Introduction to Container's

Learn with Bilal : https://www.youtube.com/@bilalmazhar100
GitHub : https://github.com/BilalMaz/DevOps-Architect-BootCamp

About me

Hi there, my name is Bilal and I will Welcome you to DevOps boot camp! I am thrilled to have you join us for this exciting journey of learning and discovery.

In this boot camp, we will be exploring the principles and practices of DevOps, which is a set of methodologies and tools that aims to bridge the gap between software development and operations. DevOps is an increasingly important area in the field of software engineering, as it helps organizations to streamline their processes, improve their agility, and deliver better value to their customers.

By the end of this boot camp, you will have gained a comprehensive understanding of DevOps and its key concepts, as well as practical skills in areas such as infrastructure automation, continuous integration and delivery, monitoring and logging, and more. You will be equipped with the knowledge and tools to apply DevOps principles in your own work and contribute to the success of your organization.

I am always looking to connect with other professionals in the field, share ideas and insights, and stay up to date on the latest trends and developments. I welcome the opportunity to connect with you and explore ways in which we can collaborate and support each other.

GitHub : https://github.com/BilalMaz/DevOps-Architect-BootCamp

LinkedIn : https://www.linkedin.com/in/bilalmazhar-cyber-security-consultant/

What is container ?

- Containerization is a method of OS virtualization that packages applications along with dependencies, libraries, and configurations into self-contained units called containers.
- Containers provide lightweight and isolated environments, ensuring consistent application execution across different computing environments.
- **Key Features and Benefits:**
  1. Portability,
  2. Scalability,
  3. Efficiency,
  4. Dependency Management,
  5. Reproducibility.

Container Tools

Sr.	Tools	Description
1	Docker	Docker is a widely adopted containerization platform that provides a comprehensive set of tools for building, managing, and deploying containers. It emphasizes ease of use and portability across different environments.
2	Podman	Podman is an alternative container engine to Docker that focuses on providing a secure and lightweight container runtime. It offers a CLI-compatible interface with Docker and is designed to be daemon less.
3	rkt	rkt (pronounced "rocket") is a container runtime developed by CoreOS. It aims to provide security, simplicity, and composability. rkt is known for its focus on security and its ability to run containers without a central daemon.
	Contained	Contained is an open-source container runtime that provides a low-level interface for managing container execution and image distribution. It is designed to be embedded in higher-level container orchestration platforms.

Docker

Docker is a popular containerization platform that simplifies building, deploying, and managing containers,

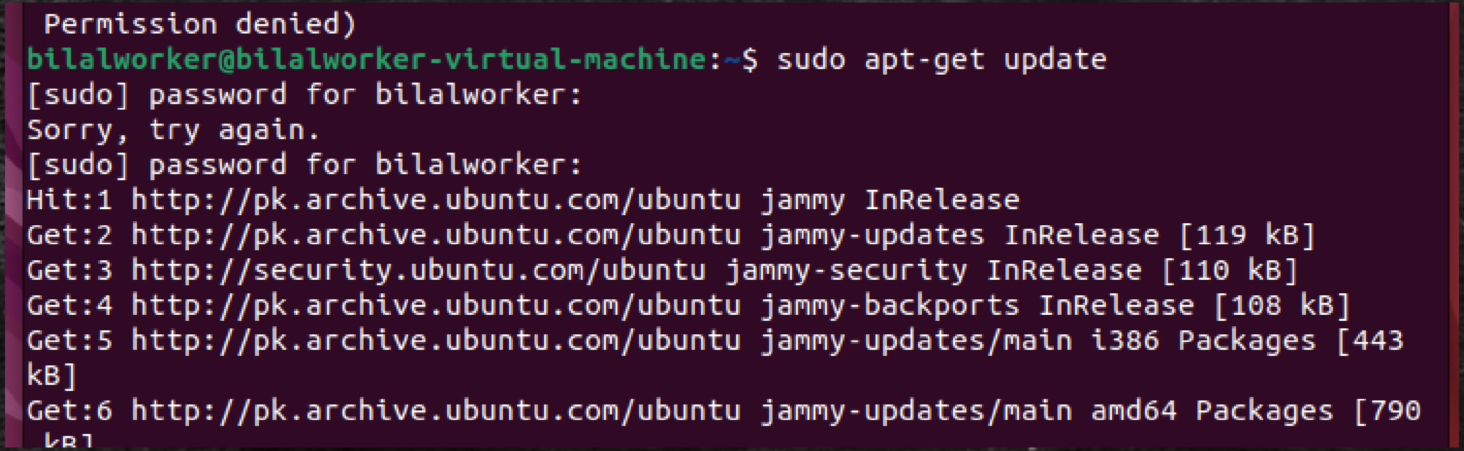
Key Features:

1. Container Images,
2. Docker file,
3. Containerization,
4. Docker Engine,
5. Container Registry,
6. Docker Compose,
7. Docker Swarm

Docker enables easy application portability, scalability, and consistent deployment across environments.

