

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

Sr no	Name of the experiment	Pg no	Grade	Date
A	Self Study	3	A++	11/07/2022
B	Case Study - Jenkins	9	A+++	25/07/2022
1	Exp 1 : Version Control Using Git & GitHub	11	A++	01/08/2022
2	Exp 2 : Installation Of Jenkins And Creation Of Different Styles Of Project Namely Freestyle, Maven.	24		12/09/2022
3	Exp 3 : Docker installation & to perform basic docker & container commands	35		26/09/2022
i	Assignment 1 : Docker Volume Creation	56		10/10/2022
ii	Assignment 2 : Installation Of Selenium & performing automatic testing	62		17/10/2022

EXPERIMENT A - Self Study

Date of Experiment: 11/07/2022

Prerequisite:

C, Python, Java, Software Engineering, Cloud

Objective:

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

Aim:

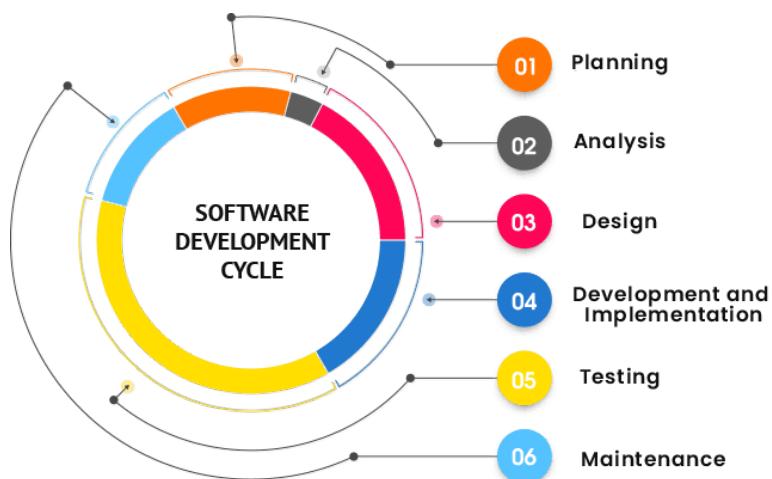
To do self-study on basics understanding of DevOps

Procedure:

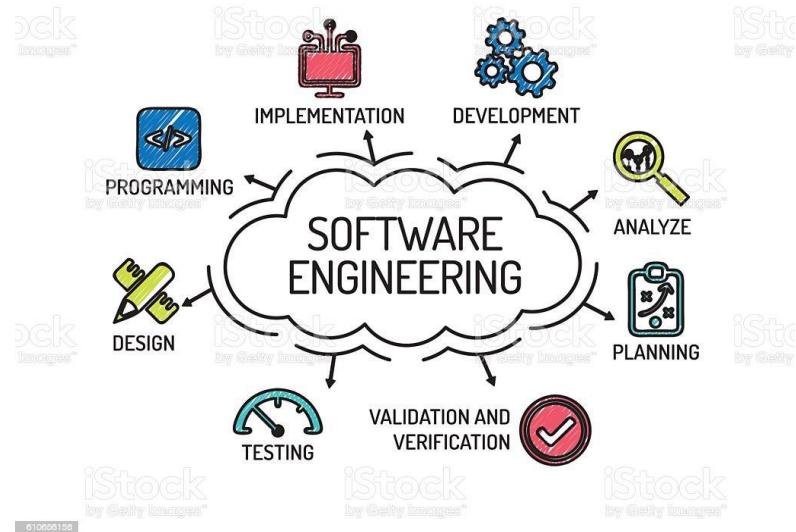
Individual students will perform this self-study over internet resources and will submit a report on their study and give a viva voce on or before 18/07/2022, and they will be graded based on rubrics.

The topics for self-study are :

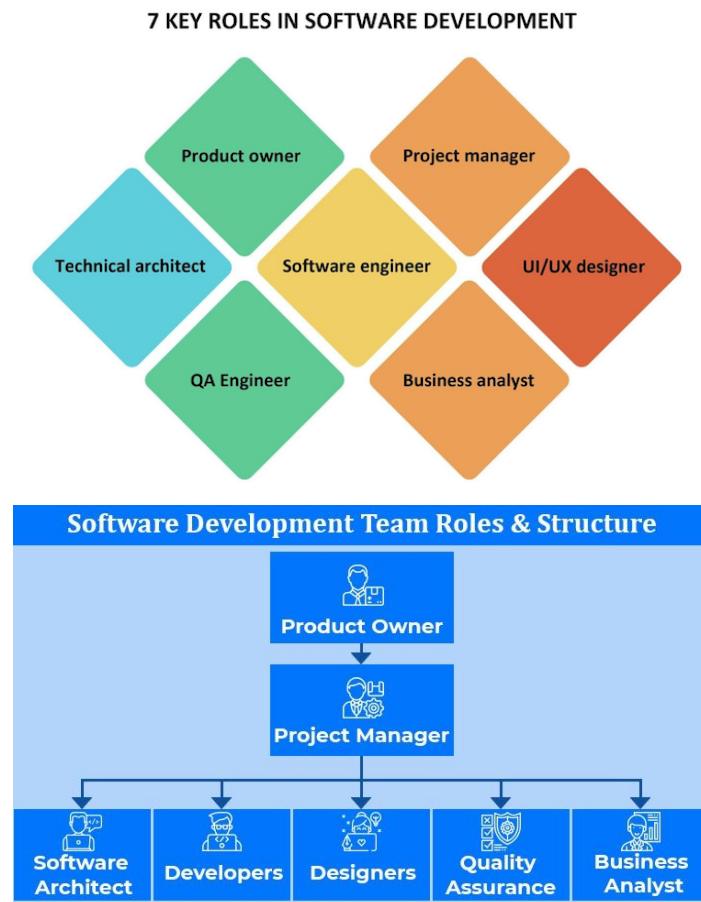
1. Development of Software



2. Software Engineering



3. Development team



4. Operation team



5. Operations Step



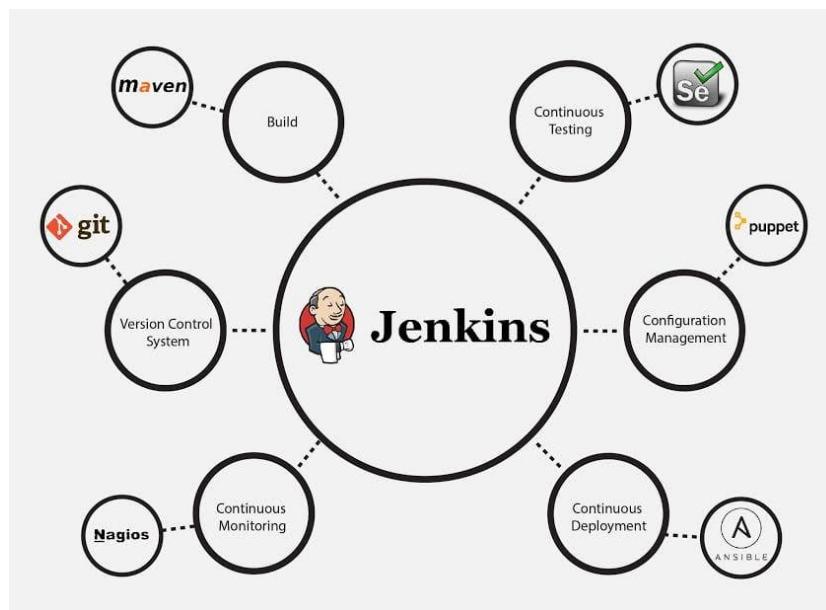
6. GitHub



7. Jenkins

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

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



8. Docker

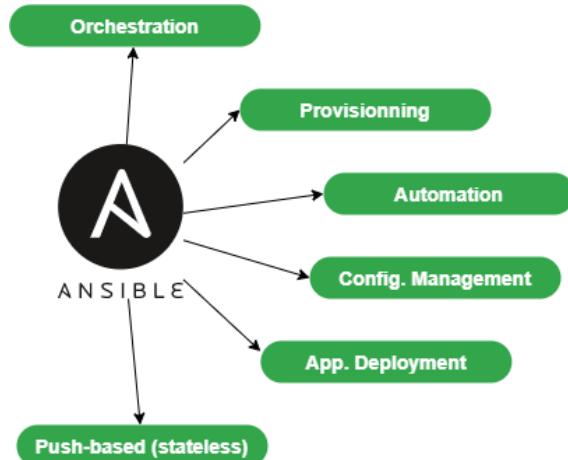
Docker is an open platform for developing, shipping, and running applications.

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

Features of Docker:

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

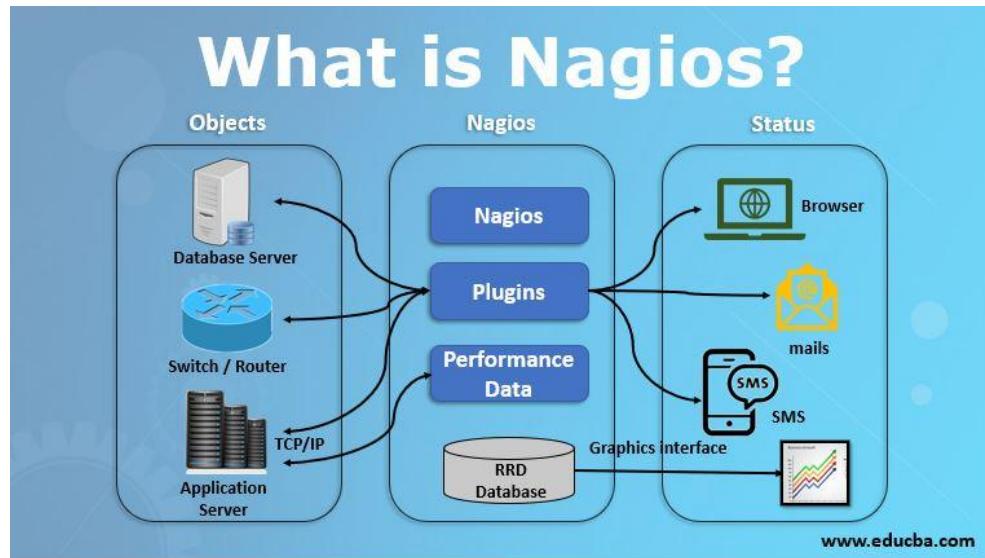
9. Ansible



Features:

- Open source
- Simple
- Versatile
- Powerful
- Agentless

10. NegiOs



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

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

References:

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

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

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

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

[What is GitHub - javatpoint](#)

EXPERIMENT B

Case Study - “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:

What is Azure? [2]

- ❖ Microsoft Azure, often referred to as Azure, is a cloud computing service operated by Microsoft for application management via Microsoft-managed data centers.
- ❖ It supports many different programming languages, tools, and frameworks, including both Microsoft-specific and third-party software and systems.
- ❖ Azure mainly offers 4 features :

1. Compute 2.Network 3.Storage 4.Database

3

When was Azure developed? [2]

- ❖ Azure, announced at Microsoft's Professional Developers Conference (PDC) in October 2008, went by the internal project codename "Project Red Dog".
- ❖ It formally released on 1st February 2010, as Windows Azure before being renamed Microsoft Azure on March 25, 2014.

4

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).

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.

5

How was Azure developed? [3]

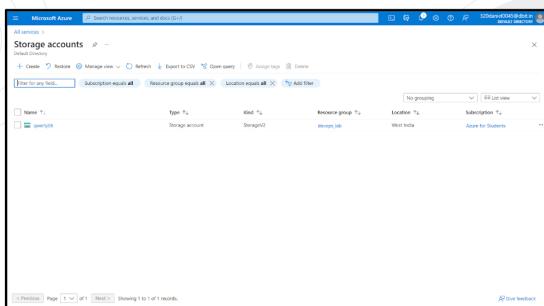
- ❖ The very first version of Windows Azure had a limited number of services that formed the four pillars of the platform.
- ❖ The first pillar was the compute service. Developers could package and run ASP.NET web applications and APIs that ran in the context of a web role while the worker role was designed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

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.

7

Creation of Storage Account



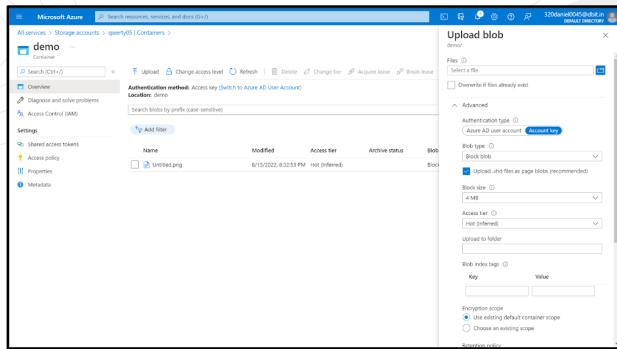
9

Properties of Storage Account

A screenshot of the Microsoft Azure Storage account properties page for 'qweerty05'. The left sidebar shows a tree view of account components: Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Metrics and logs, Alerts, Storage browser, Data storage, Containers, File shares, Queues, Tables, Security + networking, Networking, Azure CDN, Access keys, Shared access signature, and Encryption. The main pane displays the 'Overview' tab with details like Resource group (qweerty05), Location (West India), Primary/Secondary location (Bengaluru/India, South India), Subscription (qweerty05 - Azure for Students), and Subscriptions (qweerty05 - Azure for Students). The 'Networking' section shows a single endpoint '10.0.0.101:8080'. The 'Security' section includes options for Hierarchy namespace, Default access tier (Hot), Blob public access (Enabled), and Shared access signature (Enabled). The 'Encryption' section shows 'Encryption at rest' as 'Enabled (AES-256)'. The bottom navigation bar includes tabs for Properties, Monitoring, Capabilities (7), Recommendations, Tutorials, and Developer Tools.

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.

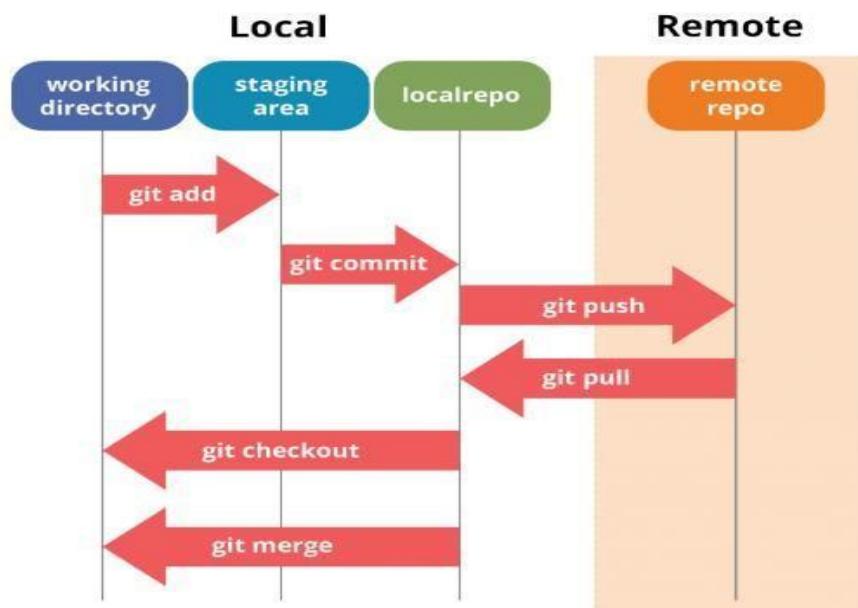
Some of the basic operations in Git are:

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:ghp_9NIAoCpLMdhZzeqE26ZFeHf8rOFVDx4c8V0k@gith  
ub.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/
```

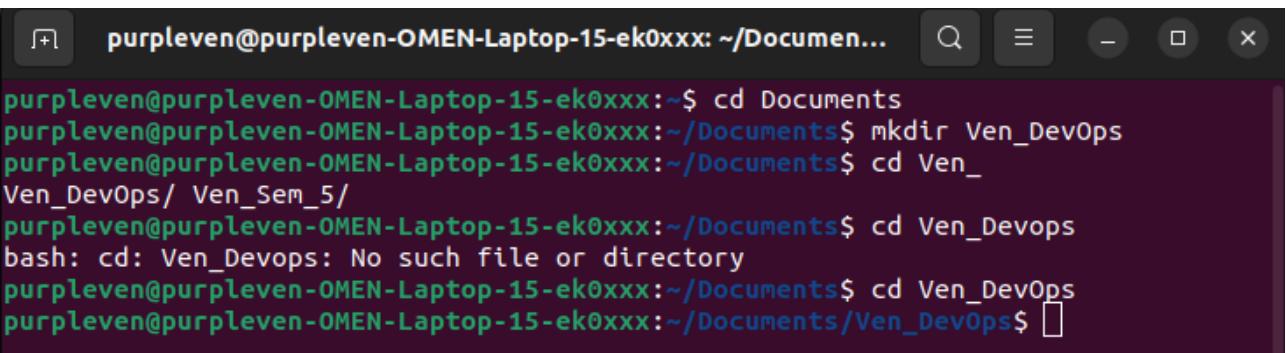
Output:

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



```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents...  
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ cd Documents  
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents$ mkdir Ven_DevOps  
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents$ cd Ven_  
Ven_DevOps/ Ven_Sem_5/  
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents$ cd Ven_DevOps  
bash: cd: Ven_Devops: No such file or directory  
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents$ cd Ven_DevOps  
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

