



Assignment 3 : Light Container

Name: Naimur Islam Navid
ID: 24141160
CSE484 (Cloud Computing)

Task-1: Install docker

First, we are going to install the docker. To install this we will follow the steps below.

(My Ubuntu version is 22.04)

First, let's update our existing list of packages:

\$sudo apt update

```
naimur@Navid-24141160:~$ sudo update
[sudo] password for naimur:
sudo: update: command not found
naimur@Navid-24141160:~$ sudo apt update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages
8 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages
6 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages
6 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages
79 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en
kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11
```

Next, install a few prerequisite packages that let apt use packages over HTTPS:

\$sudo apt install apt-transport-https ca-certificates curl software-properties-common

```
naimur@Navid-24141160:~$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203~22.04.1).
ca-certificates set to manually installed.
software-properties-common is already the newest version (0.99.22.9).
software-properties-common set to manually installed.
The following NEW packages will be installed:
  apt-transport-https curl
0 upgraded, 2 newly installed, 0 to remove and 9 not upgraded.
Need to get 195 kB of archives.
After this operation, 625 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.12-1 [510 B]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 curl amd64 7.81.0-1ubuntu1.1 [169 kB]
Fetched 170 kB in 1s (141 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
naimur@Navid-24141160:~$
```

Then we need to add the GPG key for the official Docker repository to our system:

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

```
naimur@Navid-24141160:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
naimur@Navid-24141160:~$
```

Now lets add the Docker repository to APT sources:

```
$echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
naimur@Navid-24141160:~$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
naimur@Navid-24141160:~$
```

Now lets update the existing list of packages again for the addition to be recognized

```
$sudo apt update
```

```
naimur@Navid-24141160:~$ sudo apt update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [4096 kB]
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [461 kB]
Hit:3 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 89.8 kB in 2s (57.8 kB/s)
Reading package lists... Done
Building dependency tree... Done
```

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

\$apt-cache policy docker-ce

You'll see output like this, although the version number for Docker may be different:

```
naimur@Navid-24141160:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:27.3.1-1~ubuntu.22.04~jammy
  Version table:
   5:27.3.1-1~ubuntu.22.04~jammy 500
     500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
   5:27.3.0-1~ubuntu.22.04~jammy 500
     500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
   5:27.2.1-1~ubuntu.22.04~jammy 500
     500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
   5:27.2.0-1~ubuntu.22.04~jammy 500
     500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
   5:27.1.2-1~ubuntu.22.04~jammy 500
     500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
   5:27.1.1-1~ubuntu.22.04~jammy 500
     500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
```

Notice that docker-ce is not installed, but the candidate for installation is from the Docker repository for Ubuntu 22.04 (jammy).

Finally, install Docker:

\$sudo apt install docker-ce

```
naimur@Navid-24141160:~$ sudo apt install docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin git git-man liberror-perl pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit
  git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
```

Docker should now be installed, the daemon started, and the process enabled to start on boot. Check that it's running:

\$sudo systemctl status docker

The output should be similar to the following, showing that the service is active and running

```
naimur@Navid-24141160:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
   Active: active (running) since Sat 2024-11-30 01:05:55 +06; 46s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 5607 (dockerd)
      Tasks: 9
     Memory: 21.1M
        CPU: 486ms
    CGroup: /system.slice/docker.service
           └─5607 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>

Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.2351458>
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.2367233>
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.3646556>
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.7468577>
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.7777729>
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.7780831>
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.7782191>
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.7784673>
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.8365405>
Nov 30 01:05:55 Navid-24141160 systemd[1]: Started Docker Application Container>
lines 1-22/22 (END)
```

And that's it docker is installed!

Task-2: Show outputs of basic Docker commands (i.e pull, search, run, build, commit, rm, rmi, etc.. find more from Google)

In this task, I'm going to show some basic commands of docker.

Using docker consists of passing it a chain of options and commands followed by arguments. The syntax takes this form:

```
$docker [option] [command] [arguments]
```

Here to run a docker command we need arguments. In this task I'm going to only show some basic commands by running and we will see more on the next tasks.

To view system-wide information about Docker, use:

\$docker info

```
nainur@Navid-24141160:~$ docker info
Client: Docker Engine - Community
Version: 27.3.1
Context: default
Debug Mode: false
Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version: v0.17.1
    Path: /usr/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version: v2.29.7
    Path: /usr/libexec/docker/cli-plugins/docker-compose
Server:
ERROR: permission denied while trying to connect to the Docker daemon
unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/
dial unix /var/run/docker.sock: connect: permission denied
sssss pretty printing info
```

To view all available subcommands, type:

\$docker

```
naimur@Navid-24141160:~$ docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
run      Create and run a new container from an image
exec     Execute a command in a running container
ps       List containers
build    Build an image from a Dockerfile
pull     Download an image from a registry
push     Upload an image to a registry
images   List images
login    Authenticate to a registry
logout   Log out from a registry
search   Search Docker Hub for images
version  Show the Docker version information
info     Display system-wide information

Management Commands:
builder  Manage builds
buildx*  Docker Buildx
checkpoint Manage checkpoints
```

Im also attaching all the docker commands

```

Commands:
attach      Attach local standard input, output, and error streams to a running container
commit     Create a new image from a container's changes
cp         Copy files/folders between a container and the local filesystem
create     Create a new container
diff       Inspect changes to files or directories on a container's filesystem
events     Get real time events from the server
export     Export a container's filesystem as a tar archive
history    Show the history of an image
import     Import the contents from a tarball to create a filesystem image
inspect    Return low-level information on Docker objects
kill       Kill one or more running containers
load      Load an image from a tar archive or STDIN
logs      Fetch the logs of a container
pause     Pause all processes within one or more containers
port      List port mappings or a specific mapping for the container
rename    Rename a container
restart   Restart one or more containers
rm        Remove one or more containers
rmi       Remove one or more images
save      Save one or more images to a tar archive (streamed to STDOUT by default)
start     Start one or more stopped containers
stats     Display a live stream of container(s) resource usage statistics
stop      Stop one or more running containers
tag       Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top       Display the running processes of a container
unpause   Unpause all processes within one or more containers
update    Update configuration of one or more containers
wait      Block until one or more containers stop, then print their exit codes

```

```

Common Commands:
run      Create and run a new container from an image
exec     Execute a command in a running container
ps       List containers
build    Build an image from a Dockerfile
pull     Download an image from a registry
push     Upload an image to a registry
images   List images
login    Authenticate to a registry
logout   Log out from a registry
search   Search Docker Hub for images
version  Show the Docker version information
info     Display system-wide information