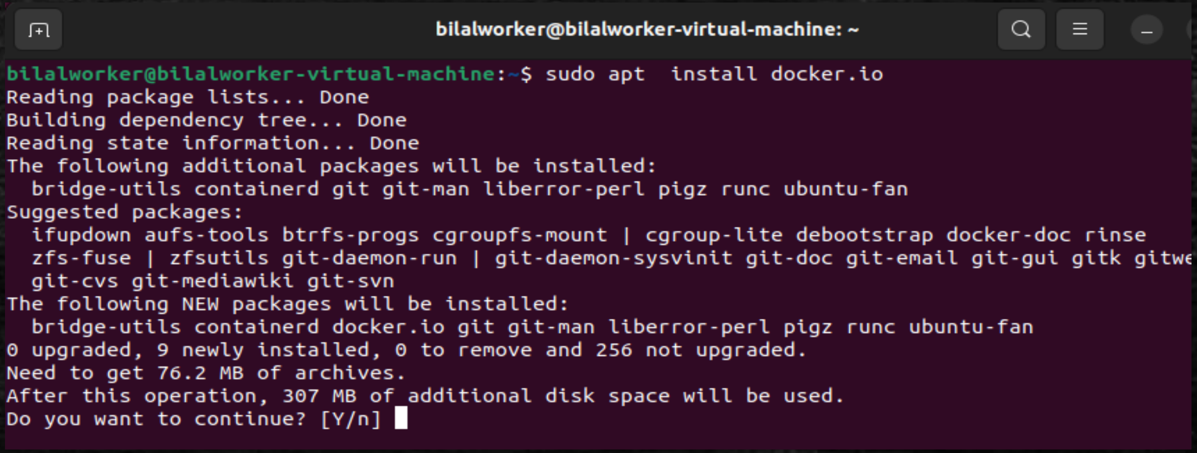
Docker Installation

Sudo apt-get update



```
 Permission denied)
bilalworker@bilalworker-virtual-machine:~$ sudo apt-get update
[sudo] password for bilalworker:
Sorry, try again.
[sudo] password for bilalworker:
Hit:1 http://pk.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://pk.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:4 http://pk.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:5 http://pk.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [443
kB]
Get:6 http://pk.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [790
 kB]
```

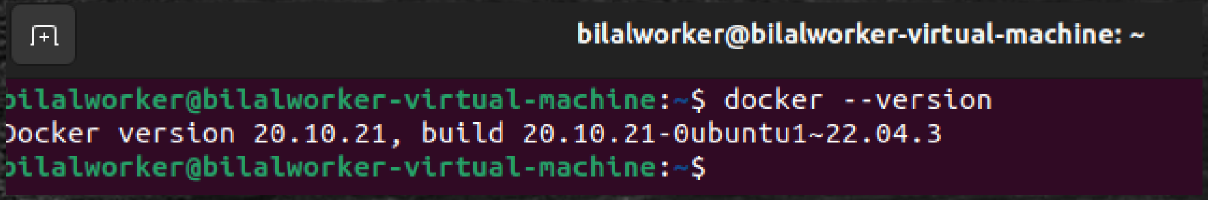
Sudo apt-get install docker.io



```
bilalworker@bilalworker-virtual-machine:~$ sudo apt  install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse
  zfs-fuse | zfsutils git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitwe
  git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 256 not upgraded.
Need to get 76.2 MB of archives.
After this operation, 307 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Docker Installation

docker --version

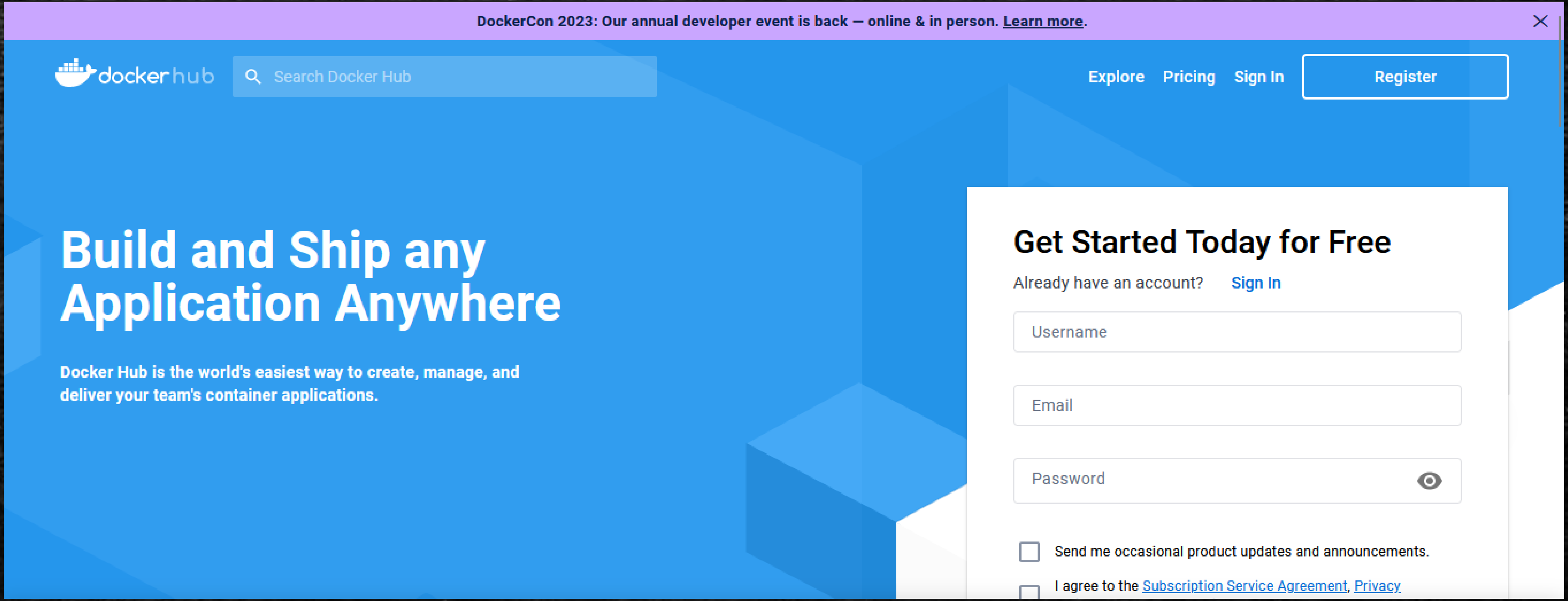


bilalworker@bilalworker-virtual-machine: ~

