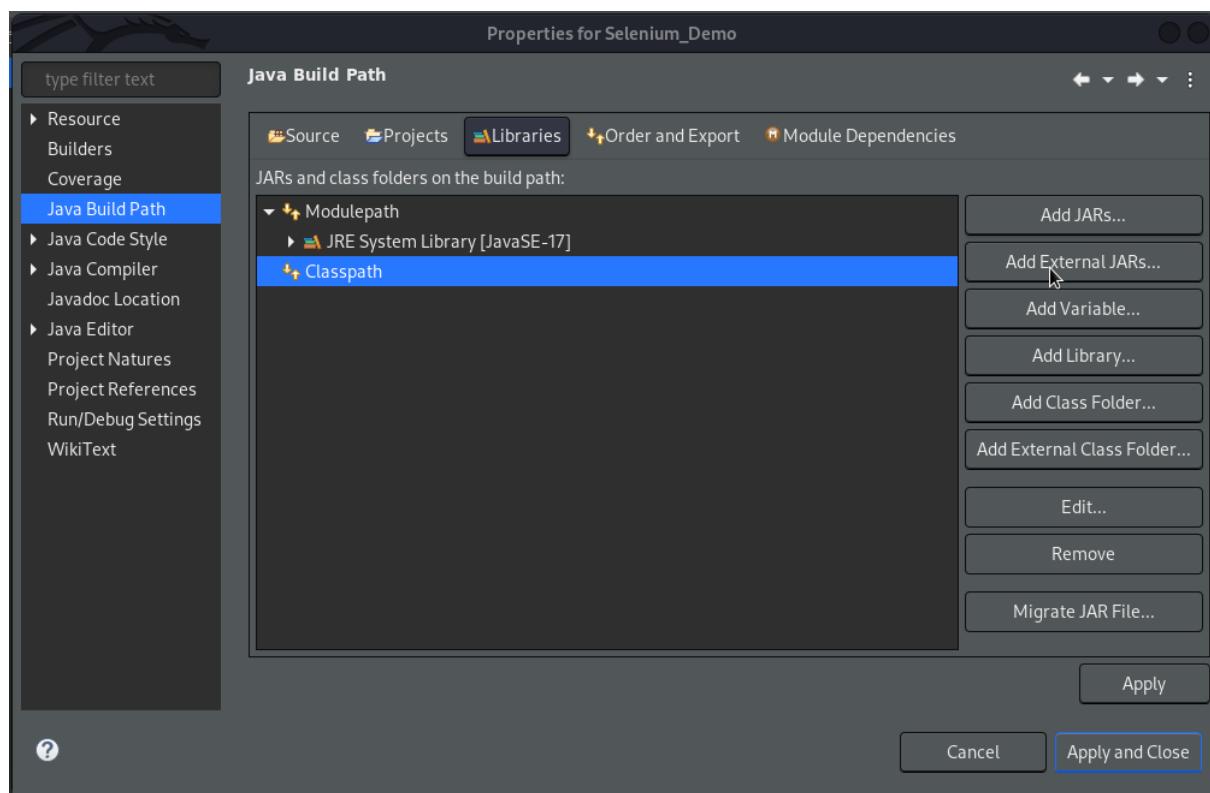
The screenshot shows the Eclipse IDE interface with the title "eclipse-workspace - selenium_demo/src/com/selenium/learn>HelloSelenium.java - Eclipse IDE". The left side has a "Package Explorer" view showing a project named "selenium_demo" with a "src" folder containing a "com.selenium.learn" package and a "HelloSelenium.java" file. The right side has an "Outline" view showing the class "HelloSelenium" and a method "test()". The code editor window displays the following Java code:

```
1 package com.selenium.learn;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class HelloSelenium {
6
7     @Test
8     void test() {
9         assertEquals("Hello Selenium", "Hello Selenium");
10    }
11
12 }
13
14 }
15
```

