Experiment 3: Docker Installation & basic commands of Docker

Part A:

Steps for Installing Docker:

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

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

After entering the above command, you will need to enter the password of the root and press enter.
3. Check if the system is up-to-date using the following command:

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

4. Install Docker using the following command:

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

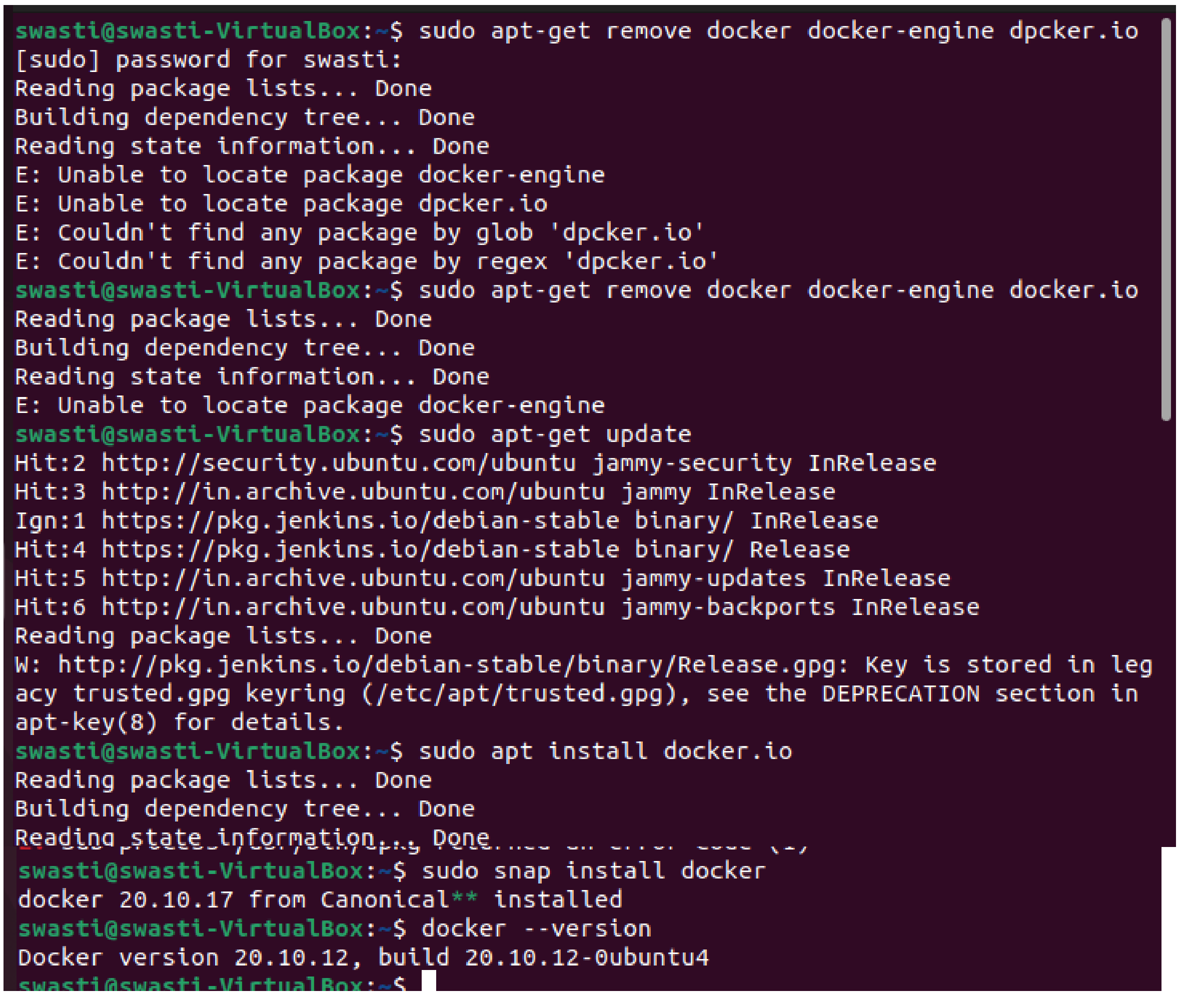
You'll then get a prompt asking you to choose between y/n - choose *y*