```
ilalworker@bilalworker-virtual-machine:~$ docker --version
ocker version 20.10.21, build 20.10.21-0ubuntu1~22.04.3
ilalworker@bilalworker-virtual-machine:~$
```

Docker Hub

https://hub.docker.com/


DockerCon 2023: Our annual developer event is back — online & in person. Learn more.

docker hub

Search Docker Hub

Explore Pricing Sign In Register

Build and Ship any Application Anywhere

Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.

Get Started Today for Free

Already have an account? Sign In

Username

Email

Password

Send me occasional product updates and announcements.

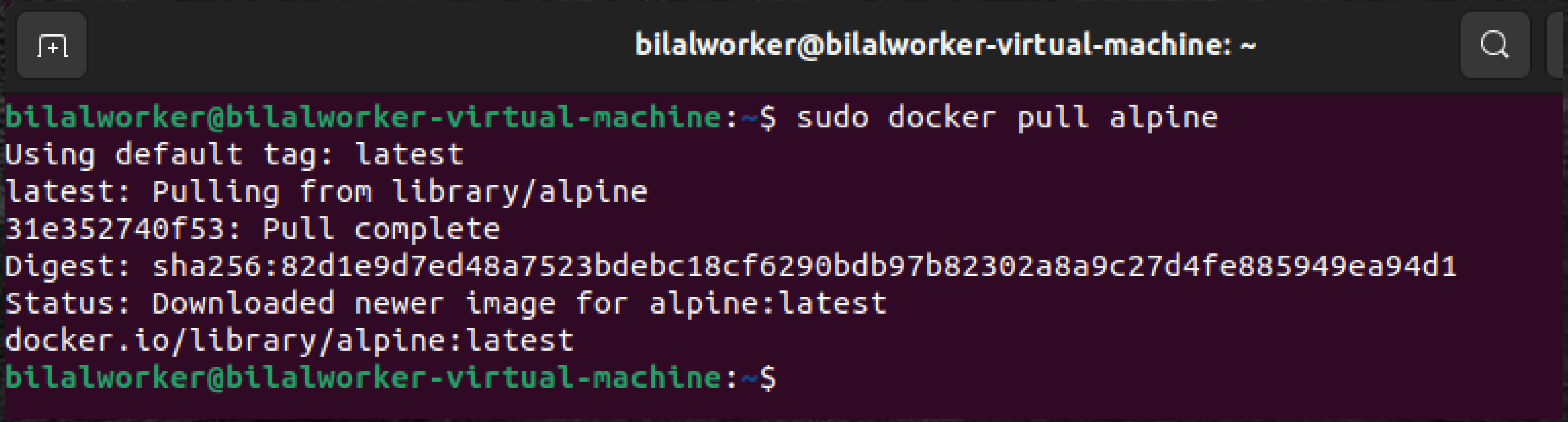
I agree to the Subscription Service Agreement, Privacy

Docker basic Commands

Commands	Description
docker run	Creates and runs a new container from an image
docker start	Starts a stopped container
docker stop	Stops a running container
docker restart	Stops and starts a container
docker rm	Removes one or more containers
docker ps	Lists running containers
docker ps -a	Lists all containers (including stopped ones)
docker images	Lists available images
docker pull	Pulls an image from a registry to the local machine
docker push	Pushes an image to a registry
docker build	Builds a Docker image from a Dockerfile

Lab 1 : Pull and Run a Linux Container

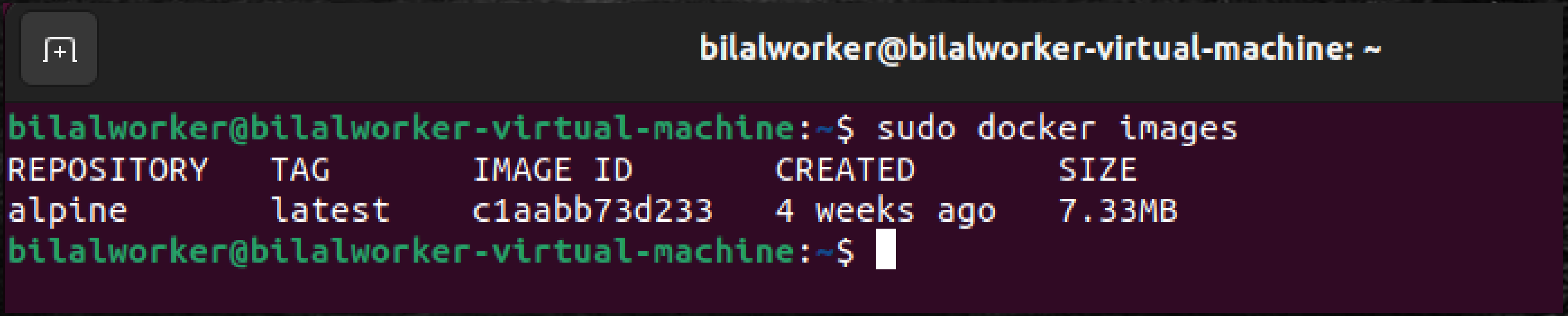
docker pull <linux_image_name>



bilalworker@bilalworker-virtual-machine: ~

