Don Bosco Institute Of Technology, (DBIT), Mumbai

Department Of Information Technology

LAB JOURNAL

On

ITL503 : DevOps Lab

By

11 Suhani Desale

Academic Year : Nov, 2022

INDEX

EXPERIMENT A - Self Study

Date of Experiment: 11/07/2022

Prerequisite:
C, Python, Java, Software Engineering, Cloud

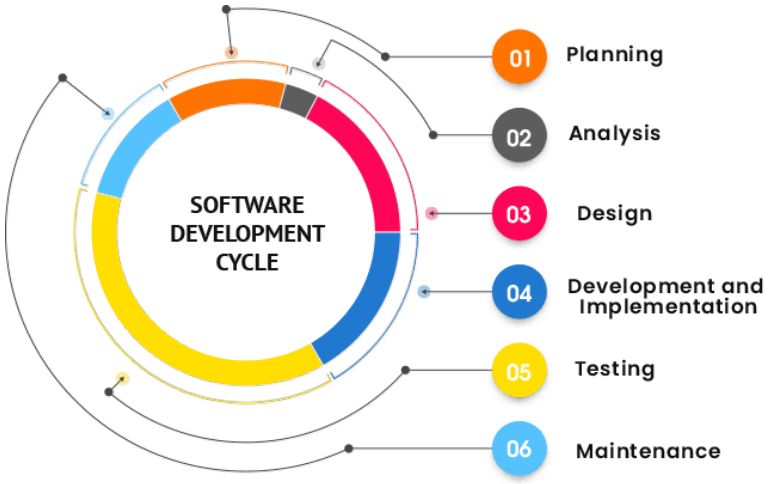
Objective:
To make students understand DevOps, for what? Why? and by whom?

Aim:
To do self-study on basics understanding of DevOps

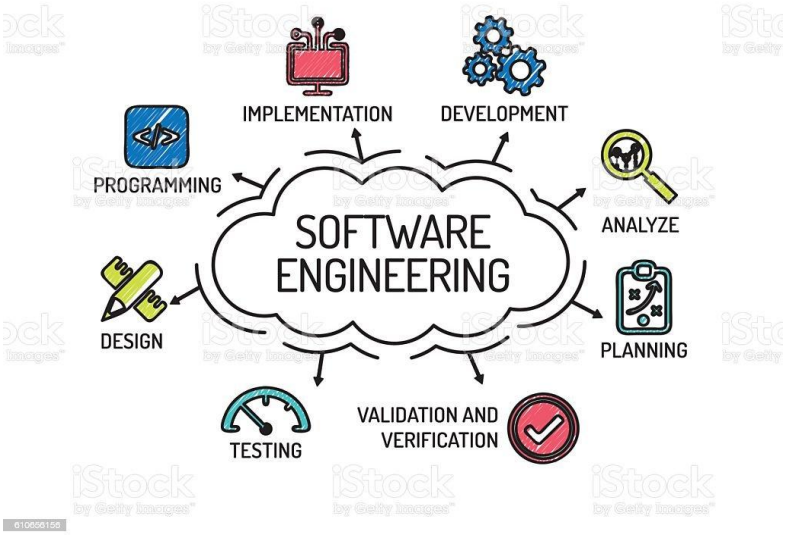
Procedure:
Individual students will perform this self-study over internet resources and will submit a report on their study and give a viva voce on or before 18/07/2022, and they will be graded based on rubrics.

The topics for self-study are :

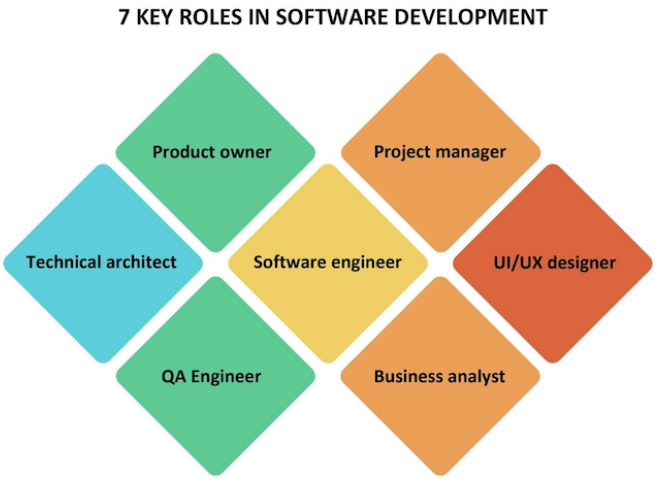
1. Development of Software

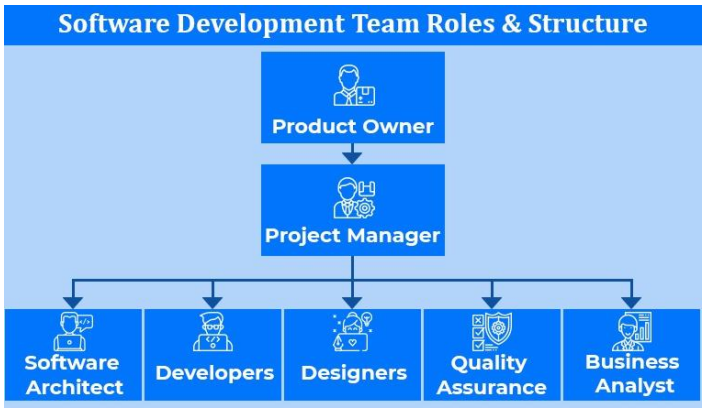


2. Software Engineering



3. Development team





4. Operation team

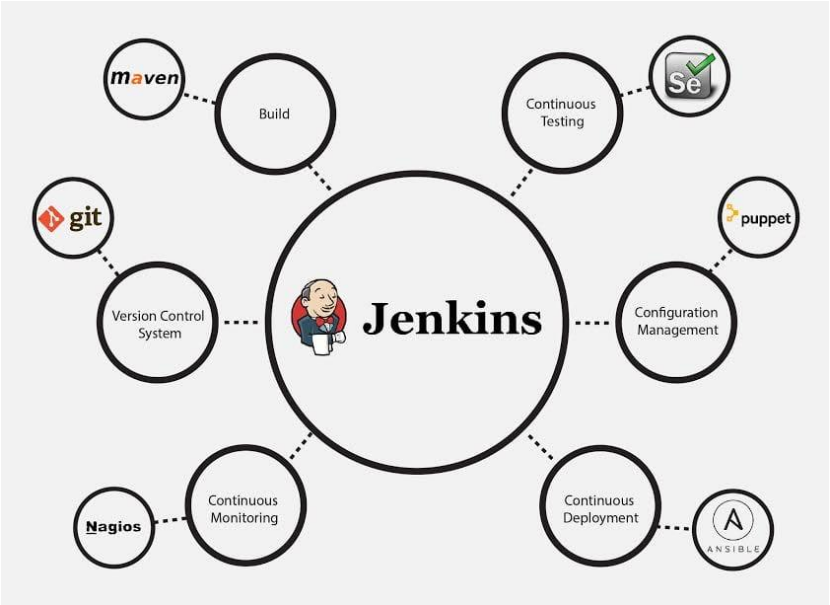
5. Operations Step

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.



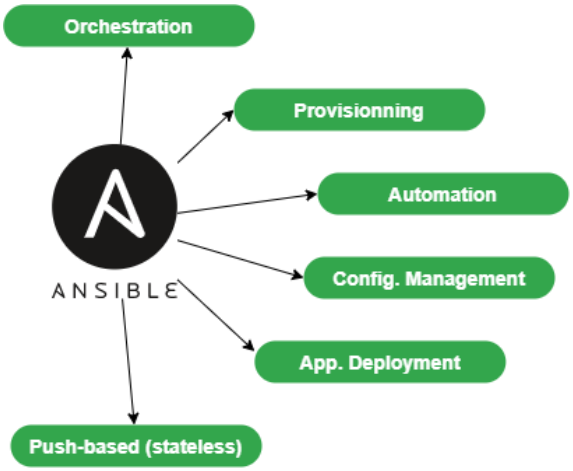
8. Docker

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.

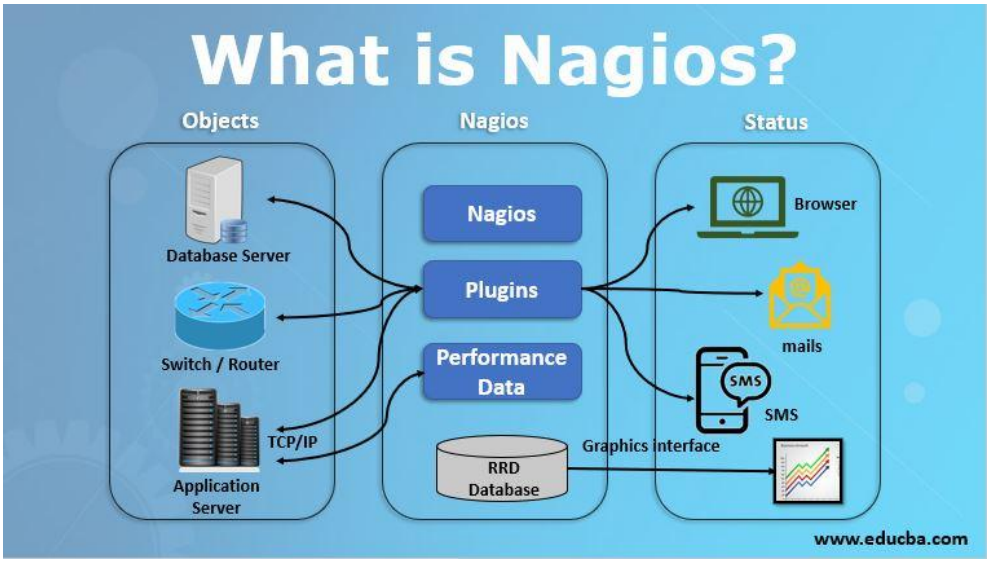
9. Ansible



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 - “Azure”

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.

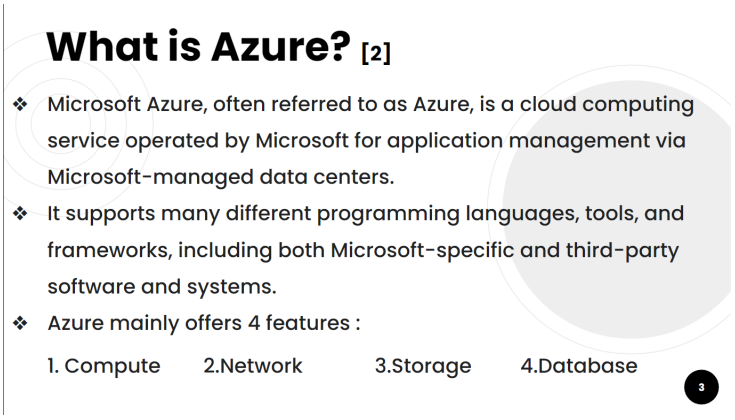
Our team members are :

Suhani Desale

Daniel Colaco

Arsh Farooqui

Presentation:



Who Developed Azure? [1]

- Windows Azure was introduced at the Professional developers conference (PDC) in 2008 by Ray Ozzie, a former Chief Software Architect at Microsoft.
- Ray Ozzie was one of the pioneers and most well known proponents of the ground-breaking, web based software delivery paradigm that later came to be known as "Software as a Service" (SaaS).

5

continued

- Ozzie outlined his plan for developing a disruptive platform that would imitate the look of the Microsoft Windows OS on the Internet in a Microsoft internal letter dated October 28, 2005.

6

How was Azure developed?[3]

- The very first version of Windows Azure had a limited number of services that formed the four pillars of the platform.
- The first pillar was the compute service. Developers could package and run ASP.NET web applications and APIs that ran in the context of a web role while the worker role was designed for long-running processes with no UI.

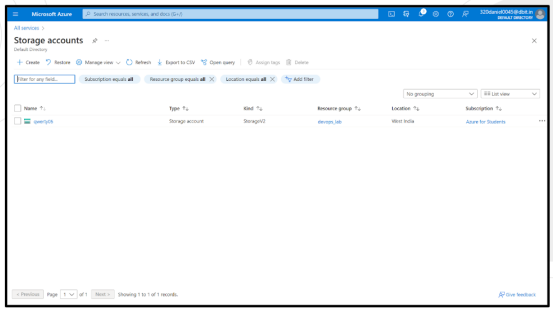
7

continued

- Azure Blob storage was the second pillar of Windows Azure that delivered persistence and durability to services.
- The third pillar was a database service branded as SQL Azure.
- The fourth pillar was the Azure Service Bus, a message bus that was born out of BizTalk Server.

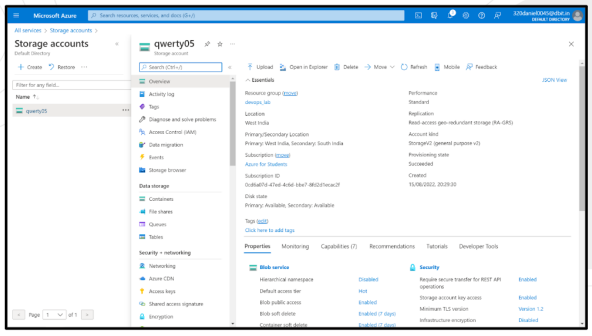
8

Creation of Storage Account



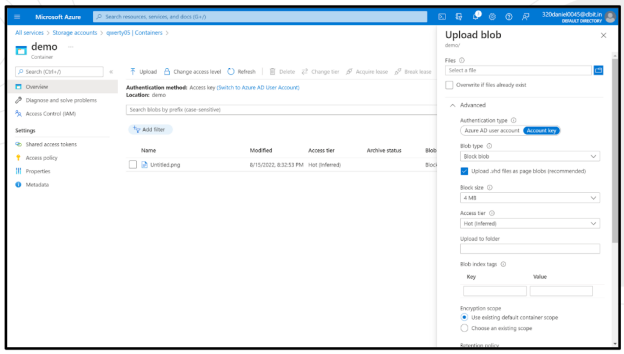
9

Properties of Storage Account



10

Creation of Container



11

Conclusion :

Presentation was done on the respective topic and a document was prepared

References:

1. “A Look back at Ten Years of Microsoft Azure” www.forbes.com (slide 5,6) https://www.forbes.com/sites/janakirammsv/2020/02/03/a-look-back-at-ten-years-of-microsoft-azure/?sh=21144f9c4929 (accessed on Aug 3 , 2022)
2. “Microsoft Azure” www.wikipedia.org (slide 3,4) https://en.wikipedia.org/wiki/Microsoft_Azure (accessed on Aug 10, 2022)
3. “What is Azure | Azure for Beginners | Intellipaat - YouTube” https://youtu.be/eN799RtzmY8 (slide 7,8)(accessed on Aug 10,2022)

EXPERIMENT 1

Version Control Using Git

Date of 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, analyze 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 mentioned 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.

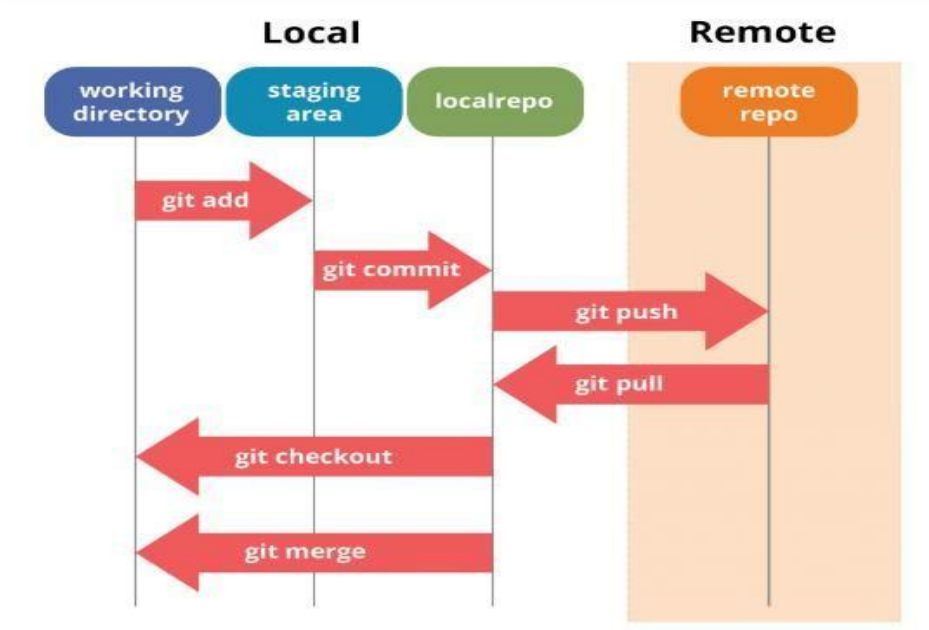
Some of the basic operations in Git are:

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 its 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/

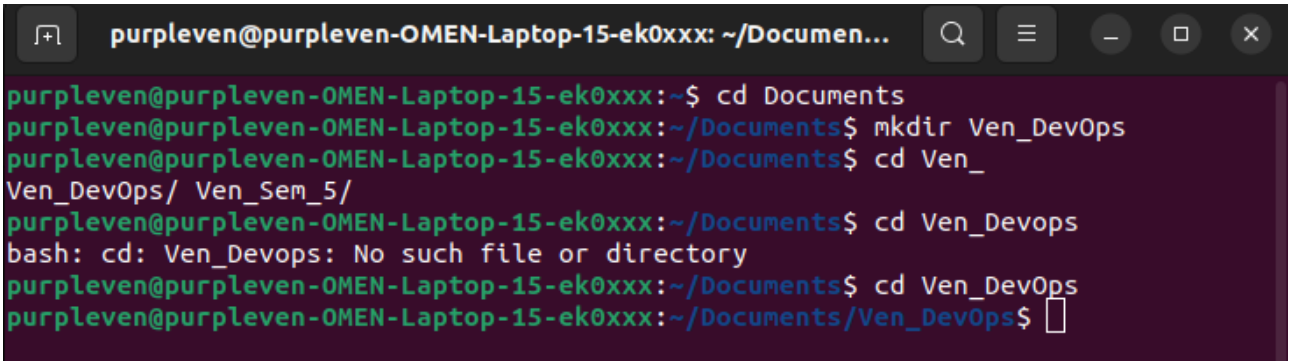
Output:

Installing git with `sudo apt install git`

Checking if git is installed with `git –version`

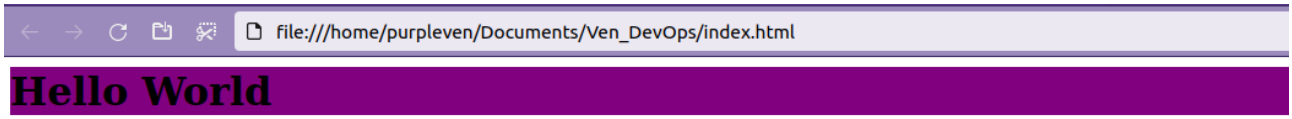
Now with the help of git:

1) Creating a directory



2) Creating two files with the help of the touch command

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ touch index.html
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ cat >> index.html
<html><body><h1>Hello World</h1></body></html>^Z
[1]+  Stopped                 cat >> index.html
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ touch index.html
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ cat index.html<html><body><h1>Hello World</h1></body></html>
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ touch style.css
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ cat >> style.css
<link rel="stylesheet" href="style.css">^Z
[2]+  Stopped                 cat >> style.css
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ cat style.css
h1{
background-color: purple;
}
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```



3) Initializing git with `git init`

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git init
hint: Using 'master' as the name for the initial branch. This default branch na
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/purpleven/Documents/Ven_DevOps/.git/
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

4) Now config the git with your username and email

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git config --global user.name "Vendra"
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git config --global user.email "vendra0408@gmail.com"
```

5) Checking the status of the files with the help of `git status`

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

6) Adding the untracked files with `git add .`

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git add .
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git status
On branch master

No commits yet

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

purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

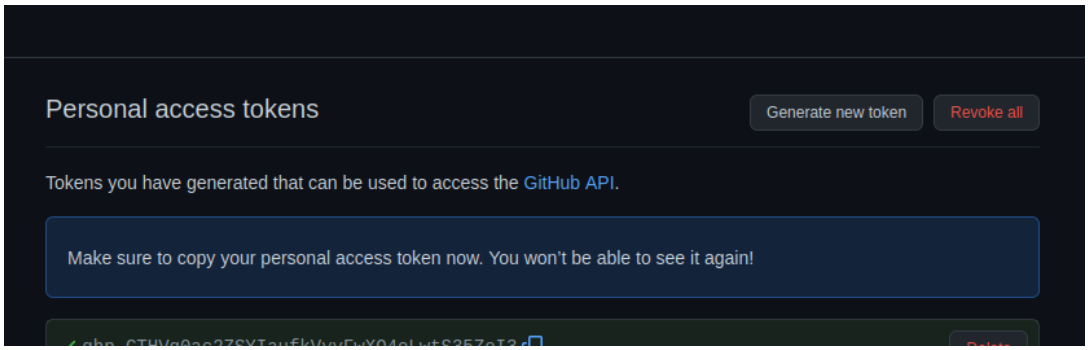
7) Now commiting the added files with a message

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git commit -m "Initial Commit"
[master (root-commit) 97e898b] Initial Commit
 2 files changed, 5 insertions(+)
 create mode 100644 index.html
 create mode 100644 style.css
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

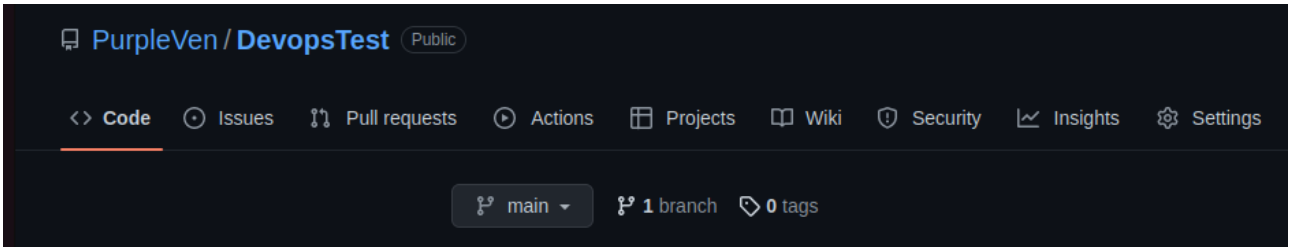
8) Checking with the help of log if the changes are committed

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git log
commit 97e898b81ae580edad91574c6fb9ea9c14368cec (HEAD -> master)
Author: Vendra <vendra0408@gmail.com>
Date:   Sun Aug 7 13:07:04 2022 +0530

    Initial Commit
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

Now with the help of GitHub:
Go to GitHub ->User accounts-> setting -> developers portal ->generate token
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



1) Create a repository



2) Adding remote origin

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git remote add origin https://github.com/PurpleVen/DevopsTest.git
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

3) Cloning the repository

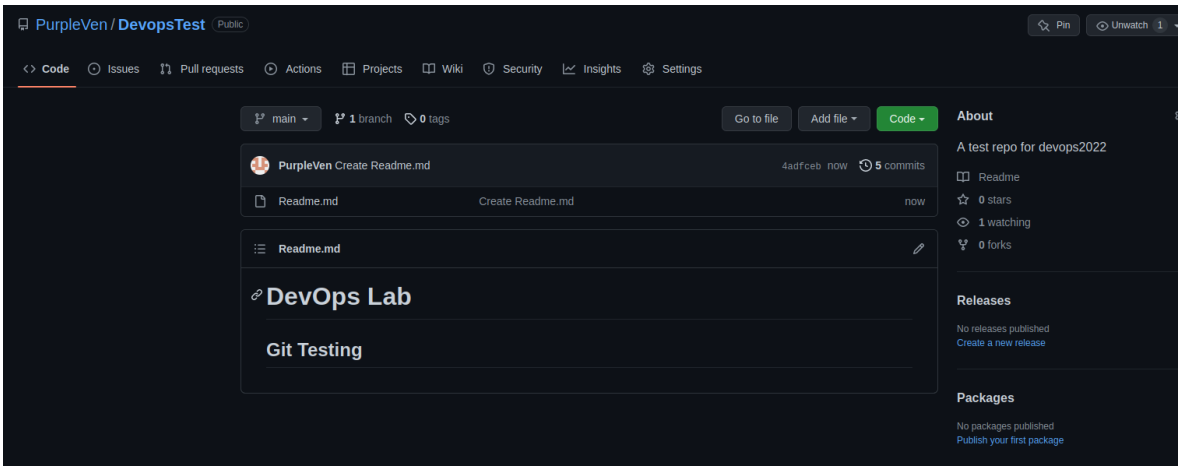
```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git clone https://github.com/PurpleVen/DevopsTest.git
Cloning into 'DevopsTest'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 10 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), done.
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

4) git remote set url

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

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git remote set-url origin https://PurpleVen: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@github.com/PurpleVen/DevopsTest.git
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