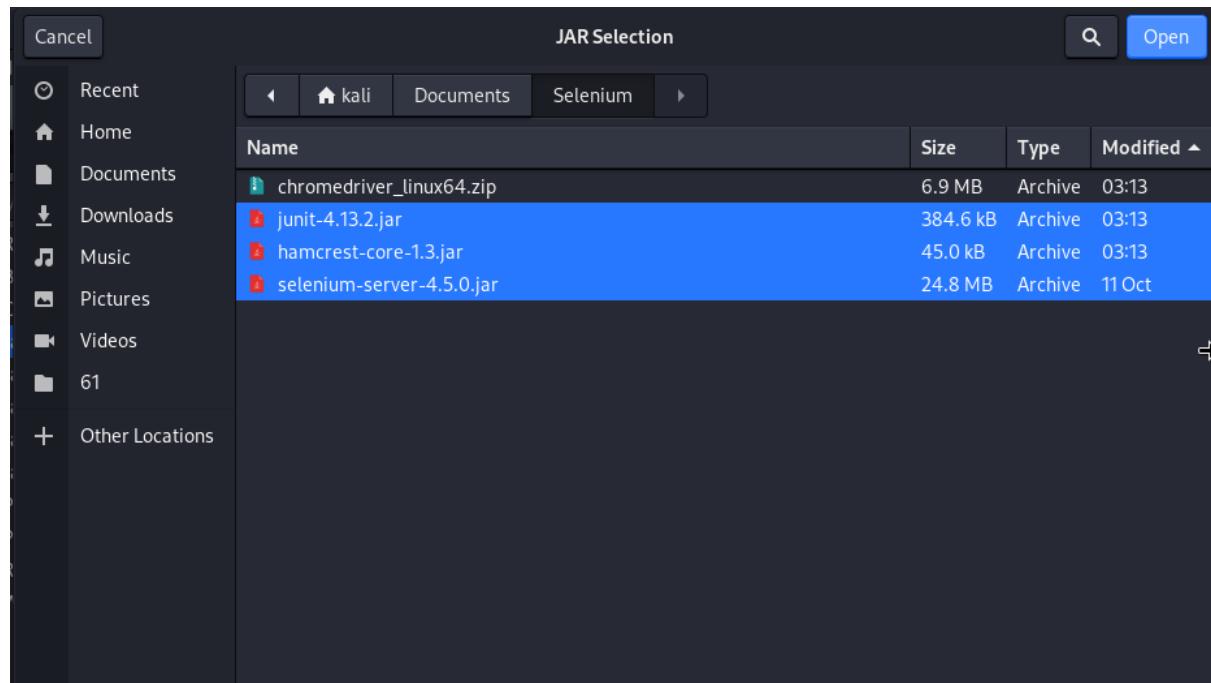
Now selenium WebDriver's into Java Build Path

In this step,

1. Right-click on your project and select **Properties**.
2. On the Properties dialog, click on “Java Build Path”.
3. Click on the **Libraries** tab, and then classpath
4. Click on “Add External JARs..”