```
bilalworker@bilalworker-virtual-machine:~$ sudo docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
31e352740f53: Pull complete
Digest: sha256:82d1e9d7ed48a7523bdebc18cf6290bdb97b82302a8a9c27d4fe885949ea94d1
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
bilalworker@bilalworker-virtual-machine:~$
```

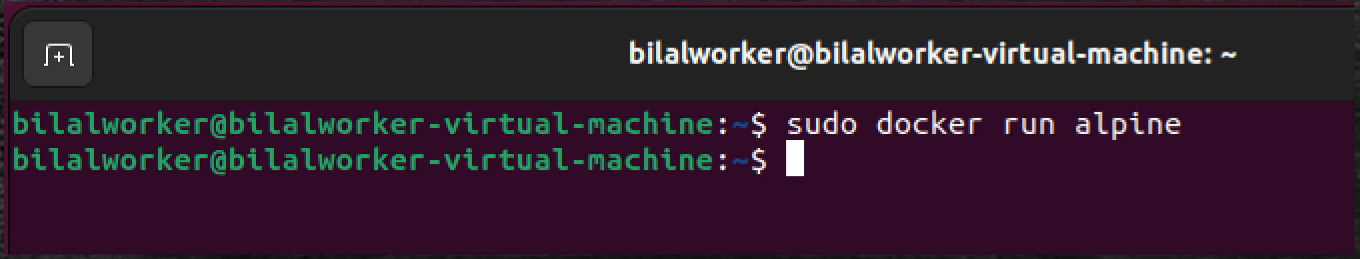
sudo docker images



bilalworker@bilalworker-virtual-machine: ~

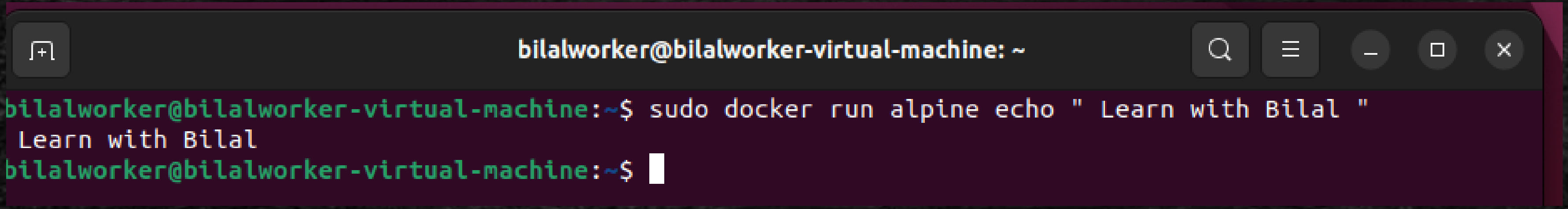
```
bilalworker@bilalworker-virtual-machine:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
alpine        latest    c1aabb73d233   4 weeks ago   7.33MB
bilalworker@bilalworker-virtual-machine:~$
```

sudo docker run <container name>



```
bilalworker@bilalworker-virtual-machine: ~
bilalworker@bilalworker-virtual-machine:~$ sudo docker run alpine
bilalworker@bilalworker-virtual-machine:~$ 
```

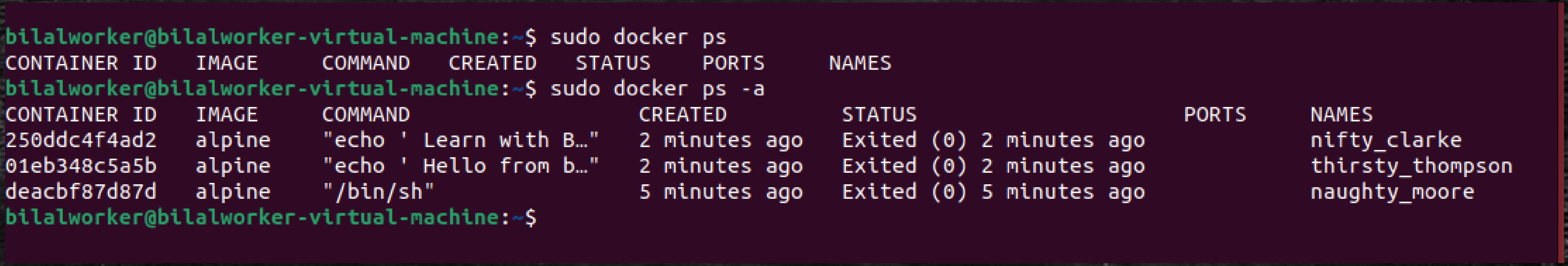
sudo docker run alpine echo " Learn with Bilal "



```
bilalworker@bilalworker-virtual-machine: ~
bilalworker@bilalworker-virtual-machine:~$ sudo docker run alpine echo " Learn with Bilal "
 Learn with Bilal
bilalworker@bilalworker-virtual-machine:~$ 
```

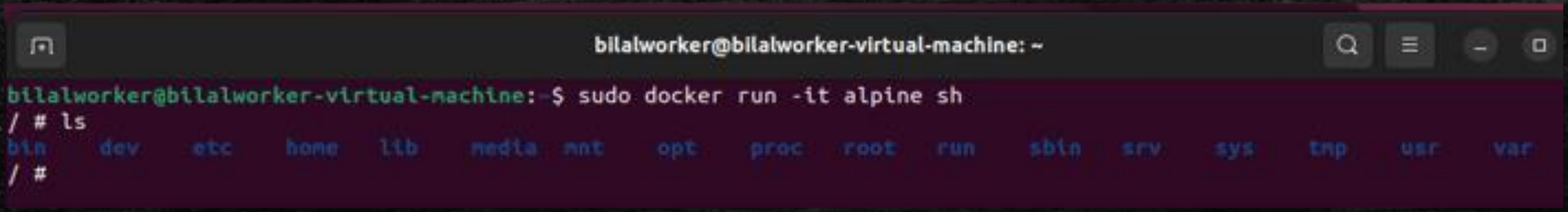
Docker process

sudo docker ps