```

```

Management Commands:
builder  Manage builds
buildx*  Docker Buildx
checkpoint Manage checkpoints
compose* Docker Compose
container Manage containers
context  Manage contexts
image    Manage images
manifest Manage Docker image manifests and manifest lists
network  Manage networks
plugin   Manage plugins
system   Manage Docker
trust    Manage trust on Docker images
volume   Manage volumes

```

Here are all the commands of docker and its usage are also given. Now I'm going to run a few docker commands to show how it works

First, lets check if our docker is running

\$sudo systemctl status docker

```
naimur@Navid-24141160:~$ sudo systemctl status docker
[sudo] password for naimur:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor
   Active: active (running) since Sat 2024-11-30 01:05:55 +06; 49min ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 5607 (dockerd)
      Tasks: 9
     Memory: 20.4M
        CPU: 1.040s
    CGroup: /system.slice/docker.service
            └─5607 /usr/bin/dockerd -H fd:// --containerd=/run/container
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.2
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.2
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.3
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.7
Nov 30 01:05:55 Navid-24141160 dockerd[5607]: time="2024-11-30T01:05:55.7
```

\$docker --version

```
naimur@Navid-24141160:~$ docker --version
Docker version 27.3.1, build ce12230
```

This command shows the docker version you are using.

\$sudo docker pull hello-world

```
naimur@Navid-24141160:~$ docker pull hello-world
Using default tag: latest
permission denied while trying to connect to the Docker daemon socket at unix:///var/
cker.sock/v1.47/images/create?fromImage=hello-world&tag=latest": dial unix /var/run/d
naimur@Navid-24141160:~$ sudo docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:305243c734571da2d100c8c8b3c3167a098cab6049c9a5b066b6021a60fcb966
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

This command will pull the latest hello-world image from the docker

\$sudo docker run hello-world

```
naimur@Navid-24141160:~$ sudo 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/
```

This will run the image I pulled and will also give an output.

\$sudo docker ps

```
naimur@Navid-24141160:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

This command will show us all the details of our created container.

\$sudo docker images

```
naimur@Navid-24141160:~$ sudo docker images
REPOSITORY      TAG         IMAGE ID      CREATED        SIZE
hello-world     latest     d2c94e258dcb  19 months ago  13.3kB
```

This command will show the list of images we created.

I think all of these commands will be enough since more commands will be shown in the rest of the tasks.

Task-3 Create a Docker image using Dockerfile.

To create a docker image using docker file we have to follow the following steps:

First of all, we have to create a directory to store the dockerfile. Using the following command

\$mkdir Dockerfiles

```
naimur@Navid-24141160:~$ mkdir dockerfiles
naimur@Navid-24141160:~$ vim dockerfiles/dockerfile
Command 'vim' not found, but can be installed with:
sudo apt install vim          # version 2:8.2.3995-1ubuntu2.19, or
sudo apt install vim-tiny     # version 2:8.2.3995-1ubuntu2.19
sudo apt install neovim       # version 0.6.1-3
sudo apt install vim-athena   # version 2:8.2.3995-1ubuntu2.19
sudo apt install vim-gtk3     # version 2:8.2.3995-1ubuntu2.19
sudo apt install vim-nox      # version 2:8.2.3995-1ubuntu2.19
```

In the dockerfile directory create a file named dockerfile and edit the file with the commands that want to execute and save the file. Use the following command

\$vim Dockerfiles/dockerfile

```
Processing triggers for nano-25 (2.10.1-1) ...
naimur@Navid-24141160:~$ vim dockerfiles/dockerfile
naimur@Navid-24141160:~$
```

If this comes error then you have to install docker vim files.
For that use the command:

\$sudo apt update

\$sudo apt install vim -y

```
naimur@Navid-24141160:~$ sudo apt update
sudo apt install vim -y
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:3 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [43.1 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 DEP-11 Metadata [208 B]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [125 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 DEP-11 Metadata [208 B]
Get:9 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [103 kB]
```

After doing this you retype the command

\$vim Dockerfiles/dockerfile

It will work now.

Here we will build an apache web server image as an example. In the dockerfile we have to write the following commands to execute the server and show output.

FROM httpd

RUN apt-get update

CMD ["echo", "CSE484"]

```
FROM httpd
RUN apt-get update
CMD ["echo", "CSE484"]

~
~
```

Now to build the dockerfile run the following command from the terminal

\$sudo docker build dockerfiles

```
naimur@Navid-24141160:~$ sudo docker build dockerfiles/
[+] Building 2.5s (6/6) FINISHED                                docker:default
=> [internal] load build definition from dockerfile             0.0s
=> => transferring dockerfile: 95B                             0.0s
=> [internal] load metadata for docker.io/library/httpd:latest 2.4s
=> [internal] load .dockerignore                               0.0s
=> => transferring context: 2B                                   0.0s
=> [1/2] FROM docker.io/library/httpd:latest@sha256:6bdbdf5ac16ac3d6ef54 0.0s
=> CACHED [2/2] RUN apt-get update                             0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                          0.0s
=> => writing image sha256:2be862428a265072bed13d24f5bb2640d59e059278601 0.0s
naimur@Navid-24141160:~$
```

After completing the command “successfully built” message will be shown.

Now check the images with the following command

\$sudo docker images

```
naimur@Navid-24141160:~$ sudo docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
<none>              <none>         90e2d9aec2e8   5 minutes ago   168MB
<none>              <none>         2be862428a26   5 minutes ago   168MB
ubuntu              latest         fec8bfd95b54   6 weeks ago     78.1MB
hello-world         latest         d2c94e258dcb   19 months ago   13.3kB
naimur@Navid-24141160:~$ sudo docker run 90e2d9aec2e8
CSE484 cloud Computing
naimur@Navid-24141160:~$ sudo docker run 2be862428a26
CSE484
naimur@Navid-24141160:~$
```

Here we can see the new image ID after successful build of our image from the dockerfile.

We have to run the image to see our output with the following command:

\$sudo docker run 2be862428a26

After that we can see our output CSE484 .

So we have successfully created a docker image from the dockerfile and run it.

Task-4 Run a container as a single task, show outputs, and show the status of all containers (using docker ps -a)

In this task we are going to run a single container to show its status and the outputs.