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`

```
initialized empty Git repository in /home/purpleven/Documents/Ven_DevOps/.git/
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 .`

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

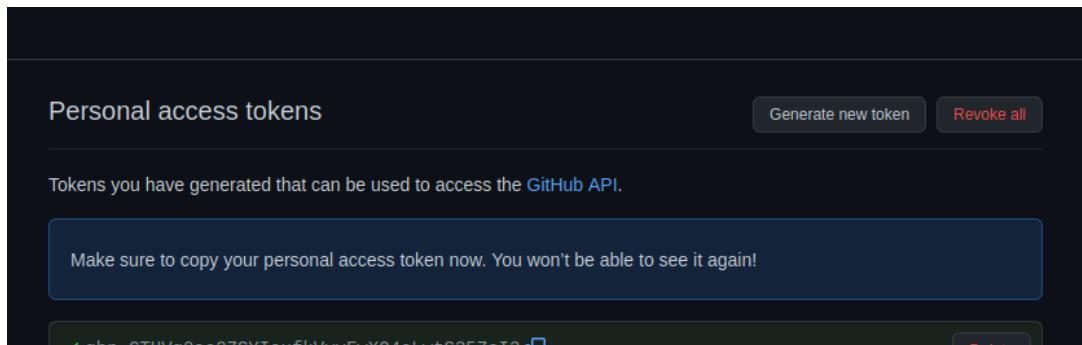
    Initial Commit
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

Now with the help of GitHub:

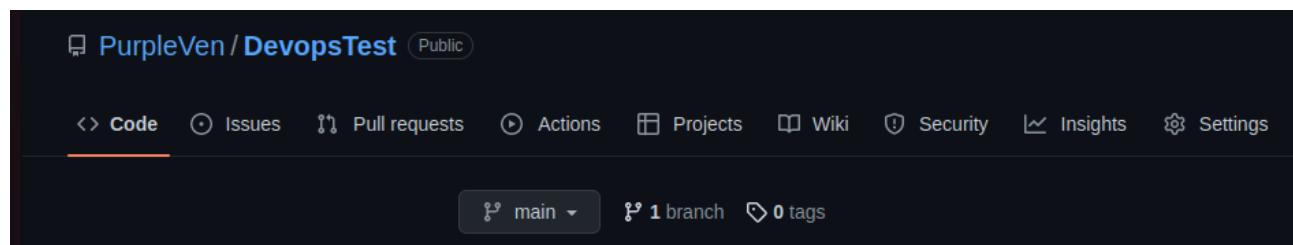
Go to GitHub ->User accounts-> setting -> developers portal

->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

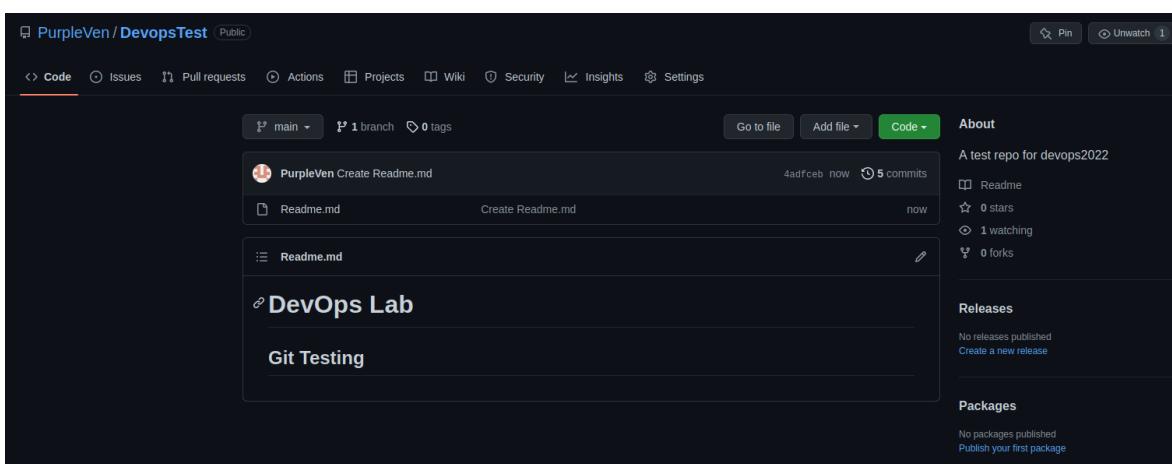
```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$ git clone https://github.com/PurpleVen/DevopsTest.git
Cloning into 'DevopsTest'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 10 (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:ghp_CTHVg0ac2ZSYIaufkVyyFwX04oLwtS35ZeI3@github.com/PurpleVen/DevopsTest.git
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps$
```

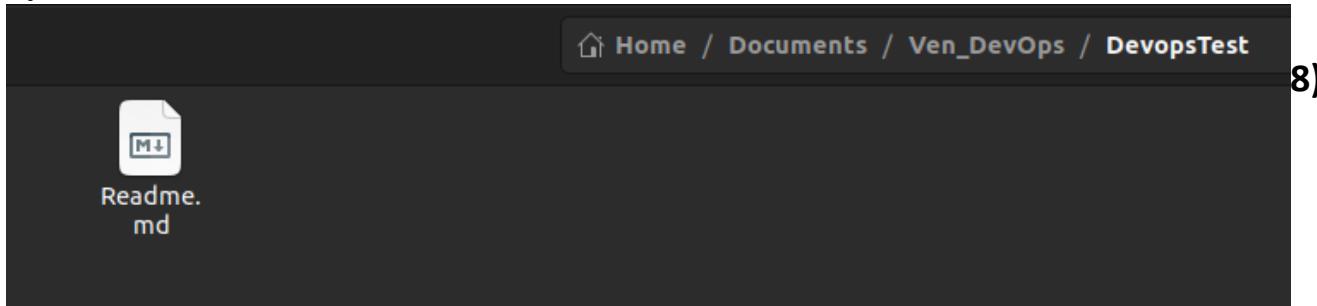
5) Added a file in github



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

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ touch sample.html
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git remote add origin https://github.com/PurpleVen/DevopsTest.git
error: remote origin already exists.
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git remote set-url origin https://PurpleVen:ghp_CTHVg0ac22SYIaufkVyyFwX04oLwt535ZeI3@github.com/PurpleVen/DevopsTest.git
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git push origin main
Everything up-to-date
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git add .
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git commit -m "Commit sample html file"
[main e2e7f7f] Commit sample html file
 1 file changed, 2 insertions(+)
 create mode 100644 sample.html
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3) 361 bytes | 361.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/PurpleVen/DevopsTest.git
 4adfcab..e2e7f7f main -> main
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~/Documents/Ven_DevOps/DevopsTest$
```

9) Displayed in GitHub

A screenshot of a GitHub repository page for 'PurpleVen / DevopsTest'. The repository is public. The 'Code' tab is selected. At the top, it shows 'main' branch, '1 branch', and '0 tags'. There are buttons for 'Go to file', 'Add file', and 'Code'. Below this, a list of commits is shown:

- PurpleVen Commit sample html file** (commit e2e7f7f, 38 seconds ago)
 - Readme.md** Create Readme.md (32 minutes ago)
 - sample.html** Commit sample html file (38 seconds ago)

A preview of the 'Readme.md' file is shown below, containing the text 'DevOps Lab' and 'Git Testing'.

20

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
```

This screenshot shows a GitHub repository interface. At the top, there are navigation links: 'master' (with a dropdown arrow), '2 branches' (with a dropdown arrow), '0 tags', 'Go to file', 'Add file' (with a dropdown arrow), and 'Code' (with a dropdown arrow). Below this, a message states 'This branch is 6 commits ahead, 6 commits behind main.' On the right, there is a 'Contribute' button with a dropdown arrow.

The main area displays a list of commits from a user named 'PurpleVen'. The commits are as follows:

File	Commit Message	Time Ago
DevopsTest	Commit	21 minutes ago
hello.txt	Committing hello.txt	34 minutes ago
index.html	Initial Commit	1 hour ago
style.css	Initial Commit	1 hour ago
trial.txt	added file	2 minutes ago

At the bottom of the commit list, there is a call-to-action: 'Help people interested in this repository understand your project by adding a README.' followed by a green 'Add a README' button.

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

The screenshot shows a GitHub repository interface. At the top, a message says "ven_branch had recent pushes less than a minute ago" and a green button says "Compare & pull request". Below this, there are buttons for "main" (selected), "3 branches", "0 tags", and "Code". A sidebar titled "Switch branches/tags" shows "main" is the default branch. The main area displays a commit history for the "ven_branch" starting with "Create Readme.md" and "Commit sample html file", both pushed 8 days ago. On the right, there's a cartoon character holding a wrench. At the bottom, there's a section for "Readme.md" with the title "DevOps Lab" and "Git Testing". A "Switch branches or tags" button is located in the bottom right corner of this section.

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

```
10 lines (9 sloc) | 212 Bytes
1 <link rel="stylesheet" href="style.css">
2 <html><body><h1>Hello World!!!!</h1></body></html>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <title>Hello World!</title>
7 <link rel="stylesheet" href="bluestyle.css">
8 </head>
9 <body>
```

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:

```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ 
```

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.

```
binary/ > /etc/apt/sources.list.d/jenkins.list
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.
W: Skipping acquire of configured file 'main/binary-i386/Packages' as repository 'http://apt.postgresql.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 keyring (/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/jenkins.service.
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ 
```

Step 2 — Starting Jenkins

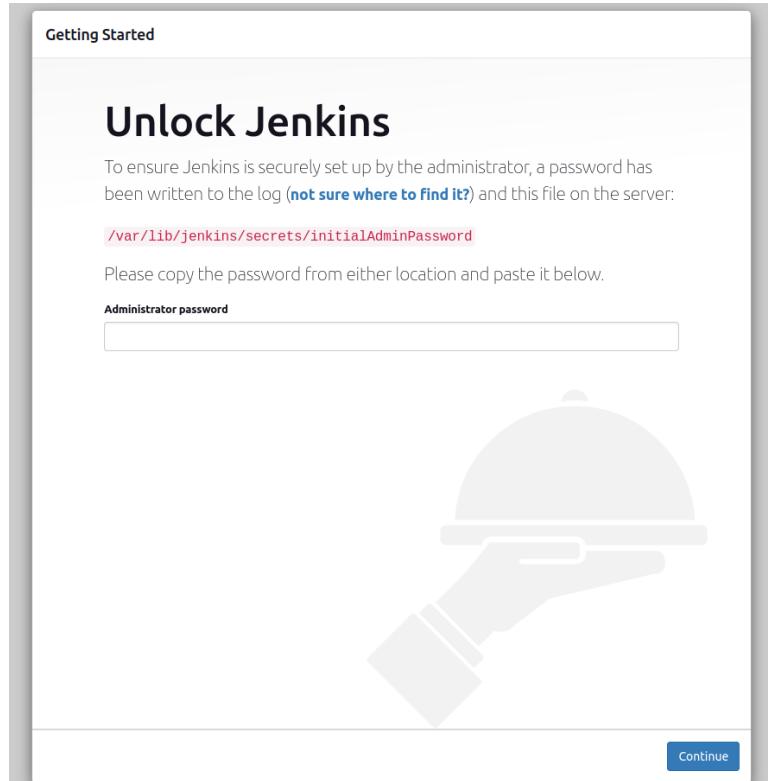
Let's start Jenkins by using systemctl

```
jenkins.service
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ sudo systemctl start jenkins
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2022-09-18 22:41:59 IST; 1min 15s ago
     Main PID: 33116 (java)
        Tasks: 49 (limit: 9181)
       Memory: 1.9G
          CPU: 1min 51.125s
        CGroup: /system.slice/jenkins.service
                  └─33116 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/www/html

Sep 18 22:41:30 purpleven-OMEN-Laptop-15-ek0xxx jenkins[33116]: This may also be found at: /var/lib/jenkins/...
Sep 18 22:41:30 purpleven-OMEN-Laptop-15-ek0xxx jenkins[33116]: ****
Sep 18 22:41:30 purpleven-OMEN-Laptop-15-ek0xxx jenkins[33116]: ****
Sep 18 22:41:30 purpleven-OMEN-Laptop-15-ek0xxx jenkins[33116]: ****
Sep 18 22:41:30 purpleven-OMEN-Laptop-15-ek0xxx jenkins[33116]: 2022-09-18 17:11:59.863+0000 [id=41]
Sep 18 22:41:59 purpleven-OMEN-Laptop-15-ek0xxx jenkins[33116]: 2022-09-18 17:11:59.905+0000 [id=24]
Sep 18 22:41:59 purpleven-OMEN-Laptop-15-ek0xxx systemd[1]: Started Jenkins Continuous Integration S...
Sep 18 22:42:02 purpleven-OMEN-Laptop-15-ek0xxx jenkins[33116]: 2022-09-18 17:12:02.546+0000 [id=60]
Sep 18 22:42:02 purpleven-OMEN-Laptop-15-ek0xxx jenkins[33116]: 2022-09-18 17:12:02.547+0000 [id=60]
Sep 18 22:42:02 purpleven-OMEN-Laptop-15-ek0xxx jenkins[33116]: 2022-09-18 17:12:02.548+0000 [id=60]
lines 1-20/20 (END)
```

Step 3 — Setting Up Jenkins

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



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

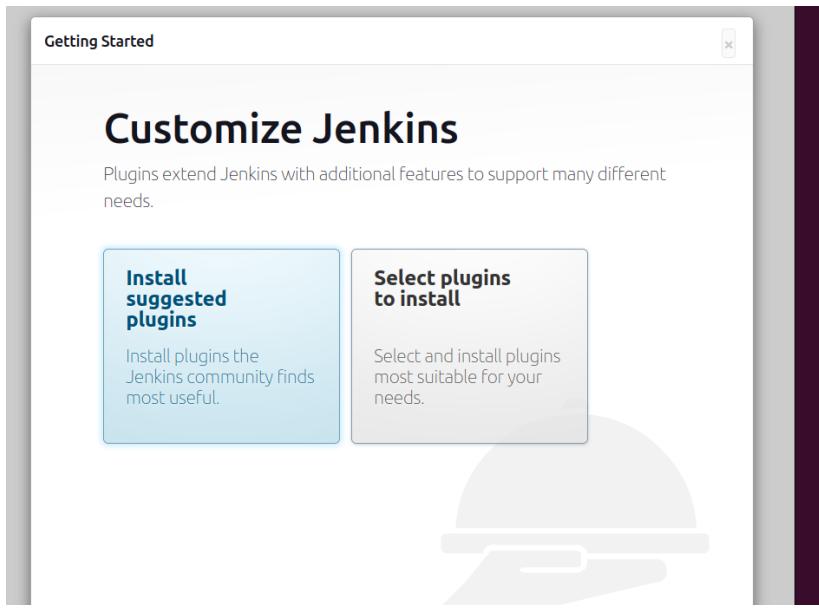
```
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ sudo nano /var/lib/jenkins/secrets/initialAdminPassw
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ sudo nano /var/lib/jenkins/secrets/initialAdminPassw
purpleven@purpleven-OMEN-Laptop-15-ek0xxx:~$ 
```

A screenshot of a terminal window. The user has run the command 'sudo nano /var/lib/jenkins/secrets/initialAdminPassword' twice. The terminal is dark-themed, and the text is white and green.

```
GNU nano 6.2          /var/lib/jenkins/secrets/initialAdminPassword
1c3223780d3e433f87dbc439b272baec
```

A screenshot of a terminal window showing the output of the 'nano' command. The file '/var/lib/jenkins/secrets/initialAdminPassword' contains the single line '1c3223780d3e433f87dbc439b272baec'. The terminal is dark-themed, and the text is white and green.

The next screen presents the option of installing suggested plugins



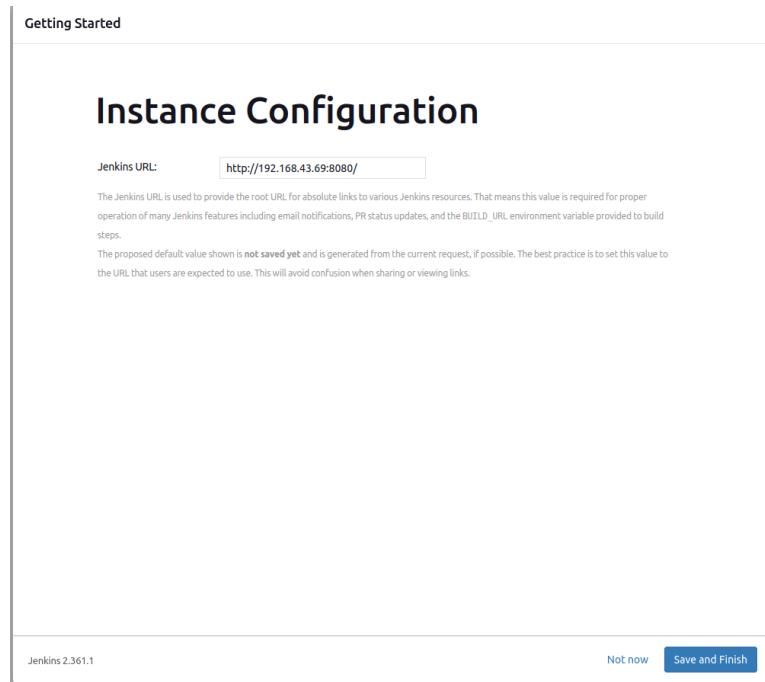
Enter the name and password for your user

The screenshot shows the 'Create First Admin User' form. It has fields for Username, Password, Confirm password, Full name, and E-mail address. At the bottom, there are buttons for 'Skip and continue as admin' and 'Save and Continue'.

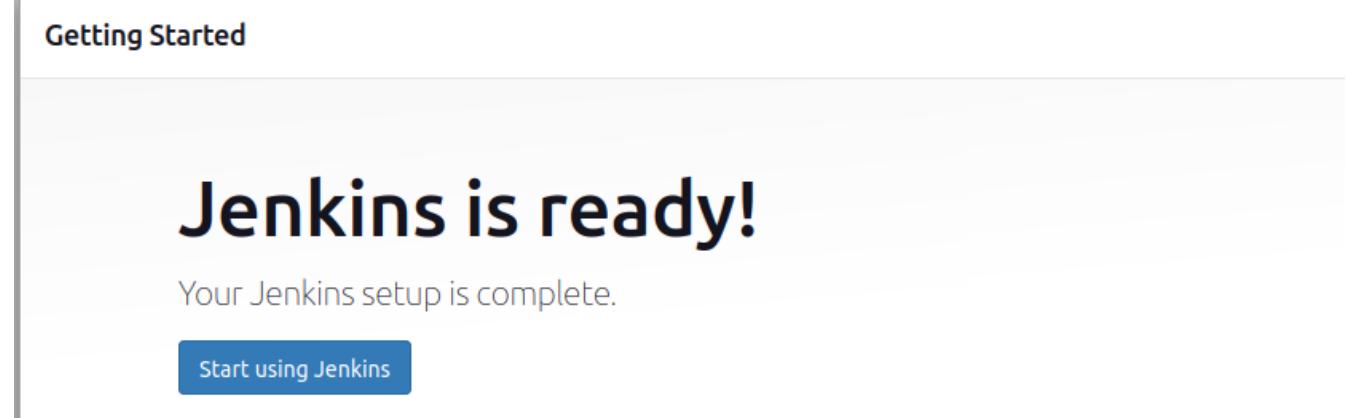
Username:	<input type="text"/>
Password:	<input type="password"/>
Confirm password:	<input type="password"/>
Full name:	<input type="text"/>
E-mail address:	<input type="text"/>

Jenkins 2.36.1 [Skip and continue as admin](#) [Save and Continue](#)

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 screenshot shows the Jenkins interface for a build named 'HelloWorld' under '#1'. On the left, there's a sidebar with links: 'Back to Project', 'Status', 'Changes', 'Console Output' (which is highlighted), 'View as plain text', 'Edit Build Information', and 'Delete build '#1''. The main content area is titled 'Console Output' with a green checkmark icon. It displays the following log output:

```
Started by user Vendra Sekar
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorld
[HelloWorld] $ /bin/sh -xe /tmp/jenkins16832578894953194148.sh
+ echo Hello world!
Hello world!
Finished: SUCCESS
```

The output is successful

The screenshot shows the Jenkins dashboard. At the top, there are buttons for 'All', '+', 'Add description', and a search bar. Below is a table with columns: S, W, Name (sorted by name), Last Success, Last Failure, and Last Duration. The 'HelloWorld' project is listed with a green checkmark icon. The 'Last Success' column shows '2 min 36 sec #1'. The 'Last Failure' column shows 'N/A'. The 'Last Duration' column shows '87 ms'. At the bottom, there are icons for 'Icon legend', 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	⌚	HelloWorld	2 min 36 sec #1	N/A	87 ms

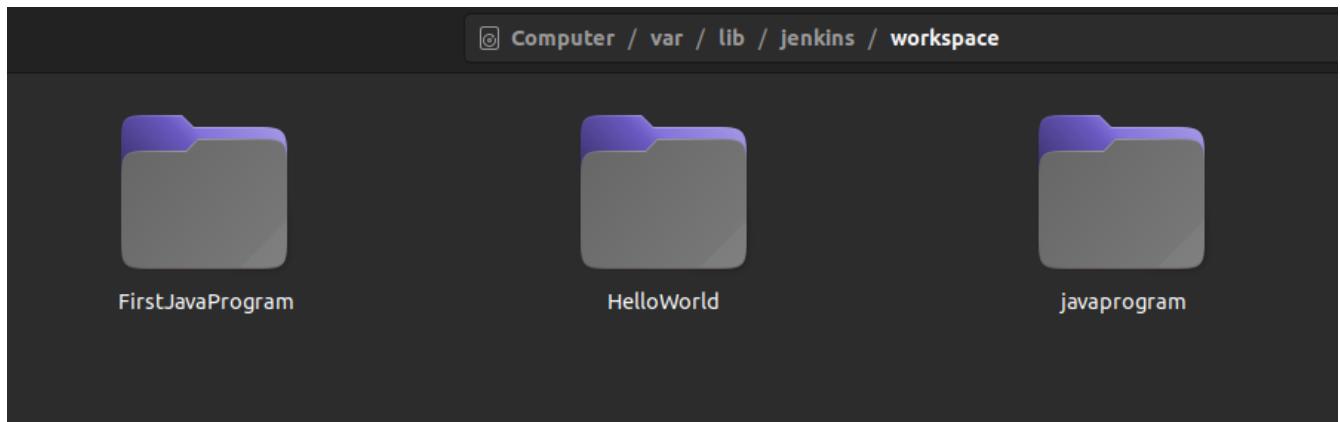
2. Exercise to create and run a java program using terminal

Create a new freestyle project

The screenshot shows the Jenkins dashboard after creating three new projects: 'FirstJavaProgram' (failed), 'HelloWorld' (successful), and 'javaprogram' (in progress). The table structure is identical to the previous screenshot. The 'FirstJavaProgram' row has a red 'X' icon and a failed status. The 'HelloWorld' row has a green checkmark icon and a successful status. The 'javaprogram' row has a blue circle icon and an 'IN PROGRESS' status. The bottom of the screen includes the same footer icons as the previous screenshot.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✗	⚡	FirstJavaProgram	N/A	6 days 5 hr #5	17 ms
✓	⌚	HelloWorld	6 days 16 hr #1	N/A	87 ms
...	⌚	javaprogram	N/A	N/A	N/A

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

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

The screenshot shows the Jenkins configuration interface for a job named 'javaprogram'. In the 'Build Steps' section, there is one step named 'Execute shell' containing the command:

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

Below the build steps is a 'Post-build Actions' section with a single button:

Save

Click on build now

The screenshot shows the Jenkins dashboard for the 'javaprogram' project. The sidebar on the left has several options: Back to Dashboard, Status, Changes, Workspace, Build Now (which is highlighted), Configure, Delete Project, and Rename. The main area is titled 'Build History' and lists the following builds:

Build	Status	Date
#4	Success	Sep 25, 2022, 4:14 PM
#3	Failure	Sep 25, 2022, 4:11 PM
#2	Failure	Sep 25, 2022, 4:10 PM
#1	Success	Sep 25, 2022, 3:48 PM

Project javaprogram

Check the output

Console Output

```
Started by user Vendra Sekar
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/javaprogram
[javaprogram] $ /bin/sh -xe /tmp/jenkins4284829222897009505.sh
+ sudo javac javahello.java
+ java javahello
Hello!!!
Finished: SUCCESS
```

The java program is successfully demonstrated on jenkins



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

Enter an item name

bash-f1
» Required field

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

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

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

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

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

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

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

Copy from autocomplete

OK

```
purpleven@purpleven:~/Documents$ cat shelljen.sh
#!/bin/bash
echo "Heyyaa!"
purpleven@purpleven:~/Documents$ 
```

```
chmod: changing permissions of 'shelljen.sh': operation not permitted
purpleven@purpleven:~/Documents$ sudo chmod 777 shelljen.sh
purpleven@purpleven:~/Documents$ ./shelljen.sh
Heyyaa!
purpleven@purpleven:~/Documents$ 
```

Build Steps

≡ Execute shell ? ×

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

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

[Advanced...](#)

[Add build step ▾](#)

Post-build Actions

[Add post-build action ▾](#)

[Save](#)

[Apply](#)

 Jenkins

Dashboard > bash-f1 > #4

[↑ Back to Project](#)

[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](#)

 Jenkins

Dashboard > bash-f1 > #4

[↑ Back to Project](#)

[Status](#)

[Changes](#)

[Console Output](#)

[View as plain text](#)

[Edit Build Information](#)

[Delete build '#4'](#)

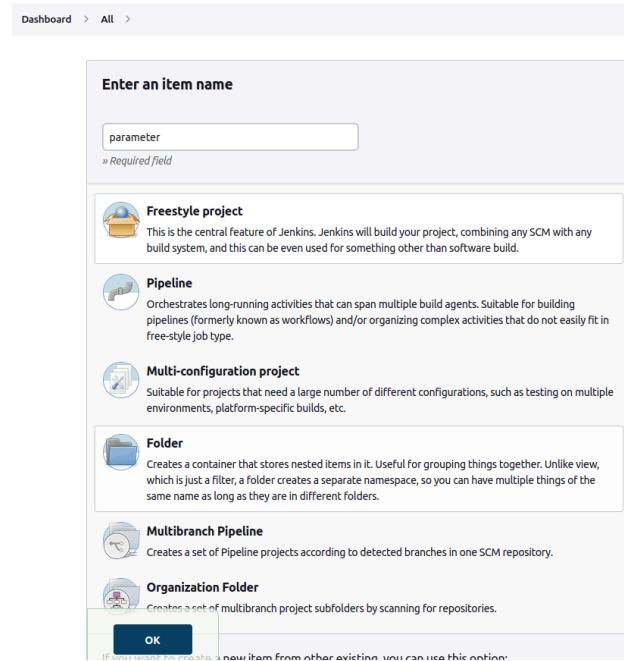
[← Previous Build](#)

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
Heyya!
Finished: SUCCESS

4. Exercise to Create a parameterized project



The screenshot shows the Jenkins Parameter configuration screen. It displays two parameters:

- String Parameter**:
 - Name**: Name
 - Default Value**: Vendra
 - Description**: (empty)
 - [Plain text] Preview**: (empty)
 - Trim the string**
- Choice Parameter**:
 - Name**: City
 - Choices**: Mumbai, Chennai, Kolkata
 - Description**: (empty)

At the bottom are 'Save' and 'Apply' buttons.

Build Steps

Execute shell [?](#) X

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](#)

[!\[\]\(43208d6b203cbb8d2e833386ceb48fa5_img.jpg\) Status](#)

[!\[\]\(e1fb41f4b2b70194bf6a365468b84fdd_img.jpg\) Changes](#)

[!\[\]\(84c418c60a6a559d256a380c0687ebe5_img.jpg\) Workspace](#)

[!\[\]\(4c42653d0f6de2f47487e45e00f6d532_img.jpg\) Build with Parameters](#)

[!\[\]\(f6662514069ff48bdef07a1000762f95_img.jpg\) Configure](#)

[!\[\]\(52bf2d3ad6161796fa8d42c289183aca_img.jpg\) Delete Project](#)

[!\[\]\(95a21ae262ab622b33baea7568c95416_img.jpg\) 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

[Add description](#)

Took 30 ms

</> 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:
  libjsr305-java libplexus-sec-dispatcher-java libplexus-utils2-java libsisu-inject-java
  libsisu-plexus-java libwagon-file-java libwagon-provider-api-java
```

```
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-amd64
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

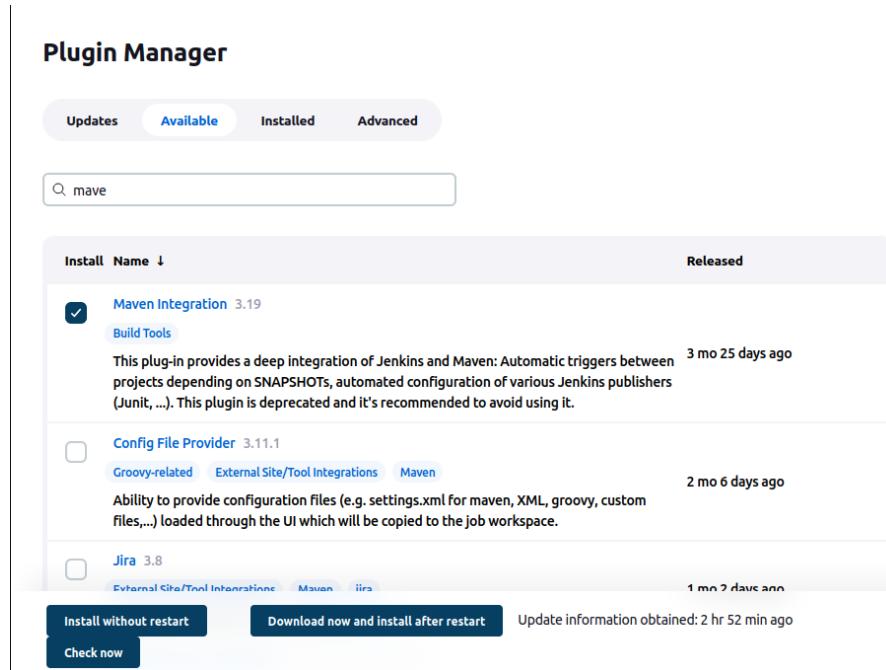
Plugin Manager

Updates Available Installed Advanced

Q maven

Install	Name ↓	Released
<input checked="" type="checkbox"/>	Maven Integration 3.19 Build Tools	3 mo 25 days ago
<input type="checkbox"/>	Config File Provider 3.11.1 Groovy-related External Site/Tool Integrations Maven	2 mo 6 days ago
<input type="checkbox"/>	Jira 3.8 External Site/Tool Integrations Maven Jira	1 mo 2 days ago

Install without restart Download now and install after restart Update information obtained: 2 hr 52 min ago
Check now



Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Javadoc

 Downloaded Successfully. Will be activated during the next boot

Maven Integration

 Installing

→ [Go back to the top page](#)

(you can start using the installed plugins right away)

→

Restart Jenkins when installation is complete and no jobs are running

[REST API](#) Jenkins 2.361.

Restart jenkins



Please wait while Jenkins is restarting ...

Your browser will reload automatically when Jenkins is ready.

Create a new project

Enter an item name

mavenProject
» Required field

Freestyle project
 This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
 Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
 Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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

Folder
 Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a Folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

OK **Organization** **Folder**

localhost:8080/job/mavenProject/configure

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

Source Code Management

None

Git [?](#)

Repositories [?](#)

Repository URL [?](#)

Credentials [?](#) [▼](#)

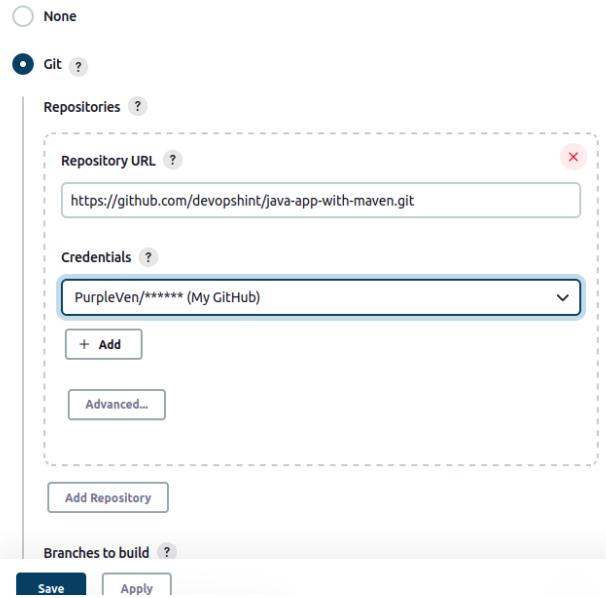
+ Add

Advanced...

Add Repository

Branches to build [?](#)

Save **Apply**



Build Triggers

- Build whenever a SNAPSHOT dependency is built [?](#)
- Schedule build when some upstream has no successful builds [?](#)
- Trigger builds remotely (e.g., from scripts) [?](#)
- Build after other projects are built [?](#)
- Build periodically [?](#)
- GitHub hook trigger for GITScm polling [?](#)
- Poll SCM [?](#)

Build Environment

- Delete workspace before build starts
[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 [?](#)

Maven Version

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

Root POM ?
my-app/pom.xml

Goals and options ?
clean package

[Advanced...](#)

Post Steps

Run only if build succeeds

Run only if build succeeds or is unstable

Run regardless of build result

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



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/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/devopshint/java-app-with-maven.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
```

```

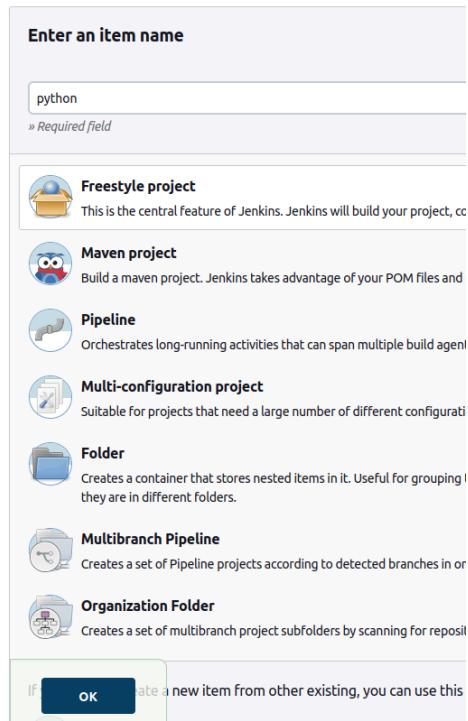
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-compress/1.11/commons-compress-1.11.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.1.1/maven-archiver-3.1.1.jar (24 kB at 46 kB/s)
[INFO] Downloading 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/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/iq80/snappy/snappy/0.4/snappy-0.4.jar (50 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

	javaprogram	2 hr 13 min #4	2 hr 17 min #5	0.0 sec	
	mavenProject	3 min 24 sec #3	28 min #1	2 min 3 sec	

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



Git ?

Repositories ?

Repository URL ? X

Credentials ? ▼

+ Add

Advanced...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ? X

Add Branch

This screenshot shows the Jenkins Git configuration interface. It includes sections for 'Repository URL' containing 'https://github.com/PurpleVen/Python.git', 'Credentials' with 'PurpleVen/******** (My GitHub)', and 'Branches to build' with '*/master'. Buttons for 'Add' and 'Advanced...' are also visible.

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:

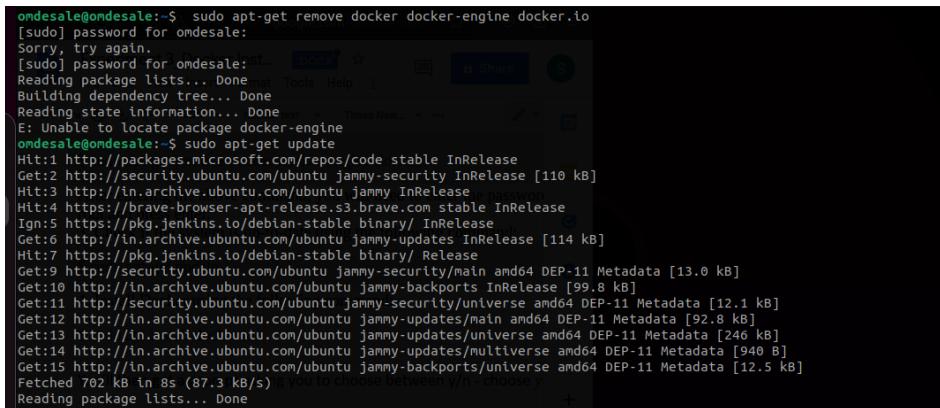
1. Open the terminal on Ubuntu.
2. Remove any [Docker files](#) that are running in the system, using the following command:

```
$ 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
```



```
omdesale@omdesale:~$ sudo apt-get remove docker docker-engine docker.io
[sudo] password for omdesale:
Sorry, try again.
[sudo] password for omdesale:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package docker-engine
omdesale@omdesale:~$ sudo apt-get update
Hit:1 http://packages.microsoft.com/repos/code stable InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 https://brave-browser-apt-release.s3.brave.com stable InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Hit:7 https://pkg.jenkins.io/debian-stable binary/ Release
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [13.0 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [12.1 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [92.8 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [92.8 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [246 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]
Get:15 http://in.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [12.5 kB]
Fetched 702 kB in 8s (87.3 kB/s)
Reading package lists... Done
```

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*

```

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 libssord-0-0 libsratom-0-0 libvidstab1.1 libzimg2
  pocketsphinx-en-us systemd-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 fuse-zfs | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd docker-to-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-to-pigz 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 2m1n 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.

```

ondesale@ondesale:~$ sudo snap install docker
docker 20.10.14 from Canonical** installed
ondesale@ondesale:~$ docker --version
docker: '-version' is not a docker command.
See 'docker --help'.
ondesale@ondesale:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu4
ondesale@ondesale:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:62af9efds15a25f84961b70f973a798d2eca956b1b2b026d0a4a63a3b0b6a3f2
Status: Downloaded newer image for hello-world:latest

Hello from Docker! This message shows that your installation appears to be working correctly.
This message was generated using the Docker Hub.
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

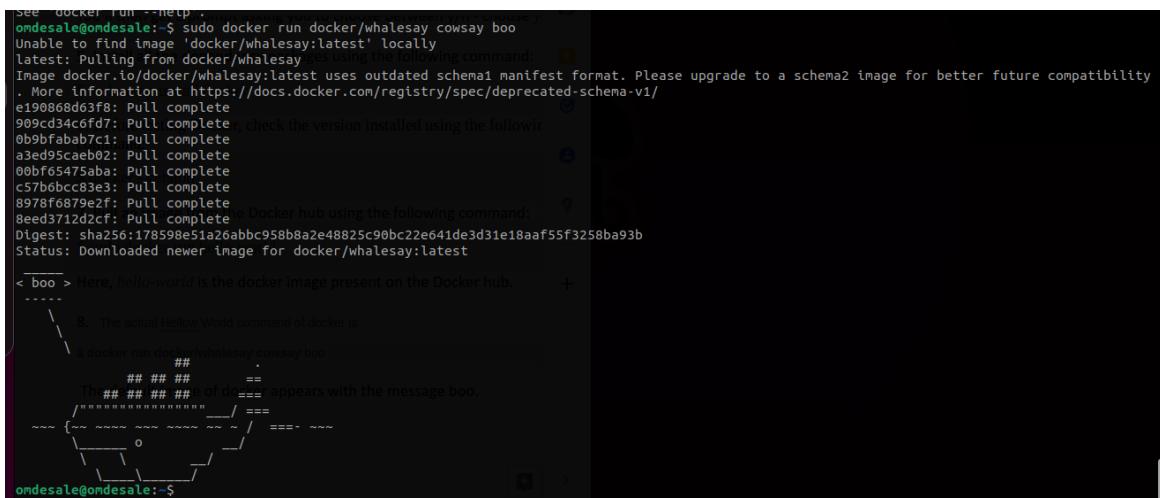
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

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
```



```
see docker run --net=host
omdesale@omdesale:~$ sudo docker run docker/whalesay cowsay boo
Unable to find image 'docker/whalesay:latest' locally
latest: Pulling from docker/whalesay
Digest: sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest

< boo > Here, hello-world is the docker image present on the Docker hub.

B... The actual Hellow World command of docker is
$ docker run docker/whalesay cowsay boo
## ## ##
## ## ## ## of d... appears with the message boo.
/   / ===
{~~ ~~~ ~~~ ~~~ ~~~ / ==-
 \_ \_ \_ \_ \_ \_ / -/ ==-
 \_ \_ \_ \_ \_ \_ / -/ ==-
omdesale@omdesale:~$
```

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:

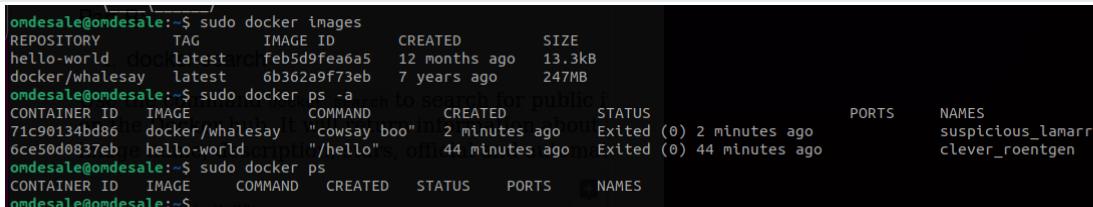
```
$ sudo docker images
```

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
```



REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	feb5d9fea6a5	12 months ago	13.3kB
docker/whalesay	latest	6b362a9f73eb	7 years ago	247MB

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
71c90134bd86	docker/whalesay	"cowsay boo"	2 minutes ago	Exited (0)	2 minutes ago	suspicious_lamarr
6ce500837eb7	hello-world	"./hello"	44 minutes ago	Exited (0)	44 minutes ago	clever_roentgen

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
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
mysql	MySQL is a widely used, open-source relation...	13234	[OK]	
mariadb	MariaDB Server is a high performing open sou...	5062	[OK]	
phpmyadmin	phpMyAdmin - A web interface for MySQL and M...	640	[OK]	
percona	Percona Server is a fork of the MySQL relati...	588	[OK]	
bitnami/mysql	Bitnami MySQL Docker Image	77		[OK]
dbatools/mysql-backup	Back up mysql databases to... anywhere!	70		
linuxserver/mysql-workbench	Stable, multi-threaded MySQL WorkBench	44		
linuxserver/mysql	A MySQL container, brought to you by LinuxSe...	37		
ubuntu/mysql	MySQL open source fast, stable, multi-thread...	36		
circleci/mysql	MySQL is a widely used, open-source relation...	27		
google/mysql	MySQL server for Google Compute Engine	21		[OK]
rapidfort/mysql	RapidFort optimized, hardened image for MySQL	13		
bitnami/mysqld-exporter	Docker image for mysqld-exporter	3		
ibmcom/mysql-s390x	Docker image for mysql-s390x	2		
newrelic/mysql-plugin	New Relic Plugin for monitoring MySQL database	1		[OK]
vitess/mysqlctld	vitess/mysqlctld	1		[OK]
hashicorp/mysql-portworx-demo	MySQL service images for Docksal - https://d...	0		
docksal/mysql	MySQL service images for Docksal - https://d...	0		
mirantis/mysql	Docker pull	0		
cimg/mysql		0		
drud/mysql	drud/mysql	0		
silint/mysql-backup-restore	Simple docker image to perform mysql backups...	0		[OK]
corposops/mysql	https://github.com/corposops/docker-images/	0		
drud/mysql-local-57	drud/mysql-local-57	0		
drud/mysql-docker-local-57	drud/mysql-docker-local-57	0		
	This repo has been deprecated, new tags are ...	0		

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
```

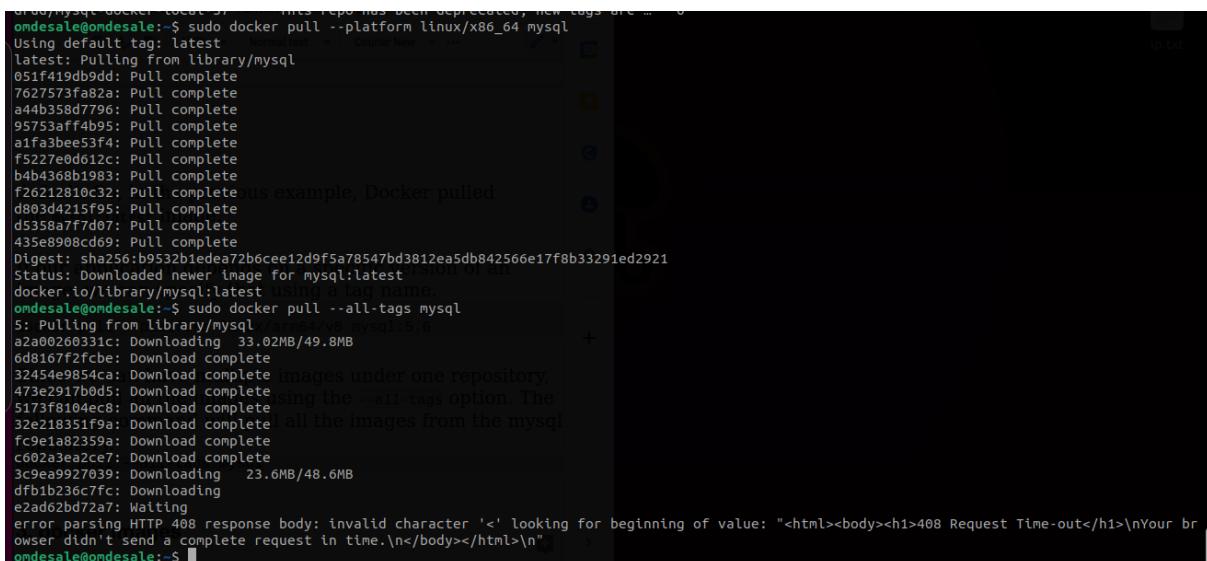
Tags are used to identify images inside a repository. If we don't specify a tag Docker engine uses the `:latest` tag by default. So, in the previous example, Docker pulled the `mysql:latest` image.

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
```

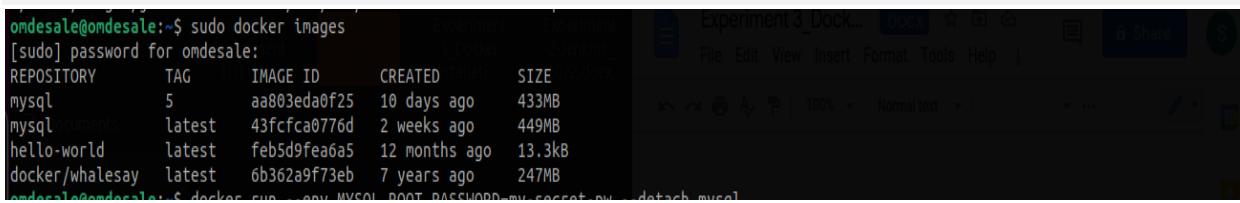


```
omdesale@omdesale:~$ sudo docker pull --platform linux/x86_64 mysql
Using default tag: latest
latest: Pulling from library/mysql
051f419db9d: Pull complete
7627573fa82a: Pull complete
a44b358d7796: Pull complete
95753aff4b95: Pull complete
a1fa3beef3f4: Pull complete
f5227e0d612c: Pull complete
b4b4368b1983: Pull complete
f26212810c32: Pull complete
d803d4215f95: Pull complete
d5358a77d607: Pull complete
435e8908cd69: Pull complete
Digest: sha256:b9532b1dea72b0cce12d9f5a78547bd3812ea5db842566e17f8b33291ed2921
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest using --tag name.
omdesale@omdesale:~$ sudo docker pull --all-tags mysql
5: Pulling from library/mysql
a2a00260031c: Downloading 33.02MB/49.8MB
6d8167ff2fcbe: Download complete
32454e9954ca: Download complete images under one repository.
473e2917b0d5: Download complete using the --all-tags option. The
5173f8104ec8: Download complete
32e218351f9a: Download complete all the images from the mysql
fc9e1a82359a: Download complete
c602a3ea2ce7: Download complete
3c9ea9927039: Downloading 23.6MB/48.6MB
dfb1b236c7fc: Downloading
e2ad62bd72a7: Waiting
error parsing HTTP 408 response body: invalid character '<' looking for beginning of value: "<html><body><h1>408 Request Time-out</h1>\nYour br
omdesale@omdesale:~$
```

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
```



```
omdesale@omdesale:~$ sudo docker images
[sudo] password for omdesale:
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
mysql           5        aa803eda0f25  10 days ago   433MB
mysql           latest   43fcfa0776d  2 weeks ago   449MB
hello-world     latest   feb5d9fea6a5  12 months ago  13.3kB
docker/whalesay latest   6b362a9f73eb  7 years ago   247MB
omdesale@omdesale:~$ docker run -e MYSQL_ROOT_PASSWORD=mu-secret -e MYSQL_DATABASE=docker mysql
```

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.

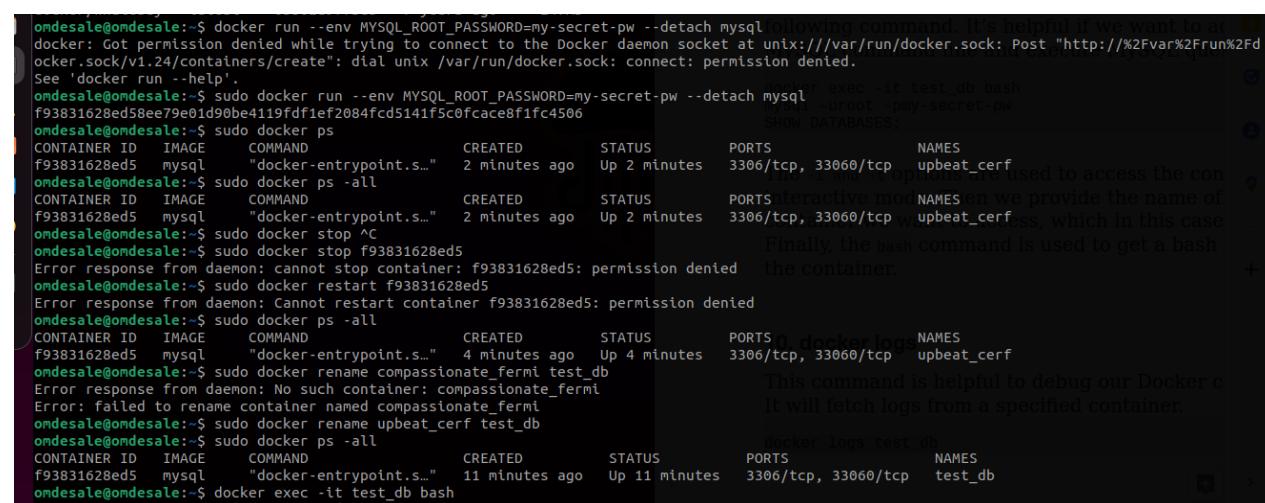
5. docker ps

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
```



The screenshot shows a terminal session on a Linux system named 'ondesale'. The user runs several Docker commands:

- `docker run --env MYSQL_ROOT_PASSWORD=my-secret-pw --detach mysql`: Fails with permission denied due to the Docker daemon socket being owned by root.
- `sudo docker run --env MYSQL_ROOT_PASSWORD=my-secret-pw --detach mysql`: Runs successfully, creating a container named 'upbeat_cerf'.
- `sudo docker ps`: Lists the running container 'upbeat_cerf'.
- `sudo docker ps -all`: Lists both the running container 'upbeat_cerf' and a stopped container 'f93831628ed5'.
- `sudo docker stop ^C`: Stops the running container.
- `sudo docker stop f93831628ed5`: Fails with permission denied.
- `sudo docker restart f93831628ed5`: Fails with permission denied.
- `sudo docker rename compassionate_fermi test_db`: Fails with 'No such container' error.
- `sudo docker rename upbeat_cerf compassionate_fermi`: Fails with 'Failed to rename container' error.
- `sudo docker ps -all`: Lists the stopped container 'compassionate_fermi'.
- `sudo docker logs test_db`: Fetches logs from the 'test_db' container.
- `docker logs test_db`: Fetches logs from the 'test_db' container.
- `sudo docker exec -it test_db bash`: Enters a bash shell in the 'test_db' container.

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 f8c52bedeccc
```

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 f8c52bedeccc
```

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
```

The screenshot shows a terminal window with the command `sudo docker logs test_db` running. The output is a stream of log messages from a MySQL container. Key messages include:

- [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.30-1.el8 started.
- [Note] [Entrypoint]: Switching to dedicated user 'mysql'
- [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.30-1.el8 started.
- [Note] [Entrypoint]: Initializing database files
- [Warning] [MY-01068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
- [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.30) initializing of server in progress as process 80
- [System] [MY-013576] [InnoDB] InnoDB initialization has started.
- [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
- [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
- [Note] [Entrypoint]: Database files initialized
- [Note] [Entrypoint]: Starting temporary server
- [Warning] [MY-01068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
- [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.30) starting as process 129
- [System] [MY-013576] [InnoDB] InnoDB initialization has started.
- [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
- [Warning] [MY-01068] [Server] CA certificate ca.pem is self signed.
- [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
- [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
- [System] [MY-011323] [Server] X Plugin ready for connections. Socket: /var/run/mysqld/mysqld.sock
- [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.30' socket: '/var/run/mysql/mysqld.sock' port: 0 MySQL Community Server GPL.
- [Note] [Entrypoint]: temporary server started.
- '/var/lib/mysql/mysql.sock' -> '/var/run/mysql/mysqld.sock'
- Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
- Warning: Unable to load '/usr/share/zoneinfo/Leapseconds' as time zone. Skipping it.
- Warning: Unable to load '/usr/share/zoneinfo/tzdata.zl' as time zone. Skipping it.
- Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
- Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.