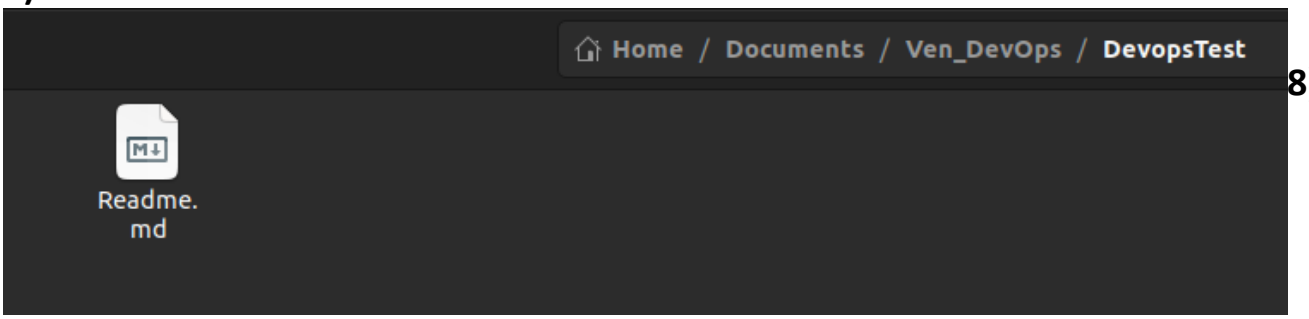
5) Added a file in github



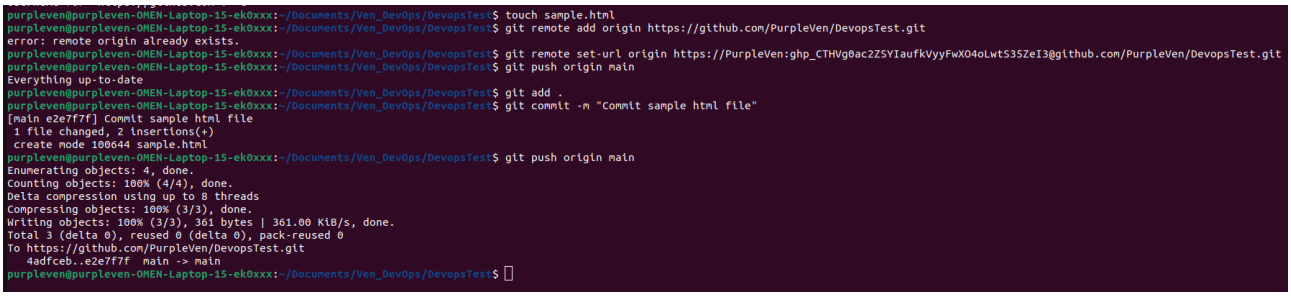
6) Trying the pull command

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ cd DevopsTest/
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 649 bytes | 649.00 KiB/s, done.
From https://github.com/PurpleVen/DevopsTest
   69e546e..4adfceb  main       -> origin/main
Updating 69e546e..4adfceb
Fast-forward
 Readme.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 Readme.md
```

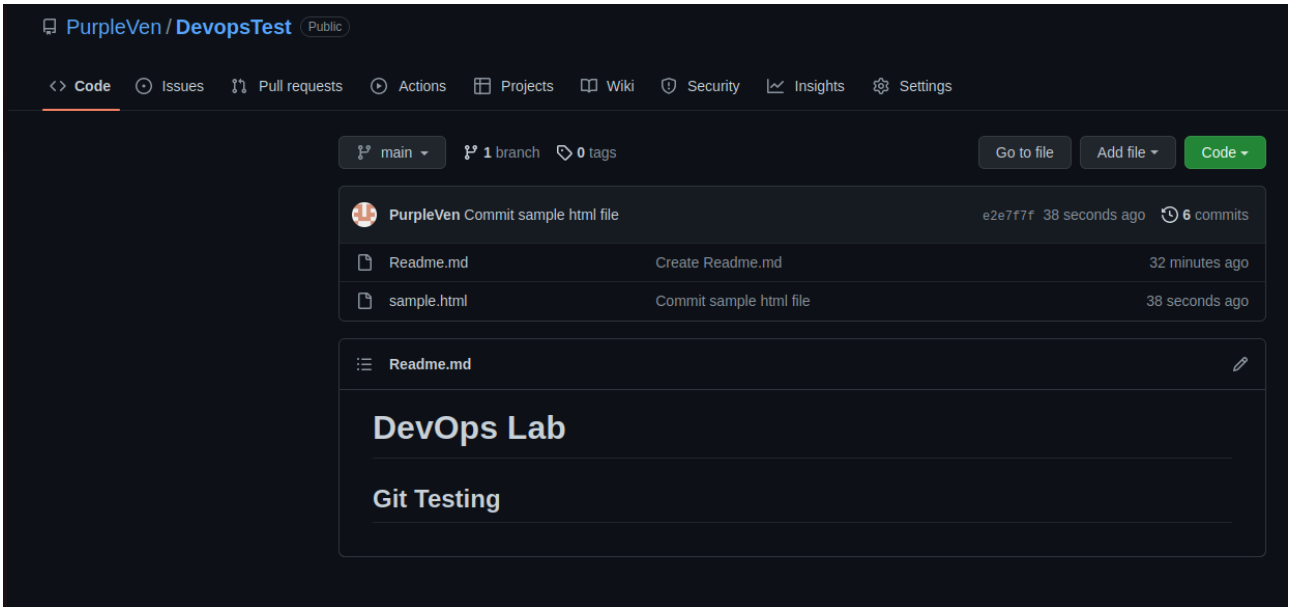
7) The readme file is visible



8) Pushing the files from local to GitHub

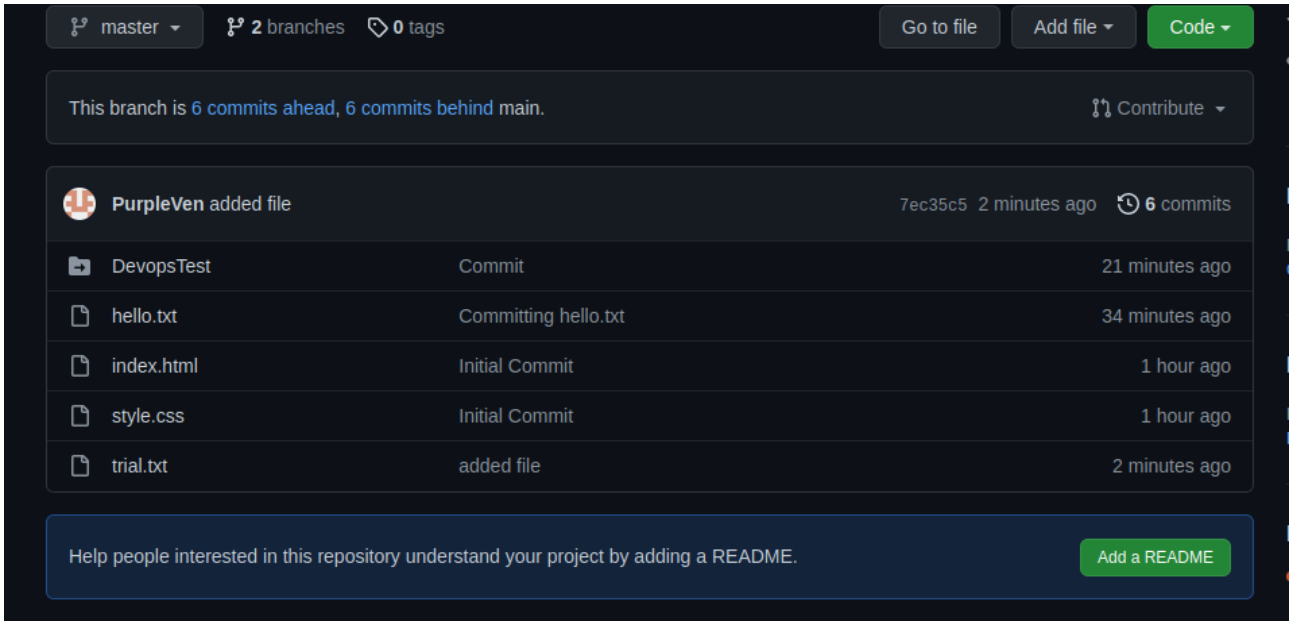


9) Displayed in GitHub



New branch master is created

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git remote add origin https://github.com/PurpleVen/DevopsTest.git
error: remote origin already exists.
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git add .
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git commit -m "git committtt"
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git push -u origin master
Branch 'master' set up to track remote branch 'master' from 'origin'.
Everything up-to-date
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/PurpleVen/DevopsTest.git'
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git push origin master
Everything up-to-date
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ touch trial.txt
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git add .
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git commit -m "added file"
[master 7ec35c5] added file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 trial.txt
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 256 bytes | 256.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/PurpleVen/DevopsTest.git
   77c7a3e..7ec35c5  master -> master
```



10) Making a new branch and adding the same file as that of main to test the git merge command

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git
 checkout -b ven_branch
Switched to a new branch 'ven_branch'
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ tou
ch sample.html
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ cat
 >> sample.html
<!DOCTYPE html>
<html>
<head>
<title>Hello World!</title>
<link rel="stylesheet" href="bluestyle.css">
</head>
<body>

<h1>Hello world!</h1></body>^C
```

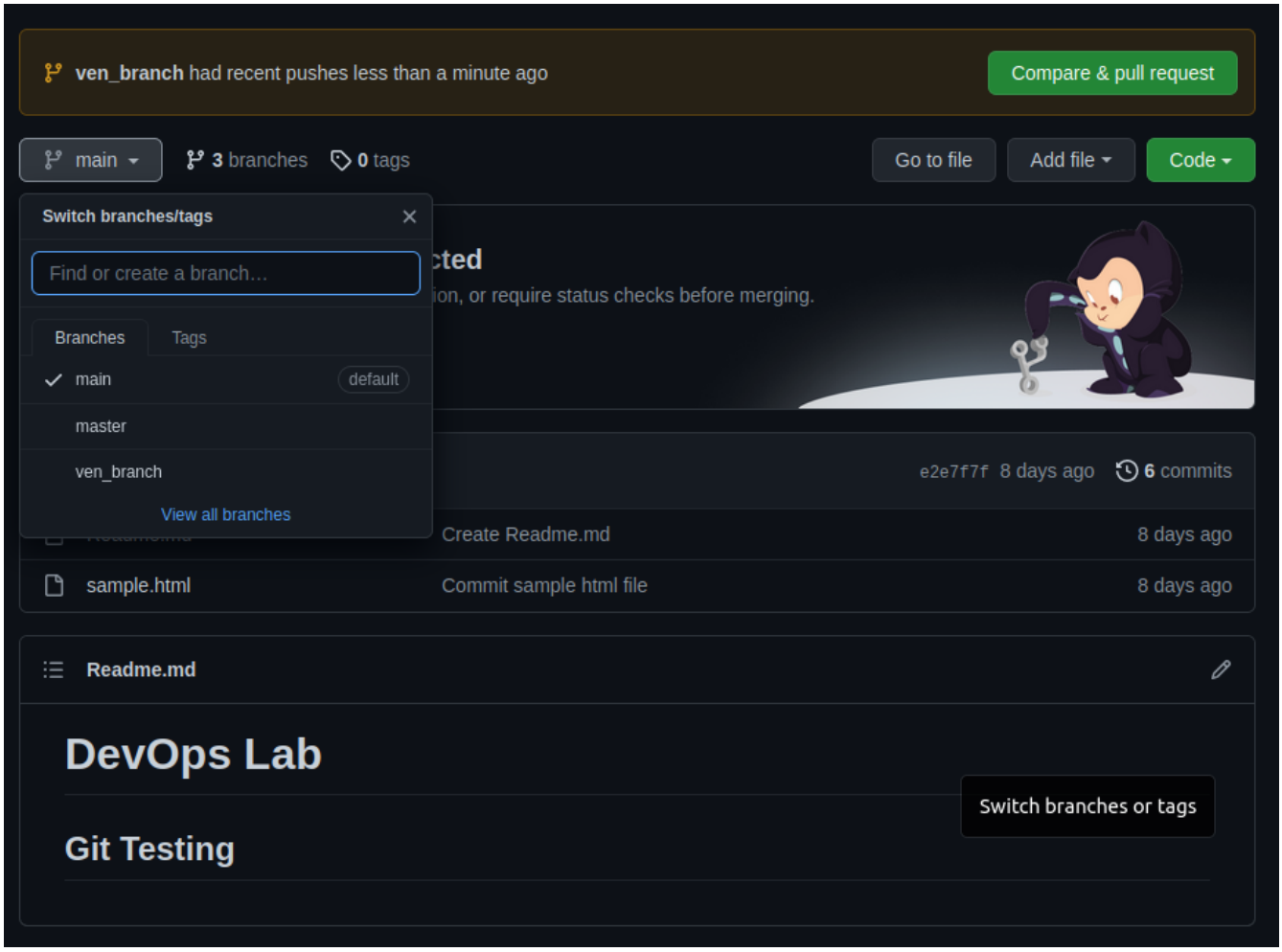
11) Committing the file in the new branch

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git
 add .
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git
 status
On branch ven_branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   sample.html

purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git
 commit -m "File sample created in new branch"
[ven_branch a46e0a7] File sample created in new branch
 1 file changed, 8 insertions(+)
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$
```

12) Git push the branch

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git
 push origin ven_branch
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), 418 bytes | 418.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'ven_branch' on GitHub by visiting:
remote:      https://github.com/PurpleVen/DevopsTest/pull/new/ven_branch
remote:
To https://github.com/PurpleVen/DevopsTest.git
 * [new branch]      ven_branch -> ven_branch
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$
```

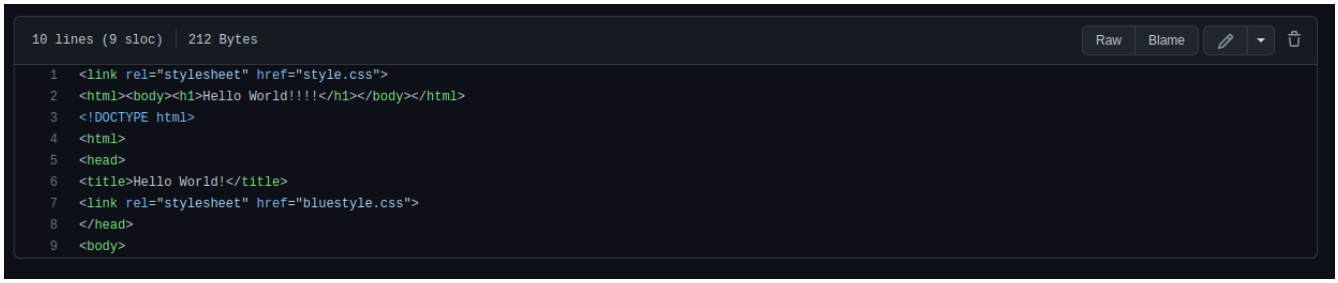
In GitHub the following changes and a new branch is visible



13) Git checkout to switch branches and git merge in the new branch with the main branch

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git
 checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git
 merge ven_branch
Updating e2e7f7f..a46e0a7
Fast-forward
 sample.html | 8 ++++++++
 1 file changed, 8 insertions(+)
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$
```

In the branch the files are merged



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

References:

1. Steps for Beginners: https://www.youtube.com/watch?v=9FOuyNt0V8I
2. For token usage as on 13 August 2022:
https://www.youtube.com/watch?v=W9zTttHeoHk
3. For forking and branching and merging use
https://spoken-tutorial.org/tutorial-search/?search_foss=Git&search_language=English

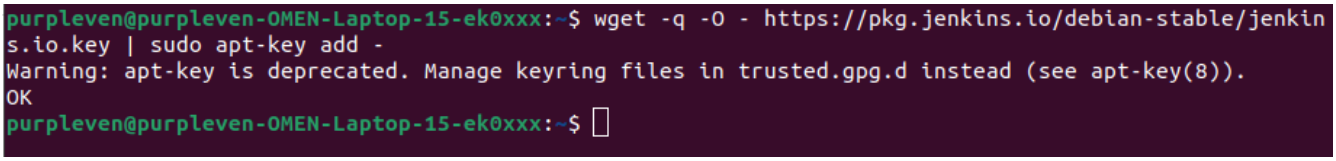
EXPERIMENT 2

Jenkins

Date of Experiment: 12/09/2022

Output:

Add the repository key to the system:



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

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable
 binary/ > /etc/apt/sources.list.d/jenkins.list'
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$
```

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

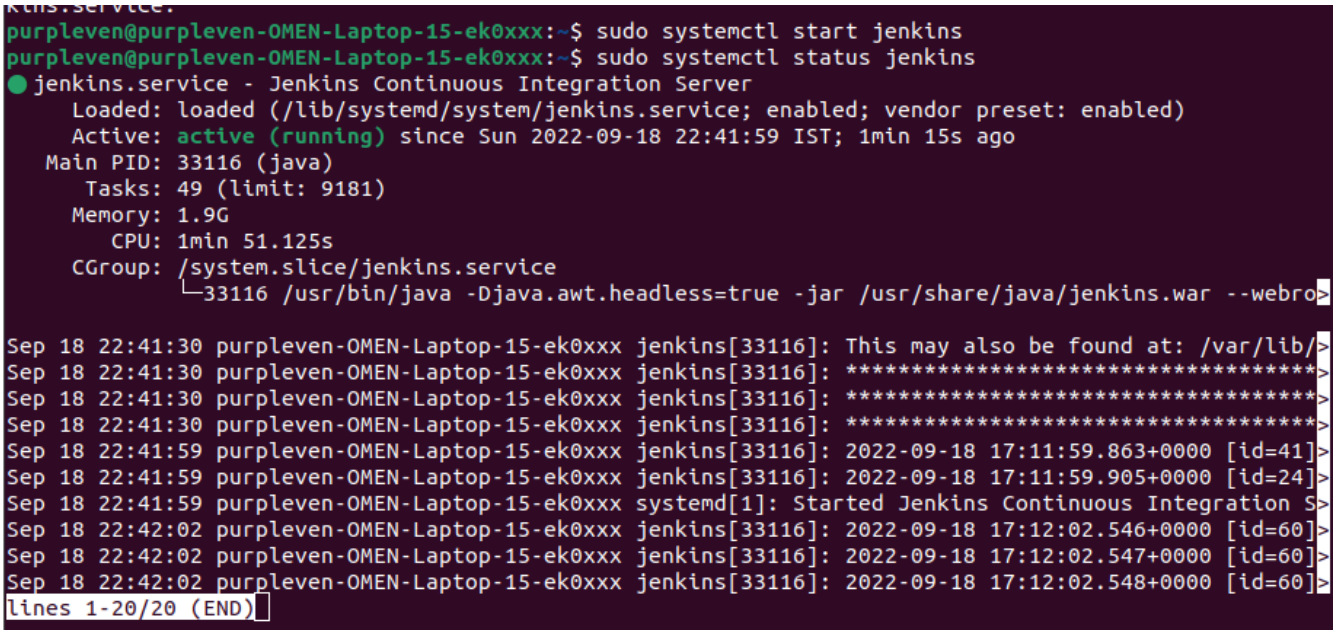
```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ sudo apt update
Hit:2 http://apt.postgresql.org/pub/repos/apt jammy-pgdg InRelease
Hit:3 https://apt.releases.hashicorp.com jammy InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:7 https://pkg.jenkins.io/debian-stable binary/ Release [2,044 B]
Get:8 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:9 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:10 https://pkg.jenkins.io/debian-stable binary/ Packages [23.0 kB]
Fetched 25.9 kB in 3s (10.3 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
N: Skipping acquire of configured file 'main/binary-i386/Packages' as repository 'http://apt.postgres
ql.org/pub/repos/apt jammy-pgdg InRelease' doesn't support architecture 'i386'
W: http://pkg.jenkins.io/debian-stable/binary/Release.gpg: Key is stored in legacy trusted.gpg keyrin
g (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$
```

Finally, we'll install Jenkins and its dependencies.

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ sudo apt install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  jenkins
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 90.7 MB of archives.
After this operation, 93.6 MB of additional disk space will be used.
Get:1 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.361.1 [90.7 MB]
Fetched 90.7 MB in 53s (1,702 kB/s)
Selecting previously unselected package jenkins.
(Reading database ... 222383 files and directories currently installed.)
Preparing to unpack .../jenkins_2.361.1_all.deb ...
Unpacking jenkins (2.361.1) ...
Setting up jenkins (2.361.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jen
kins.service.
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$
```

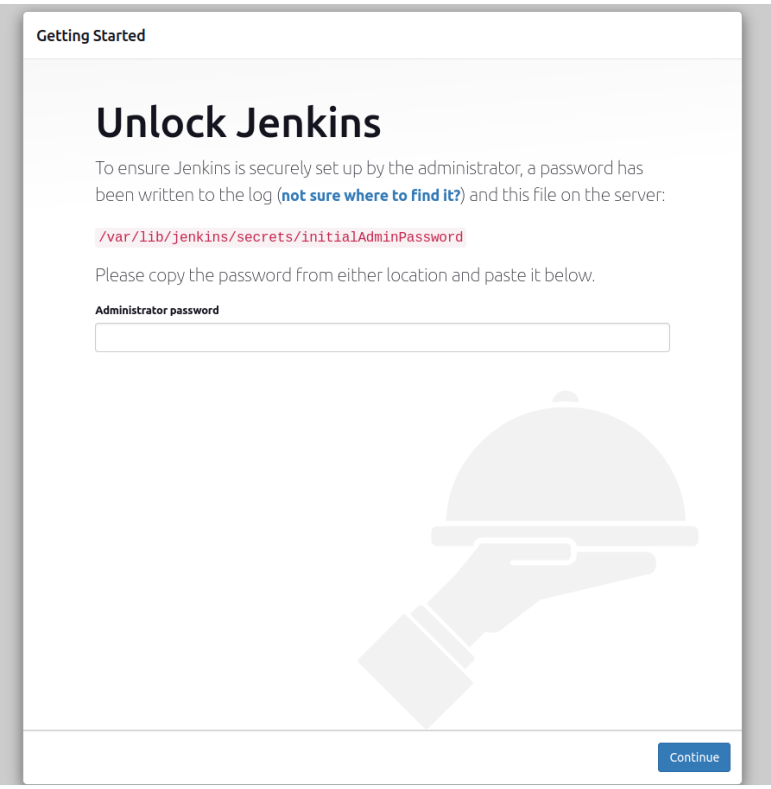
Step 2 — Starting Jenkins

Let’s start Jenkins by using systemctl

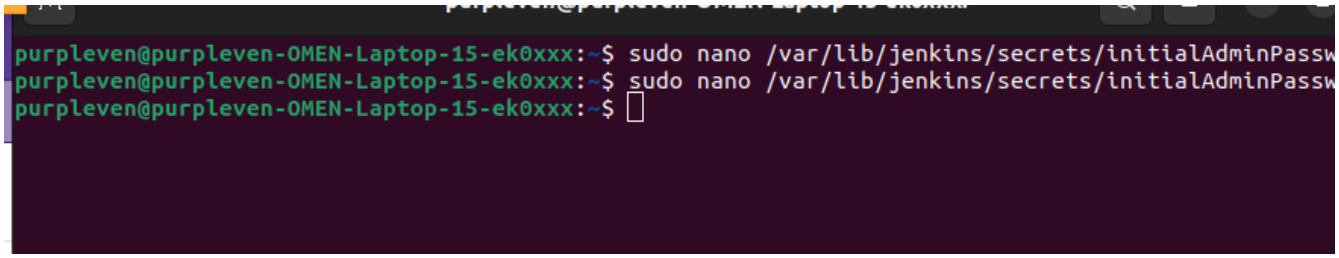


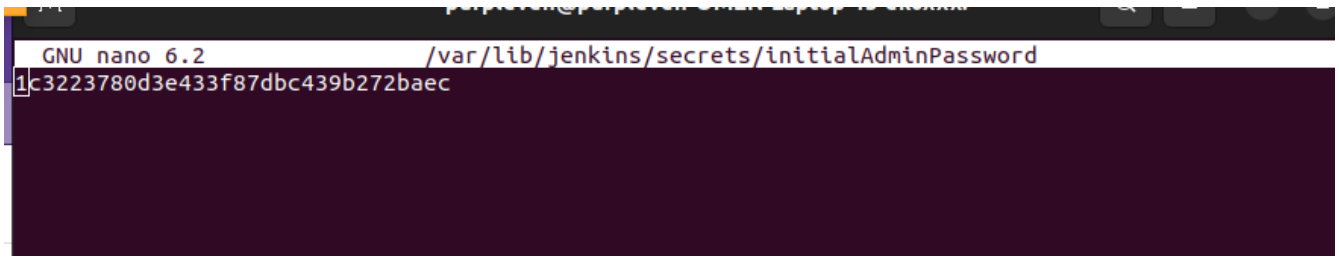
Step 3 — Setting Up Jenkins

To set up your installation, visit Jenkins on its default port, 8080, using your server domain name or IP address: http://*192.168.43.69:8080/*