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
```



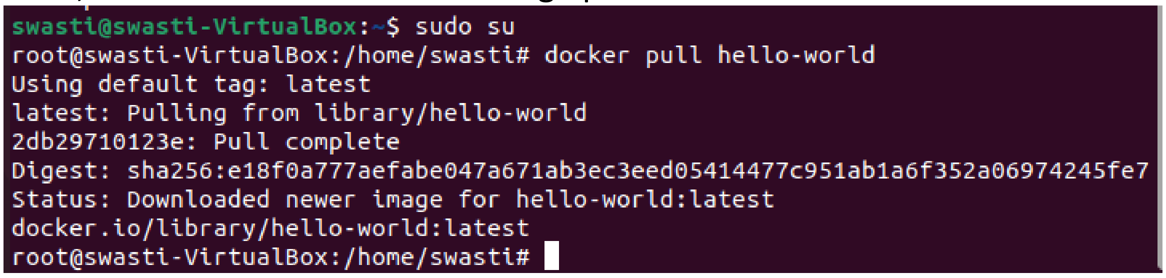
```
swasti@swasti-VirtualBox:~$ sudo apt-get remove docker docker-engine dpcker.io
[sudo] password for swasti:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package docker-engine
E: Unable to locate package dpcker.io
E: Couldn't find any package by glob 'dpcker.io'
E: Couldn't find any package by regex 'dpcker.io'
swasti@swasti-VirtualBox:~$ sudo apt-get remove docker docker-engine docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package docker-engine
swasti@swasti-VirtualBox:~$ sudo apt-get update
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:4 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
W: http://pkg.jenkins.io/debian-stable/binary/Release.gpg: Key is stored in leg
acy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in
apt-key(8) for details.
swasti@swasti-VirtualBox:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
swasti@swasti-VirtualBox:~$ sudo snap install docker
docker 20.10.17 from Canonical** installed
swasti@swasti-VirtualBox:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu4
swasti@swasti-VirtualBox:~$
```

+

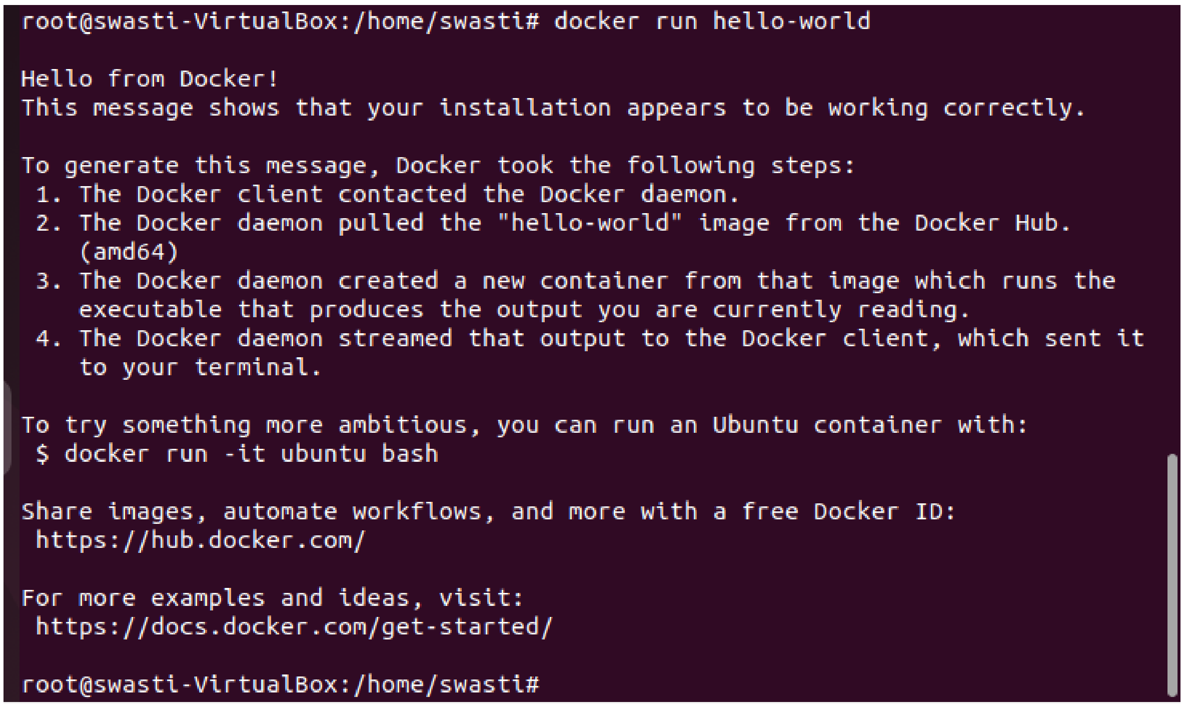
7. Pull an image from the Docker hub using the following command:

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

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