First, let's see all the container lists we created before by typing the command:

\$sudo docker ps -a

After executing this command we can see the container ID, image ID, command, created time, status, ports and names.

```
naimur@Navid-24141160:~$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
	PORTS	NAMES		
4258cd4f1d8a	2be862428a26	"echo CSE484"	12 minutes ago	Exited
0) 12 minutes ago		quizzical_golick		
7b64fdc9aa9c	2be862428a26	"echo CSE484"	30 minutes ago	Exited
0) 30 minutes ago		priceless_lamarr		
99d61b6633af	90e2d9aec2e8	"echo 'CSE484 cloud ...'"	30 minutes ago	Exited
0) 30 minutes ago		upbeat_hermann		
2179b44ed549	ubuntu	"/bin/bash"	19 hours ago	Exited
0) 19 hours ago		blissful_snyder		
b3aa42230eb7	hello-world	"/hello"	20 hours ago	Exited
0) 20 hours ago		jovial_heisenberg		
05e96df78883	ubuntu	"bash"	20 hours ago	Exited
129) 20 hours ago		xenodochial_tu		
c2144686f22b	hello-world	"/hello"	20 hours ago	Exited
0) 20 hours ago		admiring_heisenberg		

Here we can see all the containers we built. We can see the most recent container (the one we built in last task) container id: 4258cd4f1d8a and also see its details.

Now to run this single container we have to start the container with the container ID

For this we will use this command

\$sudo docker start 4258cd4f1d8a

(4258cd4f1d8a is the container id we created in our previous task)

```
naimur@Navid-24141160:~$ sudo docker start 4258cd4f1d8a
4258cd4f1d8a
```

then to show the output of the docker image we have to run the docker image file with the docker image id, with following command

\$sudo docker run 2be862428a26

After that we can see the output as “CSE484”

```
naimur@Navid-24141160:~$ sudo docker run 2be862428a26
CSE484
```

Hence we are done with running one of our single container.

Task-5: Run a container in iterative mode and install different packages in the container. Show each step.

In this task, we are going to run a container in interactive mode and install different packages in the container. Lets use a basic container ubuntu.

First let's pull it from the docker hub by using this command

\$sudo docker pull ubuntu

```
naimur@Navid-24141160:~$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:278628f08d4979fb9af9ead44277dbc9c92c2465922310916ad0c
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```

Now lets check by using this command

\$sudo docker images

```
naimur@Navid-24141160:~$ sudo docker images
REPOSITORY      TAG        IMAGE ID      CREATED        SIZE
<none>          <none>     90e2d9aec2e8  59 minutes ago 168MB
<none>          <none>     2be862428a26  59 minutes ago 168MB
ubuntu          latest     fec8bfd95b54  6 weeks ago   78.1MB
hello-world     latest     d2c94e258dcb  19 months ago 13.3kB
```

Now Start the Ubuntu container in interactive mode using the following command:

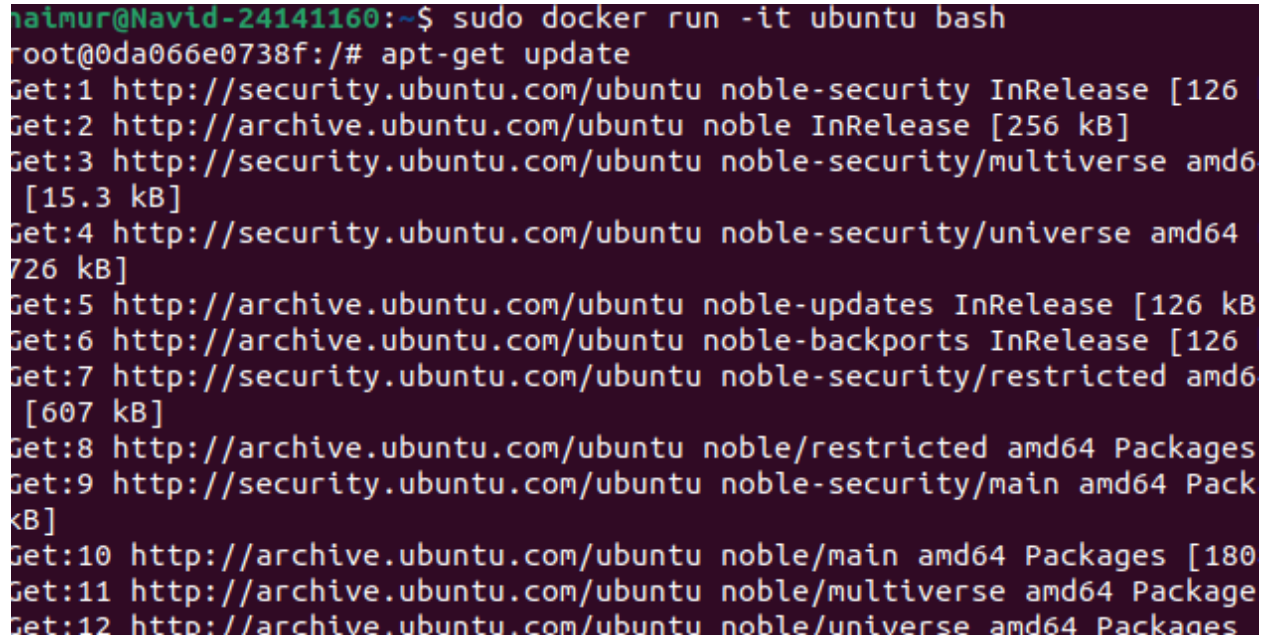
\$sudo docker run -it ubuntu bash

It will show something like this

root@0da66e0738f:/#

Now lets use this command to update

root@0da66e0738f:/# apt-get update

A terminal window with a dark background and light-colored text. The prompt is 'haimur@Navid-24141160:~\$'. The command 'sudo docker run -it ubuntu bash' has been executed, resulting in a new prompt 'root@0da66e0738f:/#'. The user then enters 'apt-get update', which produces a list of 12 package sources being updated, including security, archive, and restricted repositories for Ubuntu Noble, with their respective sizes in kB.

```
haimur@Navid-24141160:~$ sudo docker run -it ubuntu bash
root@0da66e0738f:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/multiverse amd6
[15.3 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/universe amd64
726 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126
Get:7 http://security.ubuntu.com/ubuntu noble-security/restricted amd6
[607 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Pack
kB]
Get:10 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [180
Get:11 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Package
Get:12 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages
```

Here

-it: This flag allows for interactive mode and allocates a pseudo-TTY, so you can interact with the container.

ubuntu: This specifies the image to run.

bash: Opens a shell session inside the container.