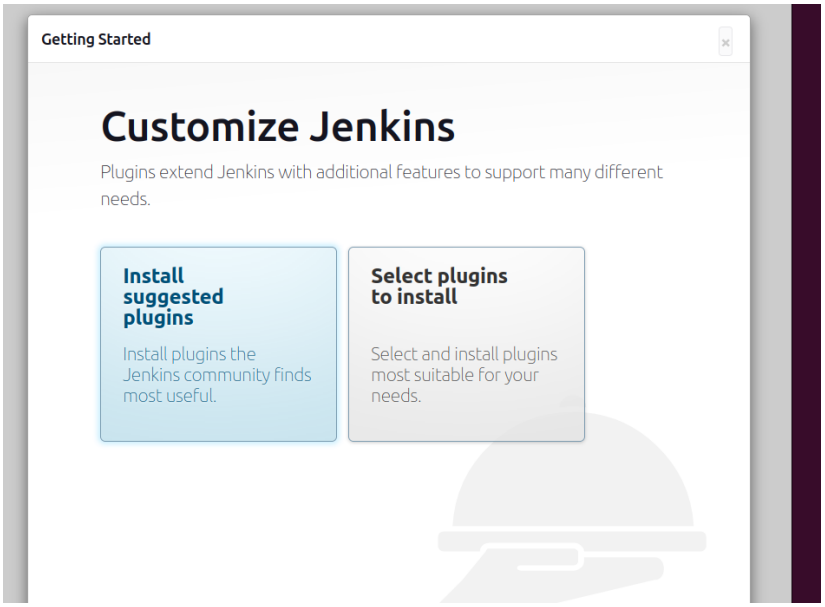


In the terminal window, use the nano command to display the password

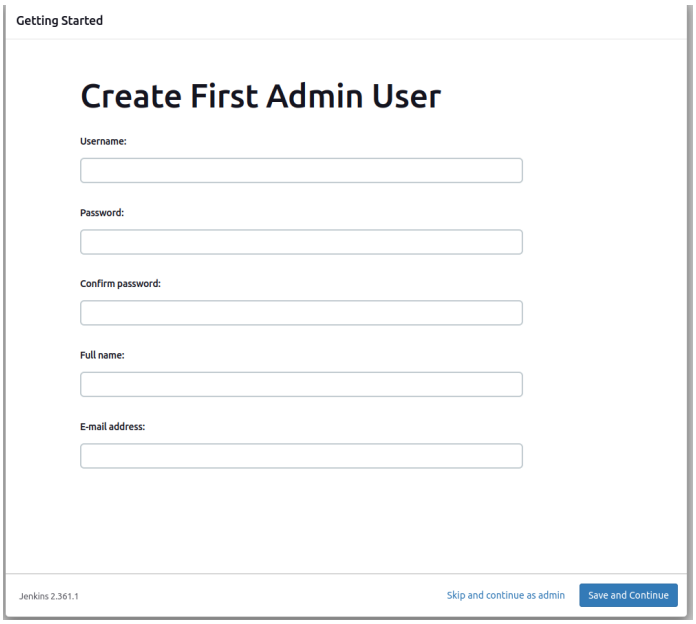




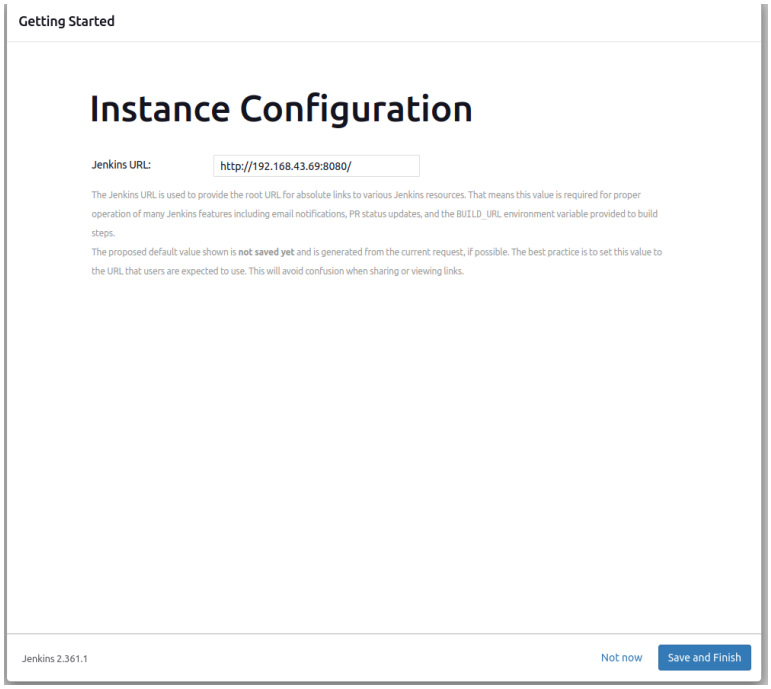
The next screen presents the option of installing suggested plugins



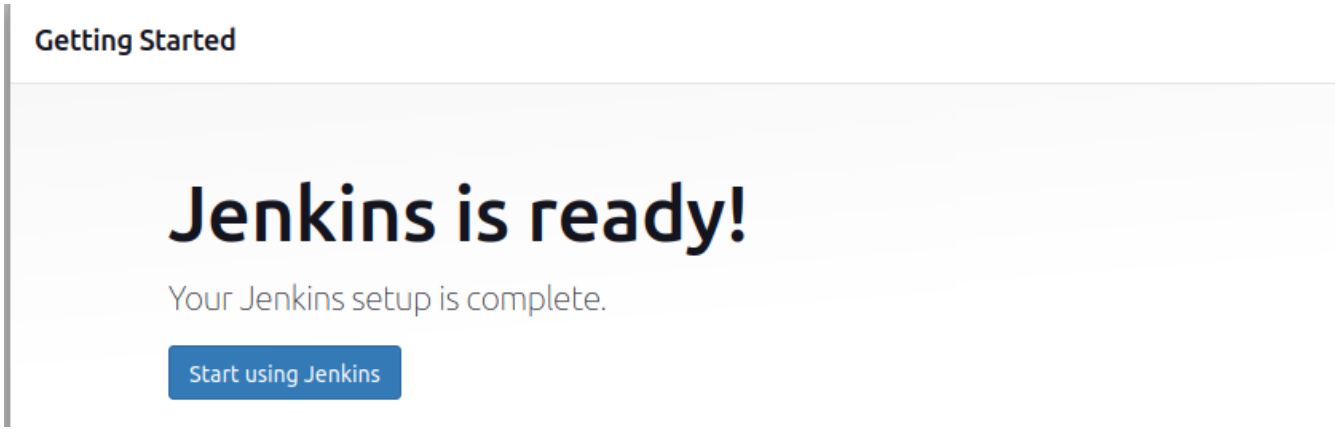
Enter the name and password for your user



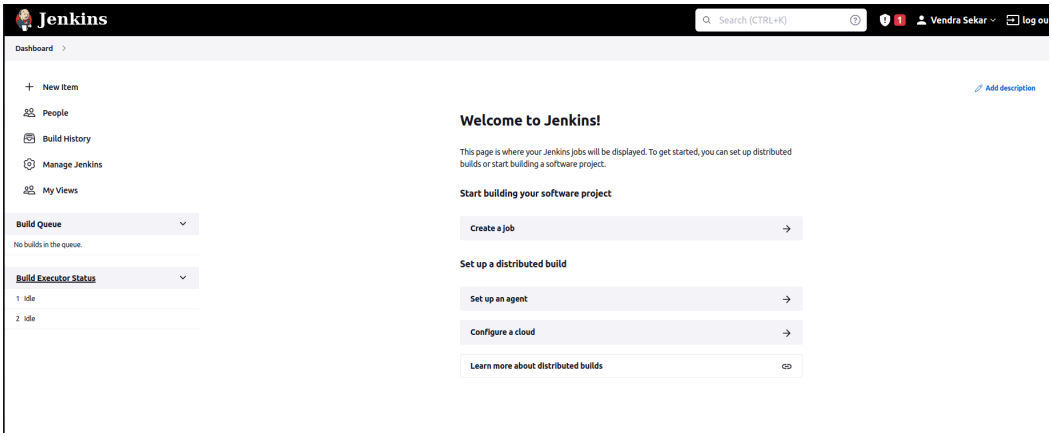
You'll receive an Instance Configuration page that will ask you to confirm the preferred URL for your Jenkins instance.



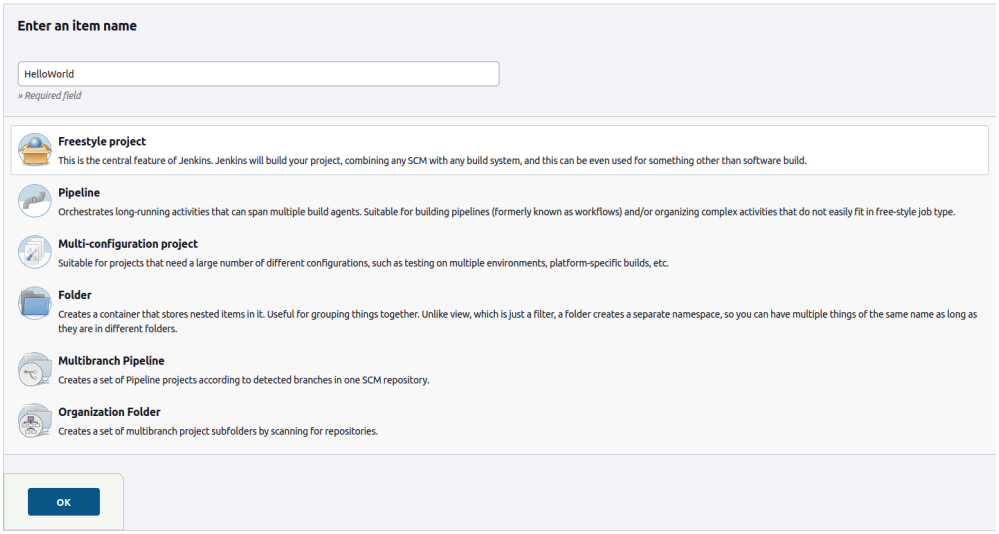
After confirming the appropriate information, click Save and Finish. You'll receive a confirmation page confirming that "Jenkins is Ready!"



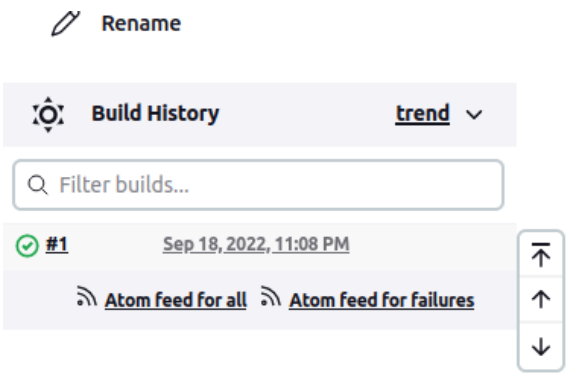
Click Start using Jenkins to visit the main Jenkins dashboard



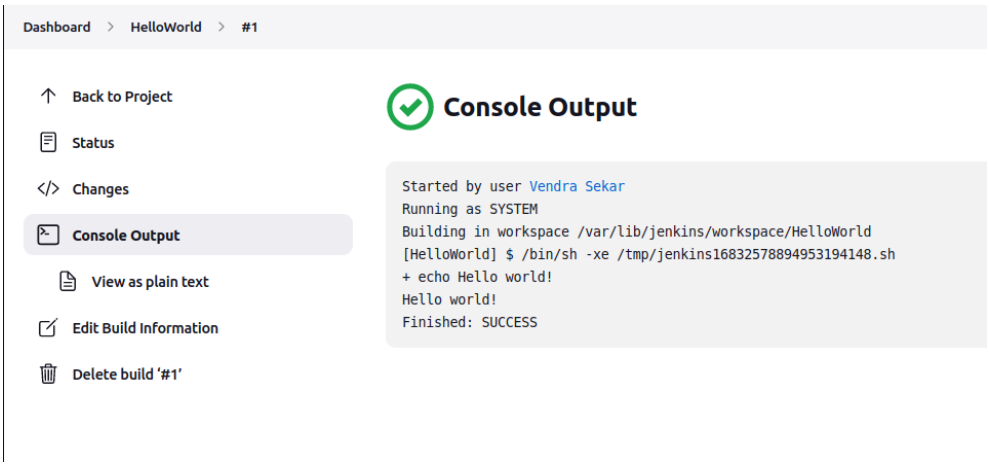
Click on new item and create a new project and use the freestyle project plugin



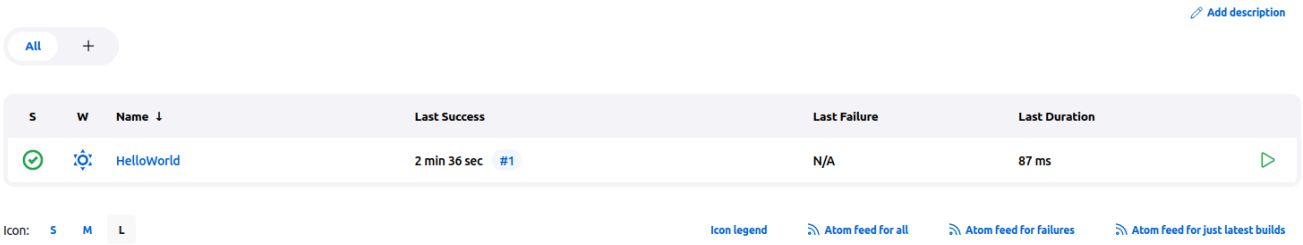
Use the execute shell feature and write a code, now click on build now



Now check on console output

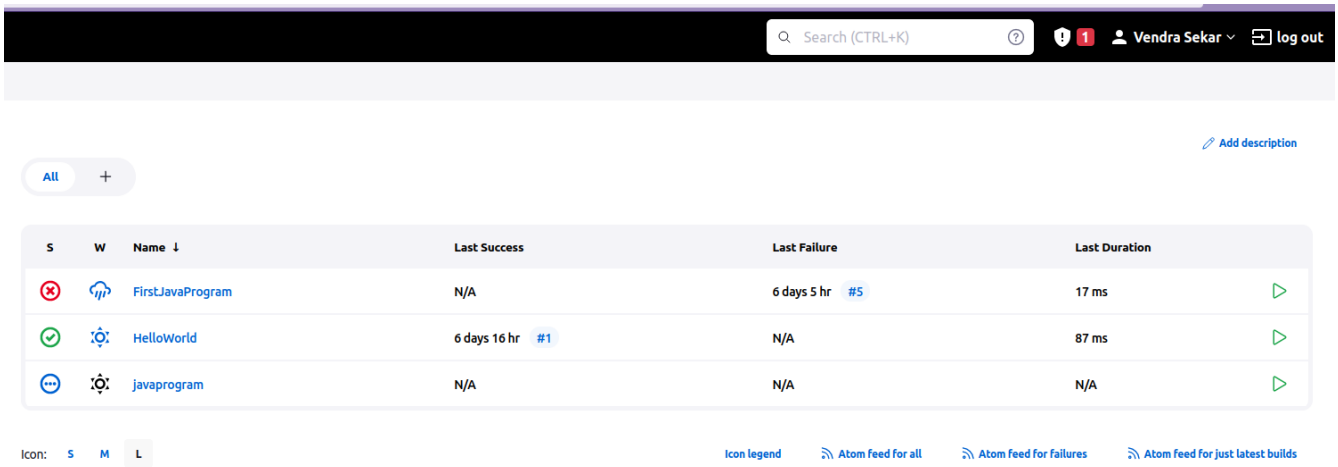


The output is successful

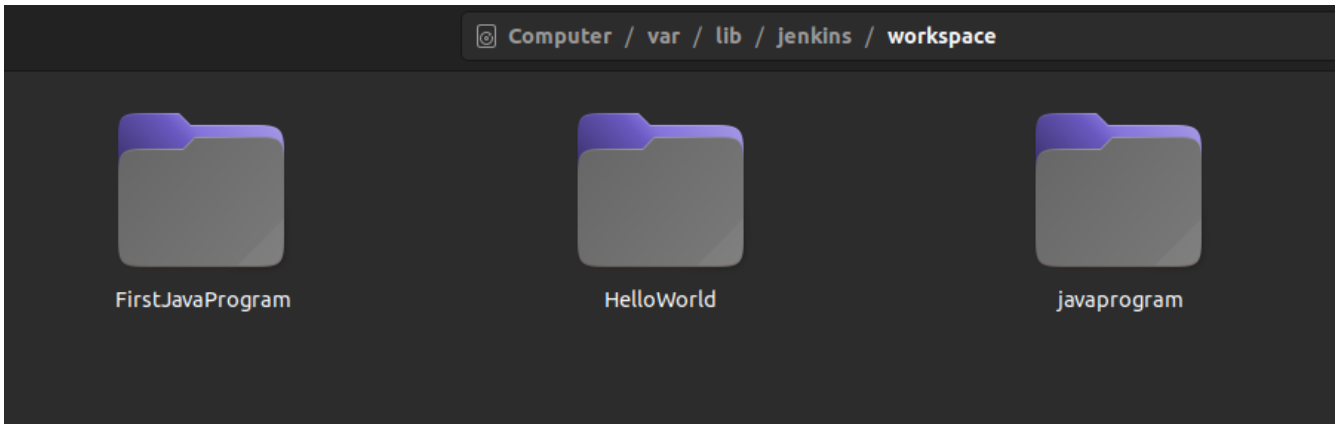


2. Exercise to create and run a java program using terminal

Create a new freestyle project



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



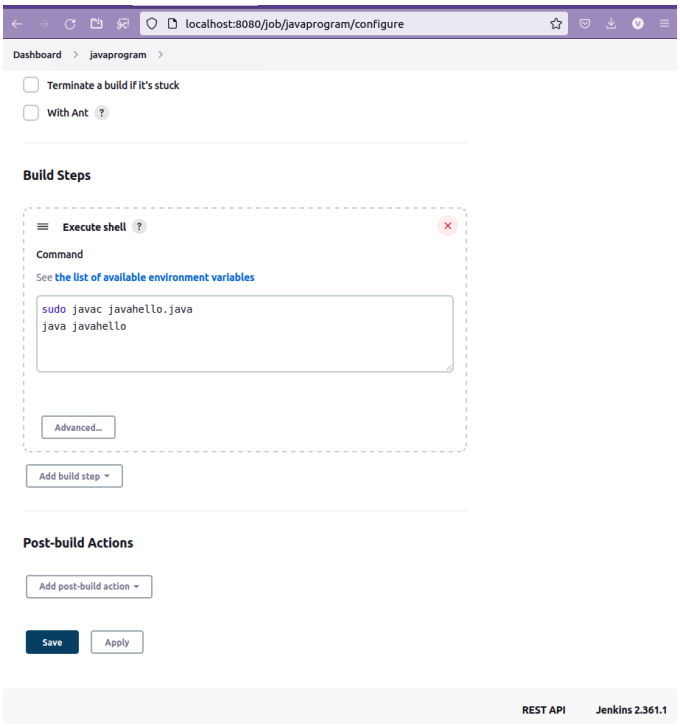
Write a java program

```
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$
```

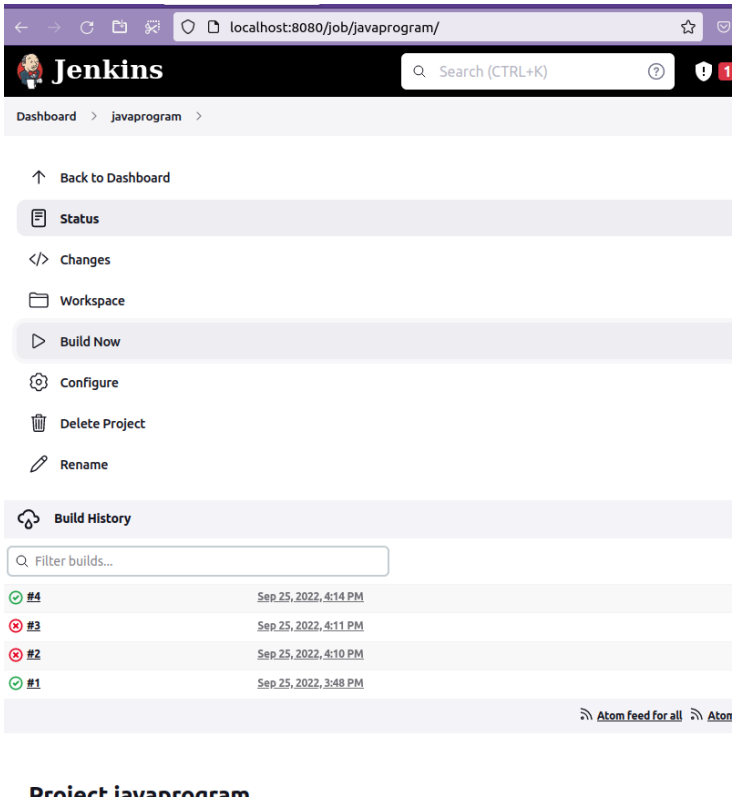
Compile the java program

```
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$
```

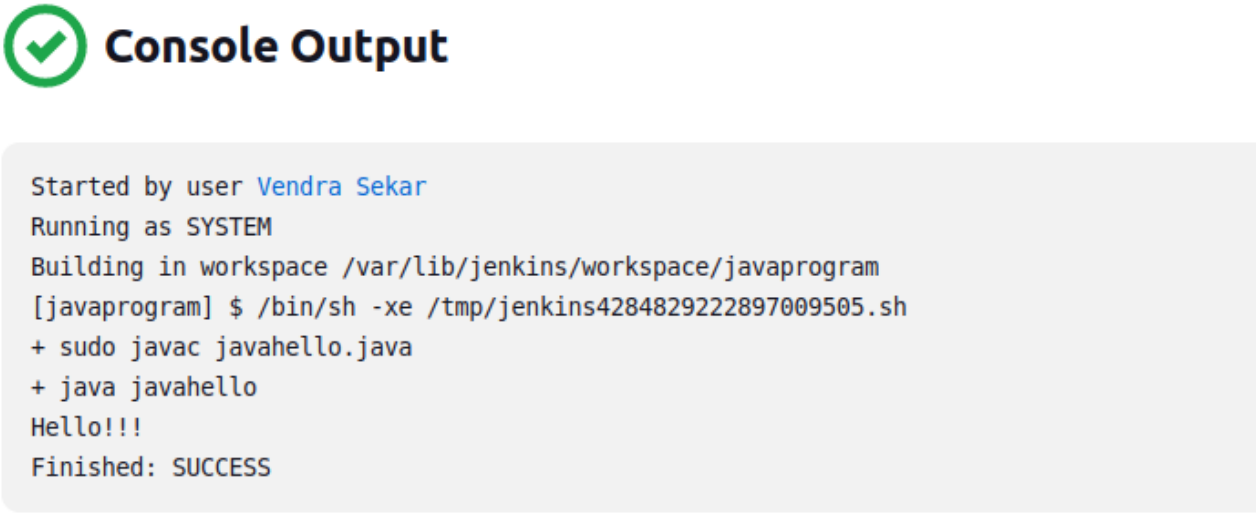
Configure the java program and add the two lines in the build step



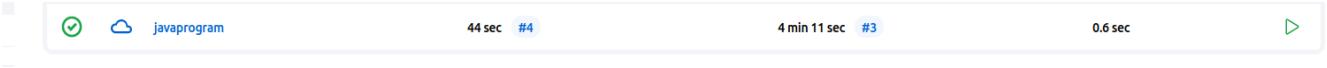
Click on build now



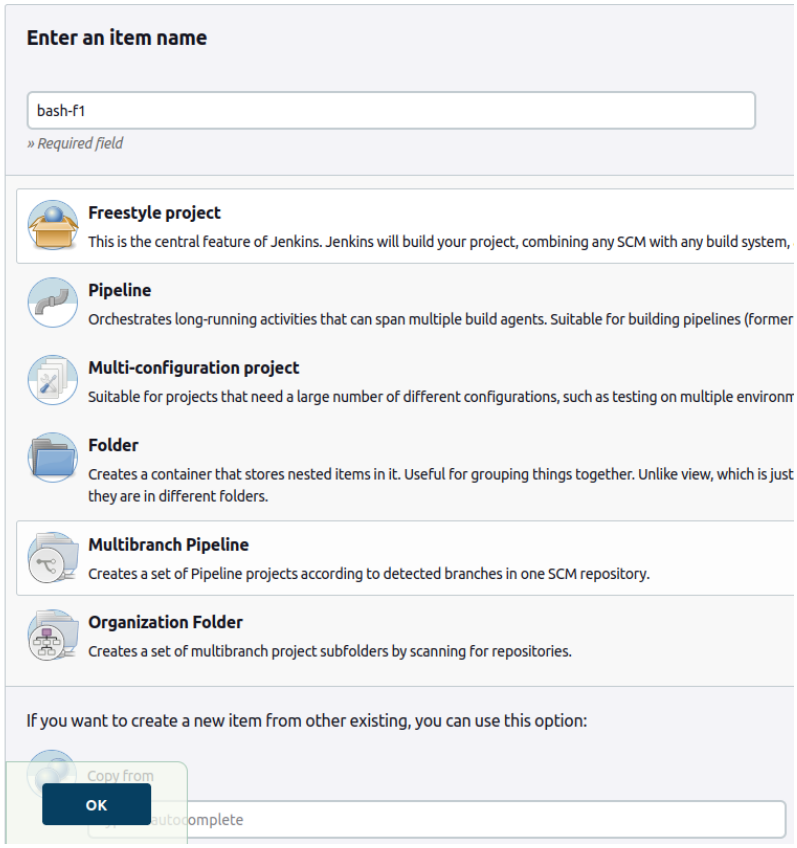
Check the output

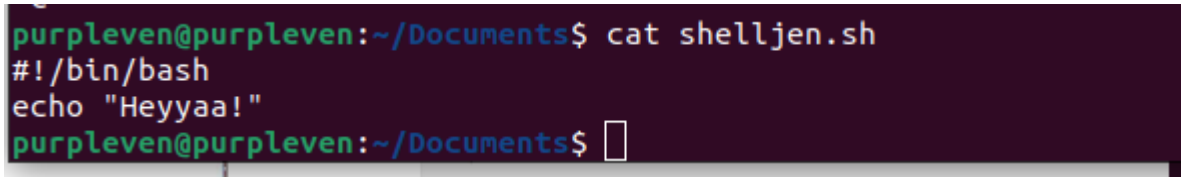


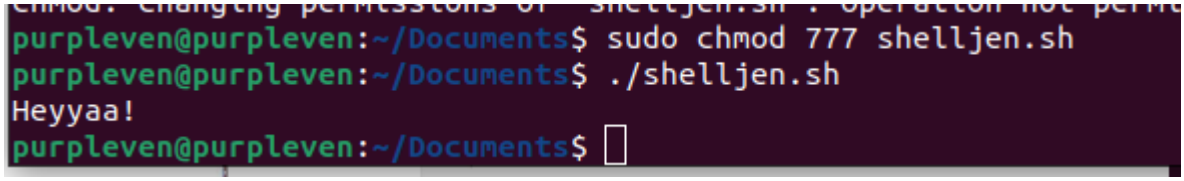
The java program is successfully demonstrated on jenkins