```
bilalworker@bilalworker-virtual-machine:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
bilalworker@bilalworker-virtual-machine:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS                     PORTS     NAMES
250ddc4f4ad2   alpine    "echo ' Learn with B…"   2 minutes ago   Exited (0) 2 minutes ago             nifty_clarke
01eb348c5a5b   alpine    "echo ' Hello from b…"   2 minutes ago   Exited (0) 2 minutes ago             thirsty_thompson
deacbf87d87d   alpine    "/bin/sh"                5 minutes ago   Exited (0) 5 minutes ago             naughty_moore
bilalworker@bilalworker-virtual-machine:~$
```

sudo docker run -it alpine sh

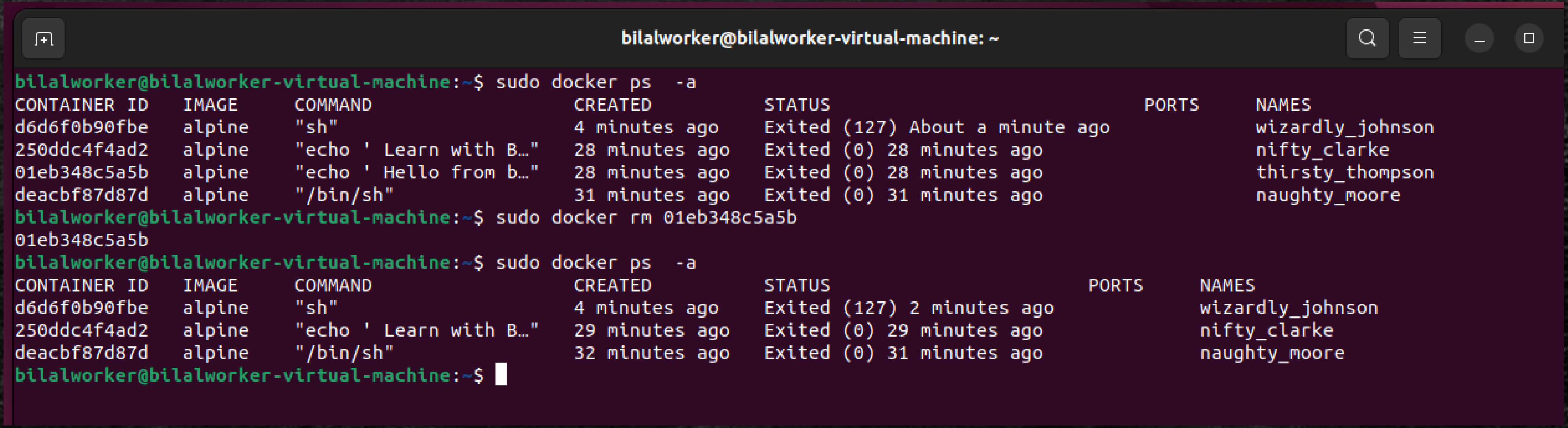


```
bilalworker@bilalworker-virtual-machine: ~
bilalworker@bilalworker-virtual-machine:~$ sudo docker run -it alpine sh
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin   srv    sys    tmp    usr    var
/ #
```

Docker remove

sudo docker ps -a

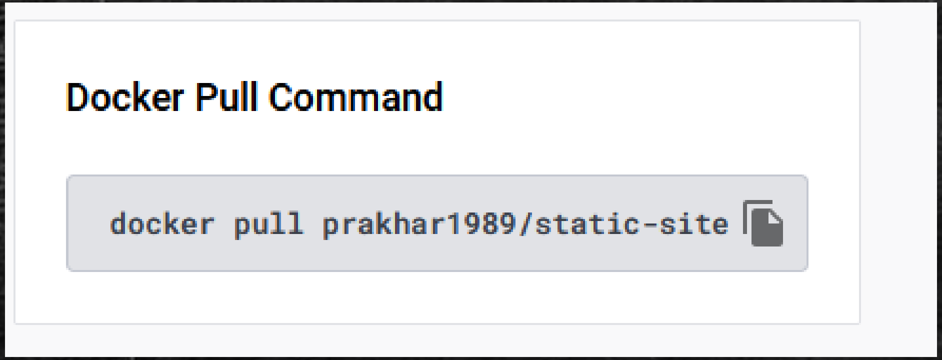
sudo docker rm <images ID >