```
swasti@swasti-VirtualBox:~$ sudo su
root@swasti-VirtualBox:/home/swasti# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:e18f0a777aefabe047a671ab3ec3eed05414477c951ab1a6f352a06974245fe7
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
root@swasti-VirtualBox:/home/swasti#
```

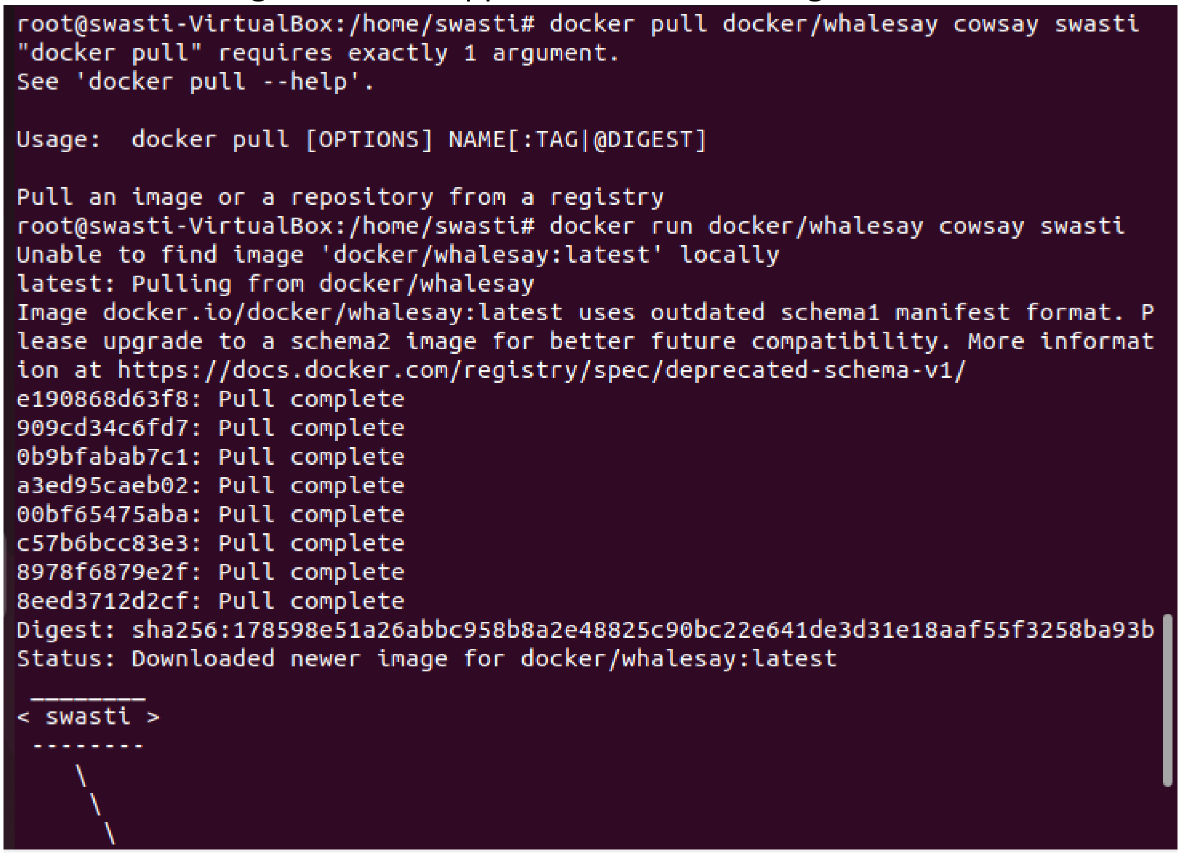


```
root@swasti-VirtualBox:/home/swasti# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