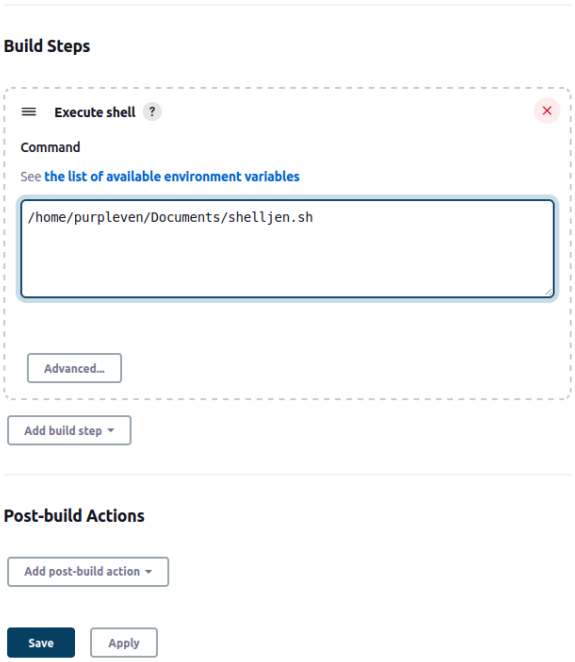


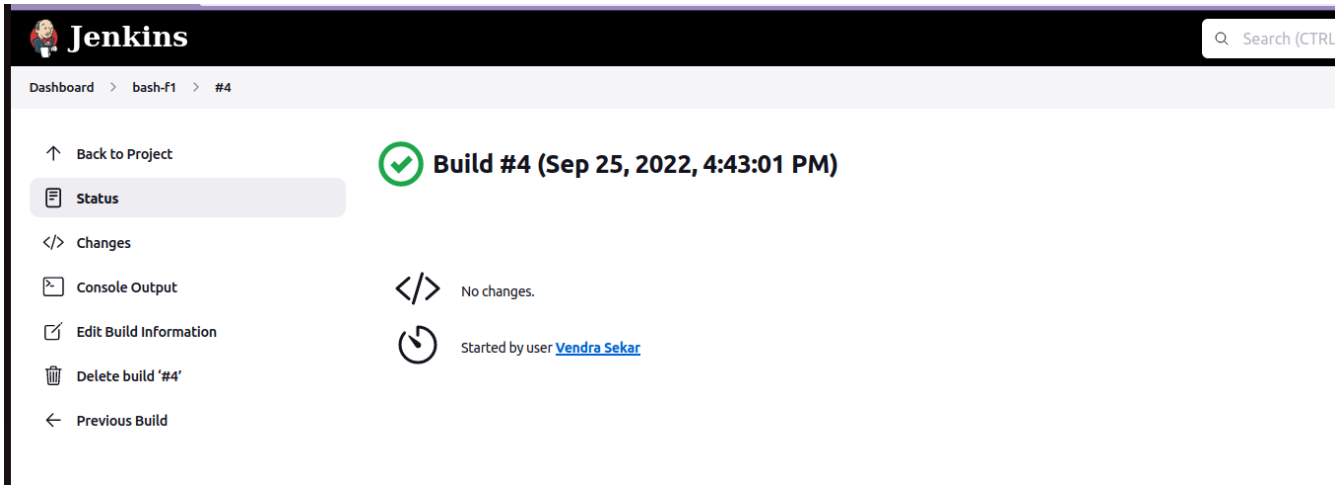
3. Exercise to create a shell script and version it in Jenkins

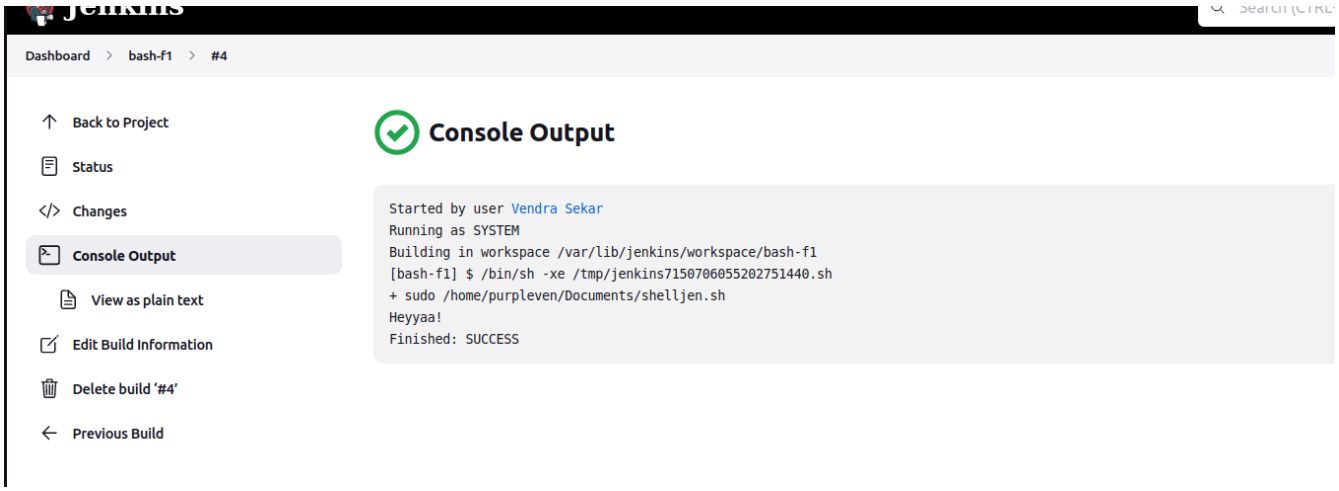




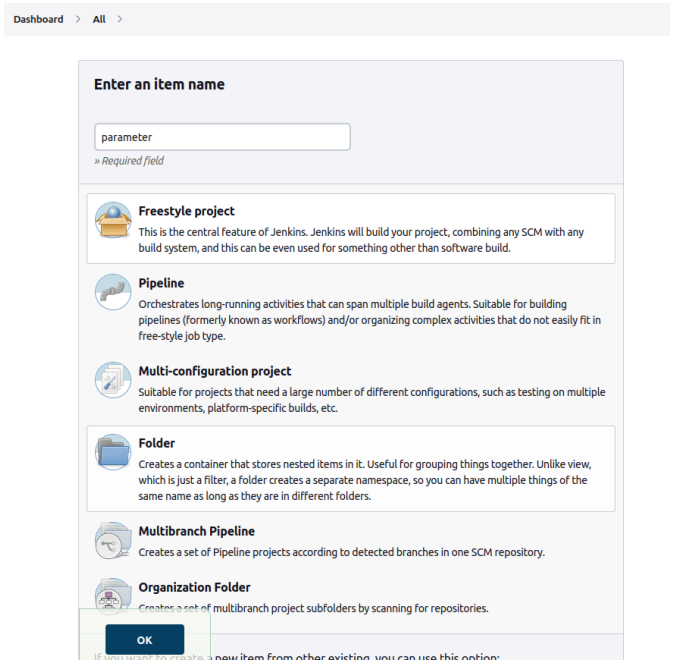


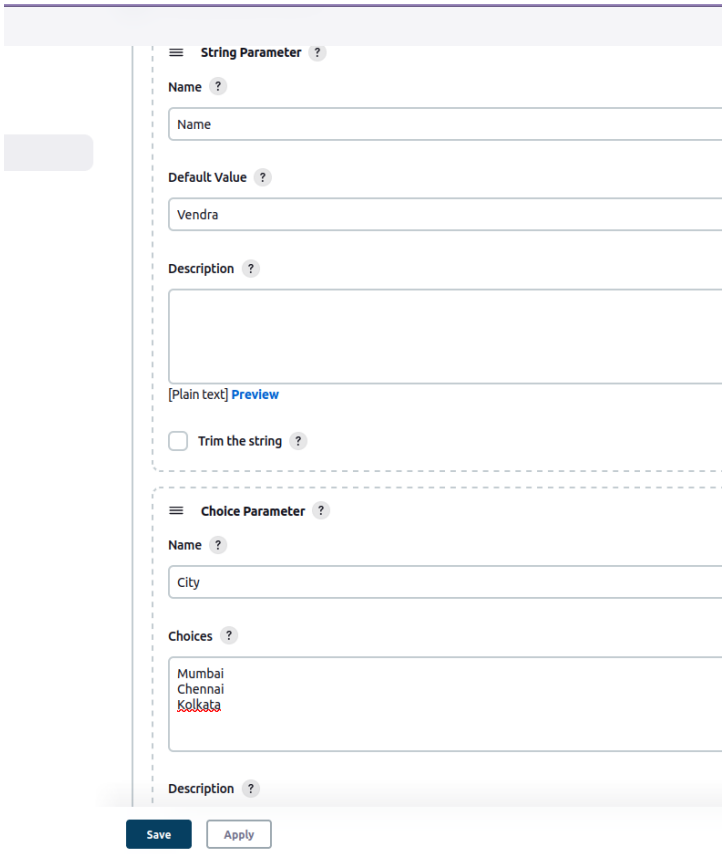

Build Steps
Execute shell
Command
See the list of available environment variables
/home/purpleven/Documents/shelljen.sh
Advanced...
Add build step
Post-build Actions
Add post-build action
Save
Apply

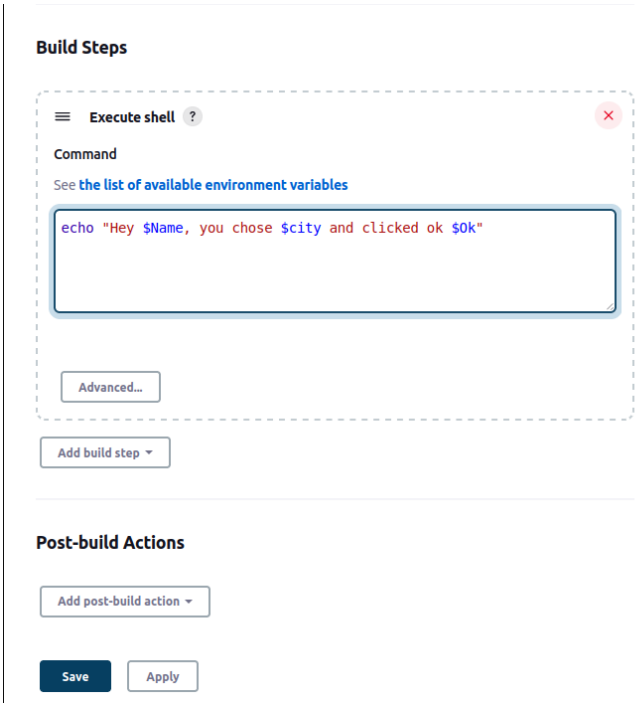

Jenkins
Search (CTRL
Dashboard
bash-f1
#4
Back to Project
Status
Changes
Console Output
Edit Build Information
Delete build '#4'
Previous Build
Build #4 (Sep 25, 2022, 4:43:01 PM)
No changes.
Started by user Vendra Sekar

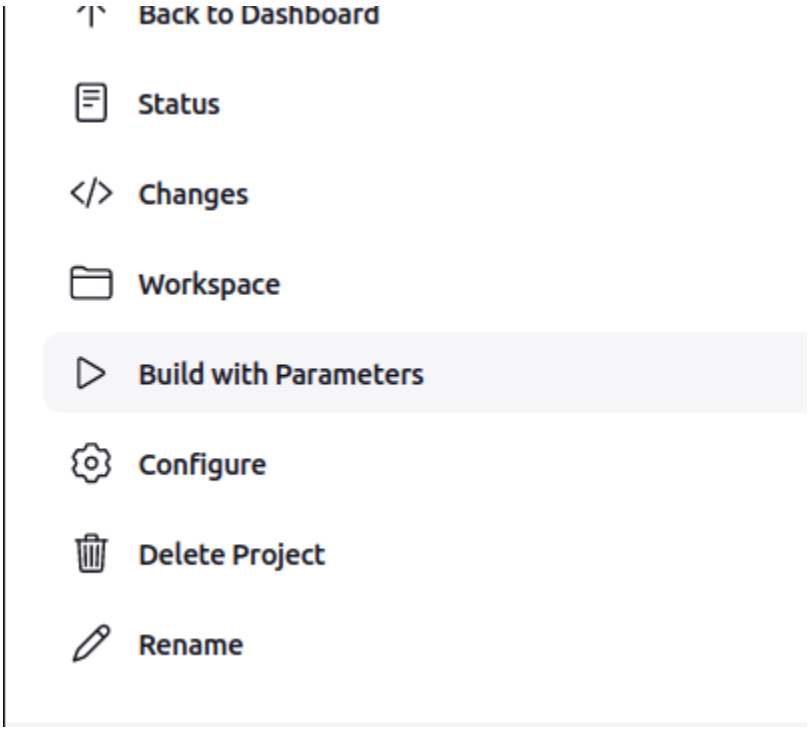

Dashboard
bash-f1
#4
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#4'
Previous Build
Console Output
Started by user Vendra Sekar
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/bash-f1
[bash-f1] $ /bin/sh -xe /tmp/jenkins7150706055202751440.sh
+ sudo /home/purpleven/Documents/shelljen.sh
Heyyaa!
Finished: SUCCESS

4. Exercise to Create a parameterized project






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


Back to Dashboard
Status
Changes
Workspace
Build with Parameters
Configure
Delete Project
Rename

Project parameter

This build requires parameters:

Name

Vendra

City

Mumbai

Ok

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 Vendra Sekar
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/parameter
[parameter] $ /bin/sh -xe /tmp/jenkins6214079698661515166.sh
+ echo Hey Vendra, you chose  and clicked ok true
Hey Vendra, you chose  and clicked ok true
Finished: SUCCESS
```

5. Exercise: create a maven Project

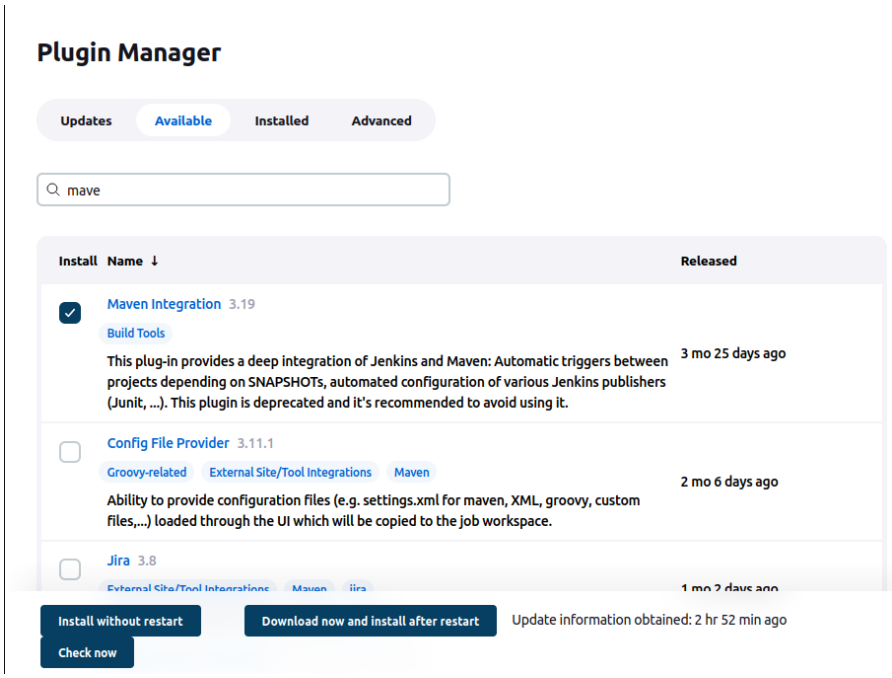
Install maven

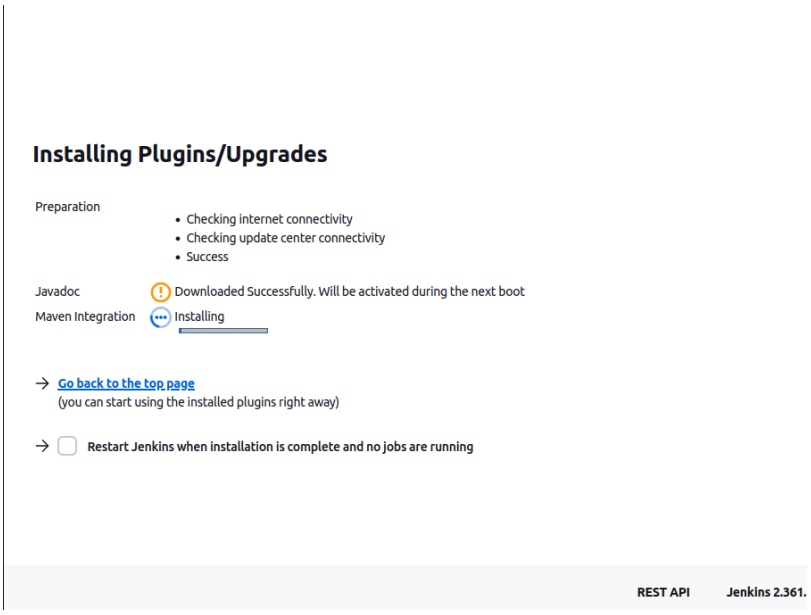
```
purpleven@purpleven:~$ sudo apt install maven
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libaopalliance-java libapache-pom-java libatinject-jsr330-api-java
  libcdi-api-java libcommons-cli-java libcommons-io-java libcommons-lang3-java
  libcommons-parent-java libgeronimo-annotation-1.3-spec-java
  libgeronimo-interceptor-3.0-spec-java libguava-java libguice-java
  libhawtjni-runtime-java libjansi-java libjansi-native-java libjsr305-java
  libmaven-parent-java libmaven-resolver-java libmaven-shared-utils-java
  libmaven3-core-java libplexus-cipher-java libplexus-classworlds-java
  libplexus-component-annotations-java libplexus-interpolation-java
  libplexus-sec-dispatcher-java libplexus-utils2-java libsisu-inject-java
  libsisu-plexus-java libslf4j-java libwagon-file-java
  libwagon-http-shaded-java libwagon-provider-api-java
Suggested packages:
```