```
bilalworker@bilalworker-virtual-machine:~$ sudo docker ps  -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                            PORTS     NAMES
d6d6f0b90fbe   alpine    "sh"                     4 minutes ago    Exited (127) About a minute ago             wizardly_johnson
250ddc4f4ad2   alpine    "echo ' Learn with B…"   28 minutes ago   Exited (0) 28 minutes ago                   nifty_clarke
01eb348c5a5b   alpine    "echo ' Hello from b…"   28 minutes ago   Exited (0) 28 minutes ago                   thirsty_thompson
deacbf87d87d   alpine    "/bin/sh"                31 minutes ago   Exited (0) 31 minutes ago                   naughty_moore
bilalworker@bilalworker-virtual-machine:~$ sudo docker rm 01eb348c5a5b
01eb348c5a5b
bilalworker@bilalworker-virtual-machine:~$ sudo docker ps  -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                        PORTS     NAMES
d6d6f0b90fbe   alpine    "sh"                     4 minutes ago    Exited (127) 2 minutes ago              wizardly_johnson
250ddc4f4ad2   alpine    "echo ' Learn with B…"   29 minutes ago   Exited (0) 29 minutes ago               nifty_clarke
deacbf87d87d   alpine    "/bin/sh"                32 minutes ago   Exited (0) 31 minutes ago               naughty_moore
bilalworker@bilalworker-virtual-machine:~$ 
```

Lab 2 : Host a Static Website using Container

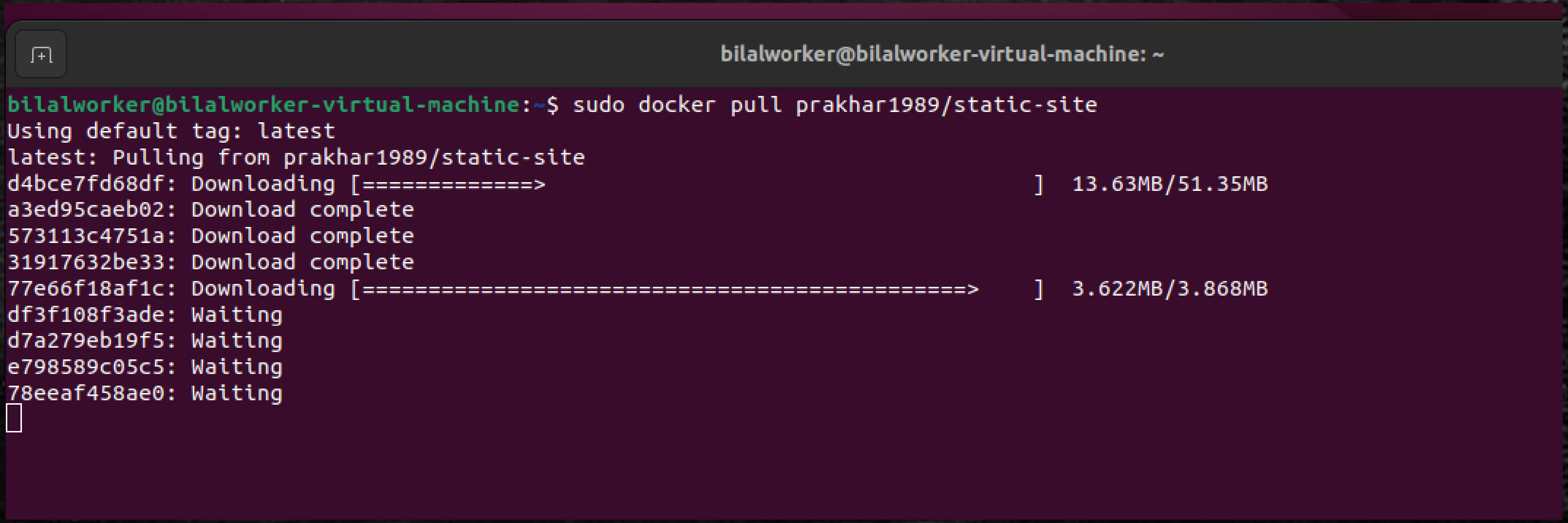
https://hub.docker.com/r/prakhar1989/static-site/



Docker Pull Command

```
docker pull prakhar1989/static-site
```

sudo docker pull prakhar1989/static-site

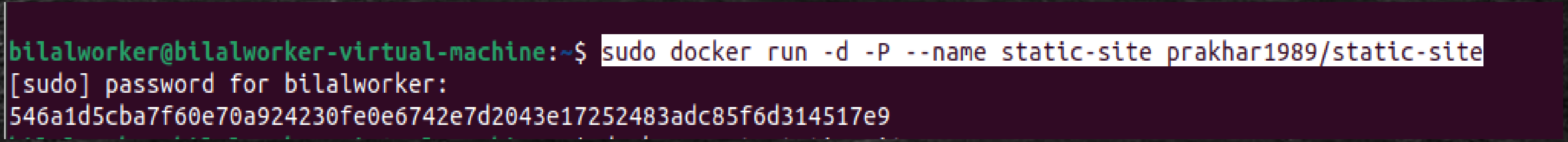


bilalworker@bilalworker-virtual-machine: ~

```
bilalworker@bilalworker-virtual-machine:~$ sudo docker pull prakhar1989/static-site
Using default tag: latest
latest: Pulling from prakhar1989/static-site
d4bce7fd68df: Downloading [==============>                                    ]  13.63MB/51.35MB
a3ed95caeb02: Download complete
573113c4751a: Download complete
31917632be33: Download complete
77e66f18af1c: Downloading [==============================================>    ]  3.622MB/3.868MB
df3f108f3ade: Waiting
d7a279eb19f5: Waiting
e798589c05c5: Waiting
78eeaf458ae0: Waiting
```

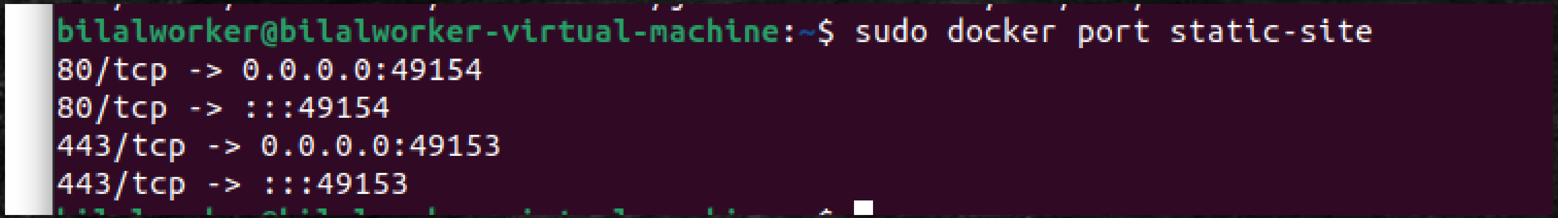
Lab 2 : Host a Static Website using Container

- sudo docker run -d -P --name static-site prakhar1989/static-site