Now lets update or install some packages

root@0da66e0738f:/# apt-get install -y curl


```

root@0da066e0738f:/# apt-get install -y curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates krb5-locales libbrotli1 libcurl4t64 libgssapi-krb5-2
  libk5crypto3 libkeyutils1 libkrb5-3 libkrb5support0 libldap2-2
  libnghttp2-14 libpsl5t64 librtmp1 libsasl2-2 libsasl2-modules
  libsasl2-modules-db libssh-4 openssl publicsuffix
Suggested packages:
  krb5-doc krb5-user libsasl2-modules-gssapi-mit
  | libsasl2-modules-gssapi-heimdal libsasl2-modules-ldap
  libsasl2-modules-sql
The following NEW packages will be installed:
  ca-certificates curl krb5-locales libbrotli1 libcurl4t64
  libk5crypto3 libkeyutils1 libkrb5-3 libkrb5support0 libl

```

root@0da66e0738f:/# apt-get install -y git

```

root@0da066e0738f:/# apt-get install -y git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  adduser git-man less libbsd0 libcbor0.10 libcurl3t64-gnutls libedit2
  liberror-perl libexpat1 libfido2-1 libgdbm-compat4t64 libgdbm6t64
  libperl5.38t64 libx11-6 libx11-data libxau6 libxcb1 libxdmcp6 libxext6
  libxmuu1 netbase openssh-client patch perl perl-modules-5.38 xauth
Suggested packages:
  liblocale-gettext-perl cron quota ecryptfs-utils gettext-base git-daemon-run
  | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn gdbm-l10n keychain libpam-ssh monkeysphere ssh-askpass
  ed diffutils-doc perl-doc libterm-readline-gnu-perl
  | libterm-readline-perl-perl make libtap-harness-archive-perl
The following NEW packages will be installed:
  adduser git git-man less libbsd0 libcbor0.10 libcurl3t64-gnutls libedit2
  liberror-perl libexpat1 libfido2-1 libgdbm-compat4t64 libgdbm6t64
  libperl5.38t64 libx11-6 libx11-data libxau6 libxcb1 libxdmcp6 libxext6

```

We install packages like that and if installed then verify

By commanding

root@0da66e0738f:/# Curl --version

```
root@0da066e0738f:/# curl --version
curl 8.5.0 (x86_64-pc-linux-gnu) libcurl/8.5.0 OpenSSL/3.0.13 zlib/1.2.11
1.0 zstd/1.5.5 libidn2/2.3.7 libpsl/0.21.2 (+libidn2/2.3.7) libssh/0.
1/zlib nghttp2/1.59.0 librtmp/2.3 OpenLDAP/2.6.7
Release-Date: 2023-12-06, security patched: 8.5.0-2ubuntu10.5
Protocols: dict file ftp ftps gopher gophers http https imap imaps l
tt pop3 pop3s rtmp rtsp scp sftp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IDN
root@0da066e0738f:/#
```

root@0da066e0738f:/# Git --version

```
root@0da066e0738f:/# git --version
git version 2.43.0
root@0da066e0738f:/#
```

And that's how you install packages in interactive mode.

Task-6: Run a database container in the background. Then show logs of the running container. After, access the container in interactive mode. show some SQL queries inside the container.

To run a database container in the background, showing logs, and some sql queries in interactive mode we have to follow below steps: Here I am using MySQL database.

Pull the MySQL Docker Image:

\$sudo docker pull mysql:latest

```
naimur@Navid-24141160:~$ sudo docker pull mysql:latest
latest: Pulling from library/mysql
2c0a233485c3: Pull complete
cb5a6a8519b2: Pull complete
570d30cf82c5: Pull complete
a841bff36f3c: Pull complete
80ba30c57782: Pull complete
5e49e1f26961: Pull complete
ced670fc7f1c: Pull complete
0b9dc7ad7f03: Pull complete
cd0d5df9937b: Pull complete
1f87d67b89c6: Pull complete
Digest: sha256:0255b469f0135a0236d672d60e3154ae2f4538b146
Status: Downloaded newer image for mysql:latest
```

To verify the image we can use the

\$sudo docker images

```
naimur@Navid-24141160:~$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>             90e2d9aec2e8       2 hours ago        168MB
<none>              <none>             2be862428a26       2 hours ago        168MB
ubuntu              latest             fec8bfd95b54       6 weeks ago        78.1MB
mysql               latest             56a8c14e1404       6 weeks ago        603MB
hello-world         latest             d2c94e258dcb       19 months ago      13.3kB
naimur@Navid-24141160:~$
```

We will see the Image ID.

Now lets Deploy the MySQL Container:

\$sudo docker run --name mysql_docker -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:latest

```
naimur@Navid-24141160:~$ sudo docker run --name mysql_docker -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:latest
e515ec7d4a6aee6fe93886c011af80c9063fa706d7ea13ea5f8c76b69c0451fe
naimur@Navid-24141160:~$
```

Explanation:

- ***--name mysql_docker:*** Names the container mysql_docker.
- ***-e MYSQL_ROOT_PASSWORD=my-secret-pw:*** Sets the root password to my-secret-pw.
- ***-d:*** Runs the container in detached mode.
- ***mysql:latest:*** Uses the official MySQL image.

I can check the MySQL container running status with the following command

\$sudo docker ps

```
naimur@Navid-24141160:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
e515ec7d4a6a   mysql:latest   "docker-entrypoint.s..." 49 seconds ago Up 48 seconds 3306/tcp, 33060/tcp      mysql_docker
naimur@Navid-24141160:~$
```

Connect to the MySQL Docker Container:

Before connecting with the MySQL server container with the host, we need to make sure the MySQL client package is installed with the following command