```
purpleven@purpleven:~$ 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-amd
64
Default locale: en_IN, platform encoding: UTF-8
OS name: "linux", version: "5.15.0-47-generic", arch: "amd64", family: "unix"
purpleven@purpleven:~$
```

Create a maven project
Eg: https://github.com/devopshint/java-app-with-maven/tree/main/my-app

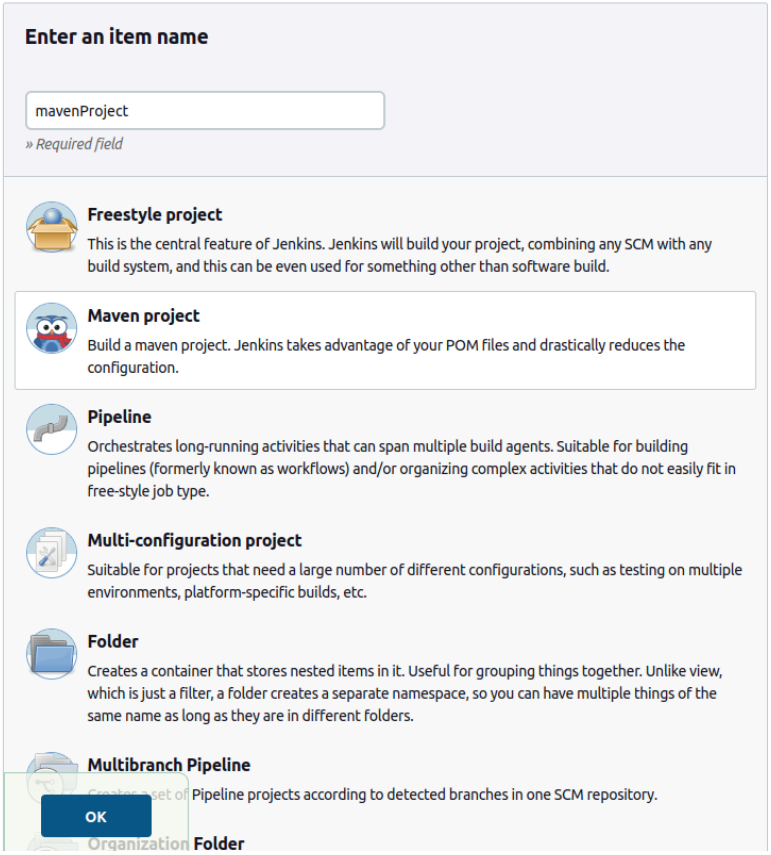
Install maven integration plugin from the plugin manager

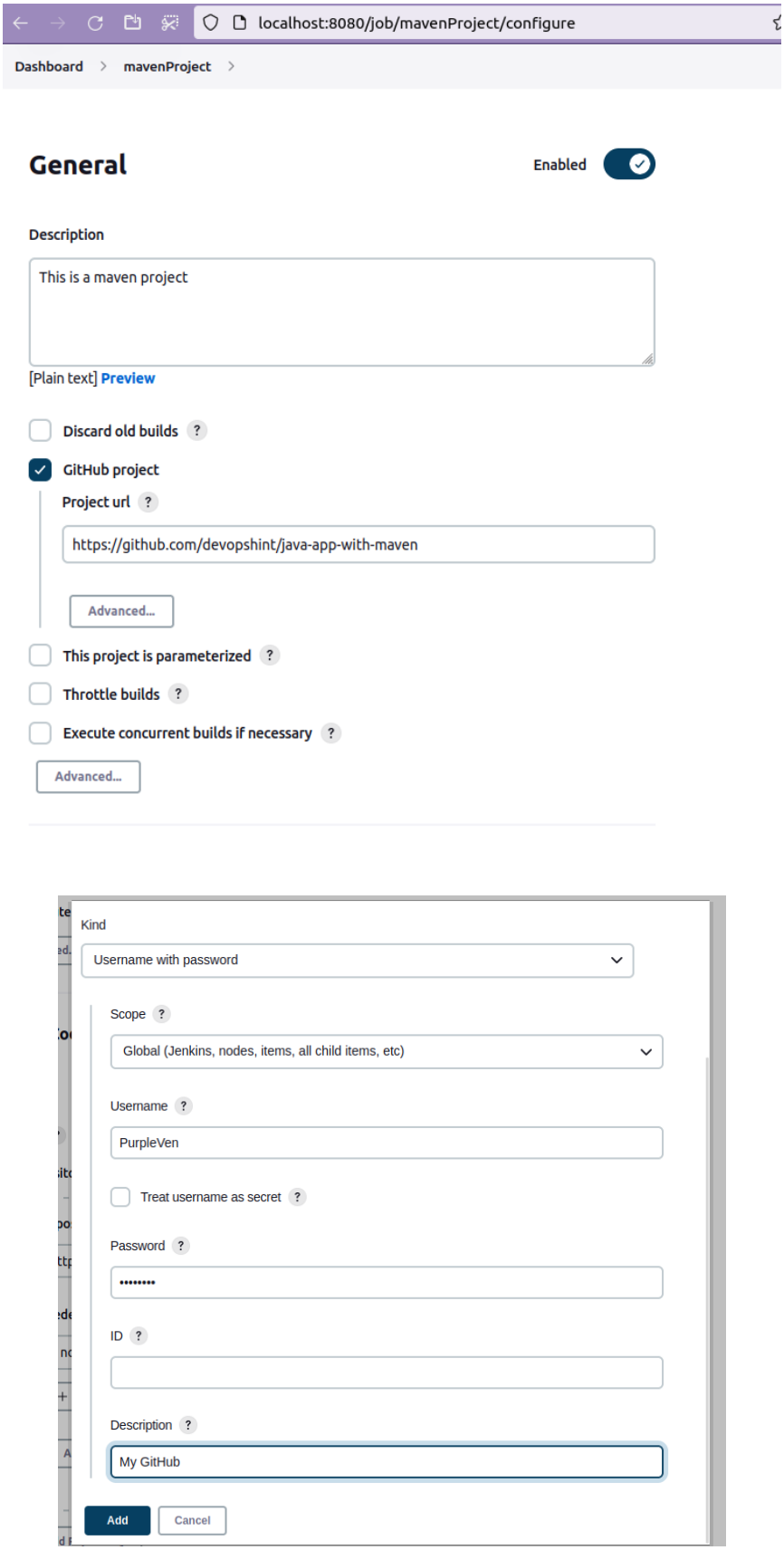


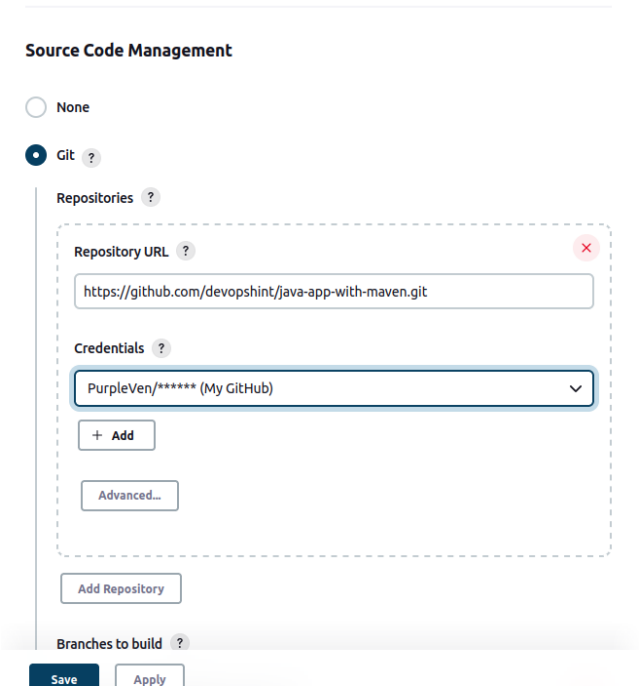


Restart jenkins

Create a new project




localhost:8080/job/mavenProject/configure
Dashboard
mavenProject
General
Enabled
Description
This is a 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...
Kind
Username with password
Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
PurpleVen
Treat username as secret
Password
ID
Description
My GitHub
Add
Cancel

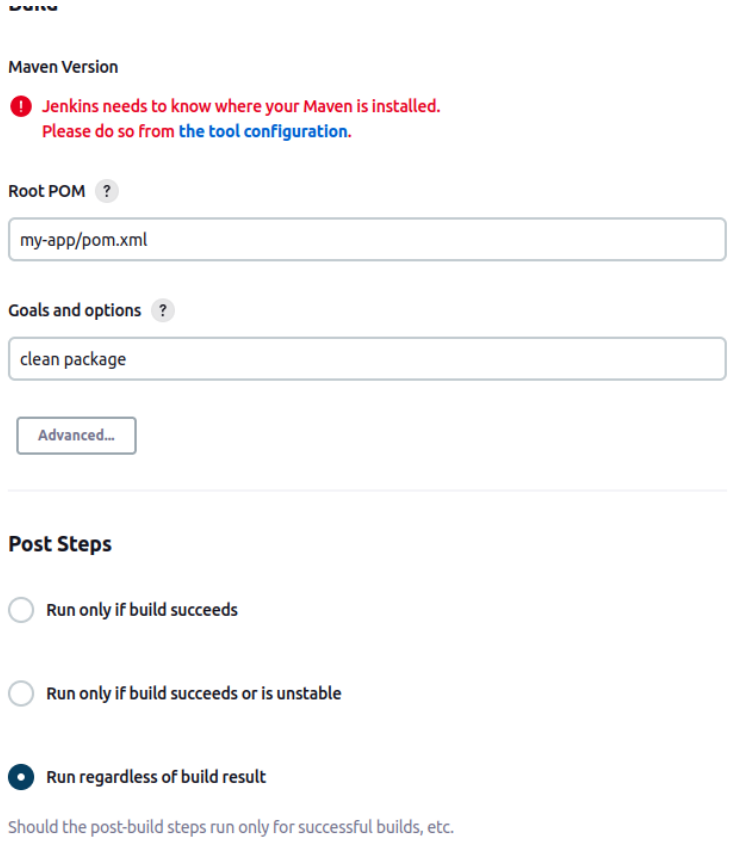


Build Triggers

- [x] 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
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

Build Environment

- [x] Delete workspace before build starts
  - Advanced...
- [ ] 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

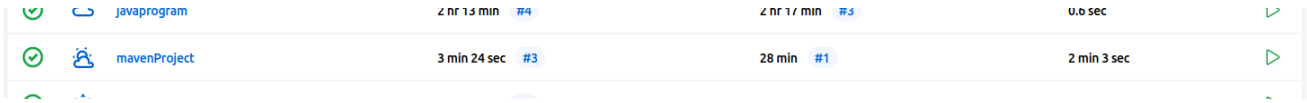


Console Output

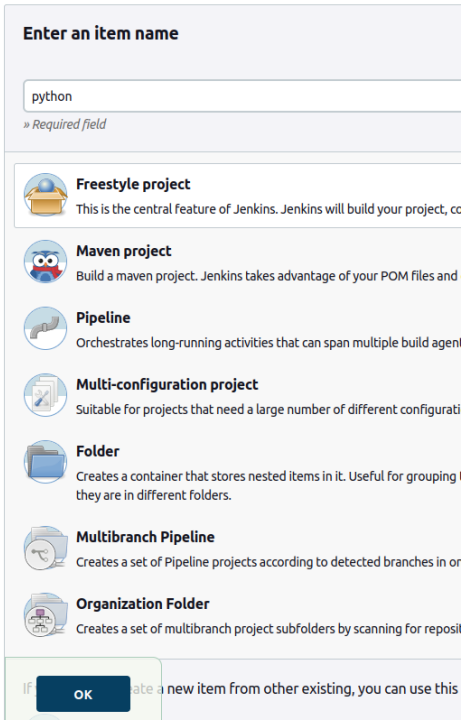
```
Started by user Vendra Sekar
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/mavenProject
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] Done
The recommended git tool is: NONE
using credential 40a1e789-e70b-4162-bfbe-a4c05f55392c
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/*:ref
 > 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
```

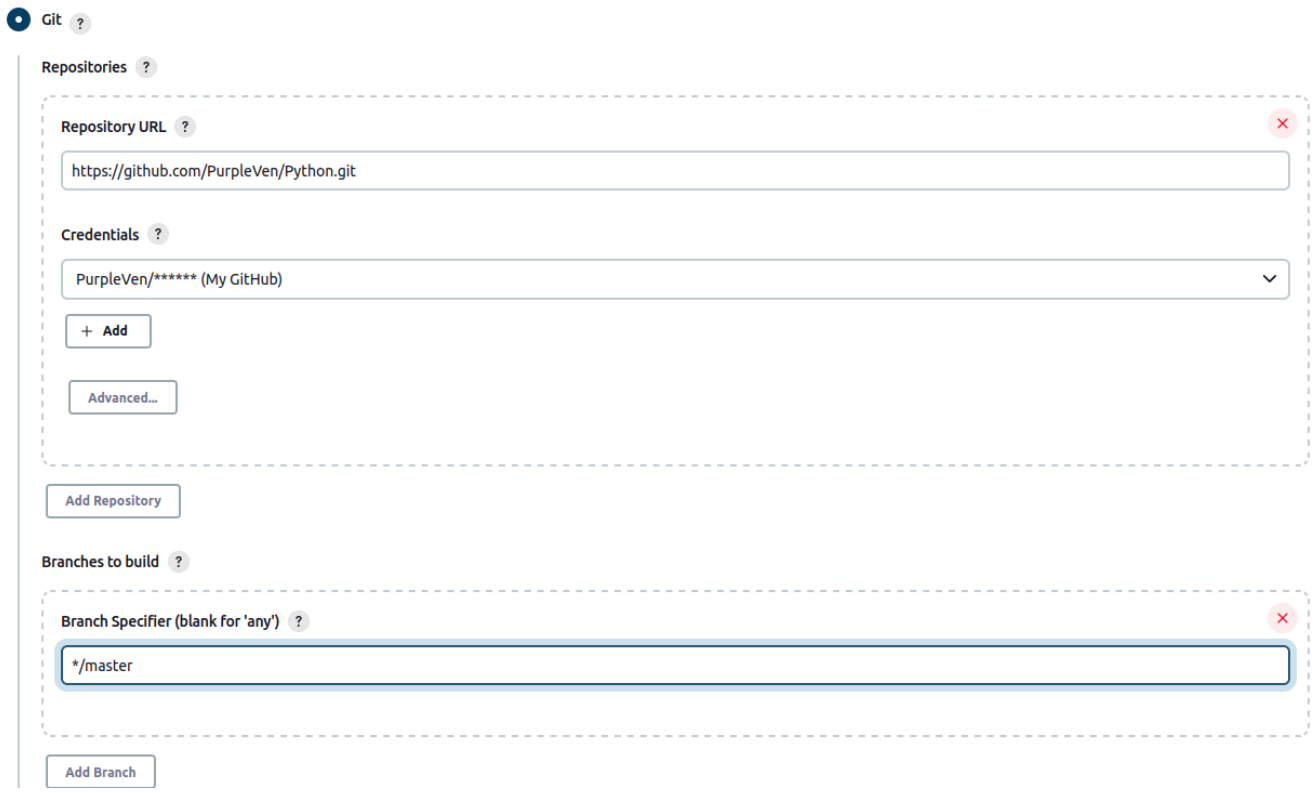
```
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-compress/1.11/commons-compress-1.11.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.1.1/maven-archiver-3.1.1.jar (24 kB at 46 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/iq80/snappy/snappy/0.4/snappy-0.4.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/2.7.1/plexus-io-2.7.1.jar (86 kB at 152 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/tukaani/xz/1.5/xz-1.5.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.0.1/maven-shared-utils-3.0.1.jar (154 kB at 228 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-archiver/3.4/plexus-archiver-3.4.jar (187 kB at 263 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/iq80/snappy/snappy/0.4/snappy-0.4.jar (58 kB at 61 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-compress/1.11/commons-compress-1.11.jar (426 kB at 429 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/tukaani/xz/1.5/xz-1.5.jar (100 kB at 82 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:  01:53 min
[INFO] Finished at: 2022-09-25T18:27:09+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



6. Exercise: to run a python simple program using freestyle





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 INSTALLATION

Experiment 3: Docker Installation & basic commands of Docker

Part A:

Steps for Installing Docker:

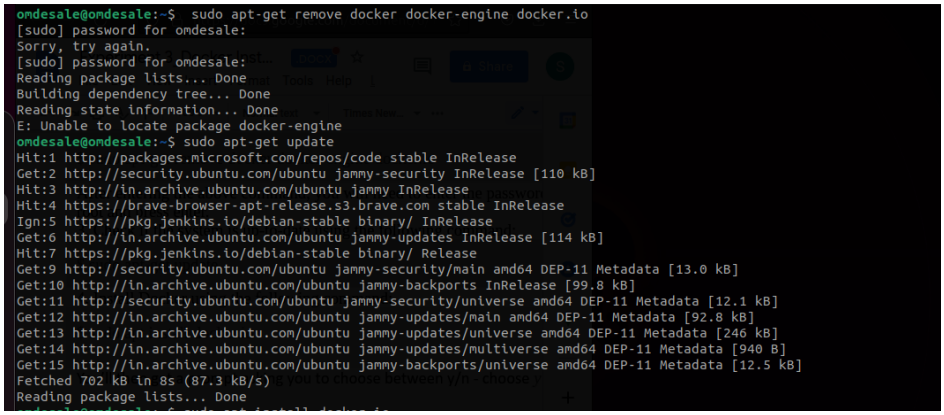
1. Open the terminal on Ubuntu.
2. Remove any Docker files that are running in the system, using the following command:

```
$ sudo apt-get remove docker docker-engine docker.io
```

After entering the above command, you will need to enter the password of the root and press enter.

3. Check if the system is up-to-date using the following command:

```
$ sudo apt-get update
```



4. Install Docker using the following command:

```
$ sudo apt install docker.io
```

You'll then get a prompt asking you to choose between y/n - choose *y*

```
omdesale@omdesale:~$ sudo apt install docker.io
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 libbs2b0 libflite1
  libgstreamer-plugins-bad1.0-0 liblilv-0-0 librubberband2
  libserd-0-0 libsord-0-0 libsratom-0-0 libvidstab1.1 libzimg2
  pocketsphinx-en-us systemd-hwe-hwdb
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  bridge-utils containerd pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite
  debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd docker.io pigz runc ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 80 not upgraded.
Need to get 65.3 MB of archives.
After this operation, 282 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 runc amd64 1.1.0-0ubuntu1 [4,087 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 containerd amd64 1.5.9-0ubuntu3 [27.0 MB]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 docker.io amd64 20.10.12-0ubuntu4 [34.0 MB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 65.3 MB in 2min 44s (397 kB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 180389 files and directories currently installed.
)
Preparing to unpack .../0-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
```

5. Install all the dependency packages using the following command:

```
$ sudo snap install docker
```

6. Before testing Docker, check the version installed using the following command:

```
$ docker –version
```

7. Pull an image from the Docker hub using the following command:

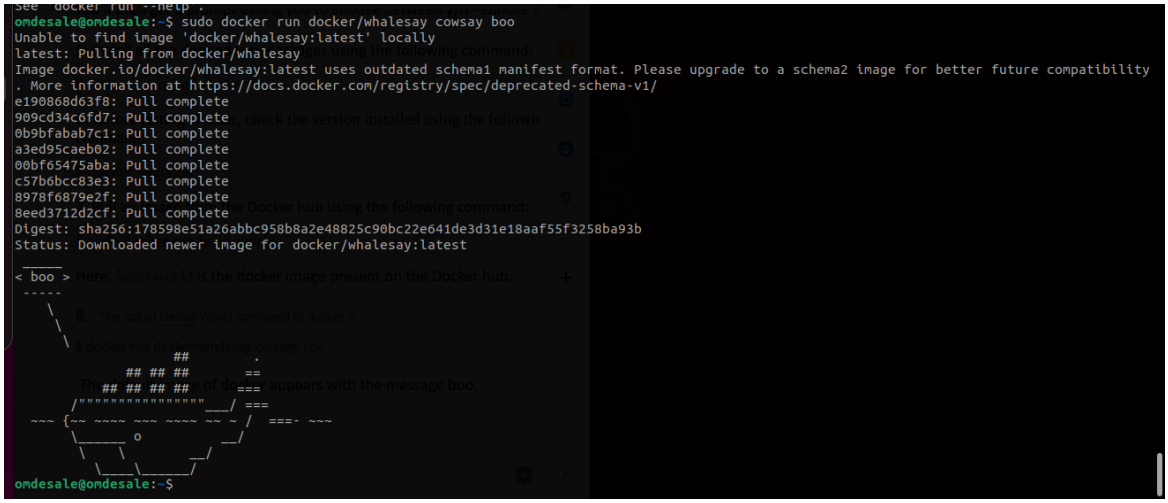
```
$ sudo docker run hello-world
```

Here, *hello-world* is the docker image present on the Docker hub.

```
omdesale@omdesale:~$ sudo snap install docker
docker 20.10.14 from Canonical** installed
omdesale@omdesale:~$ docker -version
docker: '-version' is not a docker command.
See 'docker --help'
omdesale@omdesale:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu4
omdesale@omdesale:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:62af9efd515a25f84961b70f973a798d2eca956b1b2b026d0a4a63a3b0b6a3f2
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/
```

8. The actual Hellow World command of docker is

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



The default image of docker appears with the message boo.

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

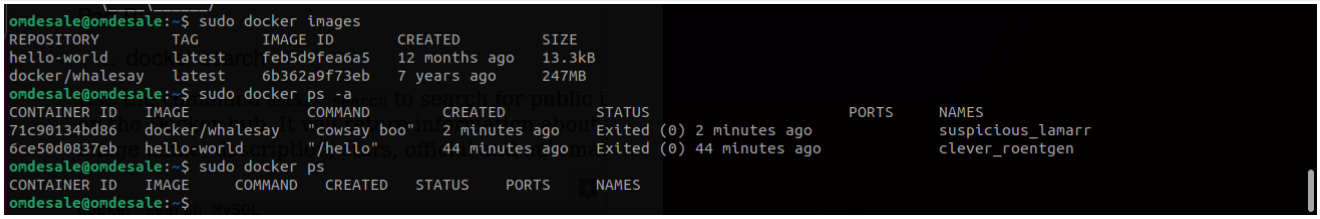
```
$ sudo docker images
```

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

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

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

```
$ sudo docker ps
```



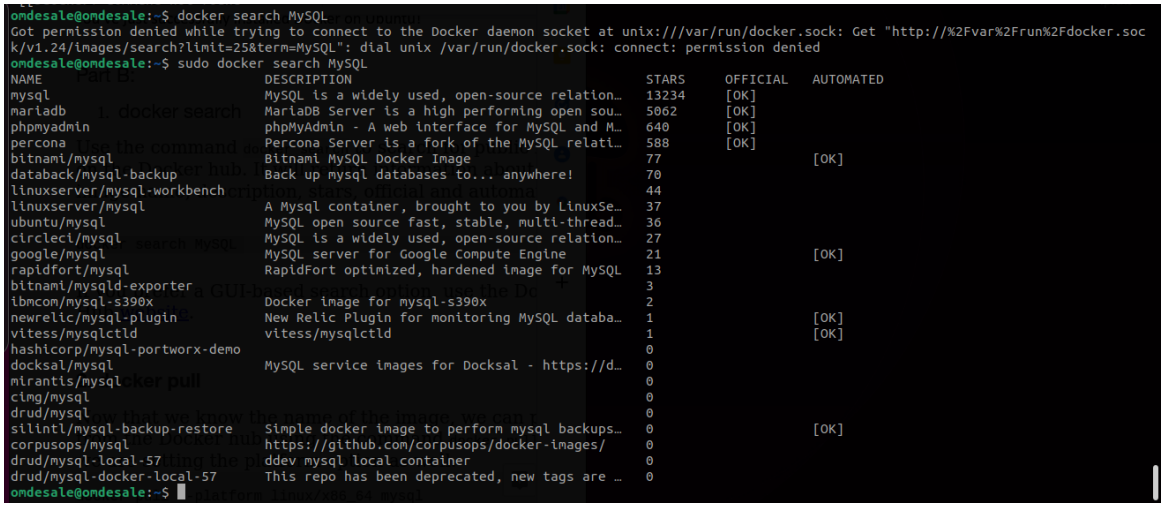
You've just successfully installed Docker on Ubuntu!

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.

```
docker search MySQL
```



If you prefer a GUI-based search option, use the Docker Hub website.

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
```

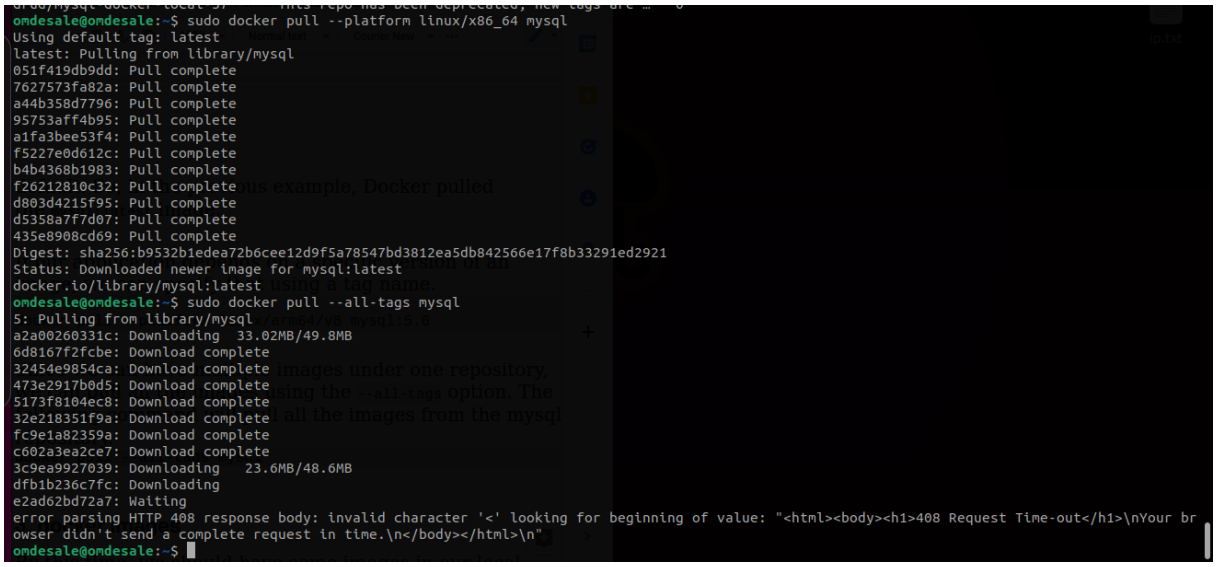
Tags are used to identify images inside a repository. If we don't specify a tag Docker engine uses the `:latest` tag by default. So, in the previous example, Docker pulled the `mysql:latest` image.

If our application depends on a specific version of an image, we can specify that using a tag name.

```
docker pull --platform linux/arm64/v8 mysql:5.6
```

Since we can have multiple images under one repository, we can pull all the images using the `--all-tags` option. The following command will pull all the images from the mysql repository.

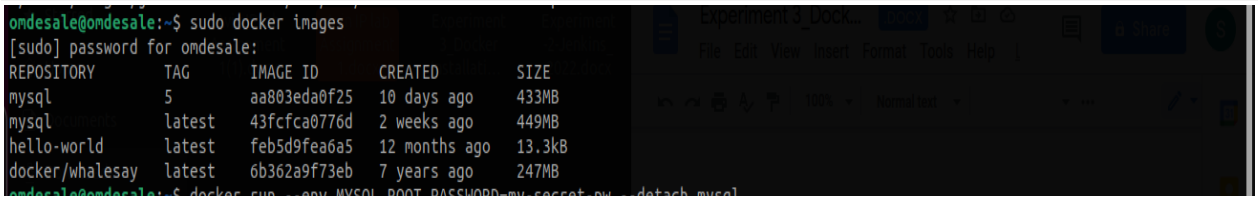
```
docker pull --all-tags mysql
```



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
```



I have two images that we downloaded in the previous step.

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
```

Moreover, we can use the `--name` option to assign a name to the container. Docker will randomly assign a name if we don't provide one.

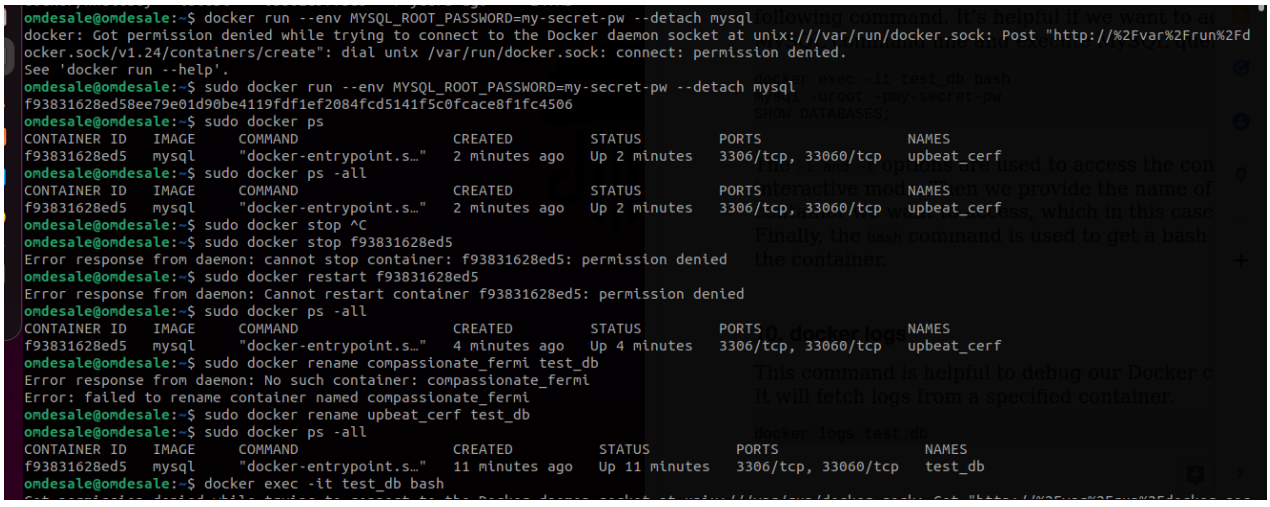
5. docker ps

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

```
docker ps
```

How about listing all the containers, including stopped once? We can do that by adding `--all` option.

```
docker ps –all
```



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 f8c52bedeecc
```

7. docker restart

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

```
docker restart f8c52bedeecc
```

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 compassionate_fermi test_db
```

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 test_db bash
mysql -uroot -pmy-secret-pw
SHOW DATABASES;
```

The `-i and -t` options are used to access the container in an interactive mode. Then we provide the name of the container we want to access, which in this case `test_db`. Finally, the `bash` command is used to get a bash shell inside the container.