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

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

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

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

root@swasti-VirtualBox:/home/swasti#
```

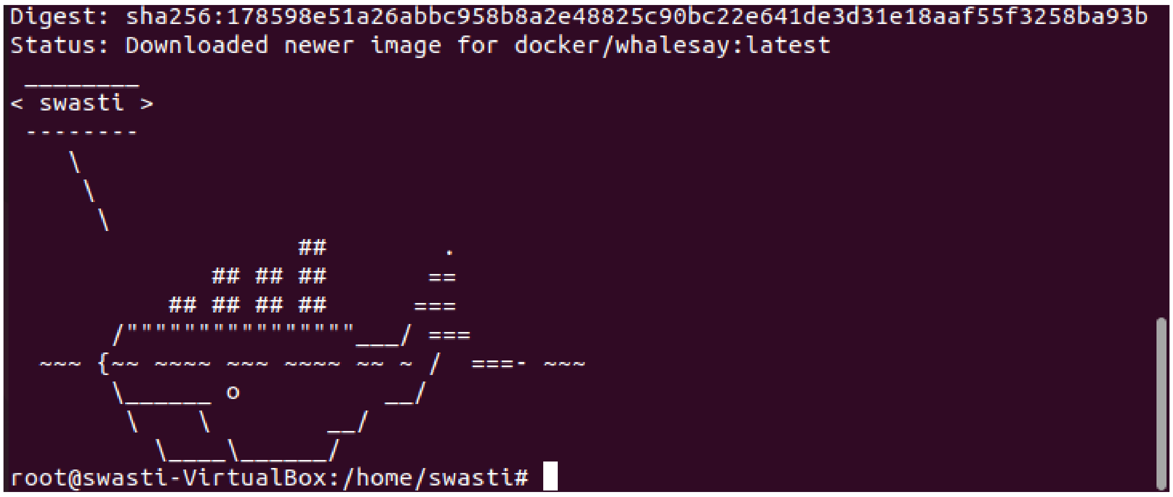
8. The actual Hellow World command of docker is

$ **docker run docker/whalesay cowsay** boo

The default image of docker appears with the message boo.



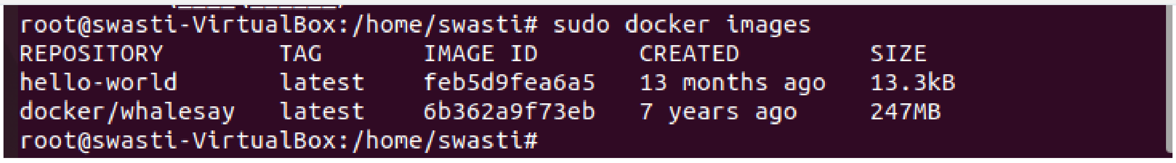
```
root@swasti-VirtualBox:/home/swasti# docker pull docker/whalesay cowsay swasti
"docker pull" requires exactly 1 argument.
See 'docker pull --help'.

Usage:  docker pull [OPTIONS] NAME[:TAG|@DIGEST]

Pull an image or a repository from a registry
root@swasti-VirtualBox:/home/swasti# docker run docker/whalesay cowsay swasti
Unable to find image 'docker/whalesay:latest' locally
latest: Pulling from docker/whalesay
Image docker.io/docker/whalesay:latest uses outdated schema1 manifest format. P
lease upgrade to a schema2 image for better future compatibility. More informat
ion at https://docs.docker.com/registry/spec/deprecated-schema-v1/
e190868d63f8: Pull complete
909cd34c6fd7: Pull complete
0b9bfabab7c1: Pull complete
a3ed95caeb02: Pull complete
00bf65475aba: Pull complete
c57b6bcc83e3: Pull complete
8978f6879e2f: Pull complete
8eed3712d2cf: Pull complete
Digest: sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest
 ________
< swasti >
 --------
    \
     \
      \
```



```
Digest: sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest
 ________
< swasti >
 --------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
root@swasti-VirtualBox:/home/swasti#
```

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

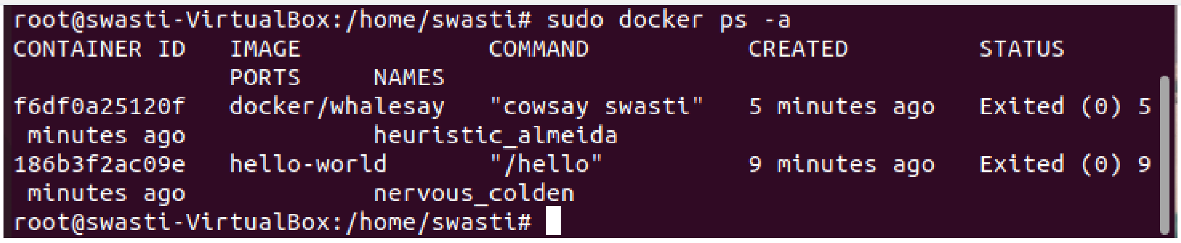
```
$ sudo docker images
```



```
root@swasti-VirtualBox:/home/swasti# sudo docker images
REPOSITORY         TAG       IMAGE ID       CREATED         SIZE
hello-world        latest    feb5d9fea6a5   13 months ago   13.3kB
docker/whalesay    latest    6b362a9f73eb   7 years ago     247MB
root@swasti-VirtualBox:/home/swasti#
```

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

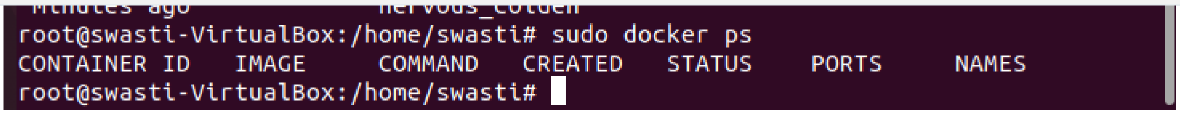
```
$ sudo docker ps -a
```