\$sudo apt-get install mysql-client

```
naimur@Navid-24141160:~$ sudo apt-get install mysql-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  mysql-client-8.0 mysql-client-core-8.0 mysql-common
The following NEW packages will be installed:
  mysql-client mysql-client-8.0 mysql-client-core-8.0 mysql-common
0 upgraded, 4 newly installed, 0 to remove and 7 not upgraded.
Need to get 2,754 kB of archives.
After this operation, 62.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ftp.ubuntu.com/ubuntu/ubuntu focal-updates/main amd64 mysql-client-8.0 [1,114 kB]
Get:2 http://ftp.ubuntu.com/ubuntu/ubuntu focal-updates/main amd64 mysql-client-core-8.0 [1,114 kB]
Get:3 http://ftp.ubuntu.com/ubuntu/ubuntu focal-updates/main amd64 mysql-common [1,114 kB]
Get:4 http://ftp.ubuntu.com/ubuntu/ubuntu focal-updates/main amd64 mysql-client [1,114 kB]
Fetched 4,456 kB in 1s (4,456 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
mysql-client-8.0 mysql-client-core-8.0 mysql-common
mysql-client mysql-client-8.0 mysql-client-core-8.0 mysql-common
0 upgraded, 4 newly installed, 0 to remove and 7 not upgraded.
Need to get 2,754 kB of archives.
After this operation, 62.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Then have to open the logs file for MySQL container to find the generated root password with the following command

\$sudo docker logs mysql_docker

```
naimur@Navid-24141160:~$ sudo docker logs mysql_docker
2024-11-30 16:54:09+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.1.0-1.el9 started.
2024-11-30 16:54:09+00:00 [Note] [Entrypoint]: Switching to dedicated MySQL user.
2024-11-30 16:54:09+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.1.0-1.el9 started.
```

From here we have to copy the generated password for the next step.

Now will go to the bash shell of the MySQL container with the following command

\$sudo docker exec -it mysql_docker bash

```
naimur@Navid-24141160:~$ sudo docker exec -it mysql_docker bash
bash-5.1#
```

Now we need to connect it.

```
bash-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 9.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input state
```

Now we will write a sql query to change the root password from the container bash. Executing the following sql query we can change the root password for the database

ALTER USER 'root'@'localhost' IDENTIFIED BY '1234';

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY '1234';
Query OK, 0 rows affected (0.01 sec)
```

A few sql queries to create a database and create a table then insert some data and show that.

CREATE DATABASE Test;

CREATE TABLE StudentInfo(

name VARCHAR(20),

id INT);

INSERT INTO StudentInfo (name, id)

VALUES

('Naimur Islam', '24141160');

```

mysql> CREATE DATABASE Test;
Query OK, 1 row affected (0.01 sec)

mysql> CREATE TABLE StudentInfo( name VARCHAR(20),id INT);
ERROR 1046 (3D000): No database selected
mysql> CREATE DATABASE Test;
ERROR 1007 (HY000): Can't create database 'Test'; database exists
mysql>
mysql> USE Test;
Database changed
mysql>
mysql> CREATE TABLE StudentInfo(
->     name VARCHAR(20),
->     id INT
-> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> INSERT INTO StudentInfo (name, id)
-> VALUES
-> ('Naimur Islam', 24141160);
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> SELECT * FROM StudentInfo;
+-----+-----+
| name      | id      |
+-----+-----+
| Naimur Islam | 24141160 |
+-----+-----+
1 row in set (0.00 sec)

mysql>

```

```

mysql> show databases
-> ;
+-----+
| Database          |
+-----+
| Test              |
| information_schema |
| mysql             |
| performance_schema |
| sys               |
+-----+
5 rows in set (0.00 sec)

mysql>

```

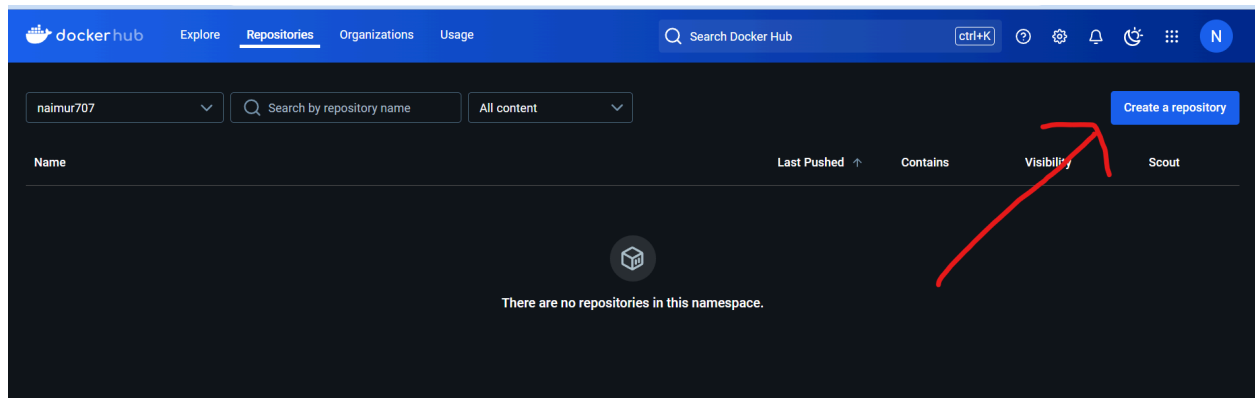
And we are done creating it.

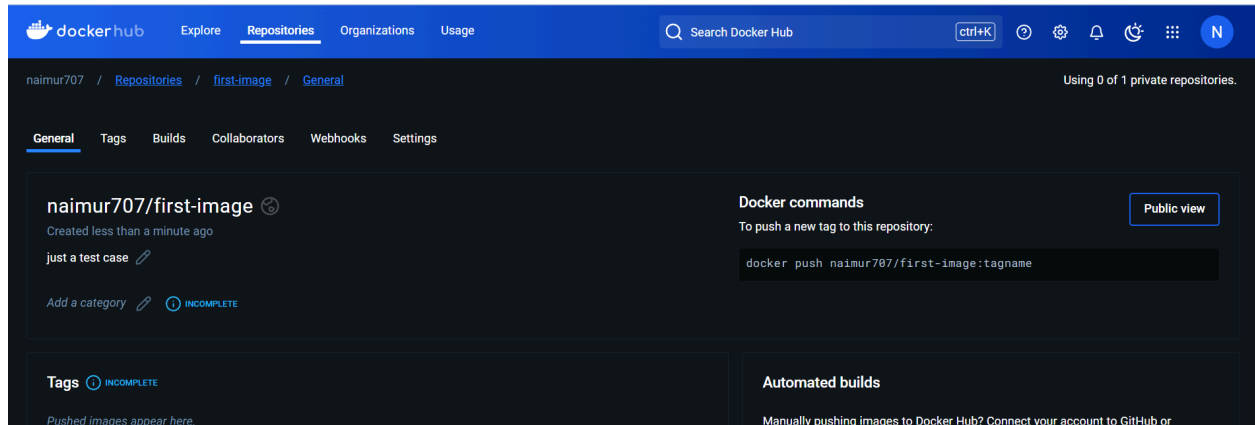
Task-7: Push your own image into the Docker public registry/Hub.