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# or docker 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.

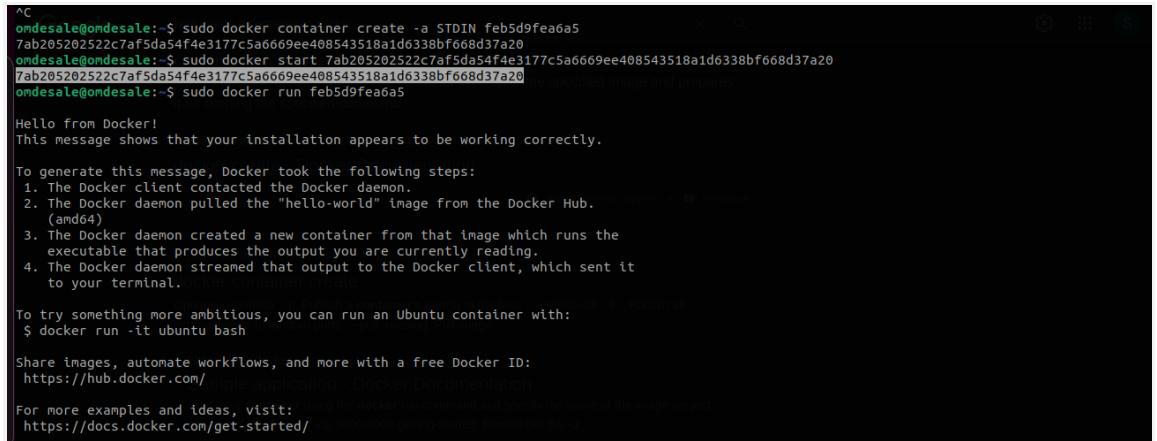
```
docker container start my_container — Start an existing container.
```

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.
```



```
^C
onedesale@ondesale:~$ sudo docker container create -a STDIN feb5d9fea6a5
7ab205202522c7af5da54f4e3177c5a6669ee408543518a1d6338bf668d37a20
onedesale@ondesale:~$ sudo docker start 7ab205202522c7af5da54f4e3177c5a6669ee408543518a1d6338bf668d37a20
7ab205202522c7af5da54f4e3177c5a6669ee408543518a1d6338bf668d37a20
onedesale@ondesale:~$ sudo docker run feb5d9fea6a5

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/
```

```
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.

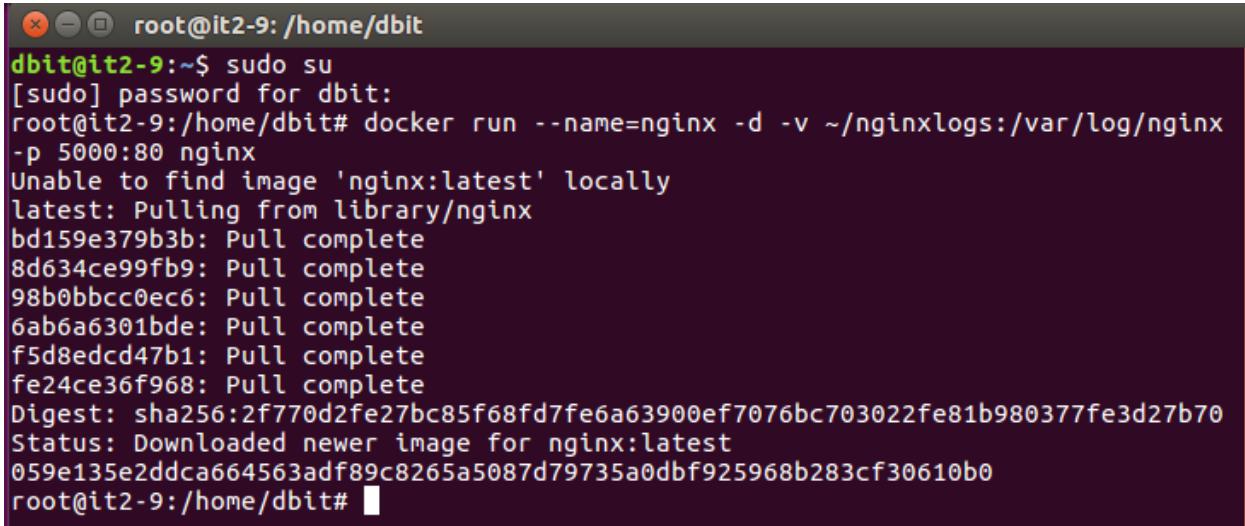
Docker volumes can be used to share files between a host system and the Docker container. For example, let's say you wanted to use the official Docker Nginx image and keep a permanent copy of Nginx's log files to analyze later. By default, the nginx Docker image will log to the /var/log/nginx directory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



```
root@it2-9: /home/dbit
dbit@it2-9:~$ sudo su
[sudo] password for dbit:
root@it2-9:/home/dbit# docker run --name=nginx -d -v ~/nginxlogs:/var/log/nginx
-p 5000:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
bd159e379b3b: Pull complete
8d634ce99fb9: Pull complete
98b0bbcc0ec6: Pull complete
6ab6a6301bde: Pull complete
f5d8edcd47b1: Pull complete
fe24ce36f968: Pull complete
Digest: sha256:2f770d2fe27bc85f68fd7fe6a63900ef7076bc703022fe81b980377fe3d27b70
Status: Downloaded newer image for nginx:latest
059e135e2ddca664563adf89c8265a5087d79735a0dbf925968b283cf30610b0
root@it2-9:/home/dbit#
```

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

- **--name=nginx** names the container so we can refer to it more easily.
- **-d** detaches the process and runs it in the background. Otherwise, we would just be watching an empty Nginx prompt and wouldn't be able to use this terminal until we killed Nginx.
- **-v ~/nginxlogs:/var/log/nginx** sets up a bind mount volume that links the `/var/log/nginx` directory from inside the Nginx container to the `~/nginxlogs` directory on the host machine. Docker uses a `:` to split the host's path from the container path, and the host path always comes first.

- `-p 5000:80` sets up a port forward. The Nginx container is listening on port 80 by default. This flag maps the container's port 80 to port 5000 on the host system.
- `nginx` specifies that the container should be built from the Nginx image, which issues the command `nginx -g "daemon off"` to start Nginx.

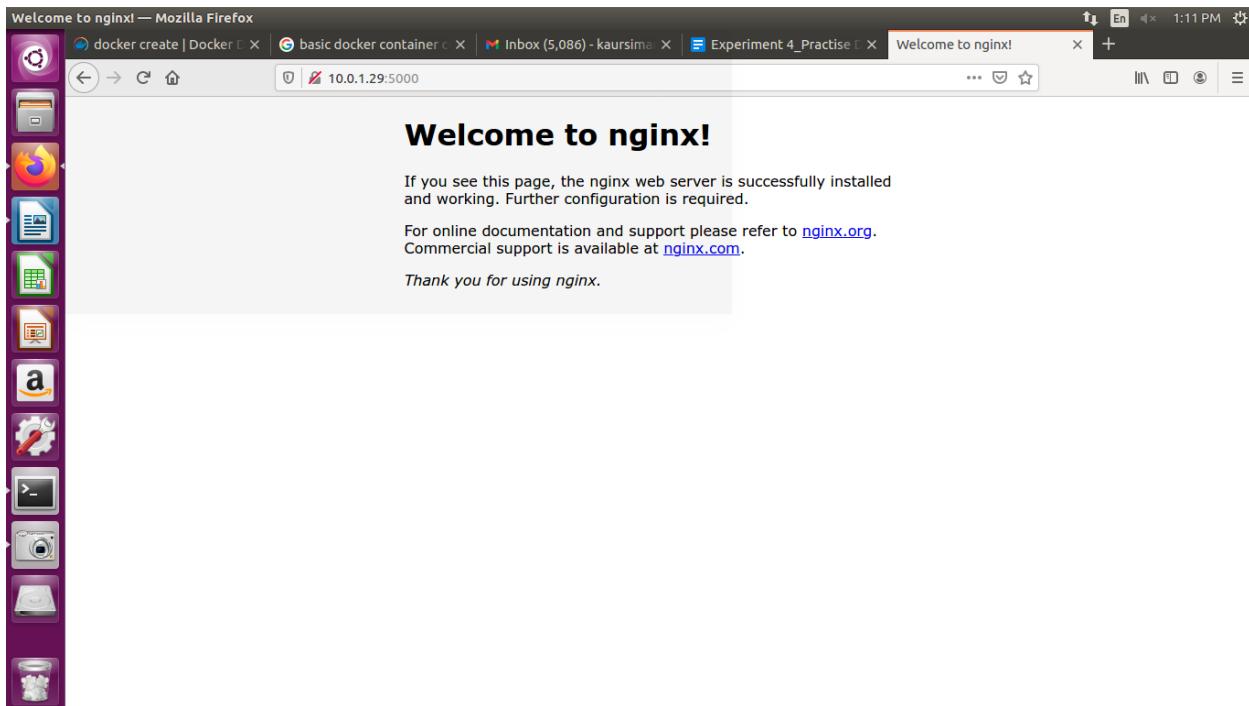
Note: The `-v` flag is very flexible . It can bindmount or name a volume with just a slight adjustment in syntax. If the first argument begins with a `/` or `~`, you're creating a bindmount. Remove that, and you're naming the volume.

- `-v /path:/path/in/container` mounts the host directory, `/path` at the `/path/in/container`
- `-v path:/path/in/container` creates a volume named `path` with no relationhip 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 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0" "-"
10.0.3.246 - - [07/Oct/2022:07:41:16 +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 "-" "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 /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
```

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

root@9effec53da72:/#
root@9effec53da72:/# cat /datavolume1/Example1.txt
Example1
root@9effec53da72:/# exit
exit
root@it2-9:/home/dbit#
```

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:/datavolume2 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:/datavolume2 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
```

```
root@it2-9:/home/dbitz# docker volume ls
DRIVER          VOLUME NAME
local           0fe006f28cff089752da6af4eccd4ca029dfb5ee740a779837b7f09d5b4bb69
local           3c2b726349aa860d8a2d2be04953a9680688cc1c34a5b4797daaed3f7755
local           7ce1b1b717dec0c1bafe3a08d364ad146cbaf7d0771a0028f72c28c3962b
local           9a8572c883e252dd1b26cc0d8fdd1ee7c459c3e3c238322f824f89c7ed3a
local           9e84015726349aa860d8a2d2be04953a9680688cc1c34a5b4797daaed3f7755
local           85f58de08777465311d9119c23e0c2538eed30160e6f74702e6d35b24bc75b5
local           99f533799dc55c19fa7c017ec229fb392fc3cbb1e8040fd3d0081be5f3c815
local           96439a05d40fb915fdb92dc1ed252fcf3c6a2431a703ec587b769629d719
local           DataVolume1
local           DataVolume2
local           c13b2290a5b1ce4b3cffb37accecde3b09abe3e5f3cd9d5f6f7bc37e482a6240
local           d0c0a135726349aa860d8a2d2be04953a9680688cc1c34a5b4797daaed3f7755
local           d7e39d50d0e405d74b9159cf19ca46ead9arf0f0bf6346b1731d95e7477555dd
local           dddc74d5c57960cccb135f4c40e8929e57da015be5e04f40fc738653a3afeb5
local           df592922c325b2d8ff4206be42c3776cc543ec2e288dd411d54161bafe3ac719
root@it2-9:/home/dbitz#
```

VOLUME NAME

Output DRIVER local

DataVolume2

And we can use docker volume rm to remove it:

```
#docker volume rm DataVolume2
```

```
root@it2-9:/home/dbitz# docker volume ls
DRIVER          VOLUME NAME
local           0fe006f28cff089752da6af4eccd4ca029dfb5ee740a779837b7f09d5b4bb69
local           3c2b726349aa860d8a2d2be04953a9680688cc1c34a5b4797daaed3f7755
local           7ce1b1b717dec0c1bafe3a08d364ad146cbaf7d0771a0028f72c28c3962b
local           9a8572c883e252dd1b26cc0d8fdd1ee7c459c3e3c238322f824f89c7ed3a
local           9e84015726349aa860d8a2d2be04953a9680688cc1c34a5b4797daaed3f7755
local           85f58de08777465311d9119c23e0c2538eed30160e6f74702e6d35b24bc75b5
local           99f533799dc55c19fa7c017ec229fb392fc3cbb1e8040fd3d0081be5f3c815
local           96439a05d40fb915fdb92dc1ed252fcf3c6a2431a703ec587b769629d719
local           DataVolume1
local           DataVolume2
local           c13b2290a5b1ce4b3cffb37accecde3b09abe3e5f3cd9d5f6f7bc37e482a6240
local           d6c0a135726349aa860d8a2d2be04953a9680688cc1c34a5b4797daaed3f7755
local           d7e39d50d0e405d74b9159cf19ca46ead9arf0f0bf6346b1731d95e7477555dd
local           dddc74d5c57960cccb135f4c40e8929e57da015be5e04f40fc738653a3afeb5
local           df592922c325b2d8ff4206be42c3776cc543ec2e288dd411d54161bafe3ac719
root@it2-9:/home/dbitz# docker volume rm DataVolume2
Error response from daemon: remove DataVolume2: volume is in use - [d464c07af51744212d935c85f675121201a7ec142c613a0ee333e27c2101b2ba]
root@it2-9:/home/dbitz#
```

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

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

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

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

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

```
root@it2-9:/home/dbit# docker run -ti --rm -v DataVolume3:/var ubuntu
root@6d6698c874da:#
root@6d6698c874da:#
root@6d6698c874da:#
exit
root@it2-9:/home/dbit#
```

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

Exit the current container:

```
$exit
```

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

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

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

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

```
root@it2-9:/home/dbit# docker run --rm -v DataVolume3:/datavolume3 ubuntu ls datavolume3  
backups  
cache  
lib  
local  
lock  
log  
mail  
opt  
run  
spool  
tmp  
root@it2-9:/home/dbit#
```

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 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:

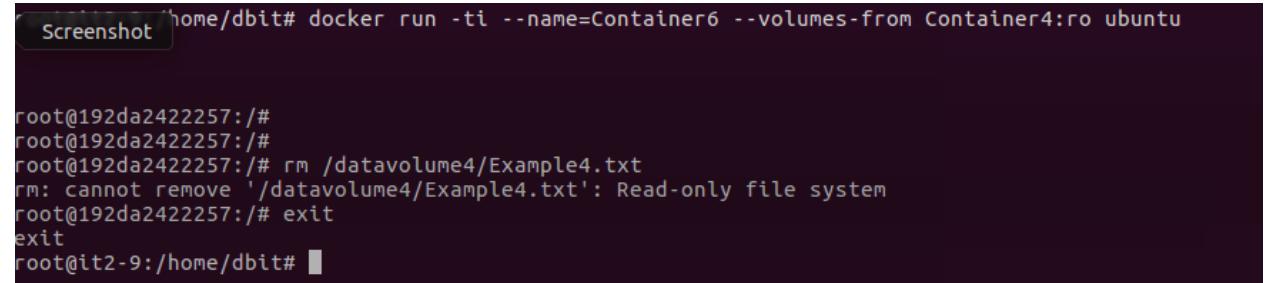
```
#docker run -ti --name=Container6 --volumes-from Container4:ro ubuntu
```

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
```



A screenshot of a terminal window titled "Screenshot". The window shows a root shell on a Docker container. The user runs the command "rm /datavolume4/Example4.txt", which fails with the error "rm: cannot remove '/datavolume4/Example4.txt': Read-only file system". The user then exits the container with "exit".

```
root@192da2422257:/#
root@192da2422257:/#
root@192da2422257:/# rm /datavolume4/Example4.txt
rm: cannot remove '/datavolume4/Example4.txt': Read-only file system
root@192da2422257:/# exit
exit
root@it2-9:/home/dbit#
```

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

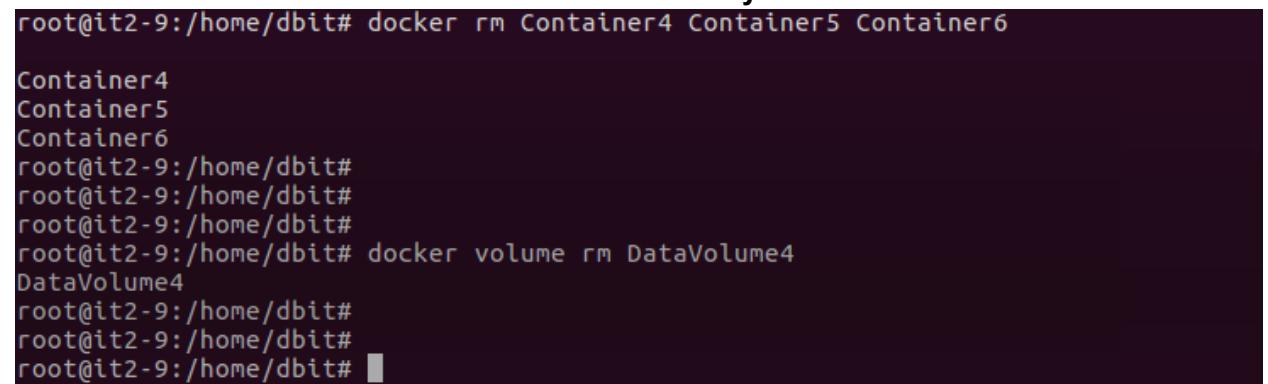
```
$exit
```

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.