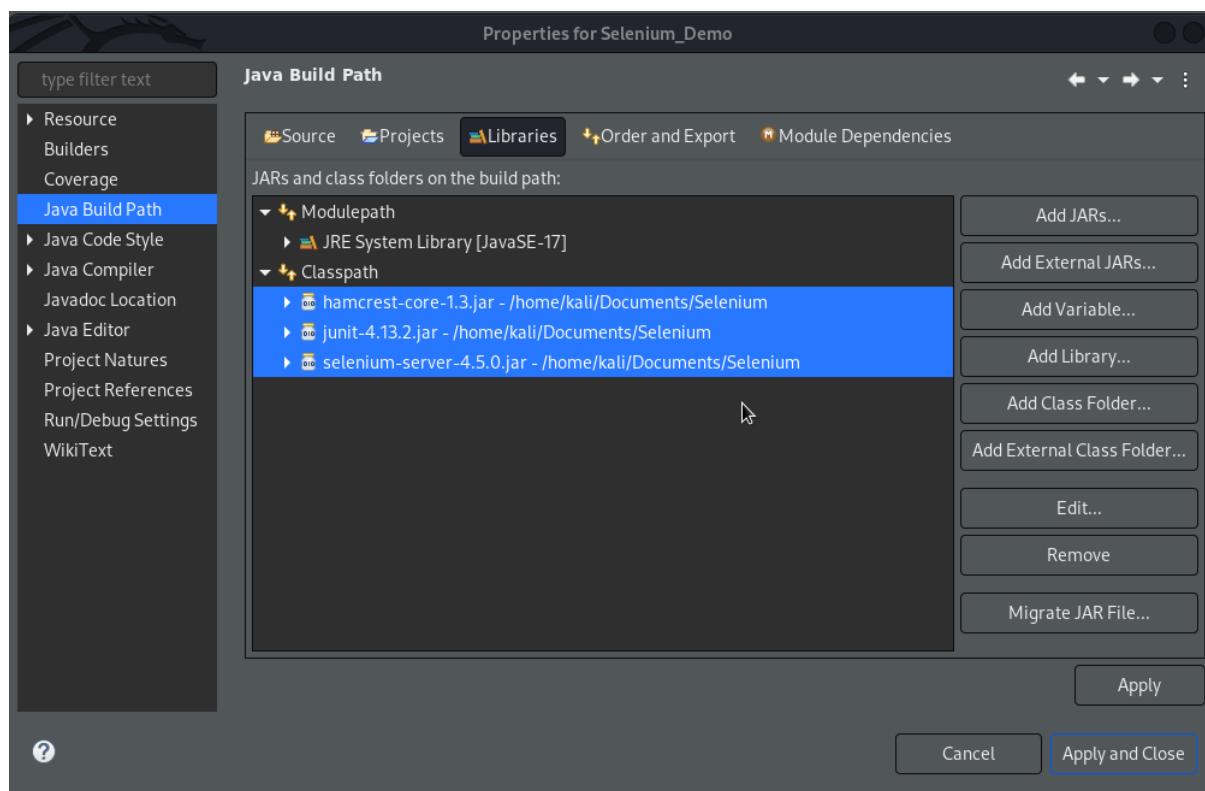


When you click on “Add External JARs..” It will open a pop-up window. Select the JAR files you want to add. (Make sure you select all three jar files including junit , hamcrest , selenium-server



After selecting jar files, click on the OK button.

6. Add all the JAR files inside and outside the “libs” folder. Your Properties dialog should now look similar to the image below.

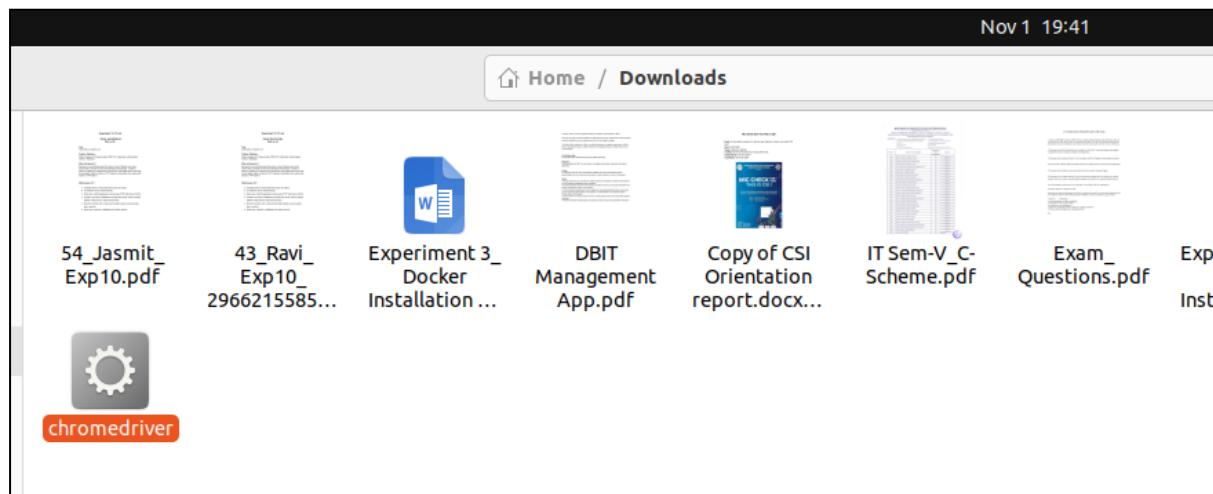


```

1 package com.selenium.learn; // Your package structure may be different or if using default package remove this.
2
3
4@import org.junit.After;
5 import org.junit.Before;
6 import org.junit.Test;
7 import org.openqa.selenium.By;
8 import org.openqa.selenium.WebDriver;
9 import org.openqa.selenium.chrome.ChromeDriver;
10
11 public class HelloSelenium {
12 WebDriver driver;
13 String url = "https://www.google.com/";
14
15@Before
16 public void setUp() {
17 //Set the key/value property according to the browser you are using.
18 System.setProperty("webdriver.chrome.driver","/home/ruthra/Selenium_Setup/chromedriver");
19
20 //Open browser instance
21 driver = new ChromeDriver();           █
22
23 //Open the AUT
24 driver.get(url);
25 }
26
27@Test
28 public void test() throws InterruptedException {
29 //Fetch the page title
30 String pageTitle = driver.getTitle();
31 System.out.println("Page title: " + pageTitle);
32
33 //Hey Google
34 driver.findElement(By.name("q")).sendKeys("Hey World");
35 Thread.sleep(3000);
36 driver.findElement(By.name("btnK")).click();
37 Thread.sleep(5000);
38
39 }