```
root@swasti-VirtualBox:/home/swasti# sudo docker ps -a
CONTAINER ID   IMAGE             COMMAND           CREATED         STATUS
               PORTS     NAMES
f6df0a25120f   docker/whalesay   "cowsay swasti"   5 minutes ago   Exited (0) 5
 minutes ago             heuristic_almeida
186b3f2ac09e   hello-world       "/hello"          9 minutes ago   Exited (0) 9
 minutes ago             nervous_colden
root@swasti-VirtualBox:/home/swasti#
```

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

```
$ sudo docker ps
```



```
root@swasti-VirtualBox:/home/swasti# sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@swasti-VirtualBox:/home/swasti#
```

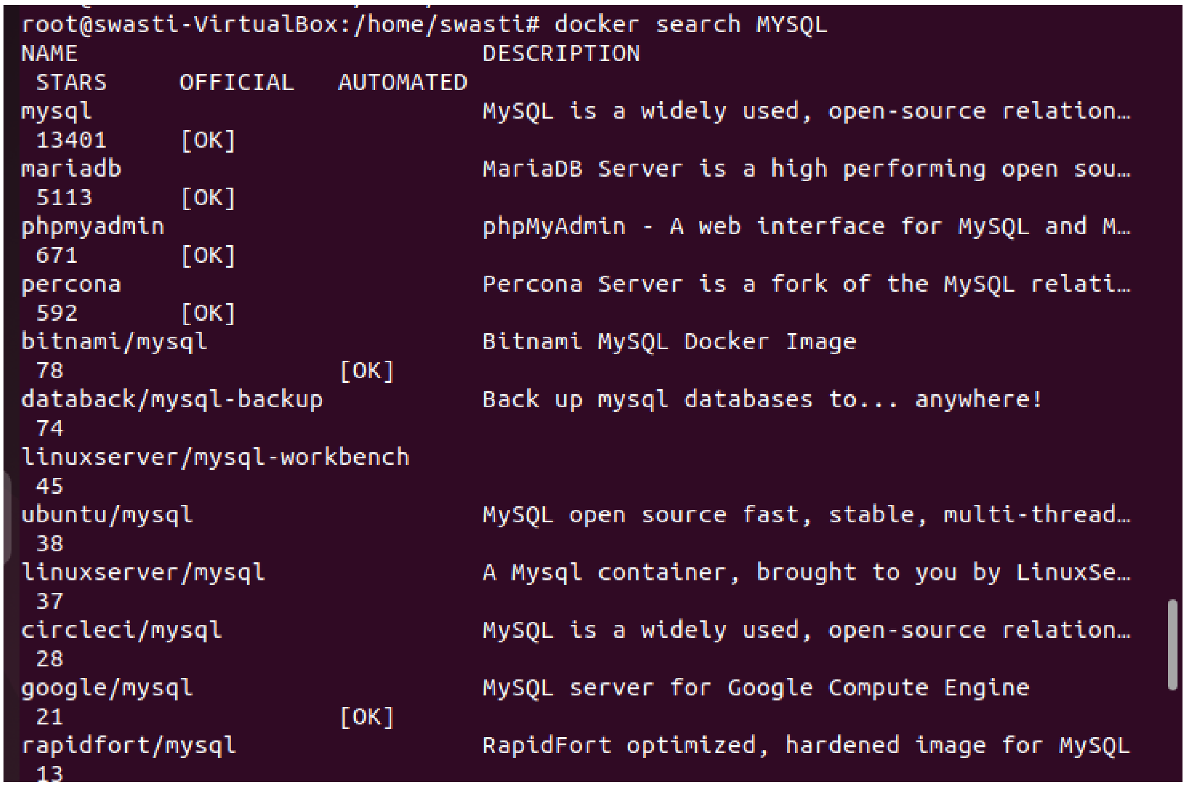
You've just successfully installed Docker on Ubuntu!

Part B:

1. docker search

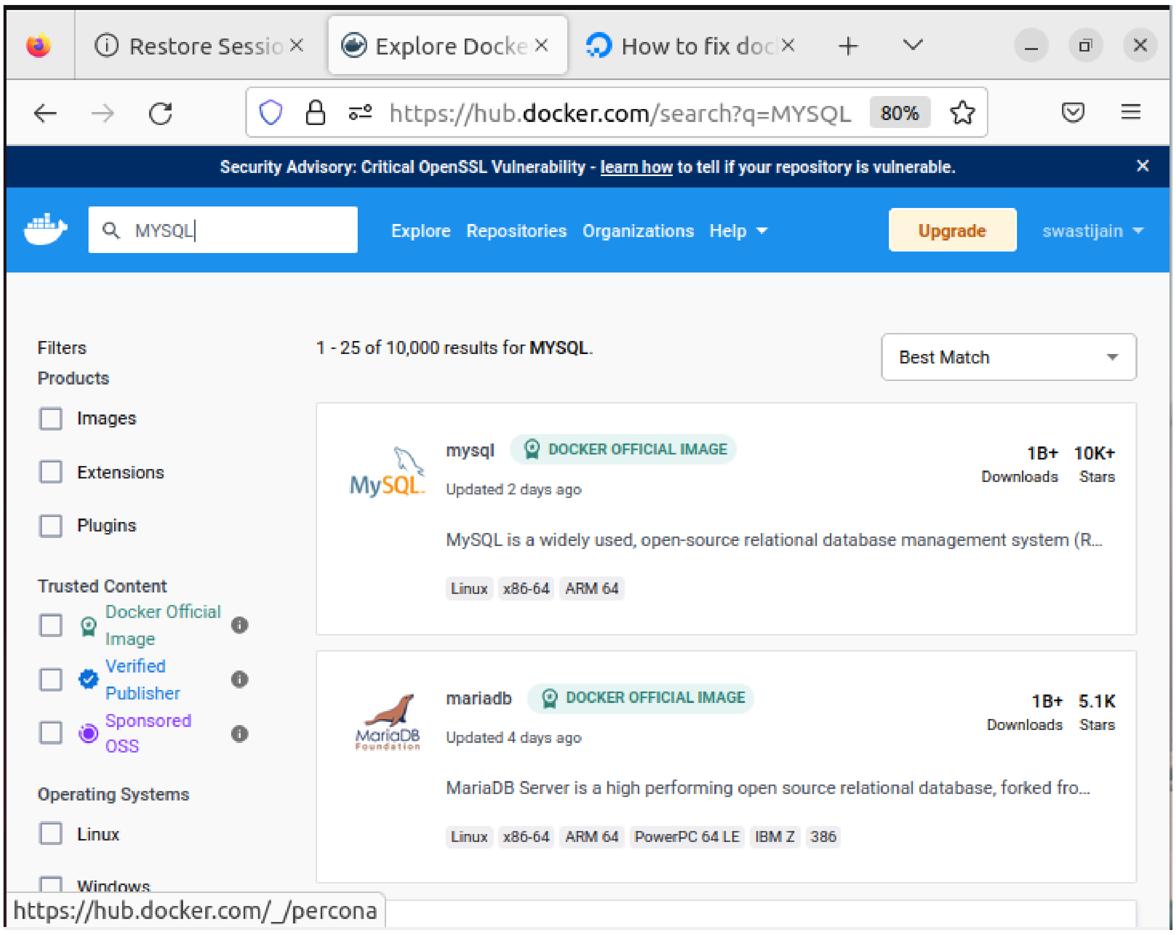
Use the command `docker search` to search for public images on the Docker hub. It will return information about the image name, description, stars, official and automated.