The very first thing to do for this task is to create an account in docker hub



After creating an account and signing in create a repository. In my case, i named it naimur707/first-image





Now go back to the terminal and first create a new dockerfile named dockerfile then edit that with desired commands and save that.

To create a new doc file use command

\$mkdir mydockfile

The change the directory

\$cd mydockfile

Create a new file named Dockerfile (no file extension):

\$touch docfile

```
naimur@Navid-24141160:~$ cd mydockerfiles
naimur@Navid-24141160:~/mydockerfiles$ touch dockerfile
naimur@Navid-24141160:~/mydockerfiles$ nano dockerfile
```

Now edit the docfile with necessary commands

\$nano Dockerfile

For example, here's a simple Dockerfile for a Node.js app:

Use the official Node.js image as the base image

FROM node:14

Set the working directory inside the container

WORKDIR /usr/src/app

Copy package.json and install dependencies

COPY package*.json ./

RUN npm install

Copy the rest of the app's files

COPY . .

Expose the port the app runs on

EXPOSE 8080

Command to run the app

CMD ["node", "app.js"]

```
naimur@Navid-24141160: ~/mydockerfiles
GNU nano 6.2 dockerfile *
# Use the official Node.js image as the base image
FROM node:14

# Set the working directory inside the container
WORKDIR /usr/src/app

# Copy package.json and install dependencies
COPY package*.json ./
RUN npm install

# Copy the rest of the app's files
COPY . .

# Expose the port the app runs on
EXPOSE 8080

# Command to run the app
CMD ["node", "app.js"]

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Now save that file by **ctrl+x**

After that we will build the docker image with my docker hub username and repository name with the following command

\$sudo docker build -t naimur707/first-image .

```
naimur@Navid-24141160:~/mydockerfiles$ sudo docker build -t naimur707/first-image .
[+] Building 2.5s (11/11) FINISHED                                docker:default
=> [internal] load build definition from dockerfile               0.0s
=> => transferring dockerfile: 399B                               0.0s
=> [internal] load metadata for docker.io/library/node:14        2.4s
=> [auth] library/node:pull token for registry-1.docker.io       0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                      0.0s
=> [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbc 0.0s
=> [internal] load build context                                  0.0s
=> => transferring context: 32B                                      0.0s
=> CACHED [2/5] WORKDIR /usr/src/app                               0.0s
=> CACHED [3/5] COPY package*.json ./                             0.0s
=> CACHED [4/5] RUN npm install                                    0.0s
=> CACHED [5/5] COPY . .                                          0.0s
=> exporting to image                                             0.0s
=> => exporting layers                                             0.0s
=> => writing image sha256:55faf92b1e5ada676d8fca48edfad6970c0c6c8c14ace9c7d5f1dbdde659e8de 0.0s
=> => naming to docker.io/naimur707/first-image                  0.0s
```

To verify check if it was created

\$sudo docker images

```
naimur@Navid-24141160:~/mydockerfiles$ sudo docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
naimur707/first-image latest          55faf92b1e5a    41 minutes ago  912MB
naimurislam/first-image latest          55faf92b1e5a    41 minutes ago  912MB
<none>              <none>         90e2d9aec2e8    4 hours ago    168MB
<none>              <none>         2be862428a26    4 hours ago    168MB
ubuntu              latest         fec8bfd95b54    6 weeks ago    78.1MB
mysql               latest         56a8c14e1404    6 weeks ago    603MB
hello-world         latest         d2c94e258dcb    19 months ago  13.3kB
naimur@Navid-24141160:~/mydockerfiles$
```

Now we have to push the image to docker hub, to do so I need to login my docker hub account with the following command

\$sudo docker login -u naimur707

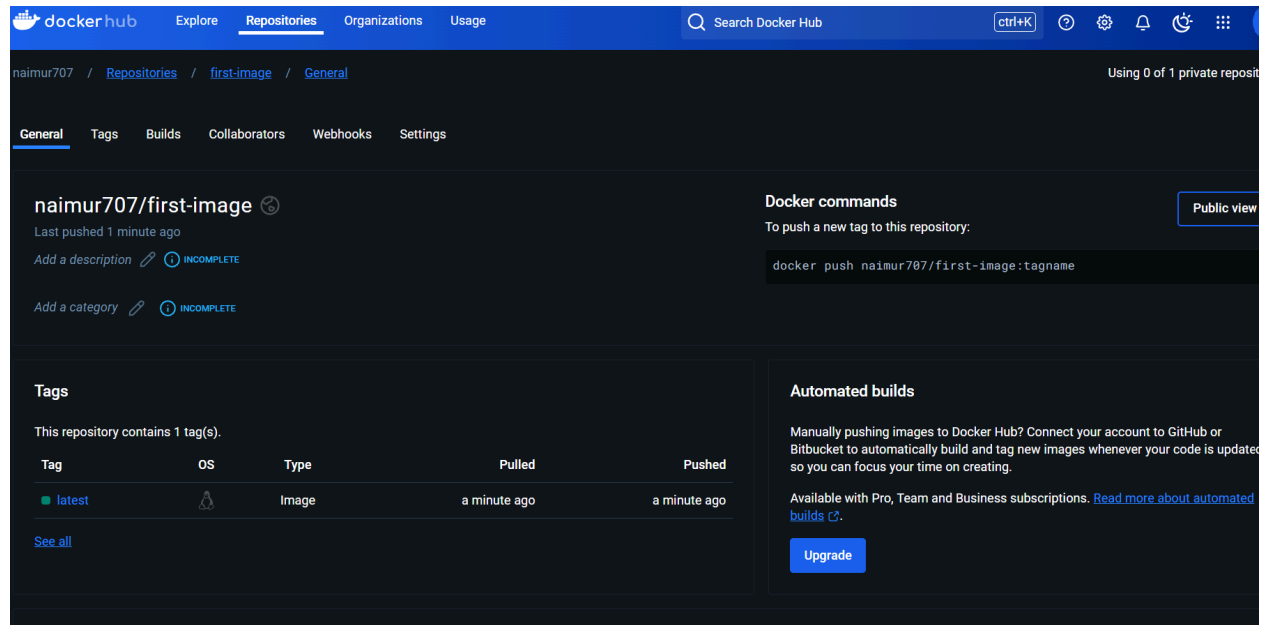
Here instead of naimur707 you will put your username.

Now push the docker image with the following command

\$sudo docker push naimur707/first-image:latest