```

8. Change the chromedriver path , remove the given path and add from your system

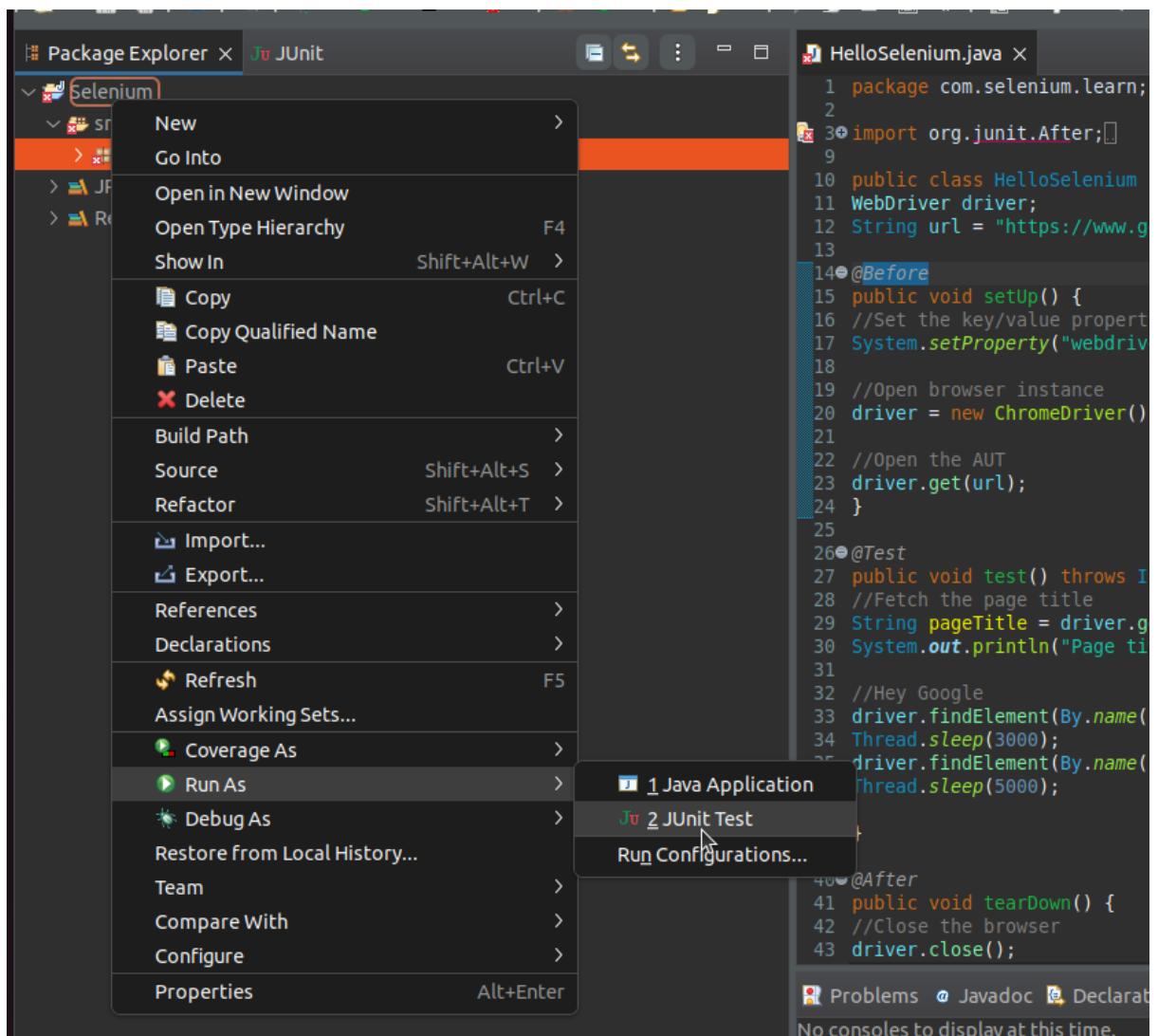


```

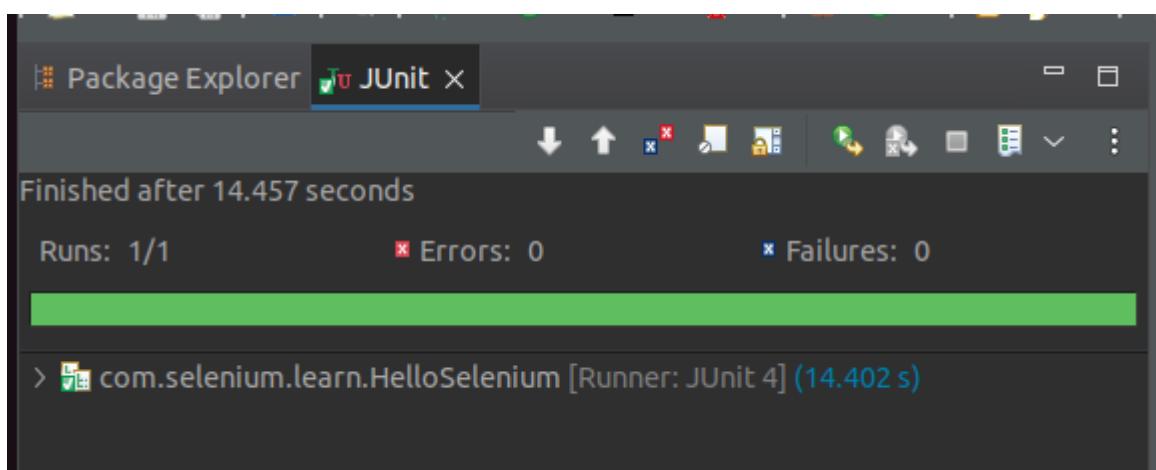
13
14@Before
15 public void setUp() {
16 //Set the key/value property according to the browser you are using.
17 System.setProperty("webdriver.chrome.driver","/home/jasmit21/Downloads/chromedriver");
18
19 //Open browser instance
20 driver = new ChromeDriver();
21

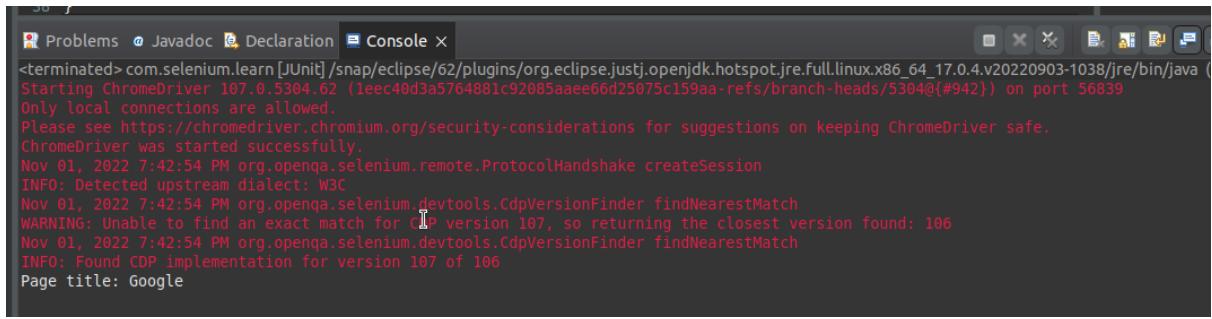
```

9. Right click on the project and run as Junit Test



Output:





The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The output window displays the following log messages:

```
<terminated> com.selenium.learn [JUnit] /snap/eclipse/62/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.4.v20220903-1038/jre/bin/java
Starting ChromeDriver 107.0.5304.62 (1eec40d3a5764881c92085aaee66d25075c159aa-refs/branch-heads/5304@(#942)) on port 56839
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Nov 01, 2022 7:42:54 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Nov 01, 2022 7:42:54 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 107, so returning the closest version found: 106
Nov 01, 2022 7:42:54 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 107 of 106
Page title: Google
```

Conclusion: Thus installation and first program on Selenium was run successfully

References :

1. <https://drive.google.com/drive/folders/113WPhiDvpeLb7lm3goo7vtNT0kTAjXIH?usp=sharing>
2. <https://www.youtube.com/watch?v=MUTBV1RJBiQ&t=511s>
3. <https://www.guru99.com/installing-selenium-webdriver.html>