```
docker search MySQL
```



```
root@swasti-VirtualBox:/home/swasti# docker search MYSQL
NAME                                DESCRIPTION
 STARS     OFFICIAL   AUTOMATED
mysql                               MySQL is a widely used, open-source relation…
 13401     [OK]
mariadb                             MariaDB Server is a high performing open sou…
 5113      [OK]
phpmyadmin                          phpMyAdmin - A web interface for MySQL and M…
 671       [OK]
percona                             Percona Server is a fork of the MySQL relati…
 592       [OK]
bitnami/mysql                       Bitnami MySQL Docker Image
 78                   [OK]
databack/mysql-backup               Back up mysql databases to... anywhere!
 74
linuxserver/mysql-workbench
 45
ubuntu/mysql                        MySQL open source fast, stable, multi-thread…
 38
linuxserver/mysql                   A Mysql container, brought to you by LinuxSe…
 37
circleci/mysql                      MySQL is a widely used, open-source relation…
 28
google/mysql                        MySQL server for Google Compute Engine
 21                   [OK]
rapidfort/mysql                     RapidFort optimized, hardened image for MySQL
 13
```

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
```

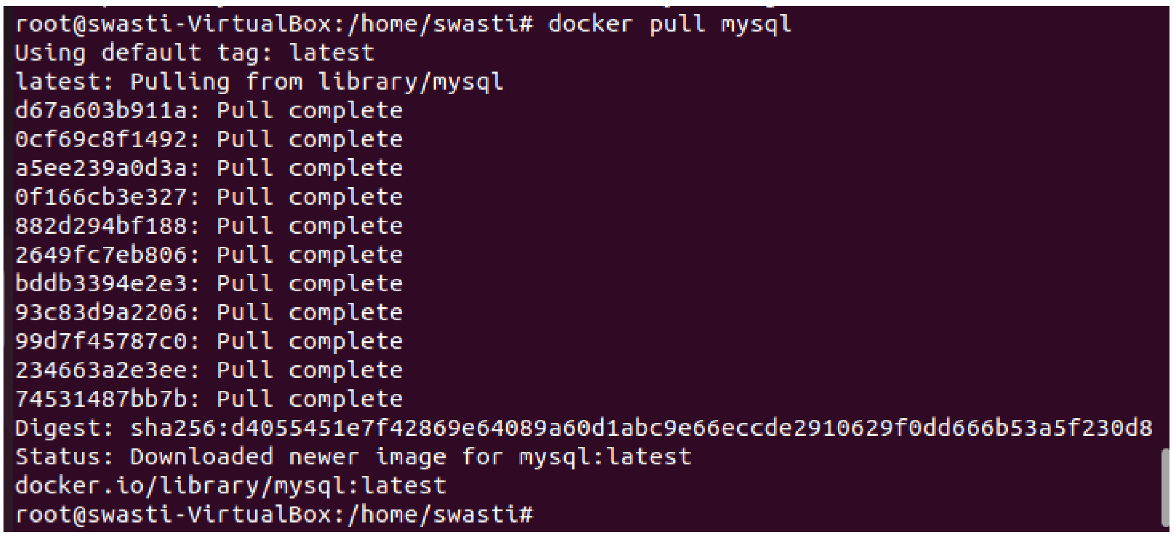
Tags are used to identify images inside a repository. If we don't specify a tag Docker engine uses the `:latest` tag by default. So, in the previous example, Docker pulled the `mysql:latest` image.

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
```

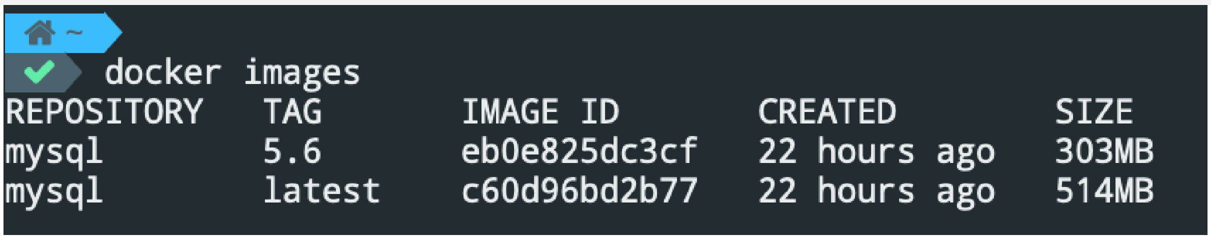


```
root@swasti-VirtualBox:/home/swasti# docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
d67a603b911a: Pull complete
0cf69c8f1492: Pull complete
a5ee239a0d3a: Pull complete
0f166cb3e327: Pull complete
882d294bf188: Pull complete
2649fc7eb806: Pull complete
bddb3394e2e3: Pull complete
93c83d9a2206: Pull complete
99d7f45787c0: Pull complete
234663a2e3ee: Pull complete
74531487bb7b: Pull complete
Digest: sha256:d4055451e7f42869e64089a60d1abc9e66eccde2910629f0dd666b53a5f230d8
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
root@swasti-VirtualBox:/home/swasti#
```

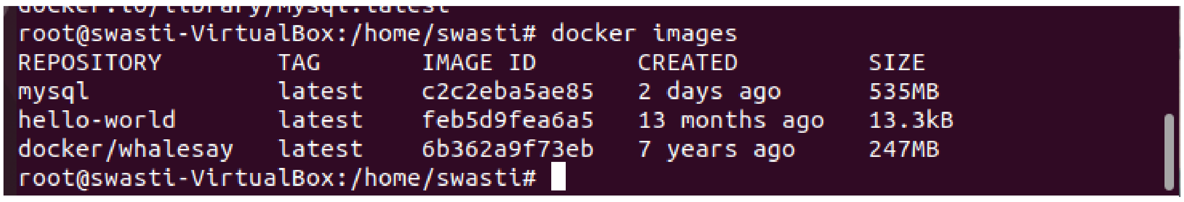
3. docker images

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
```



```
~
 docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
mysql         5.6       eb0e825dc3cf   22 hours ago   303MB
mysql         latest    c60d96bd2b77   22 hours ago   514MB
```



```
root@swasti-VirtualBox:/home/swasti# docker images
REPOSITORY         TAG       IMAGE ID       CREATED          SIZE
mysql              latest    c2c2eba5ae85   2 days ago       535MB
hello-world        latest    feb5d9fea6a5   13 months ago    13.3kB
docker/whalesay    latest    6b362a9f73eb   7 years ago      247MB
root@swasti-VirtualBox:/home/swasti#
```