```
naimur@Navid-24141160:~/mydockerfiles$ sudo docker push naimur707/first-image:latest
The push refers to repository [docker.io/naimur707/first-image]
7e42a1055e36: Pushed
9bf76c8f25eb: Pushed
5f70bf18a086: Pushed
7d1674e3317c: Pushed
0d5f5a015e5d: Pushed
3c777d951de2: Pushed
f8a91dd5fc84: Pushed
cb81227abde5: Pushed
e01a454893a9: Pushed
c45660adde37: Pushed
fe0fb3ab4a0f: Pushed
f1186e5061f2: Pushed
b2dba7477754: Pushed
latest: digest: sha256:486163b86867d94799eb423c1396e0afdbff7ee7d85af7069540422f91b009ce size: 3042
naimur@Navid-24141160:~/mydockerfiles$
```

Here is the image of the repository



Task-8: How to make your own private registry? Show steps.

In this task we are going to create my own private registry.

Here are the steps to Create a Private Docker Registry

First, install Docker Compose on your machine using the following command:

\$ sudo apt install docker-compose

```
naimur@Navid-24141160:~$ sudo apt install docker-compose
[sudo] password for naimur:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-attr python3-distutils python3-docker python3-dockerpty
  python3-docopt python3-dotenv python3-jsonschema python3-pyrsistent
  python3-setuptools python3-texttable python3-websocket
Suggested packages:
  python-attr-doc python-jsonschema-doc python-setuptools-doc
Recommended packages:
  docker.io
The following NEW packages will be installed:
  docker-compose python3-attr python3-distutils python3-docker
  python3-dockerpty python3-docopt python3-dotenv python3-jsonschema
  python3-pyrsistent python3-setuptools python3-texttable python3-websocket
0 upgraded, 12 newly installed, 0 to remove and 7 not upgraded.
Need to get 911 kB of archives.
After this operation, 4,842 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-dis
utils all 3.10.8-1~22.04 [139 kB]
```

Now Create a docker-compose.yml file and edit it with vim (or any text editor).

First open the file

\$vim docker-compose.yml

```
naimur@Navid-24141160:~$ vim docker-compose.yml
naimur@Navid-24141160:~$
```

Add the following configuration:

version: '3'

services:

registry:

image: registry:2

ports:

- "5000:5000"

```
version: '3'
services:
  registry:
    image: registry:2
    ports:
      - "5000:5000"
```

This configuration uses the official Docker Registry image (registry:2) and maps port 5000 on the container to port 5000 on your host machine. This setup allows you to access the registry by sending requests to port 5000 of your server.

Now Run the docker

Start the registry container using Docker Compose:

\$sudo docker-compose up -d

```
naimur@Navid-24141160:~$ sudo docker-compose up -d
Creating network "naimur_default" with the default driver
Pulling registry (registry:2)...
2: Pulling from library/registry
dc0decf4841d: Pull complete
6cb0aa443e23: Pull complete
813676e291ef: Pull complete
dc2fb7dcec61: Pull complete
916205650bfe: Pull complete
Digest: sha256:543dade69668e02e5768d7ea2b0aa4fae6aa7384c9a5a8dbecc2be5136079ddb
Status: Downloaded newer image for registry:2
Creating naimur_registry_1 ... done
```

Now push images to your private registry. For demonstration, use the Alpine Linux image, as it is lightweight and downloads quickly.

Pull the Alpine image:

\$sudo docker pull alpine

```
naimur@Navid-24141160:~$ sudo docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
da9db072f522: Pull complete
Digest: sha256:1e42bbe2508154c9126d48c2b8a75420c3544343bf86fd041fb7527e017a4b4a
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

Tag the image to reference your local registry:

\$sudo docker tag alpine localhost:5000/my-alpine

```
naimur@Navid-24141160:~$ sudo docker tag alpine localhost:5000/my-alpine
```

Verify the image tag by listing your Docker images:

\$sudo docker images

```
naimur@Navid-24141160:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
naimur707/first-image	latest	55faf92b1e5a	3 days ago	912MB
naimurislam/first-image	latest	55faf92b1e5a	3 days ago	912MB
<none>	<none>	2be862428a26	3 days ago	168MB
<none>	<none>	90e2d9aec2e8	3 days ago	168MB
ubuntu	latest	fec8bfd95b54	6 weeks ago	78.1MB
mysql	latest	56a8c14e1404	7 weeks ago	603MB
alpine	latest	63b790fccc90	2 months ago	7.8MB
localhost:5000/my-alpine	latest	63b790fccc90	2 months ago	7.8MB
registry	2	c18a86d35e98	14 months ago	25.4MB
hello-world	latest	d2c94e258dcb	19 months ago	13.3kB

Push the tagged image to your private registry:

\$sudo docker push localhost:5000/my-alpine

```
naimur@Navid-24141160:~$ sudo docker push localhost:5000/my-alpine
Using default tag: latest
The push refers to repository [localhost:5000/my-alpine]
75654b8eeebd: Pushed
latest: digest: sha256:3e21c52835bab96cbecb471e3c3eb0e8a012b91ba2f0b934bd0b5394cd570b9f size: 527
naimur@Navid-24141160:~$
```

Note: This setup works for a local registry. If you want to host the registry on a remote server, you will need to configure a secure SSL connection to enable secure communication.

And we are done creating a private registry and also pushed an image in it.

Task-9: Create a small website or app with minimal functionality (Could be a simple HTML website that has a button that opens a static image/file) inside a Docker container. Then run the application (inside the container) in the background of your HOST machine in any port. Browse the website from your host machine.

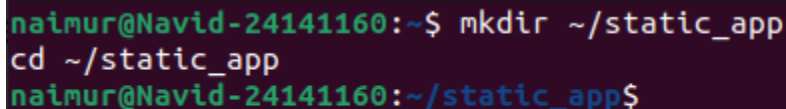
In this task, we will create a small simple web app that will run in the background of our host machine and we will browse that website from our local host machine. To do this follow the below steps:

First

Create a directory named static_app in your home directory:

\$ mkdir ~/static_app

\$ cd ~/static_app

A terminal window with a dark purple background. The prompt is 'naimur@Navid-24141160:~\$'. The first command is 'mkdir ~/static_app' and the second is 'cd ~/static_app'. The prompt changes to 'naimur@Navid-24141160:~/static_app\$' after the second command.