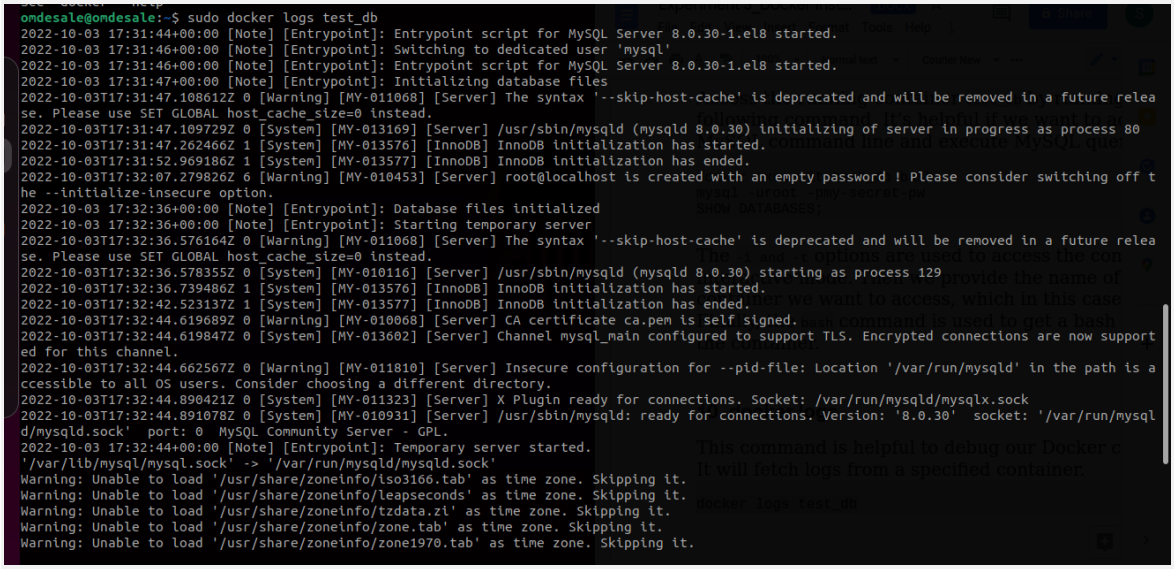
10. docker logs

This command is helpful to debug our Docker containers. It will fetch logs from a specified container.

```
docker logs test_db
```

If we want to continue to stream new output,use the option `-follow`.

```
docker logs -follow test_db
```



11. docker rm

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

```
docker rm test_db
```

You may encounter an error like

Error response from daemon: You cannot remove a running container ..... Stop the container before attempting removal or force remove

As it recommends, we can stop the container first and then remove it or use option `-f` to remove a running container forcefully.

```
docker stop test_db
docker rm test_db# ordocker rm -f test_db
```

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
```

These commands come with plenty of helpful options. If you want to know about other available options, run the `docker command_name --help` command. For example:

```
docker logs --help
```

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 my_repo/my_image:my_tag` — Create a container from an image.

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`

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

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

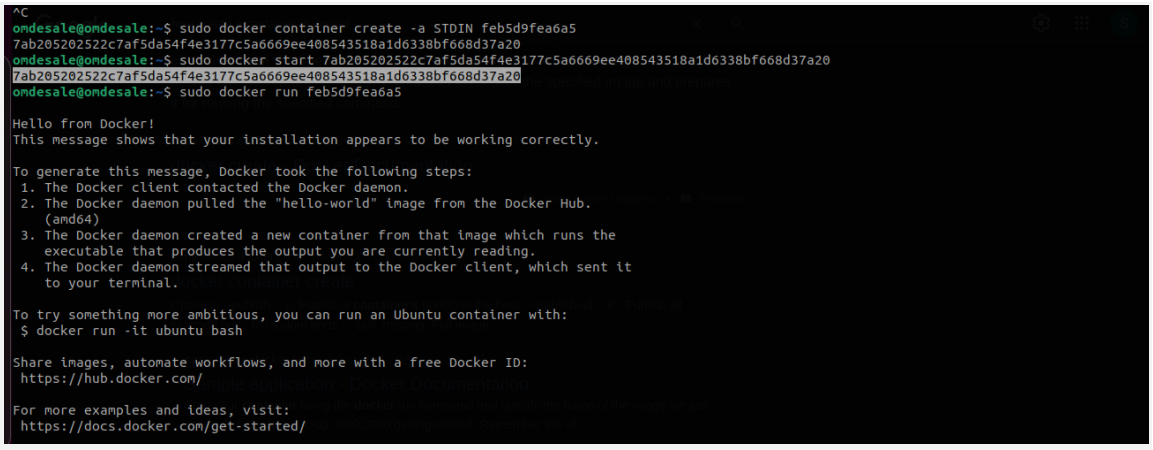
`docker container start my_container` — Start an existing container.

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

`docker container start my_container`

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.



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

`-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`

`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`

`-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`

`-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 my_container` — See lots of info about a container.

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

Container Endings

Sometimes you need to stop a running container.

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

Or if you are impatient:

`docker container kill my_container` — 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 my_container` — Delete one or more containers.

`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`.

Now, let's turn to the magical container-producing molds called images.

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

Rubrics:

Parameter	Successful installation & Hello world commands	Docker basic commands	Docker container commands
Weightage	A	A+	A++

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.

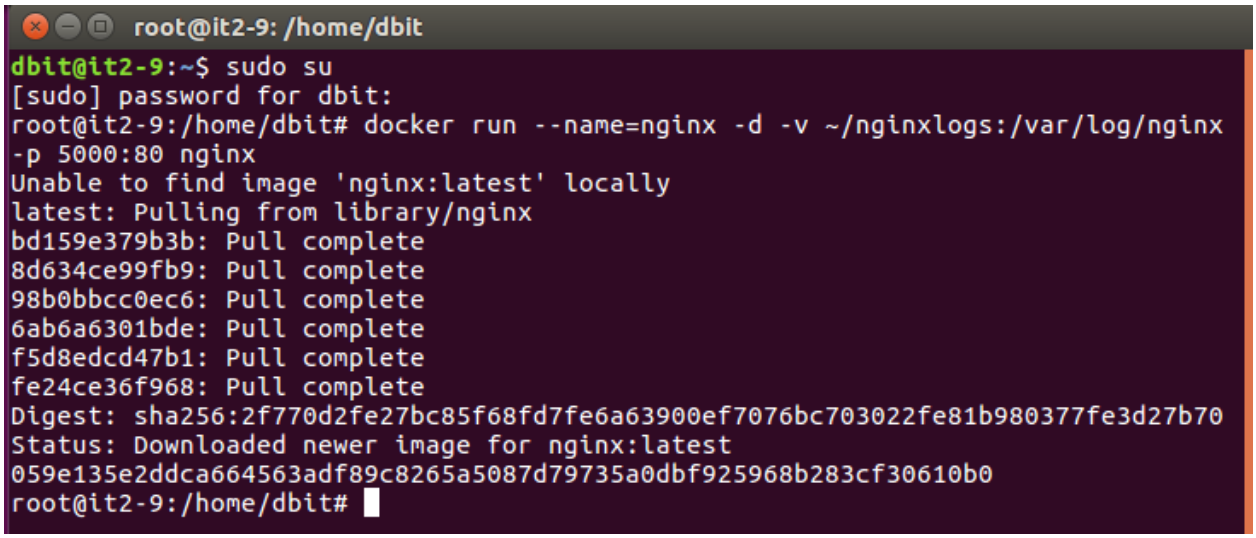
Docker volumes can be used to share files between a host system and the Docker container. For example, let's say you wanted to use the official Docker Nginx image and keep a permanent copy of Nginx's log files to analyze later. By default, the nginx Docker image will log to the /var/log/nginx directory 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
```



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 portt forward. The Nginx container is listening on po rt 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 "daem on off" to start Nginx.**

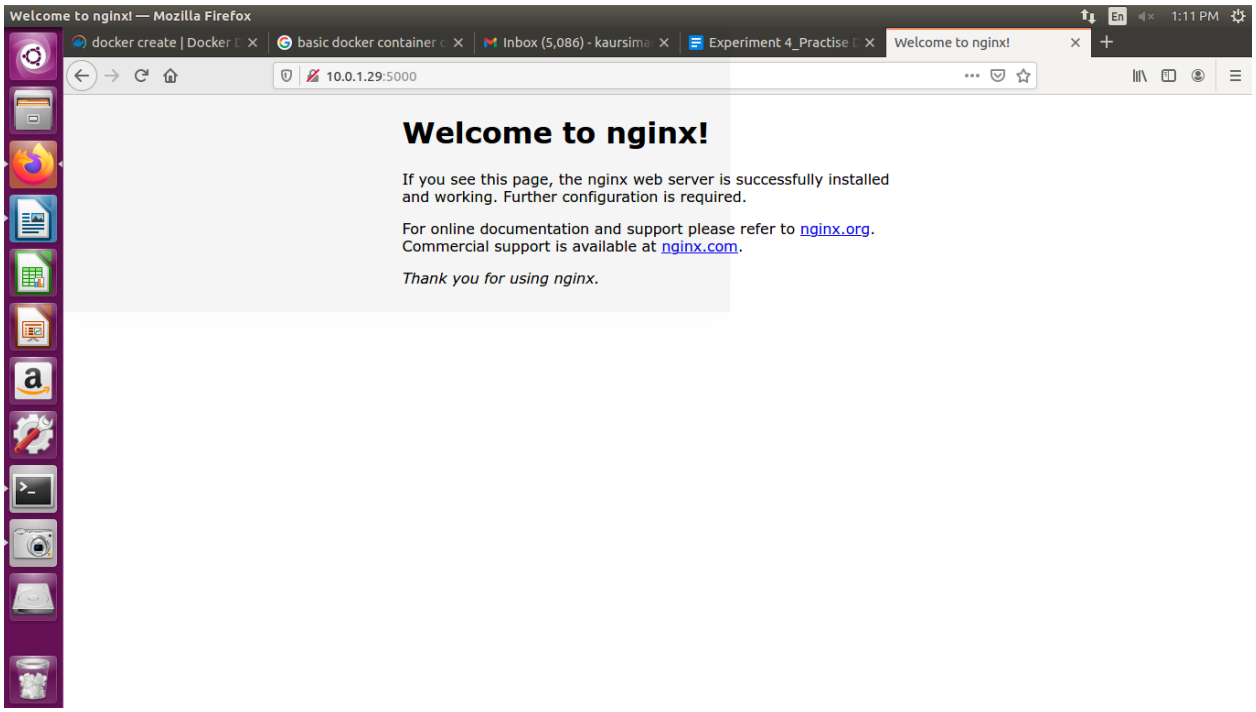
Note: The -v flag is very flexible . It can bindmount or name a volume with just a slight adjustment in syntax. If the first argument begins with a / or ~/, you're creating a bindmount. Remove that, and you're naming the volume.

- **-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 relations hip 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@it2-9:/home/dbit# cat ~/nginxlogs/access.log
10.0.3.246 - - [07/Oct/2022:07:41:16 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozill
a/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0" "-"
10.0.3.246 - - [07/Oct/2022:07:41:16 +0000] "GET /favicon.ico HTTP/1.1" 404 153
"http://10.0.1.29:5000/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko
/20100101 Firefox/88.0" "-"
10.0.3.246 - - [07/Oct/2022:07:41:39 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozill
a/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0" "-"
10.0.3.246 - - [07/Oct/2022:07:41:39 +0000] "GET /favicon.ico HTTP/1.1" 404 153
"http://10.0.1.29:5000/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko
/20100101 Firefox/88.0" "-"
root@it2-9:/home/dbit#
```

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@it2-9:/home/dbit# docker volume create --name DataVolume1
DataVolume1
root@it2-9:/home/dbit#
```

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
```

```
DataVolume1
root@it2-9:/home/dbit# docker run -ti --rm -v DataVolume1:/datavolume1 ubuntu
root@f6b60cdc600d:/#
```

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@it2-9:/home/dbit# docker volume create --name DataVolume1
DataVolume1
root@it2-9:/home/dbit# docker run -ti --rm -v DataVolume1:/datavolume1 ubuntu
root@f6b60cdc600d:/# echo "Example1" > /datavolume1/Example1.txt
root@f6b60cdc600d:/# exit
exit
root@it2-9:/home/dbit#
```

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

```
#docker volume inspect DataVolume1
```

```
root@it2-9:/home/dbit# docker volume inspect DataVolume1
[
    {
        "CreatedAt": "2022-10-07T13:19:52+05:30",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/DataVolume1/_data",
        "Name": "DataVolume1",
        "Options": {},
        "Scope": "local"
    }
]
root@it2-9:/home/dbit#
```

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:

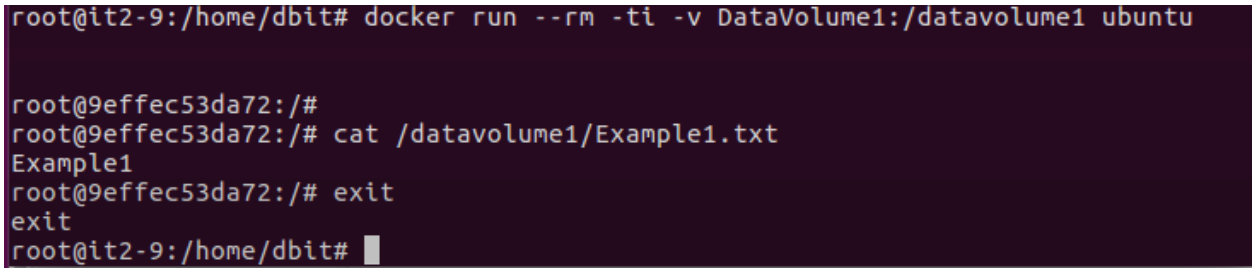
```
$cat /datavolume1/Example1.txt
```

```
root@it2-9:/home/dbit# docker run --rm -ti -v DataVolume1:/datavolume1 ubuntu

root@9effec53da72:/#
root@9effec53da72:/# cat /datavolume1/Example1.txt
Example1
root@9effec53da72:/#
```

Output
Example1

Exit the container:

```
$exit
```



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@it2-9:/home/dbit# docker run -ti --name=Container2 -v DataVolume2:/datavolu
me2 ubuntu
root@d464c07af517:/# echo "Example2" > /datavolume2/Example2.txt
root@d464c07af517:/# cat /datavolume2/Example2.txt
Example2
root@d464c07af517:/# exit
exit
root@it2-9:/home/dbit#
```

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@it2-9:/home/dbit# docker run -ti --name=Container2 -v DataVolume2:/datavolu
me2 ubuntu
root@d464c07af517:/# echo "Example2" > /datavolume2/Example2.txt
root@d464c07af517:/# cat /datavolume2/Example2.txt
Example2
root@d464c07af517:/# exit
exit
root@it2-9:/home/dbit# docker start -ai Container2
root@d464c07af517:/# cat /datavolume2/Example2.txt
Example2
root@d464c07af517:/# exit
exit
root@it2-9:/home/dbit#
```

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@it2-9:/home/dbit# docker volume rm DataVolume2
Error response from daemon: remove DataVolume2: volume is in use - [d464c07af517
44212d935c85f675121201a7ec142c613a0ee333e27c2101b2ba]
root@it2-9:/home/dbit#
```

Output

Error response from daemon: unable to remove volume: remove DataVolume2: volume is in use - [d0d2233b668eddad4986313c7a4a1bc0d2edaf0c7e1c02a6a6256de27db17a63]

We can use this ID to remove the container:

```
#docker rm d0d2233b668eddad4986313c7a4a1bc0d2edaf0c7e1c02a6a6256de27db17a63
```

Output
d0d2233b668eddad4986313c7a4a1bc0d2edaf0c7e1c02a6a6256de27db17a63

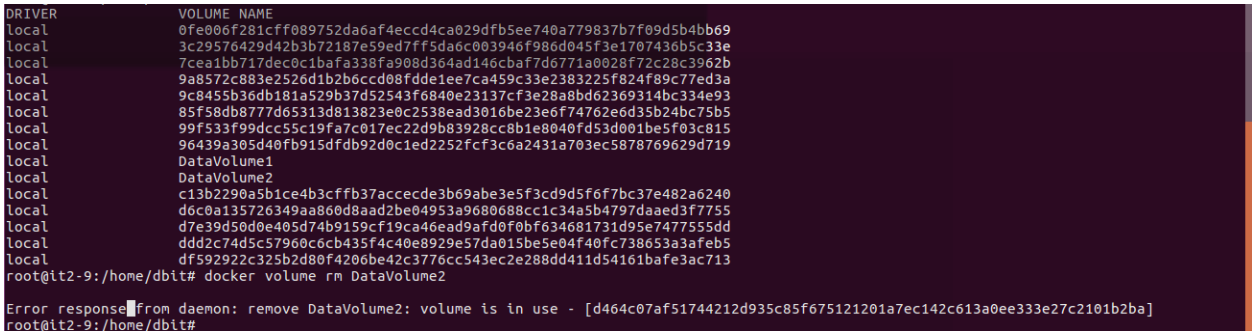
Removing the container won't affect the volume. We can see it's still present on the system by listing the volumes with docker volume ls:

```
#docker volume ls
```

Output DRIVER local **VOLUME NAME DataVolume2**

And we can use docker volume rm to remove it:

```
#docker volume rm DataVolume2
```



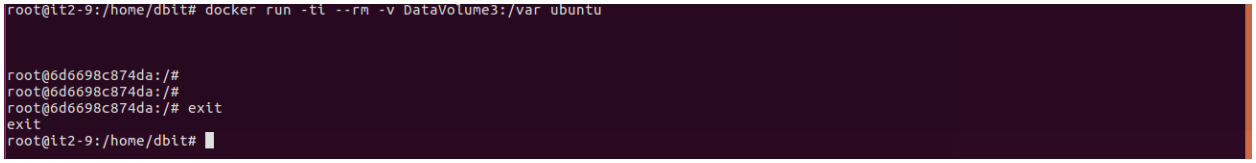
In this example, we created an empty data volume at the same time that we created a container. In our next example, we'll explore what happens when we create a volume with a container directory that already contains data.

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
```



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:

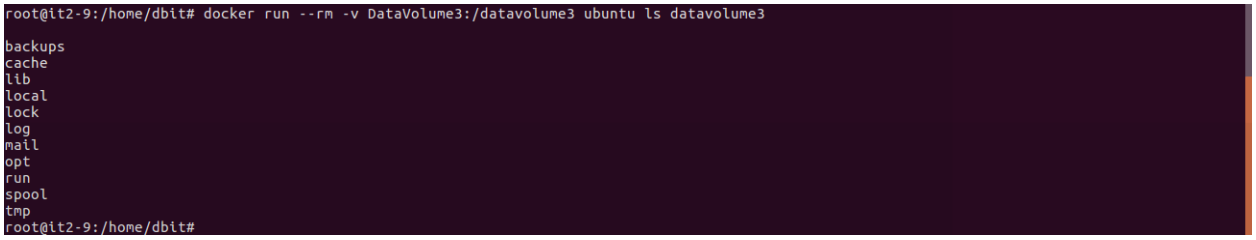
$exit

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



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@it2-9:/home/dbit# docker run -ti --name=Container4 -v DataVolume4:/datavolume4 ubuntu


root@1347948b1fed:/#
root@1347948b1fed:/# echo "This file is shared between containers" > /datavolume4/Example4.txt
root@1347948b1fed:/# exit
exit
root@it2-9:/home/dbit#
```

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

Container4 ubuntu Let's check the data persistence:

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

Output

```
root@it2-9:/home/dbit# docker run -ti --name=Container5 --volumes-from Container4 ubuntu
root@0a0e5b90e444:/# cat /datavolume4/Example4.txt
This file is shared between containers
root@0a0e5b90e444:/#
```

This file is shared between containers

Now let’s append some text from Container5:

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

```
root@0a0e5b90e444:/# echo "Both containers can write to DataVolume4" >> /datavolume4/Example4.txt


root@0a0e5b90e444:/# exit
exit
root@it2-9:/home/dbit#
```

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@it2-9:/home/dbit# docker start -ai Container4


root@1347948b1fed:/#
root@1347948b1fed:/# cat /datavolume4/Example4.txt
This file is shared between containers
Both containers can write to DataVolume4
root@1347948b1fed:/# exit
exit
root@it2-9:/home/dbit#
```

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:

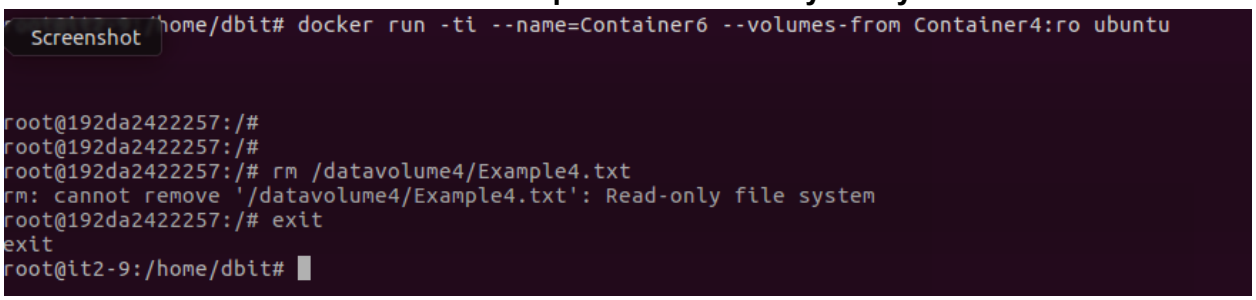
```
#docker run -ti --name=Container6 --volumes-from Container4:ro ubuntu
```

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



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

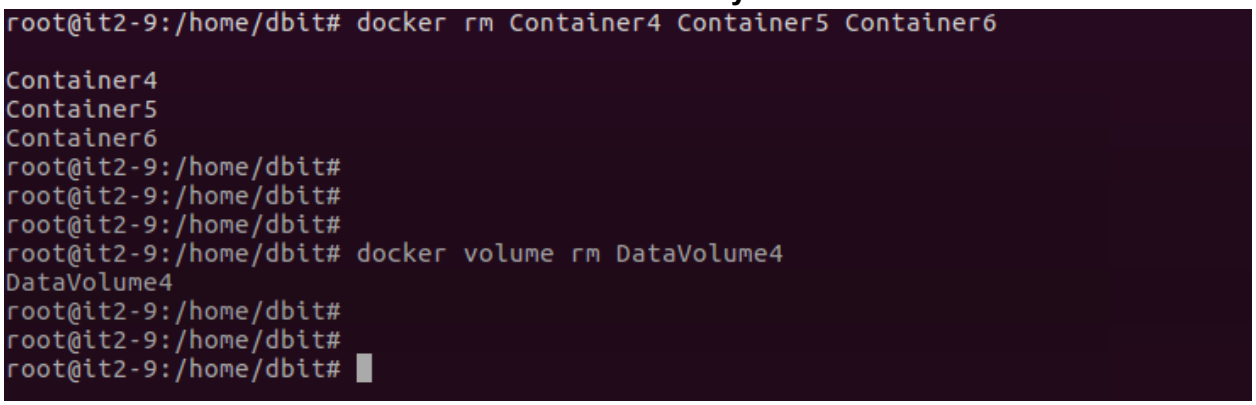
```
$exit
```

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.



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@CPDockerTEST:/home/ubuntu# docker volume create volume1
volume1
root@CPDockerTEST:/home/ubuntu# docker volume create volume2
volume2
root@CPDockerTEST:/home/ubuntu# docker volume create volume3
volume3
root@CPDockerTEST:/home/ubuntu# docker volume ls
DRIVER              VOLUME NAME
local               volume1
local               volume2
local               volume3
root@CPDockerTEST:/home/ubuntu#
```

Now delete all docker volumes at once using command:

 # docker volume prune

```
root@CPDockerTEST:/home/ubuntu# docker volume ls
DRIVER              VOLUME NAME
local               volume1
local               volume2
local               volume3
root@CPDockerTEST:/home/ubuntu# 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:
volume1
volume2
volume3

Total reclaimed space: 0B
root@CPDockerTEST:/home/ubuntu# docker volume ls
DRIVER              VOLUME NAME
root@CPDockerTEST:/home/ubuntu#
```

See? We have deleted all volumes in one go.