A screenshot of a terminal window showing the cleanup process. The user runs "docker rm Container4 Container5 Container6", which removes the three containers. Then, they run "docker volume rm DataVolume4", which removes the data volume. The terminal shows the names of the removed containers and the success message for the volume removal.

```
root@it2-9:/home/dbit# docker rm Container4 Container5 Container6
Container4
Container5
Container6
root@it2-9:/home/dbit#
root@it2-9:/home/dbit#
root@it2-9:/home/dbit#
root@it2-9:/home/dbit# docker volume rm DataVolume4
DataVolume4
root@it2-9:/home/dbit#
root@it2-9:/home/dbit#
root@it2-9:/home/dbit#
```

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 volumel
volumel
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           volumel
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           volumel
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:
volumel
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      7cea1bb717dec0c1bafa338fa908d364ad146cba7d6771a0028f72c28c3962b
local      9a8572c883e2526d1b2b6cc08fdde1ee7ca459c33e2383225f824f89c77ed3a
local      9c8455b36db181a529b37d52543f6840e23137cf3e28a8bd62369314bc334e93
local      85f58db8777d65313d813823e0c2538ead3016be23e6f74762e6d35b24bc75b5
local      99f533f99dcc55c19fa7c017ec22d9b83928cc8b1e8040fd53d001be5f03c815
local      96439a305d40fb915dfb92d0c1ed2252fcf3c6a2431a703ec5878769629d719
local      DataVolume2
local      c13b2290a5b1ce4b3cffb37accecde3b69abe3e5f3cd9d5f6f7bc37e482a6240
local      d6c0a135726349aa860d8aad2be04953a9680688cc1c34a5b4797daaed3f7755
local      d7e39d50d0e405d74b9159cf19ca46ead9afdf0bf634681731d95e7477555dd
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](#).

The screenshot shows the Oracle Java SE Downloads page. On the left, there's a sidebar with links like Java SE, Java EE, Java ME, etc. The main content area has tabs for Overview, Downloads (which is selected), Documentation, Community, Technologies, and Training. Below the tabs, it says "Java SE Downloads". There are two download options: "Java Platform (JDK) 8u121" and "NetBeans with JDK 8". The "Java Platform (JDK) 8u121" option is highlighted with a red box and a red arrow pointing to its "DOWNLOAD" button with the text "click on this button".

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

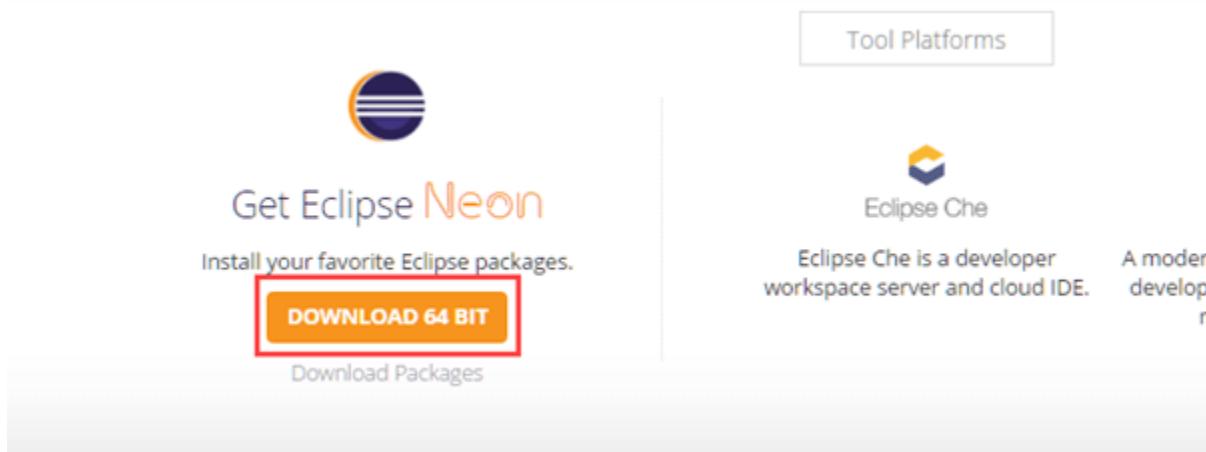
```
C:\Users\Krishna Rungta>java
Usage: java [-options] class [args...]
              (to execute a class)
      or  java [-options] -jar jarfile [args...]
              (to execute a jar file)
where options include:
  -d32      use a 32-bit data model if available
  -d64      use a 64-bit data model if available
  -server    to select the "server" VM
              The default VM is server.

  -cp <class search path of directories and zip/jar files>
  -classpath <class search path of directories and
              A ; separated list of directories,
              and ZIP archives to search for class files
  -D<name>=<value>
              set a system property
  -verbose:[class|gc|jni]
              enable verbose output
  -version     print product version and exit
  -version:<value>
              Warning: this feature is deprecated and will be removed
              in a future release
```

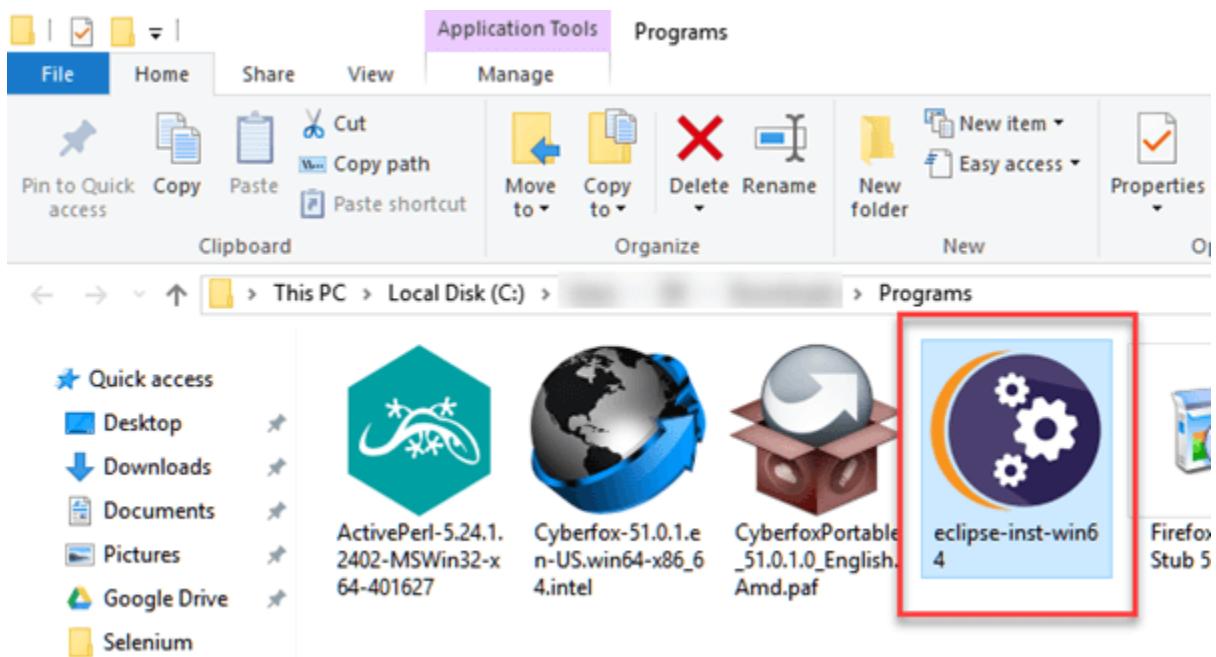
You should see this output

Step 2 – Install Eclipse IDE

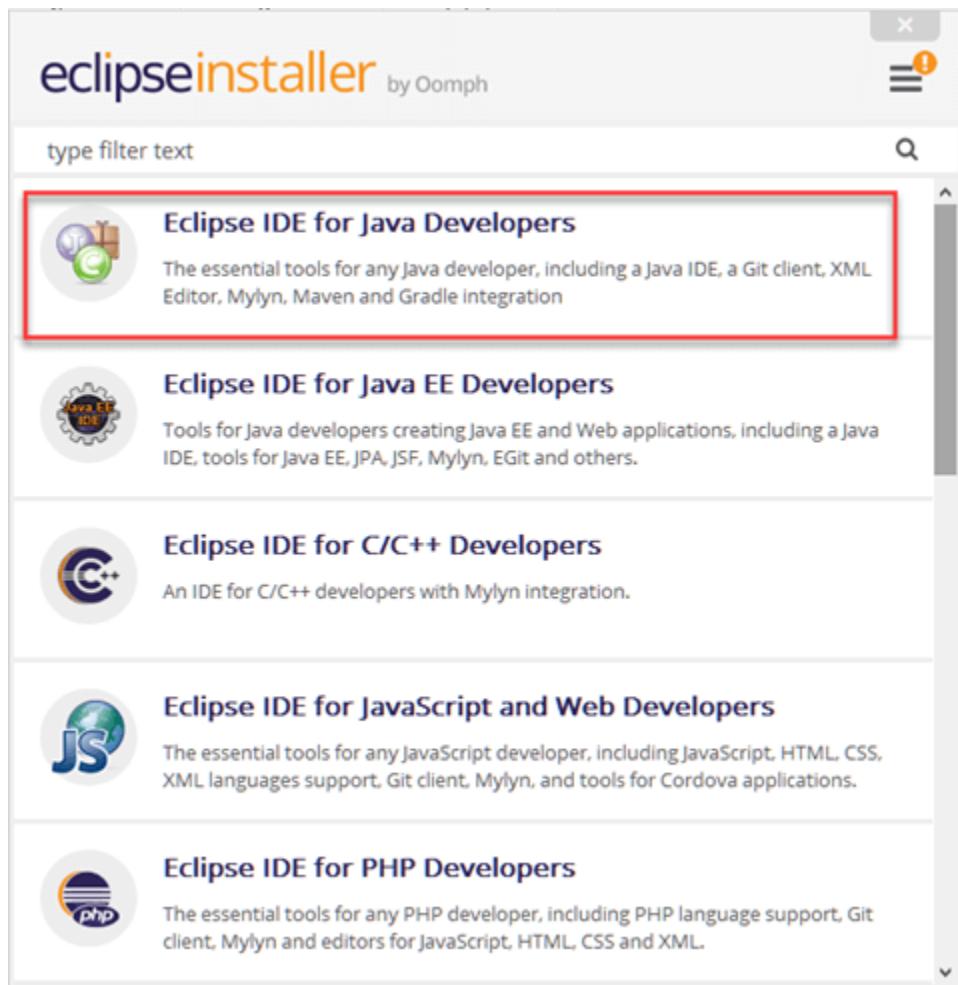
Download latest version of “**Eclipse IDE for Java Developers**” [here](#). Be sure to choose correctly between Windows 32 Bit and 64 Bit versions.



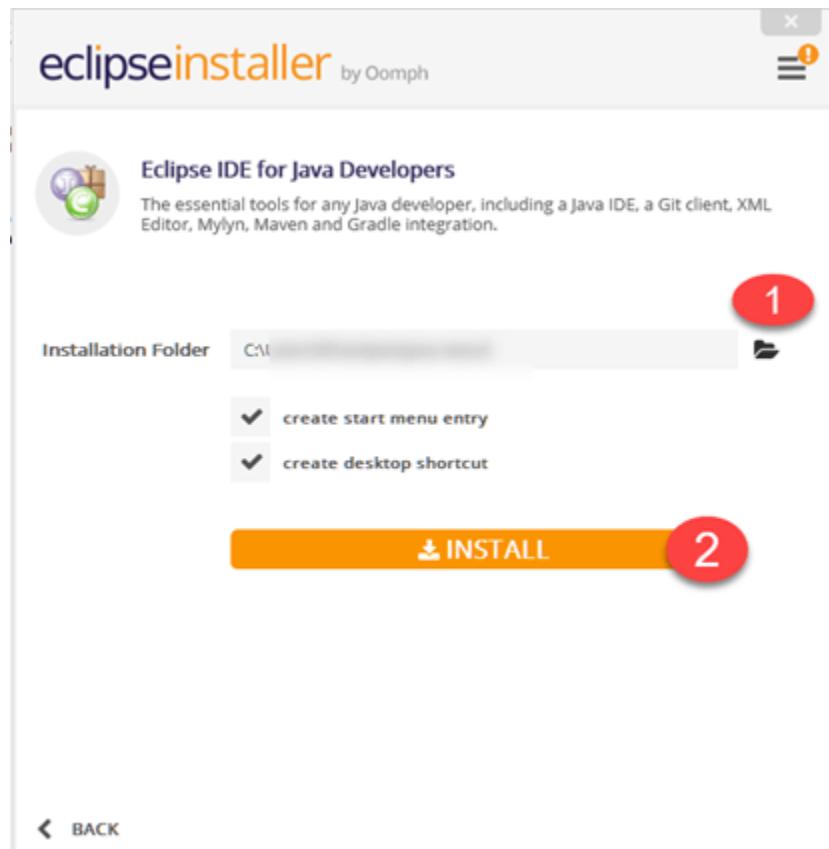
You should be able to download an exe file named “eclipse-inst-win64” for Setup.



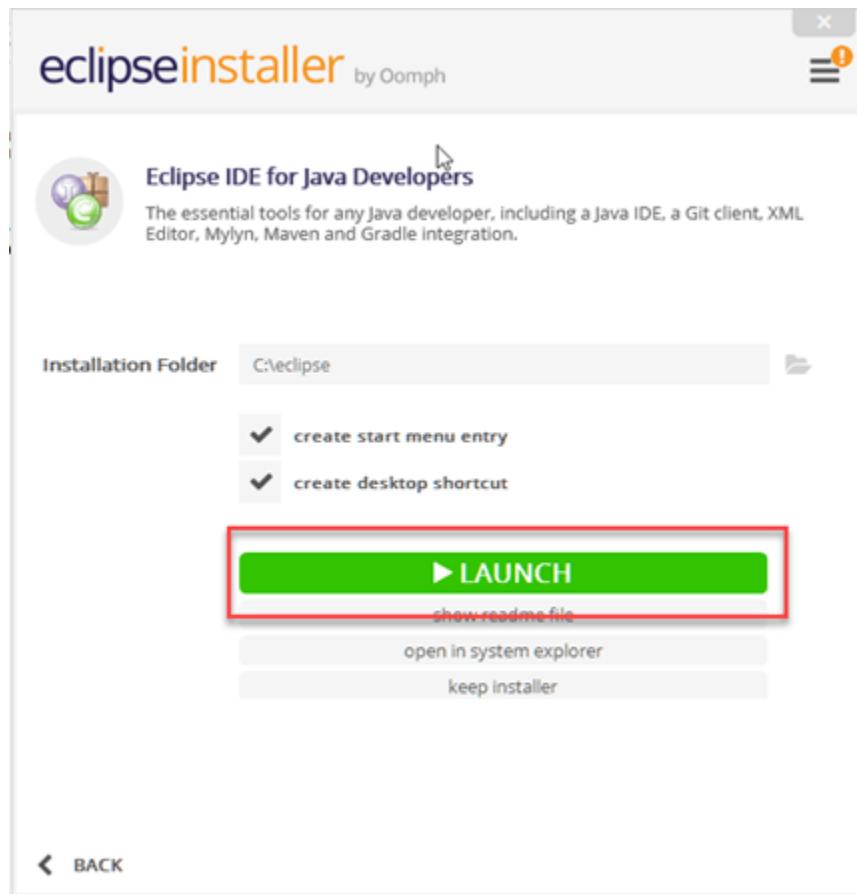
Double-click on file to Install the Eclipse. A new window will open. Click Eclipse IDE for Java Developers.



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



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



This will start eclipse neon IDE for you.

Step 3 – Download the Selenium Java Client Driver

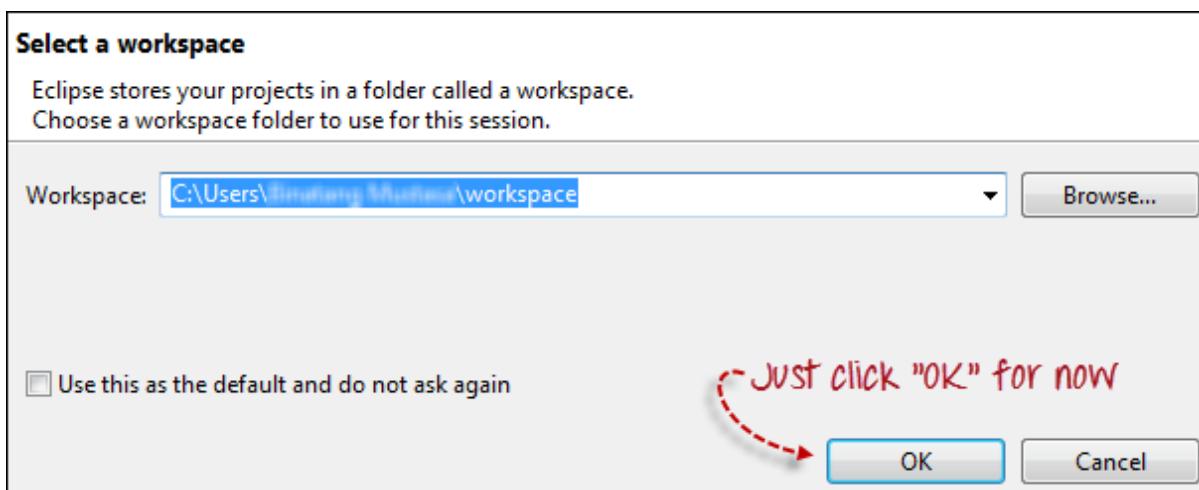
You can download **Selenium Webdriver for Java Client Driver** [here](#). You will find client drivers for other languages there, but only choose the one for Java.