```
naimur@Navid-24141160:~$ mkdir ~/static_app
cd ~/static_app
naimur@Navid-24141160:~/static_app$
```

Inside this directory, create an index.html file for the website content:

\$ vim index.html

A terminal window with a dark purple background. The prompt is 'naimur@Navid-24141160:~/static_app\$'. The command 'vim index.html' is entered.

```
naimur@Navid-24141160:~/static_app$ vim index.html
```

Add the following content:

<p>Welcome To Cloud Computing on naimurs branch <button type="button">Click Here!</button></p>


```
<p>Welcome To Cloud Computing on naimurs branch <button type="button">Click Here  
!</button></p>  
~  
~  
~  
~  
~  
~
```

Now Create a Dockerfile to define the Docker image:

\$vim Dockerfile

```
naimur@Navid-24141160:~/static_app$ vim Dockerfile  
naimur@Navid-24141160:~/static_app$
```

Add the following instructions:

FROM nginx:alpine

COPY . /usr/share/nginx/html

```
FROM nginx:alpine  
COPY . /usr/share/nginx/html  
~  
~  
~  
~  
~  
~
```

Type **:wq** and save it

This setup uses the lightweight NGINX image (nginx:alpine) to serve the static index.html file.

Now Build the Docker image using the following command:

\$sudo docker build -t static-app:v1 .

```
naimur@Navid-24141160:~/static_app$ sudo docker build -t static-app:v1 .
[sudo] password for naimur:
[+] Building 9.9s (8/8) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 85B                               0.0s
=> [internal] load metadata for docker.io/library/nginx:alpine  6.3s
=> [auth] library/nginx:pull token for registry-1.docker.io     0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 220B                                  0.0s
=> [1/2] FROM docker.io/library/nginx:alpine@sha256:41523187cf7d7a2f2677 3.2s
```

Run the container and bind port 80 of the container to port 80 of the host machine:

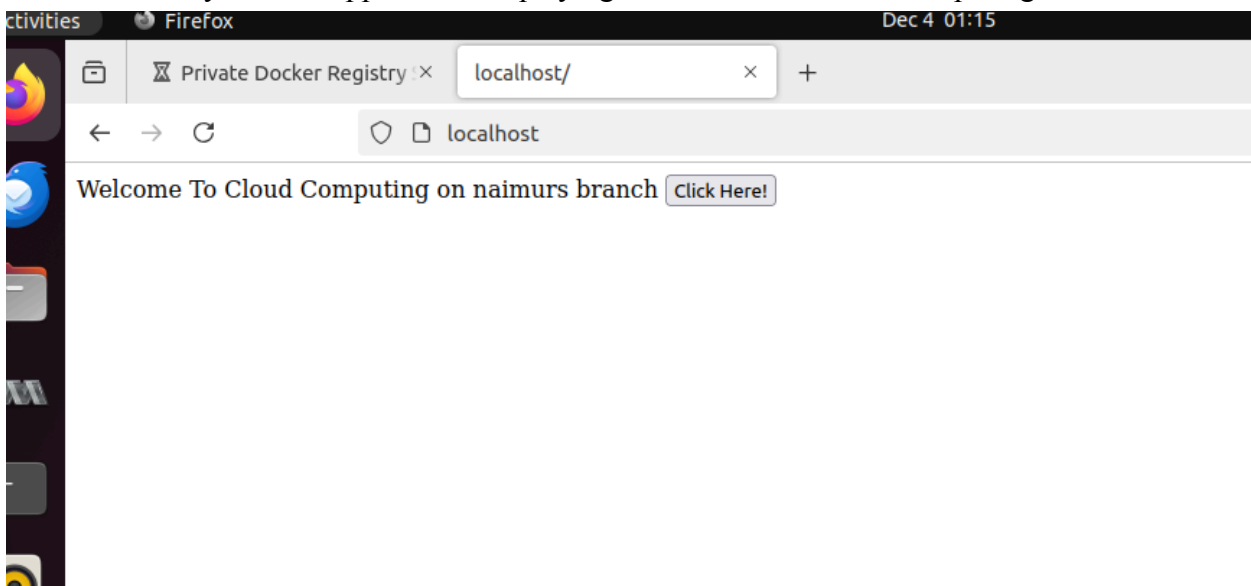
\$sudo docker run -d --name=app-container -p 80:80 static-app:v1

```
naimur@Navid-24141160:~/static_app$ sudo docker run -d --name=app-container -p 80:80 static-app:v1
77264942046eea03727cca4421a6a2481112549bd83c5ae90b70655781bcf5a4
naimur@Navid-24141160:~/static_app$
```

Now open a web browser on your host machine and navigate to:

http://localhost

You should see your web application displaying "Welcome To Cloud Computing" with a button.



Task-10: Migrate the new container having the application into another machine. Again run the container and browse the URL. It should work.

If we push the web app image to a public registry like Docker Hub, anyone with access to the registry can pull the image and run it on their machine.

By running the container, they can host the web app on their local environment and access it via <http://localhost> on their browser.

The commands to push and pull the image:

Push to Docker Hub:

```
$sudo docker tag static-app:v1 naimur707/static-app:v1  
sudo docker push naimur707/static-app:v1
```

```
naimur@Navid-24141160:~/static_app$ sudo docker tag static-app:v1 naimur707/static-app:v1  
sudo docker push naimur707/static-app:v1  
The push refers to repository [docker.io/naimur707/static-app]  
c844465cc666: Pushed  
2430c01bea64: Mounted from library/nginx  
b11b58162504: Mounted from library/nginx  
8b5ce426f73d: Mounted from library/nginx  
884b72c14f15: Mounted from library/nginx  
4a37d1b49911: Mounted from library/nginx  
4e8a0009474a: Mounted from library/nginx  
287563f25f8b: Mounted from library/nginx  
75654b8eeebd: Mounted from library/nginx  
v1: digest: sha256:e3bd4baab3d54324cf4e3cdf2481c91c8f64b6af6f69256fe4636f77c73e9dee size: 2196  
naimur@Navid-24141160:~/static_app$
```

Pull and Run on Another Machine:

```
$sudo docker pull naimur707/static-app:v1  
sudo docker run -d -p 80:80 naimur707/static-app:v1
```

This makes the web app accessible locally for anyone who pulls and runs the image.

Welcome To Cloud Computing on naimurs branch [Click Here!](#)

Note: Here instead of naimur707 use your docker username. And we are done!