```
bilalworker@bilalworker-virtual-machine:~$ sudo docker run -d -P --name static-site prakhar1989/static-site
[sudo] password for bilalworker:
546a1d5cba7f60e70a924230fe0e6742e7d2043e17252483adc85f6d314517e9
```

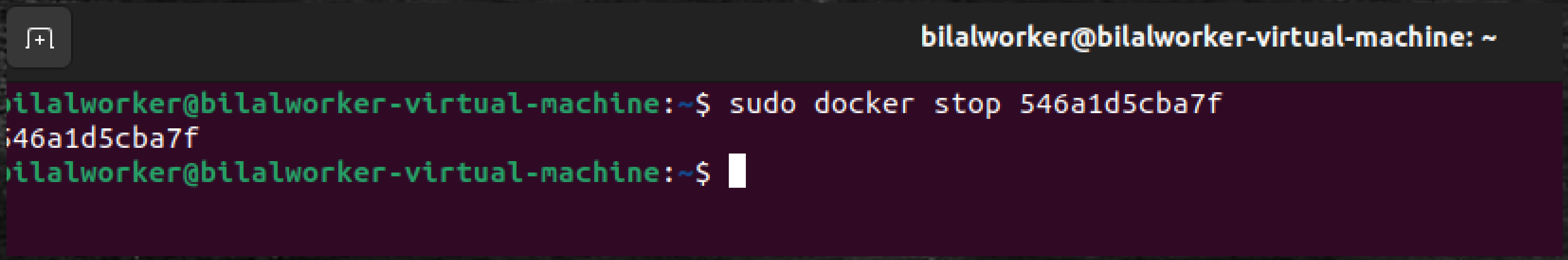
sudo docker port static-site



```
bilalworker@bilalworker-virtual-machine:~$ sudo docker port static-site
80/tcp -> 0.0.0.0:49154
80/tcp -> :::49154
443/tcp -> 0.0.0.0:49153
443/tcp -> :::49153
```

Static Docker " Stop "

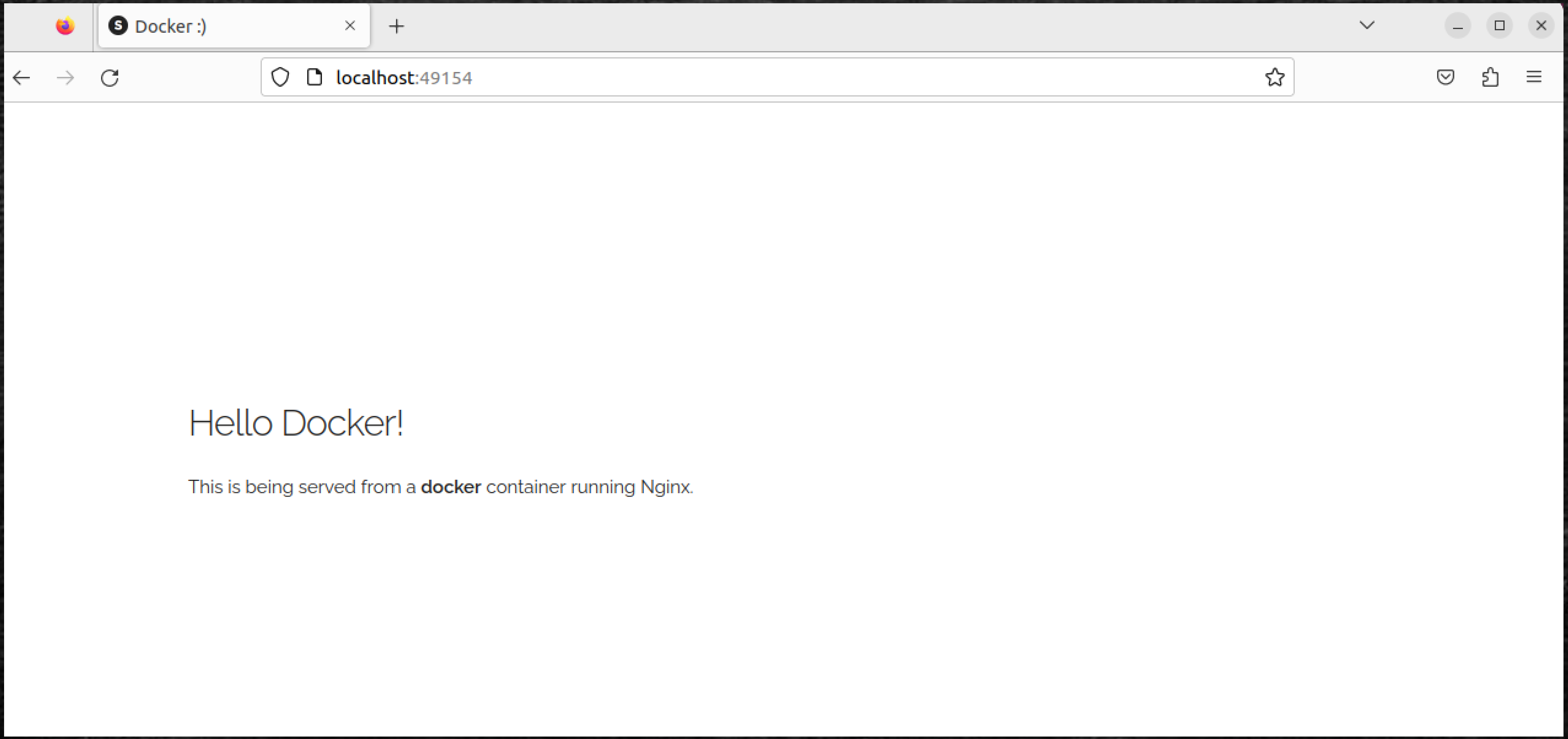
Sudo docker ps
sudo docker stop 546a1d5cba7f



bilalworker@bilalworker-virtual-machine: ~