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

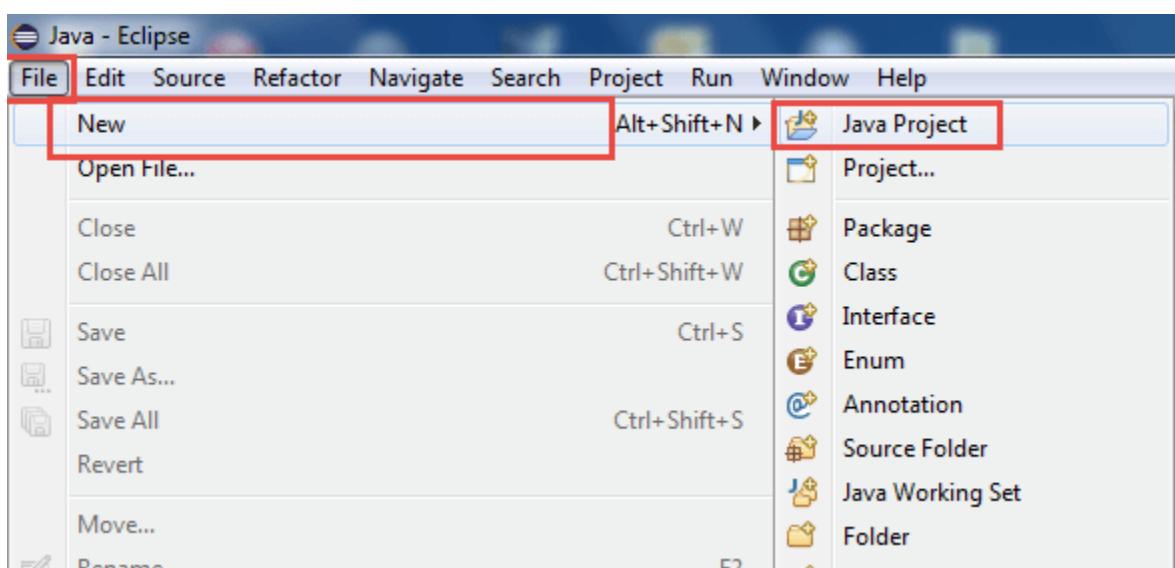
This download comes as a ZIP file named “selenium-3.14.0.zip”. For simplicity of Selenium installation on Windows 10, extract the contents of this ZIP file on your C drive so that you would have the directory “C:\selenium-3.14.0\”. This directory contains all the JAR files that we would later import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

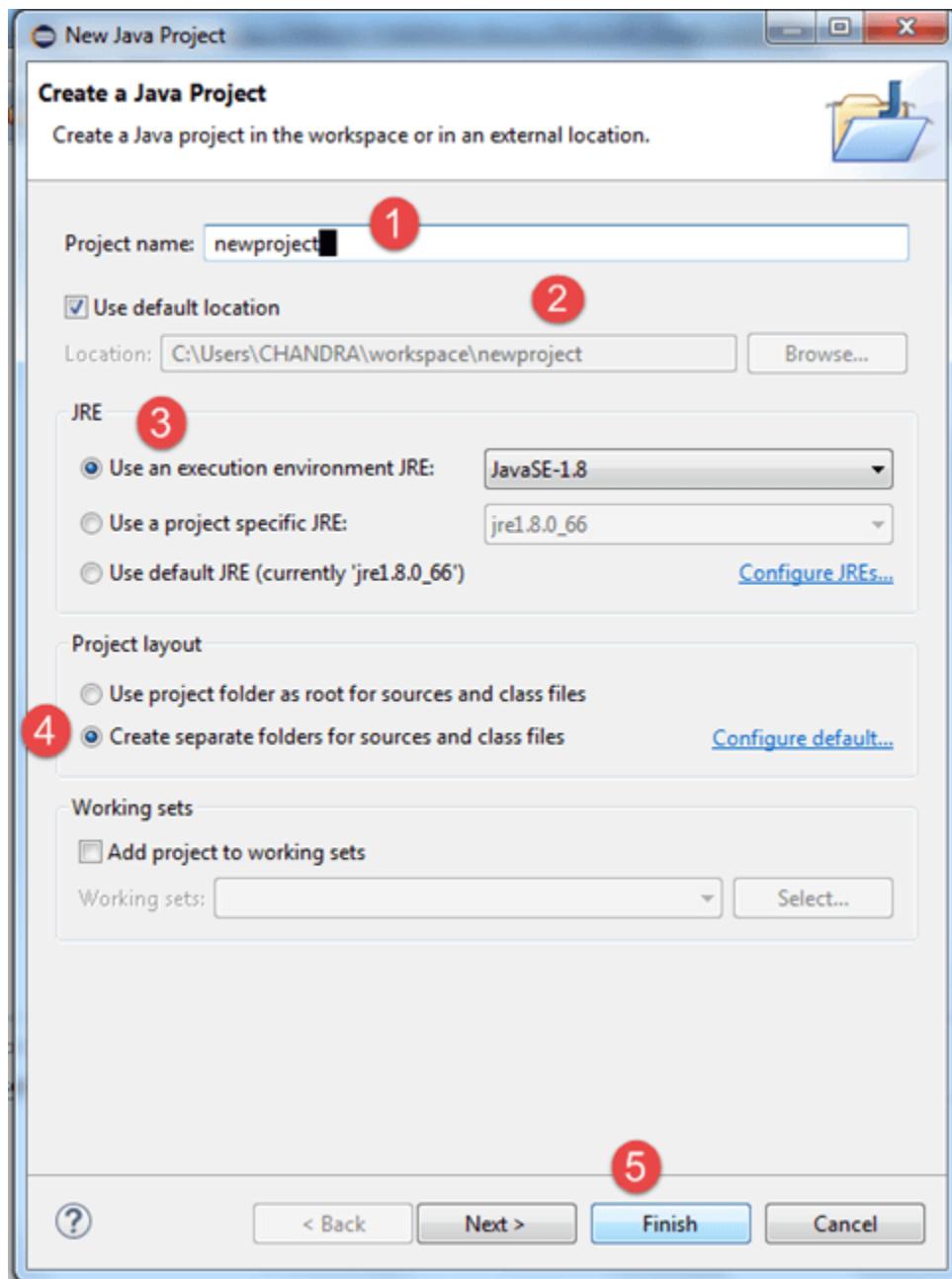


3. Create a new project through File > New > Java Project. Name the project as “newproject”.



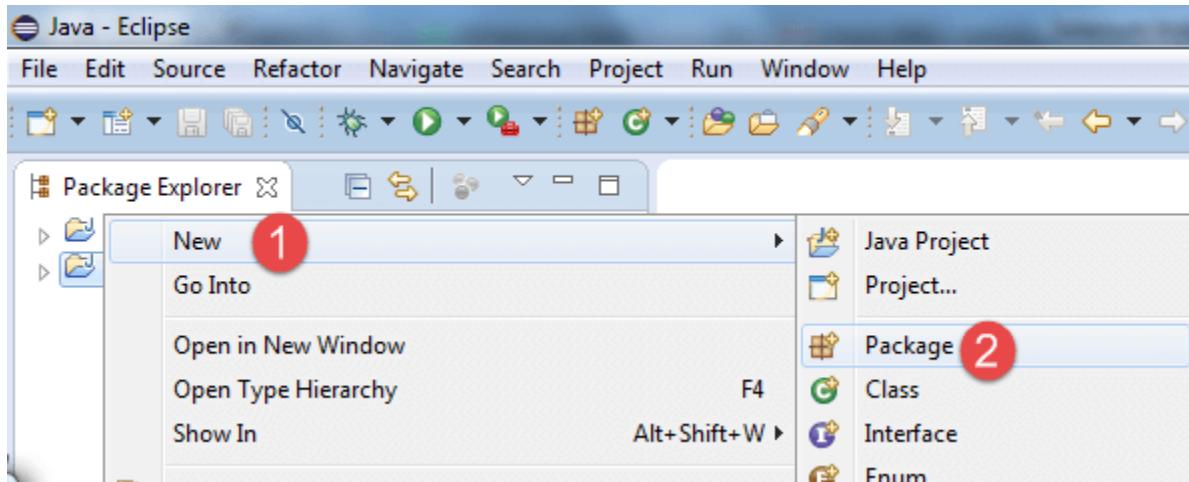
A new pop-up window will open enter details as follow

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



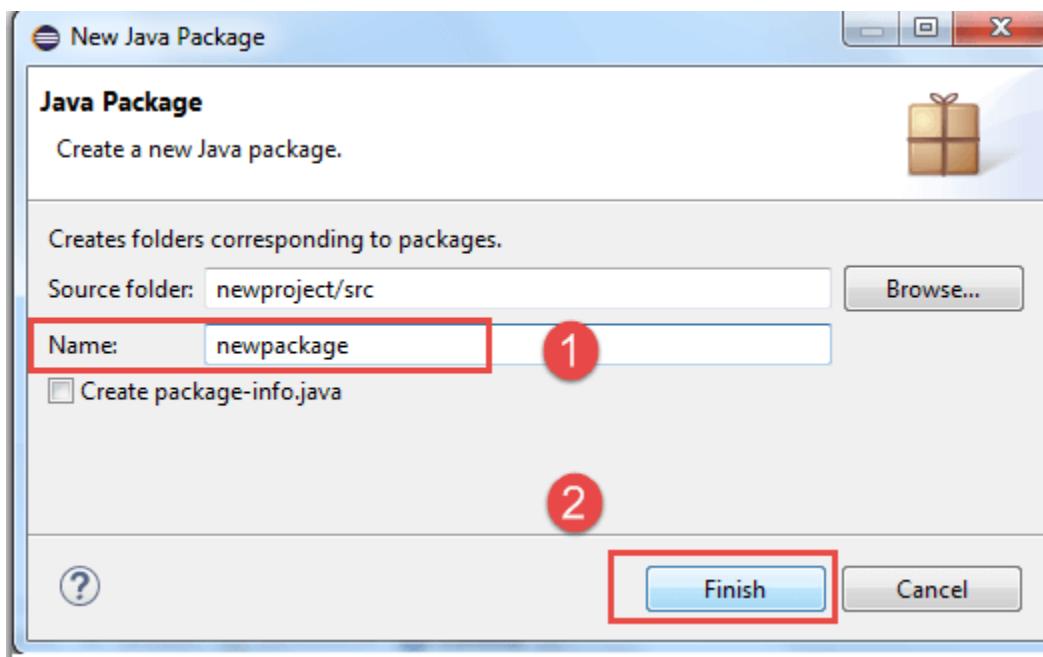
4. In this step,

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

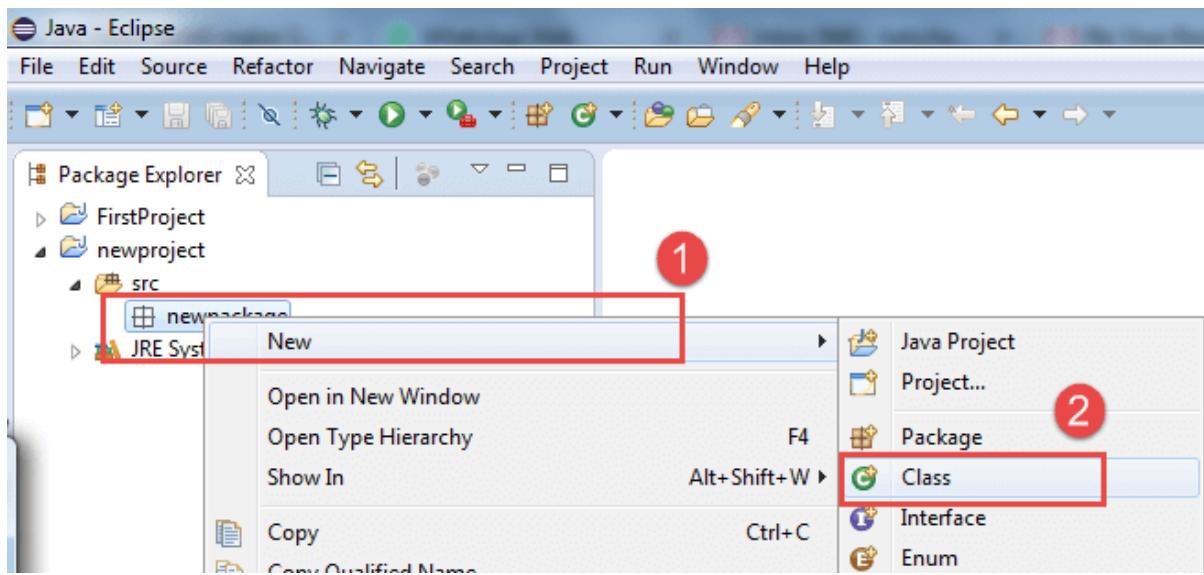


A pop-up window will open to name the package,

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

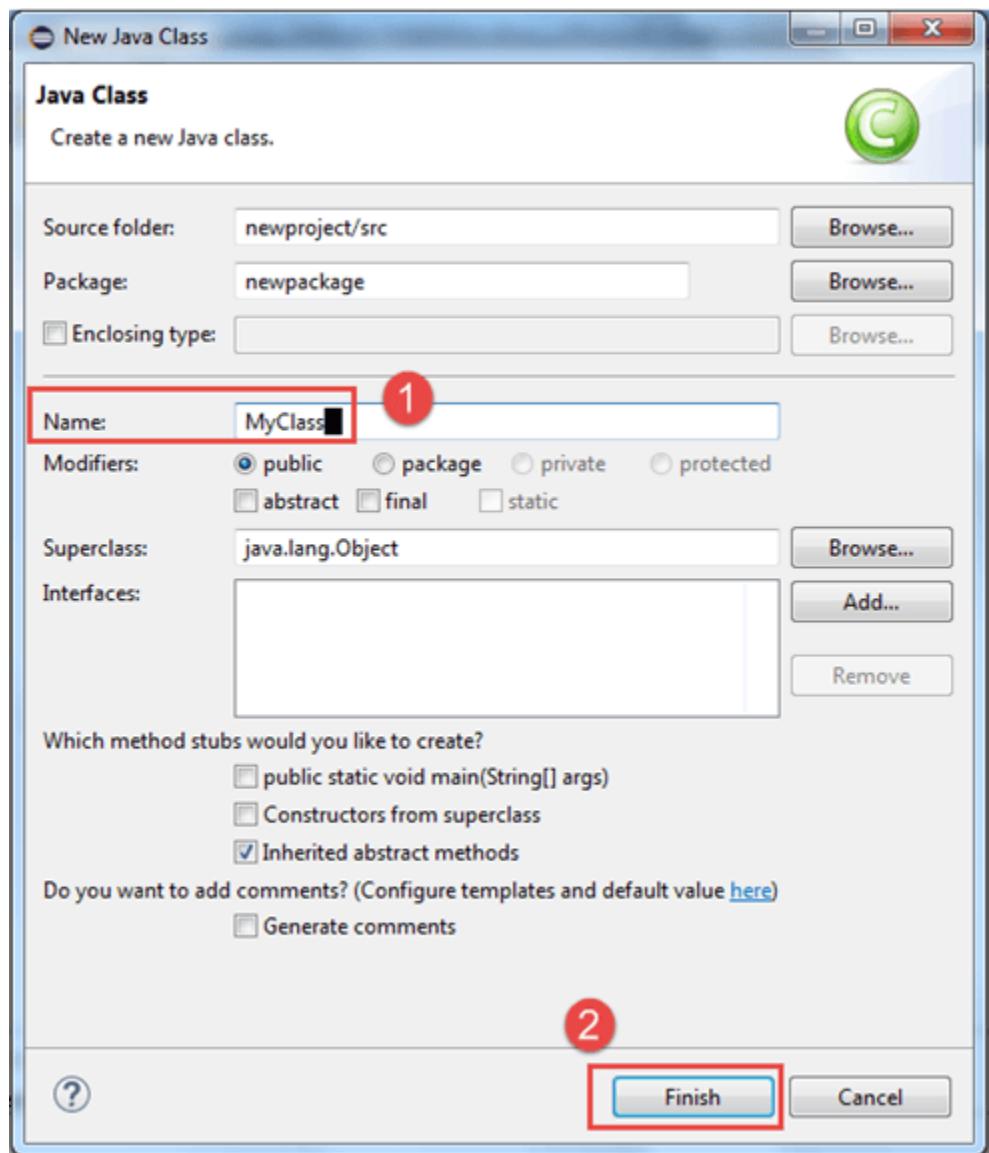


5. Create a new Java class under newpackage by right-clicking on it and then selecting New > Class, and then name it as “MyClass”. Your Eclipse IDE should look like the image below.