I have 3 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.

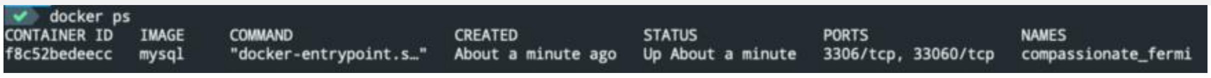
```
docker run --env MYSQL_ROOT_PASSWORD=my-secret-pw --detach mysql
```

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.

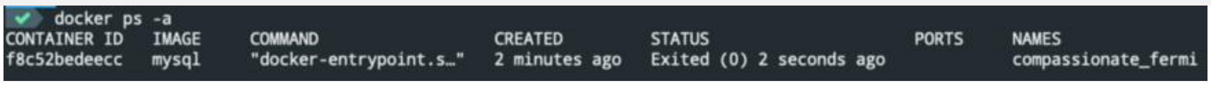
```
docker ps
```



```
 docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS               PORTS                 NAMES
f8c52bedeecc   mysql     "docker-entrypoint.s…"   About a minute ago   Up About a minute    3306/tcp, 33060/tcp   compassionate_fermi
```

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

```
docker ps -all
```



```
docker ps -a
CONTAINER ID   IMAGE   COMMAND                   CREATED         STATUS                     PORTS   NAMES
f8c52bedeecc   mysql   "docker-entrypoint.s…"   2 minutes ago   Exited (0) 2 seconds ago           compassionate_fermi
```

6. docker stop

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

```
docker stop f8c52bedeecc
```

7. docker restart

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

```
docker restart f8c52bedeecc
```

8. docker rename

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

```
docker rename compassionate_fermi test_db
```

9. docker exec

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

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

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

10. docker logs

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

```
docker logs test_db
```

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

```
docker logs -follow test_db
```

11. docker rm

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

```
docker rm test_db
```

You may encounter an error like

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

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

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

12. docker rmi

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

```
docker rmi eb0e825dc3cf
```

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

```
docker logs --help
```

Part C: Docker Container Commands:

Containers

Use `docker container my_command`

`create` — Create a container from an image.
`start` — Start an existing container.
`run` — Create a new container and start it.
`ls` — List running containers.
`inspect` — See lots of info about a container.
`logs` — Print logs.
`stop` — Gracefully stop running container.
`kill` —Stop main process in container abruptly.
`rm`— Delete a stopped container.

Images

Use `docker image my_command`

`build` — Build an image.
`push` — Push an image to a remote registry.
`ls` — List images.
`history` — See intermediate image info.
`inspect` — See lots of info about an image, including the layers.
`rm` — Delete an image.

Misc

`docker version` — List info about your Docker Client and Server versions.
`docker login` — Log in to a Docker registry.
`docker system prune` — Delete all unused containers, unused networks, and dangling images.

Containers

Container Beginnings

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

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

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

There are a lot of possible flags you could pass to `create`.

`docker container create -a STDIN my_image`

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

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

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

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

`docker container start my_container`

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

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

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

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

`-t`is short for`--tty`. Allocates a pseudo terminal that connects your terminal with the container's STDIN and STDOUT.

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

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

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

Let’s look at some more examples of `run`.

docker container run -it my_image my_command

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

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

docker container run -d my_image

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

Checking Container Status

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

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

`docker container ls -a -s`

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

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

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

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

Container Endings

Sometimes you need to stop a running container.

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

Or if you are impatient:

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

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

Then you delete the container with:

docker container rm my_container — Delete one or more containers.

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

Those are the eight essential commands for Docker containers.

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

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

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

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

Reference:

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

Rubrics:

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