```
root@it2-9:/home/dbit# 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

Total reclaimed space: 3.432MB
root@it2-9:/home/dbit# docker volume ls
DRIVER              VOLUME NAME
local               0fe006f281cff089752da6af4eccd4ca029dfb5ee740a779837b7f09d5b4bb69
local               3c29576429d42b3b72187e59ed7ff5da6c003946f986d045f3e1707436b5c33e
local               7cea1bb717dec0c1bafa338fa908d364ad146cbaf7d6771a0028f72c28c3962b
local               9a8572c883e2526d1b2b6ccd08fdde1ee7ca459c33e2383225f824f89c77ed3a
local               9c8455b36db181a529b37d52543f6840e23137cf3e28a8bd62369314bc334e93
local               85f58db8777d65313d813823e0c2538ead3016be23e6f74762e6d35b24bc75b5
local               99f533f99dcc55c19fa7c017ec22d9b83928cc8b1e8040fd53d001be5f03c815
local               96439a305d40fb915dfdb92d0c1ed2252fcf3c6a2431a703ec5878769629d719
local               DataVolume2
local               c13b2290a5b1ce4b3cffb37accecde3b69abe3e5f3cd9d5f6f7bc37e482a6240
local               d6c0a135726349aa860d8aad2be04953a9680688cc1c34a5b4797daaed3f7755
local               d7e39d50d0e405d74b9159cf19ca46ead9afd0f0bf634681731d95e7477555dd
local               ddd2c74d5c57960c6cb435f4c40e8929e57da015be5e04f40fc738653a3afeb5
local               df592922c325b2d80f4206be42c3776cc543ec2e288dd411d54161bafe3ac713
root@it2-9:/home/dbit#
```

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

Experiment 4: Assignment

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-

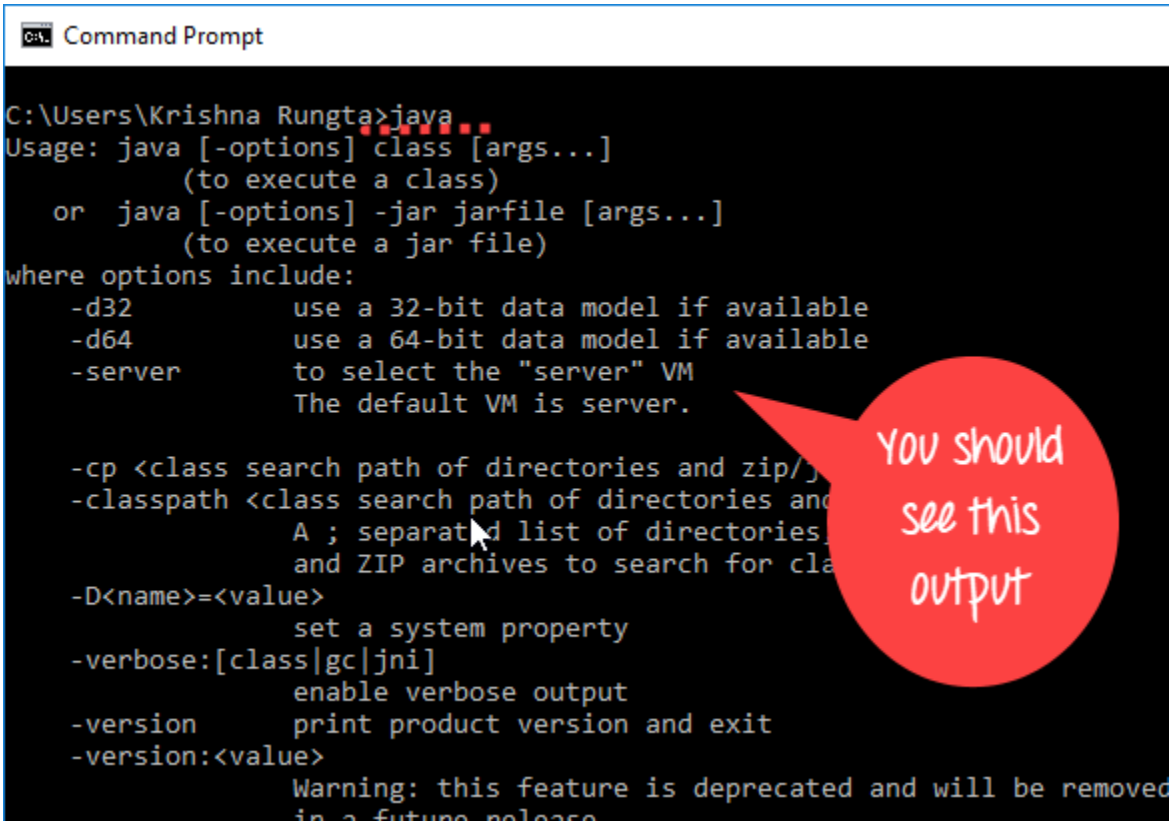
Step 1 – Install Java on your computer

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

Next –

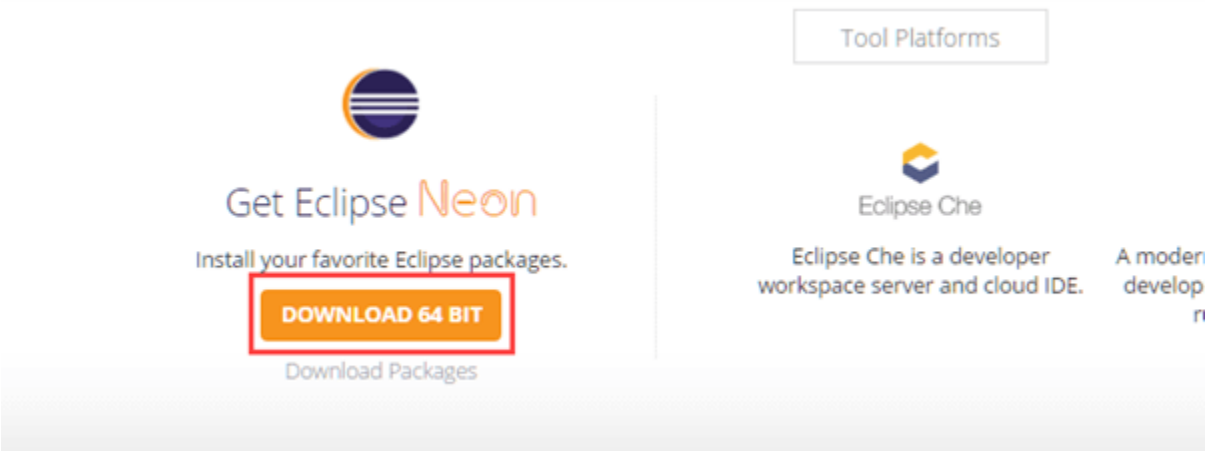
This JDK version comes bundled with Java Runtime Environment (JRE), so you do not need to download and install the JRE separately.

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

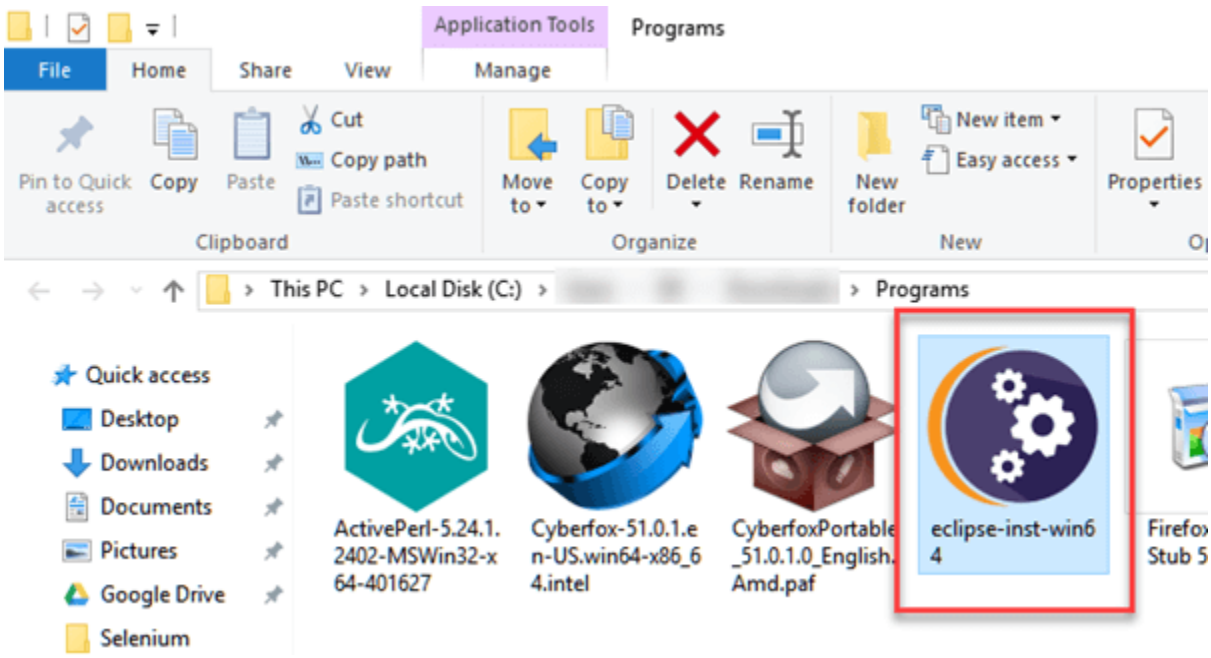


Step 2 – Install Eclipse IDE

Download latest version of **"Eclipse IDE for Java Developers"** here. Be sure to choose correctly between Windows 32 Bit and 64 Bit versions.

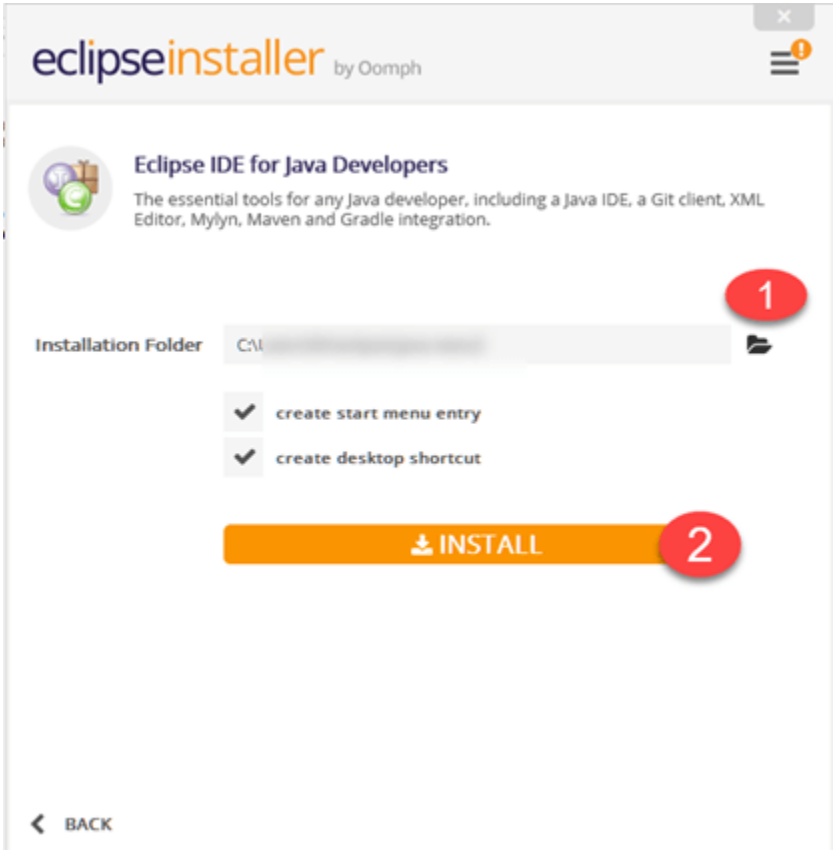


You should be able to download an exe file named "eclipse-inst-win64" for Setup.

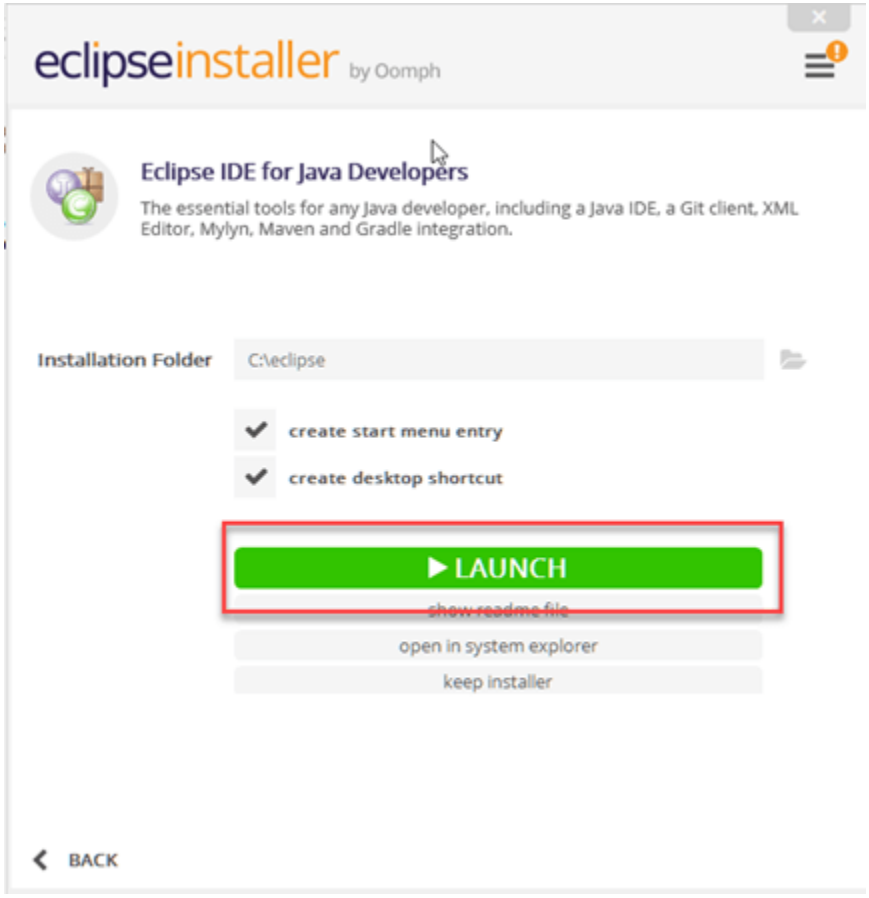


Double-click on file to Install the Eclipse. A new window will open. Click Eclipse IDE for Java Developers.

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



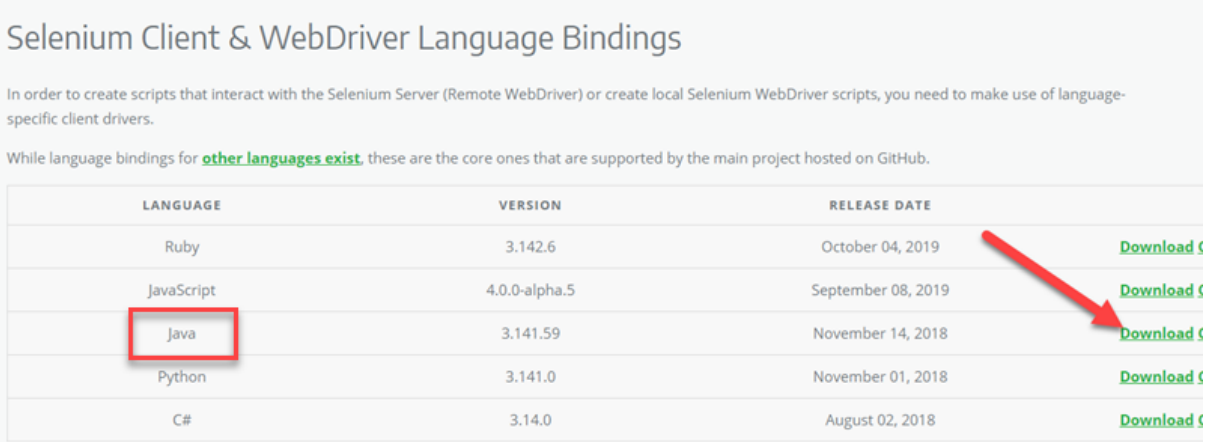
After successful completion of the installation procedure, a window will appear. On that window click on Launch



This will start eclipse neon IDE for you.

Step 3 – Download the Selenium Java Client Driver

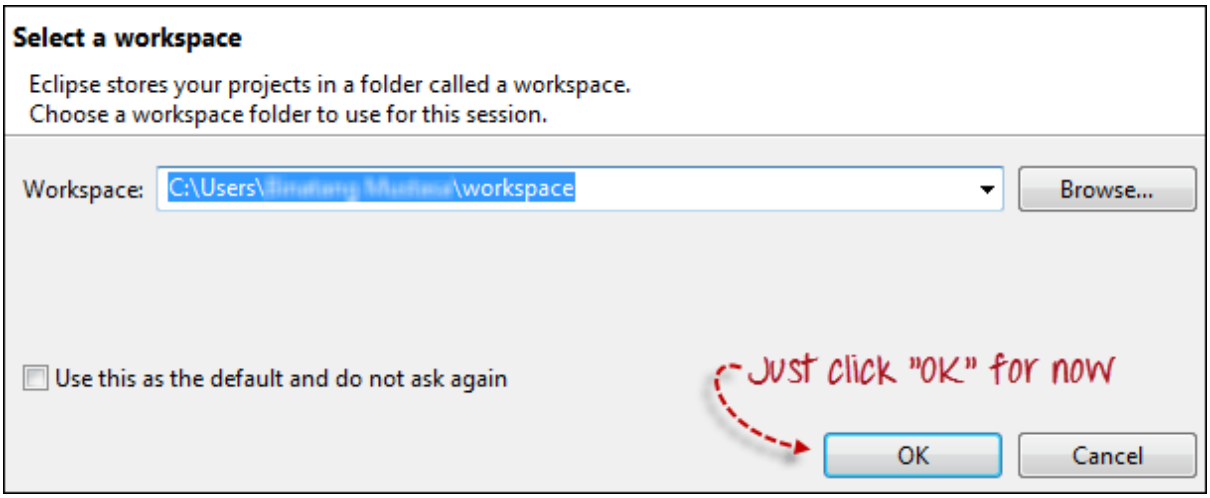
You can download **Selenium Webdriver for Java Client Driver** here. You will find client drivers for other languages there, but only choose the one for Java.



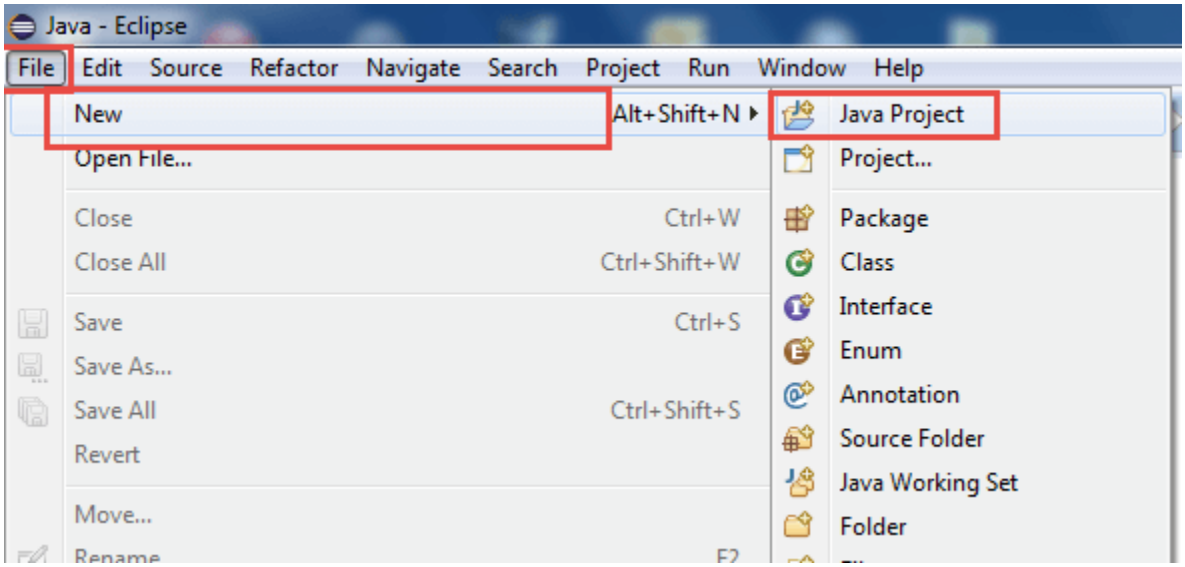
This download comes as a ZIP file named “selenium-3.14.0.zip”. For simplicity of Selenium installation on Windows 10, extract the contents of this ZIP file on your C drive so that you would have the directory “C:\selenium-3.14.0\”. This directory contains all the JAR files that we would later import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

Step 4 – Configure Eclipse IDE with WebDriver

1. Launch the “eclipse.exe” file inside the “eclipse” folder that we extracted in step 2. If you followed step 2 correctly, the executable should be located on C:\eclipse\eclipse.exe.