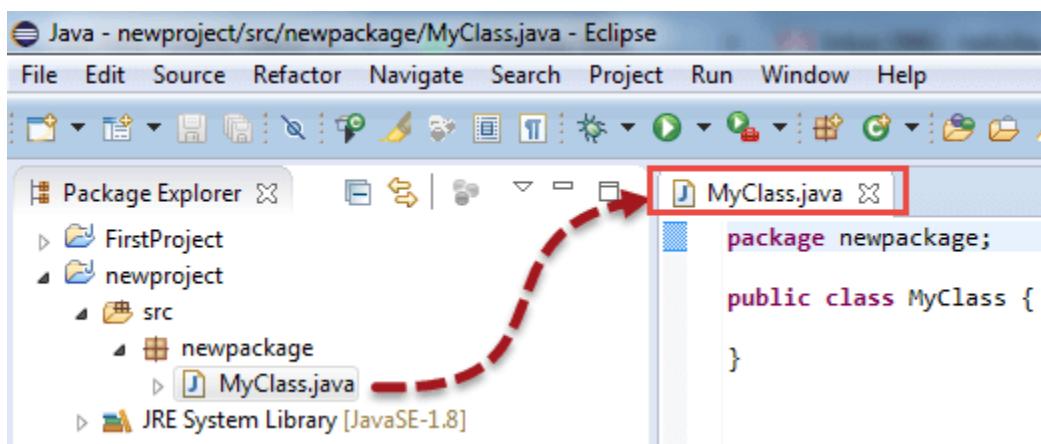


When you click on Class, a pop-up window will open, enter details as

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



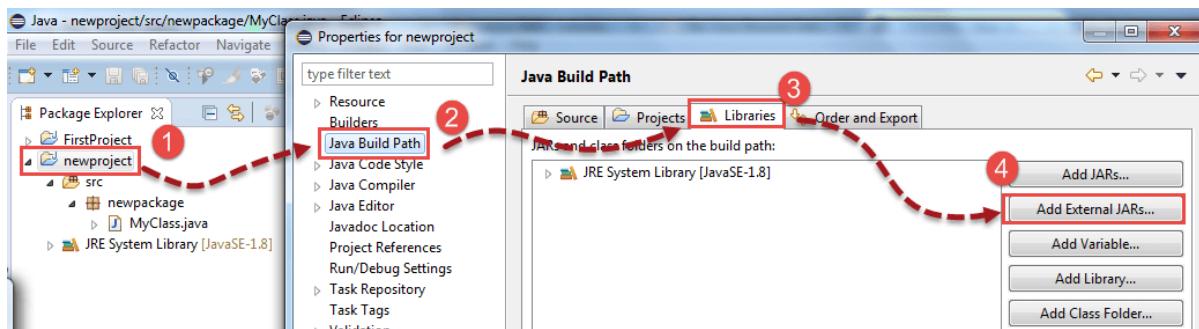
This is how it looks like after creating class.



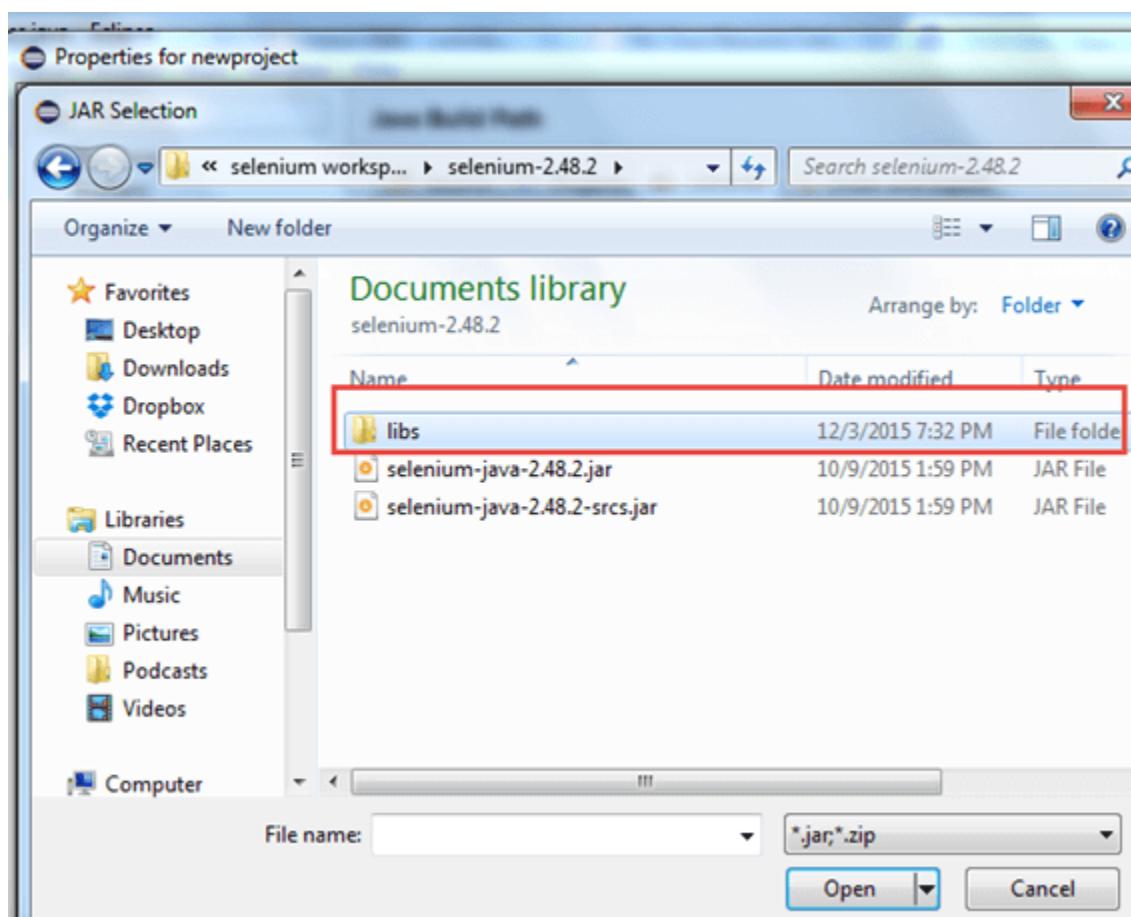
Now selenium WebDriver's into Java Build Path

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..”

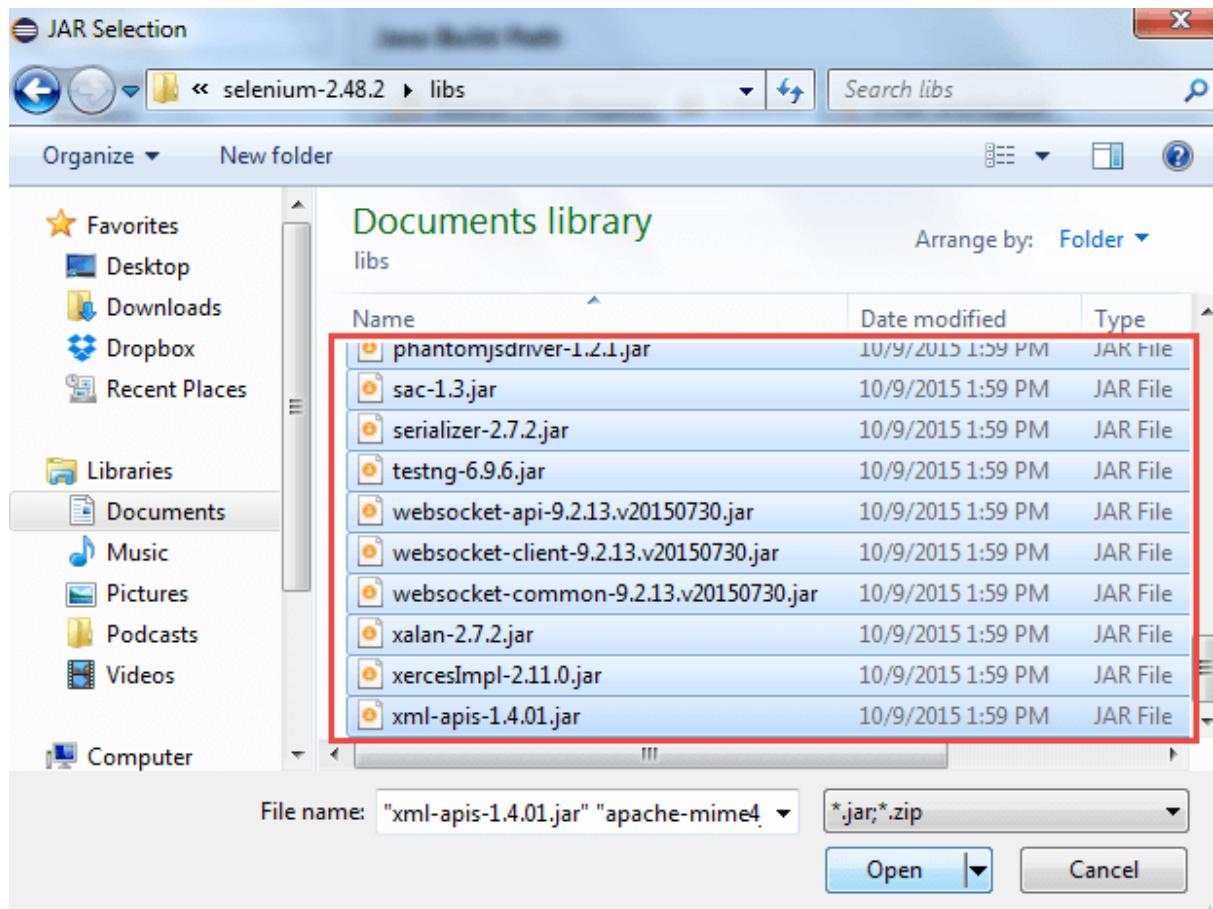


When you click on “Add External JARs..” It will open a pop-up window. Select the JAR files you want to add.

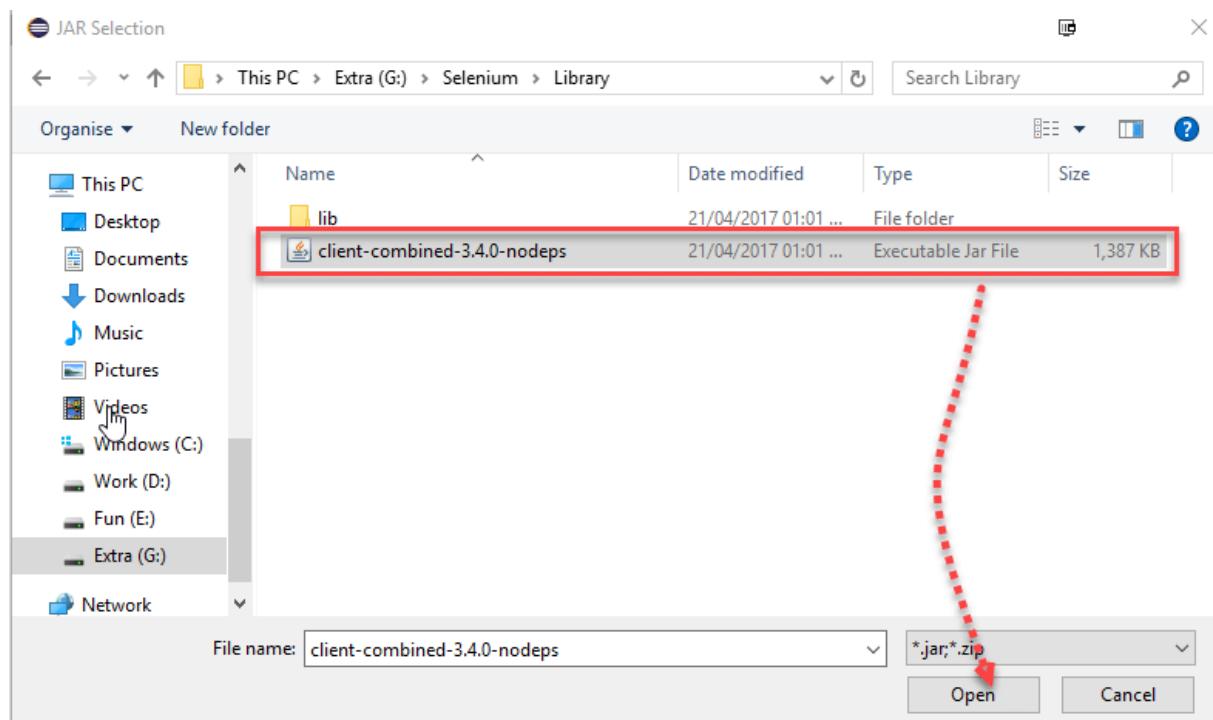


After selecting jar files, click on OK button.

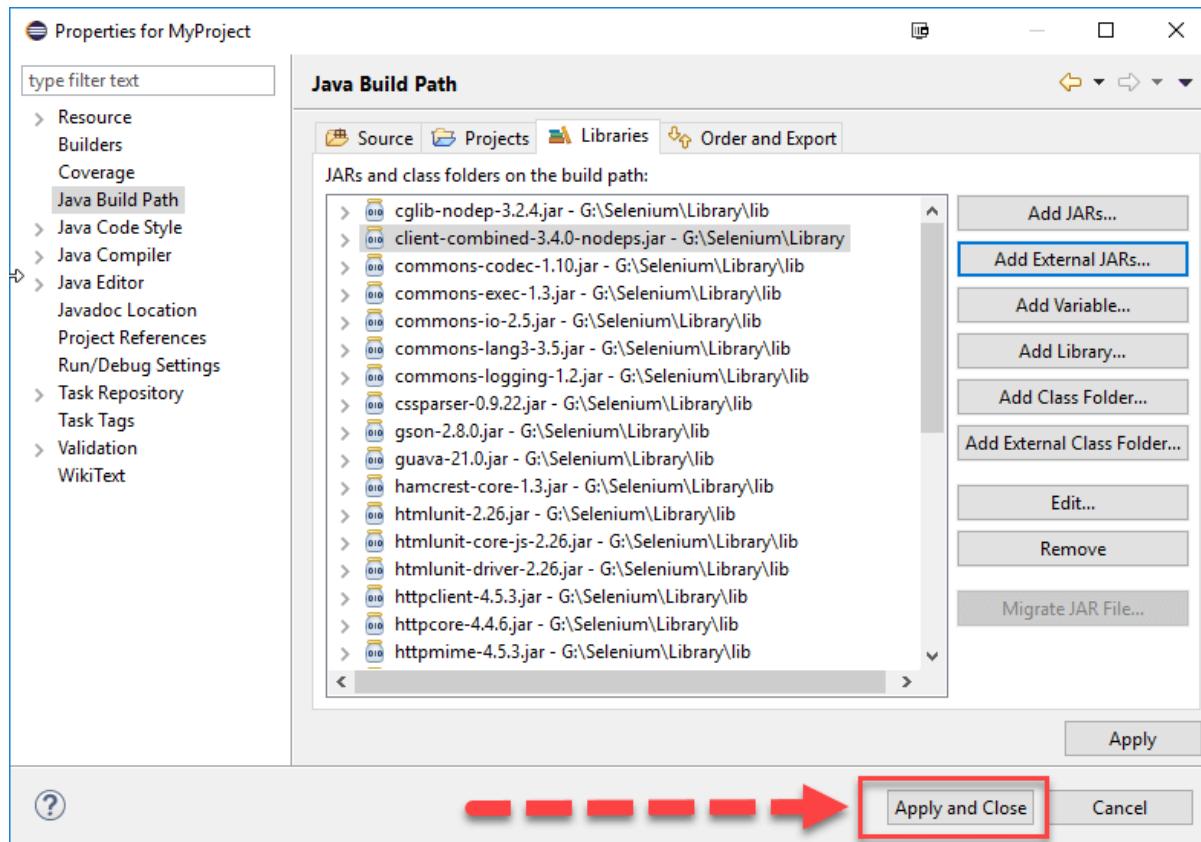
Select all files inside the lib folder.



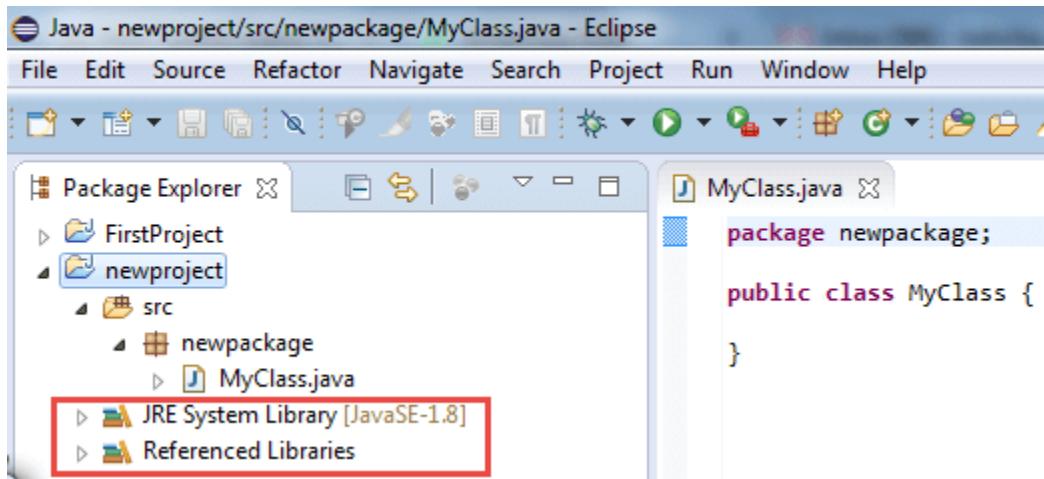
Select files outside lib folder



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/113WPhiDvpeLb7Im3goo7vtNT0kTAjXIH?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
```