2. When asked to select for a workspace, just accept the default location.

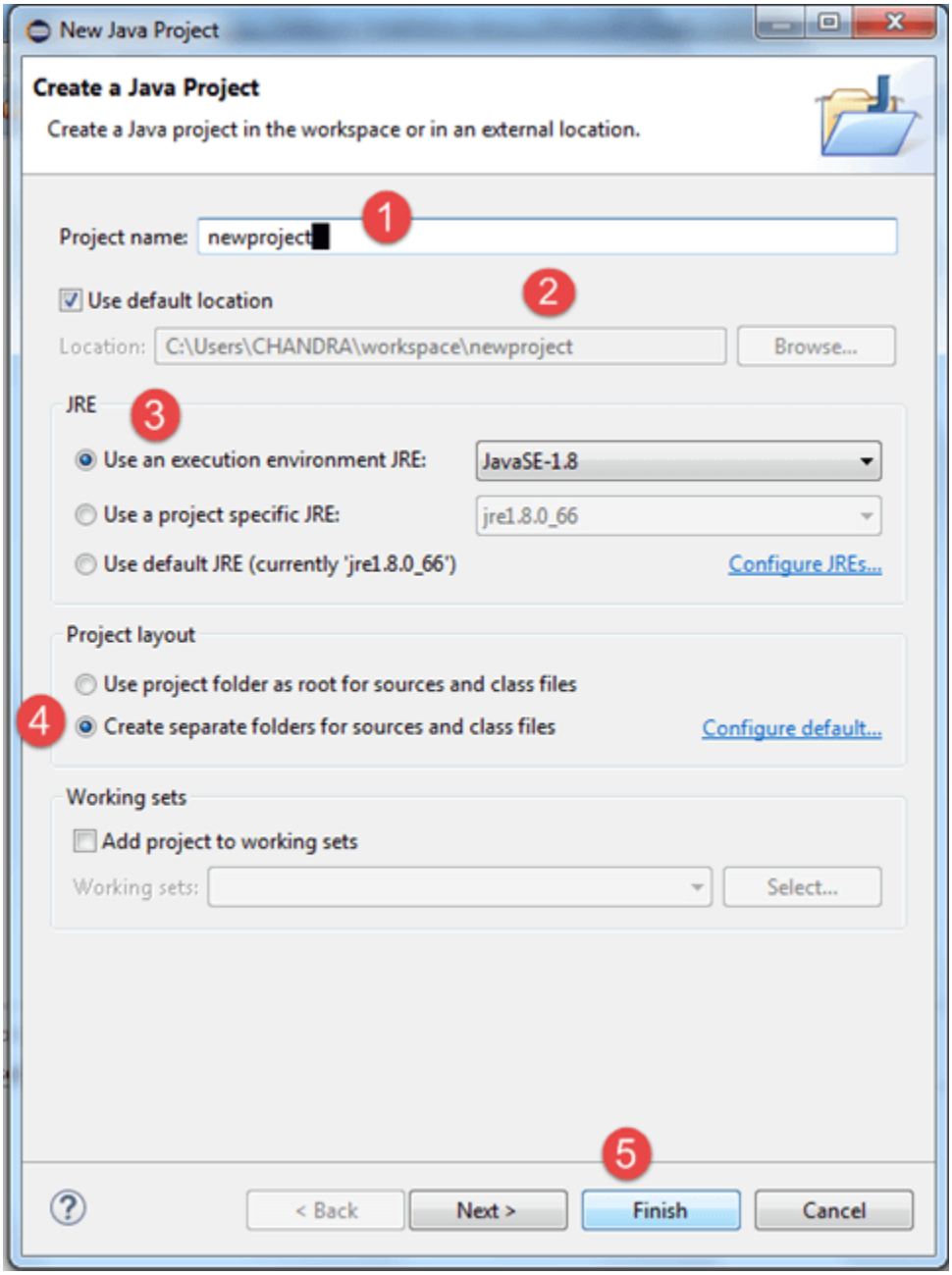


3. Create a new project through File > New > Java Project. Name the project as “newproject”.



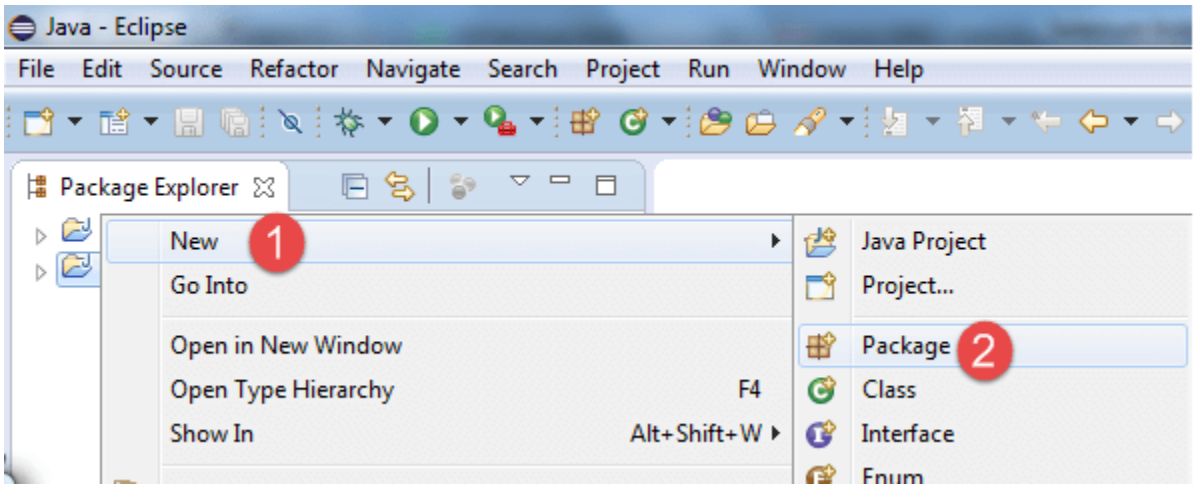
A new pop-up window will open enter details as follow

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



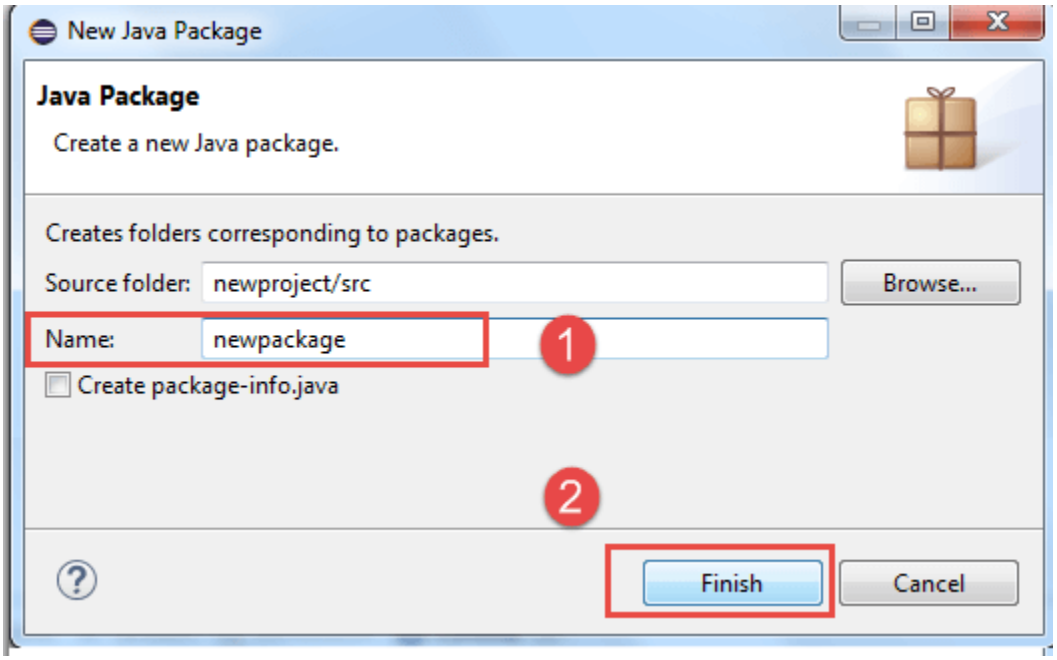
4. In this step,

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

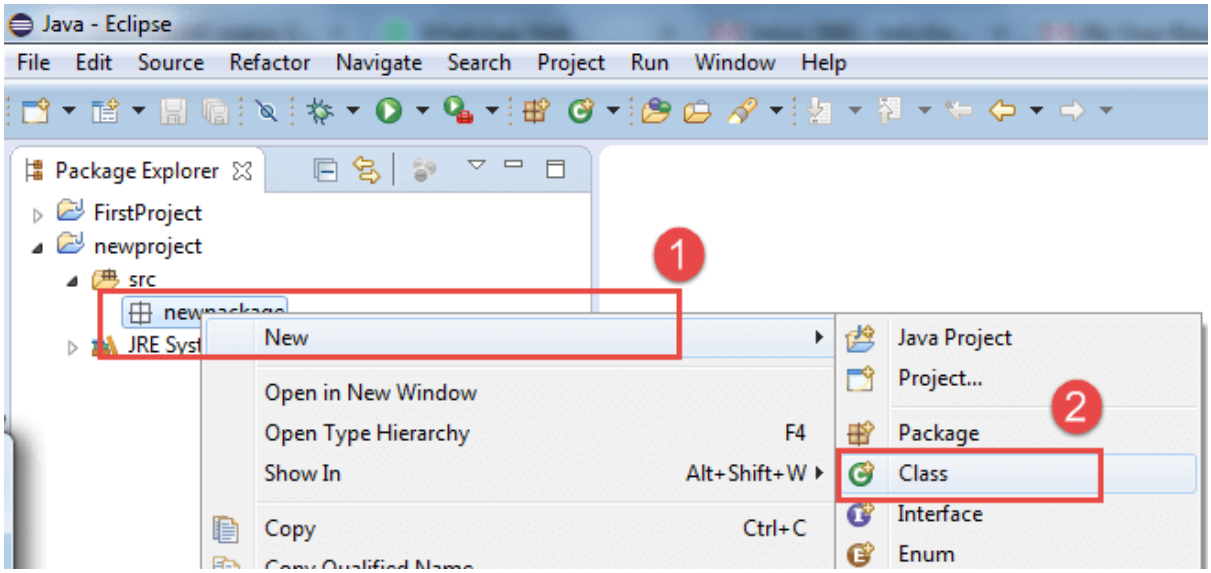


A pop-up window will open to name the package,

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

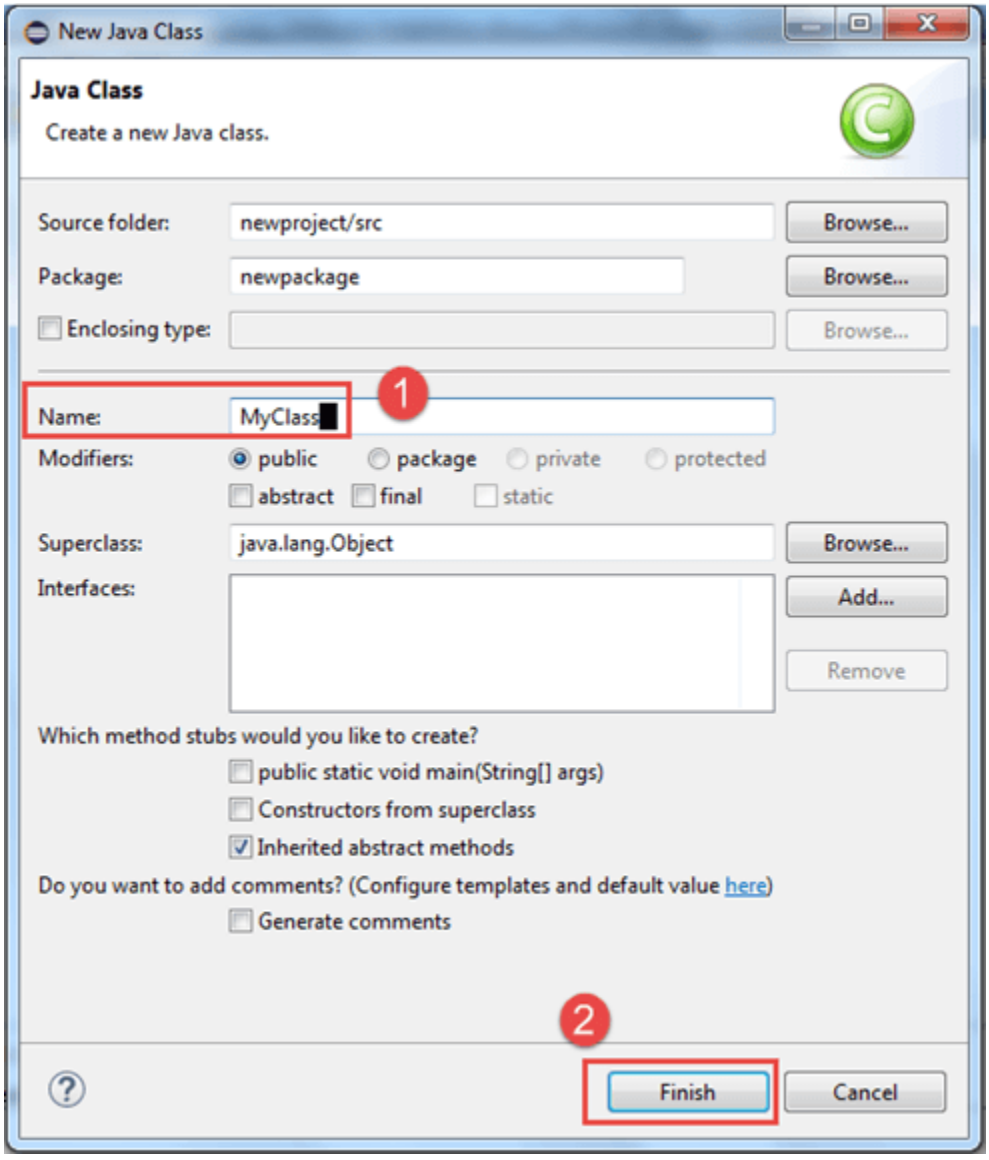


5. Create a new Java class under newpackage by right-clicking on it and then selecting- New > Class, and then name it as “MyClass”. Your Eclipse IDE should look like the image below.

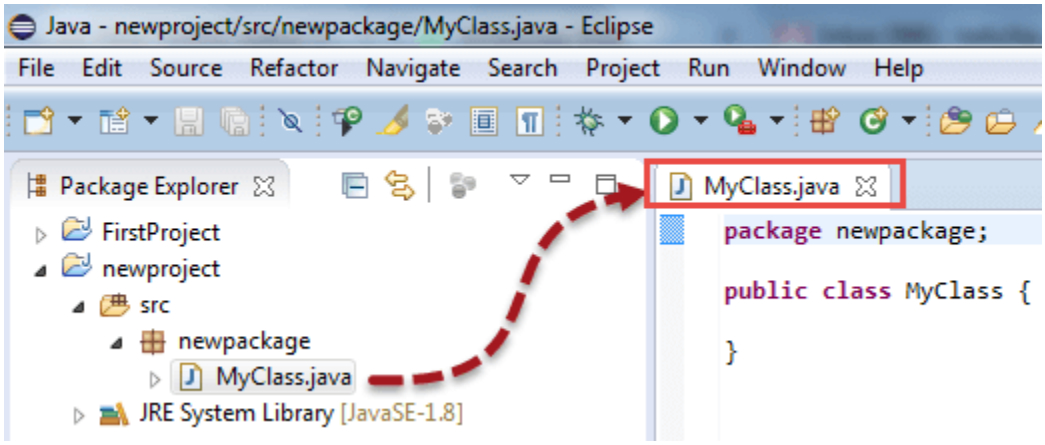


When you click on Class, a pop-up window will open, enter details as

1. Name of the class
2. Click on Finish button



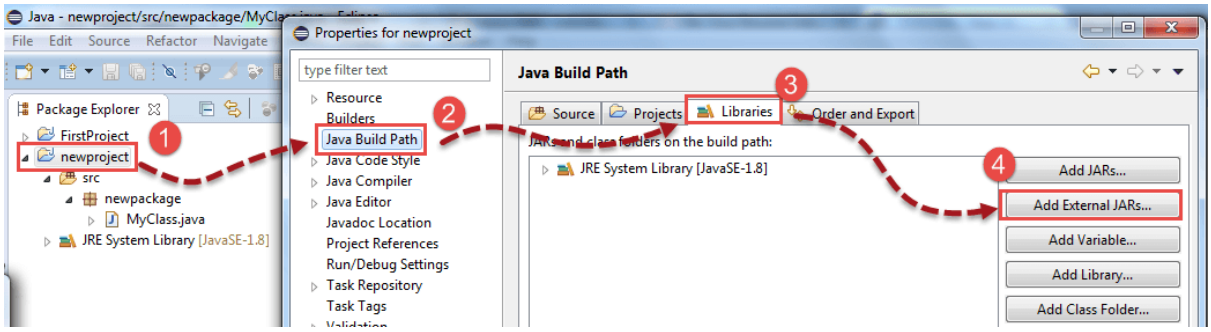
This is how it looks like after creating class.



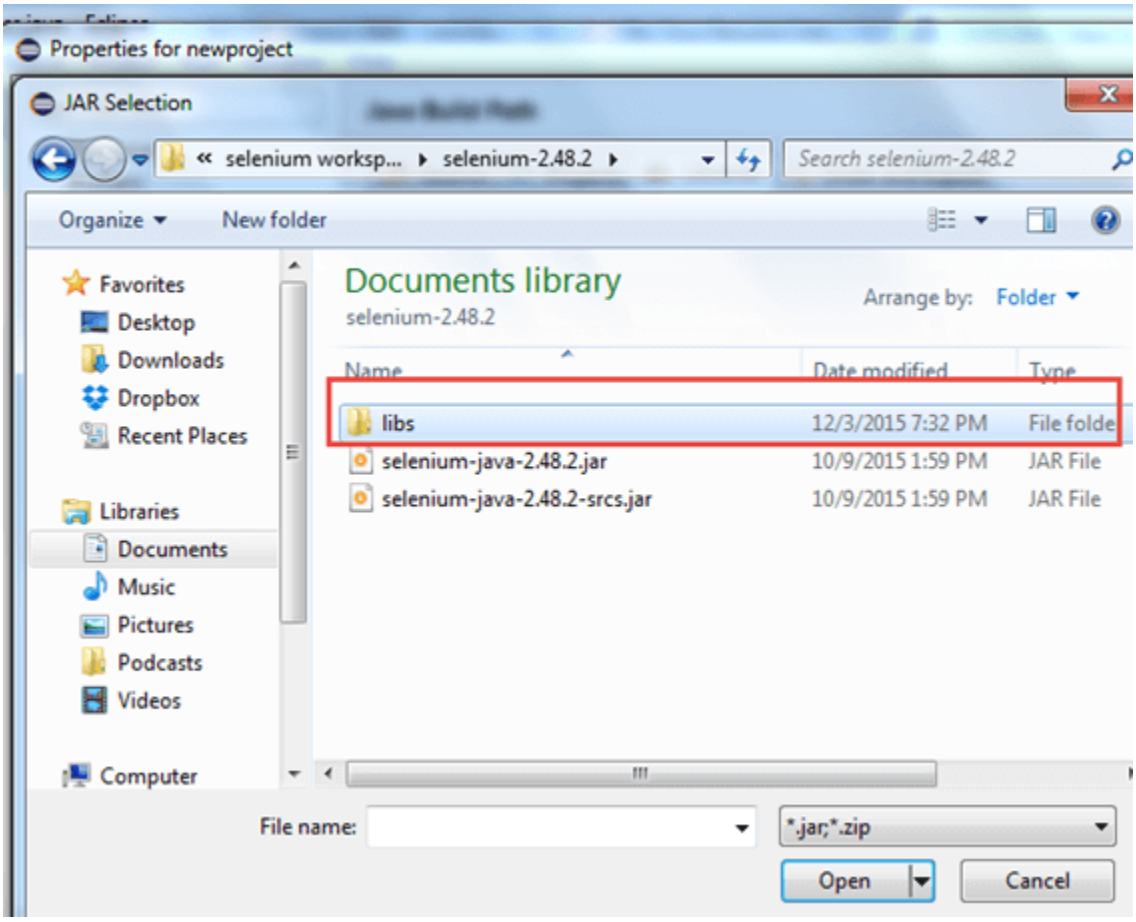
Now selenium WebDriver's into Java Build Path

In this step,

1. Right-click on "newproject" and select **Properties**.
2. On the Properties dialog, click on "Java Build Path".
3. Click on the **Libraries** tab, and then
4. Click on "Add External JARs.."

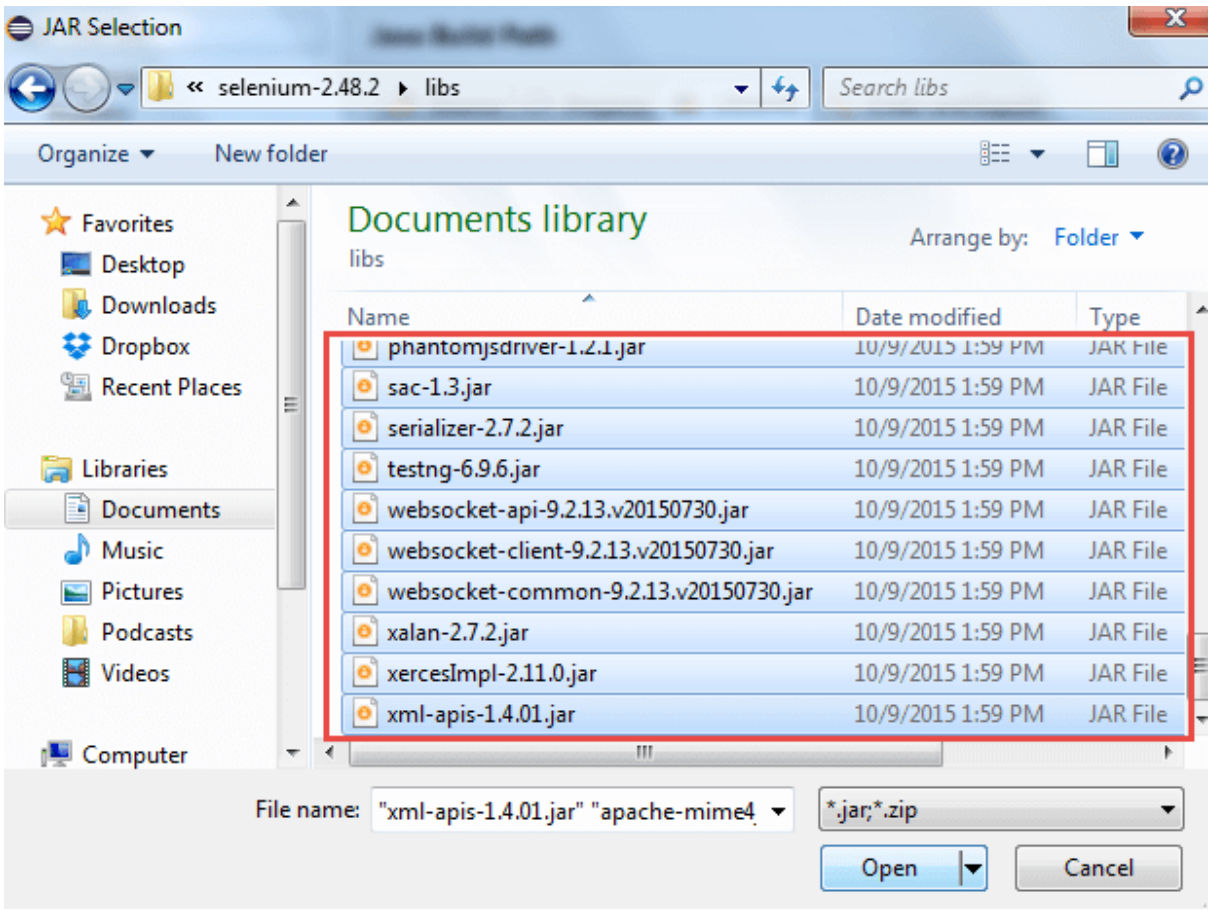


When you click on "Add External JARs.." It will open a pop-up window. Select the JAR files you want to add.

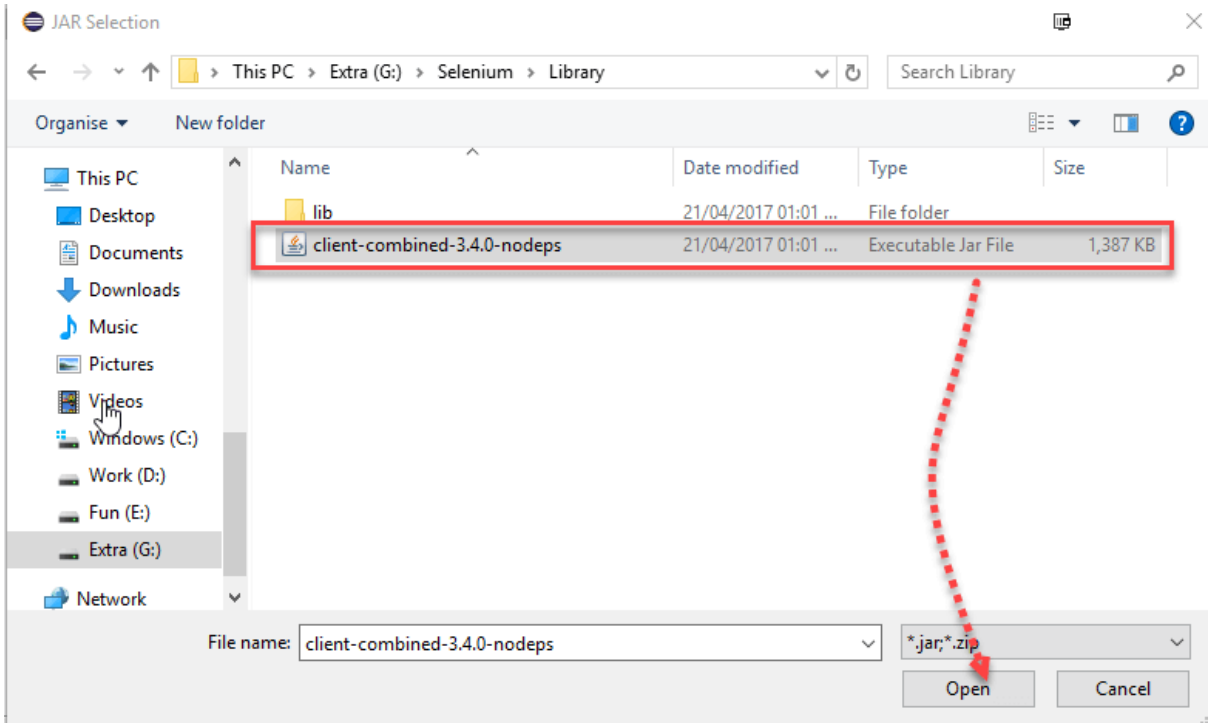


After selecting jar files, click on OK button.

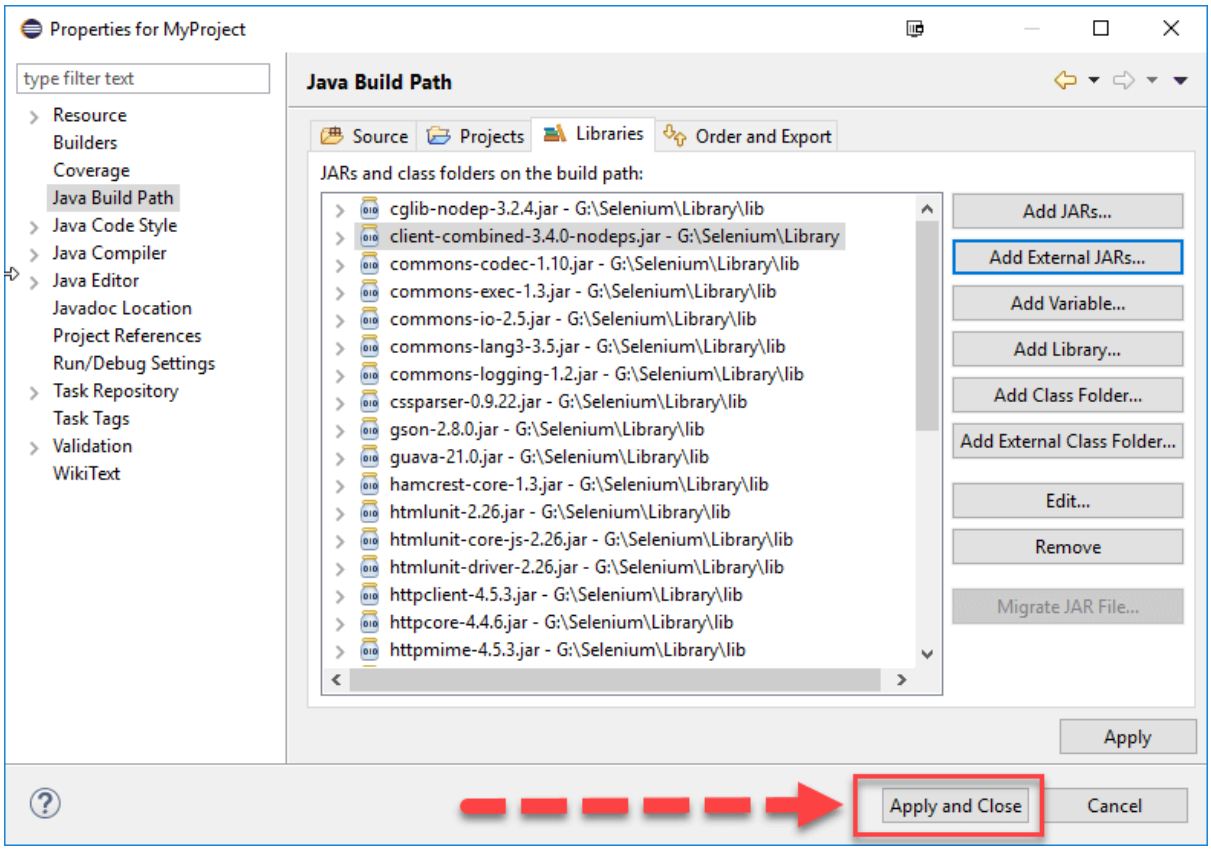
Select all files inside the lib folder.



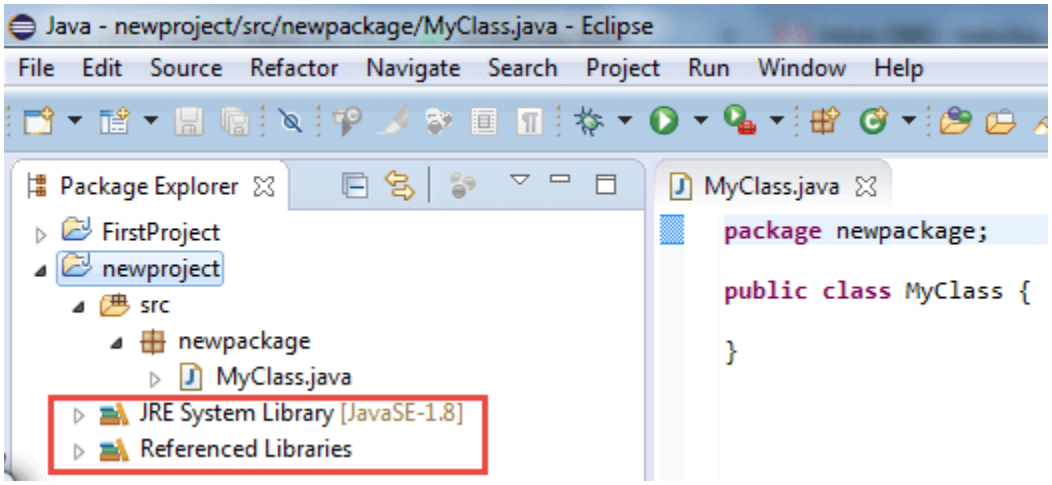
Select files outside lib folder



Once done, click “Apply and Close” button



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



7. Finally, click OK and we are done importing Selenium libraries into our project.

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

Rubrics:

Eclipse with Selenium Perspective & Junit program plugged in with error	**Eclipse with Selenium Perspective & Junit program run but did not understand the Output and function**	**Eclipse with Selenium Perspective & Junit program successfully as shown in reference 2**
A	**A+**	**A++**

```
73  docker ps
  74  docker search mongo
  75  docker pull kope/mongodb
  76  docker run suhani
  77  docker login
  78  docker run suhani
  79  docker run hello-world
  80  docker ps -all
  81  docker stop 5618136d029b
  82  docker ps -all
  83  docker ps -a
  84  docker run hello-world
  85  docker run mysql
  86  docker ps -all
  87  docker stop 4eaaff315ce9
  88  docker restart 4eaaff315ce9
  89  docker ps -all
  90  docker rename gifted_pike suhani
  91  docker rename gifted_pike suhani1
  92  docker ps -all
  93  docker logs suhani1
  94  docker rm suhani1
  95  docker ps -all
  96  docker images
  97  docker rmi c2c2eba5ae85
  98  docker images
  99  docker logs --help
 —----------------------------------------------
 101  docker create --name raj -p 80:80 hello-world
 102  docker ps -all
 103  docker ps -a
 104  docker container start raj
 105  docker container run  raj
```

```
106  docker container run feb5d9fea6a5
107  docker container ls
108  docker container inspect raj
109  docker container kill
110  docker container kill raj
111  docker container rm  raj
112  docker container ls
113  docker ps -a
```