```
ilalworker@bilalworker-virtual-machine:~$ sudo docker stop 546a1d5cba7f
46a1d5cba7f
ilalworker@bilalworker-virtual-machine:~$ 
```

Static website running from Container !



Hello Docker!

This is being served from a **docker** container running Nginx.

Docker file

A Dockerfile is a simple text file that contains a list of commands that the Docker client calls while creating an image. It's a simple way to automate the image creation process. The best part is that the commands you write in a Docker file are almost identical to their equivalent Linux commands. This means you don't really have to learn new syntax to create your own docker file

Docker file ready, you can build an image using the docker build command and the **-f** flag to specify the path to the Docker file. For example:

" docker build -t <image name> -f /path/to/Dockerfile "

Replace <image name> with the desired name for your image.

Sample Docker file

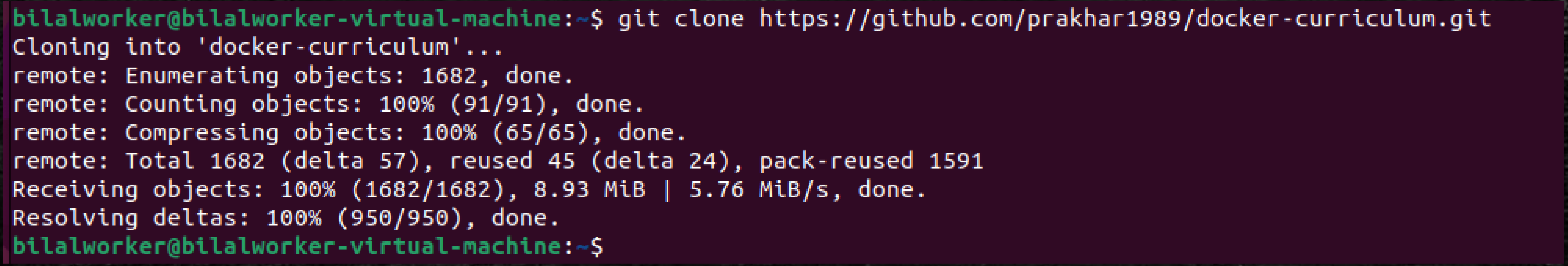
Tag	Description
FROM	Specifies the base image for the new image
WORKDIR	Sets the working directory inside the container
COPY	Copies files and directories from the host system to the container
ADD	
RUN	Executes commands inside the container during the build process
ENV	Sets environment variables for the container.
EXPOSE	Declares the ports that the container listens on at runtime.
ENTRYPOINT	Specifies the command that is run when the container starts
CMD	sets the command to be executed when the container starts.

```
# Base image
FROM <base_image>

# Set working directory
WORKDIR /app

# Copy files to the container
COPY <source> <destination>

# Run commands inside the container
RUN <command>

# Set environment variables
ENV <key>=<value>

# Expose ports
EXPOSE <port>

# Define entry point command
ENTRYPOINT ["<command>"]

# Define default command for the container
CMD ["<command>"]
```

Let's create " *Apna* " (Own) Container

Now that we have a better understanding of images, it's time to create our own. Our goal in this section will be to create an image that sandboxes a simple Flask application

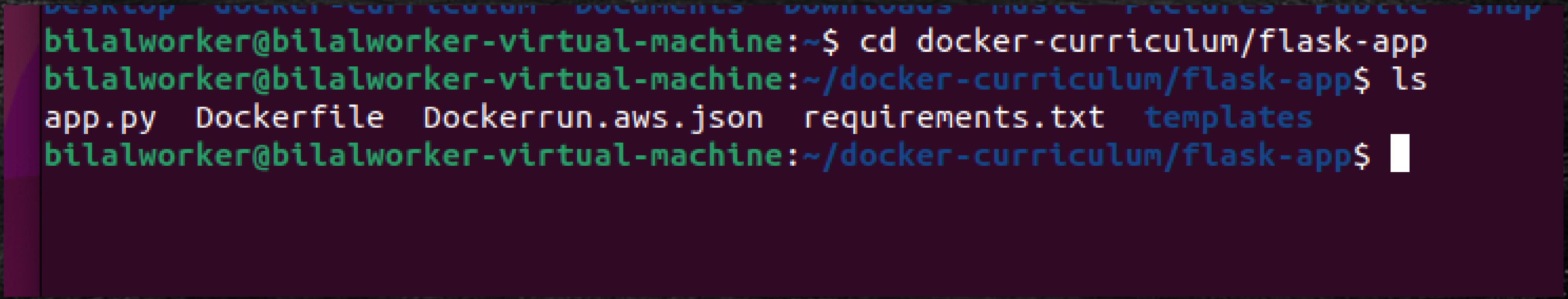
```
$ git clone https://github.com/prakhar1989/docker-curriculum.git
```



```
bilalworker@bilalworker-virtual-machine:~$ git clone https://github.com/prakhar1989/docker-curriculum.git
Cloning into 'docker-curriculum'...
remote: Enumerating objects: 1682, done.
remote: Counting objects: 100% (91/91), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 1682 (delta 57), reused 45 (delta 24), pack-reused 1591
Receiving objects: 100% (1682/1682), 8.93 MiB | 5.76 MiB/s, done.
Resolving deltas: 100% (950/950), done.
bilalworker@bilalworker-virtual-machine:~$
```

Let's create " *Apna* " (Own) Container

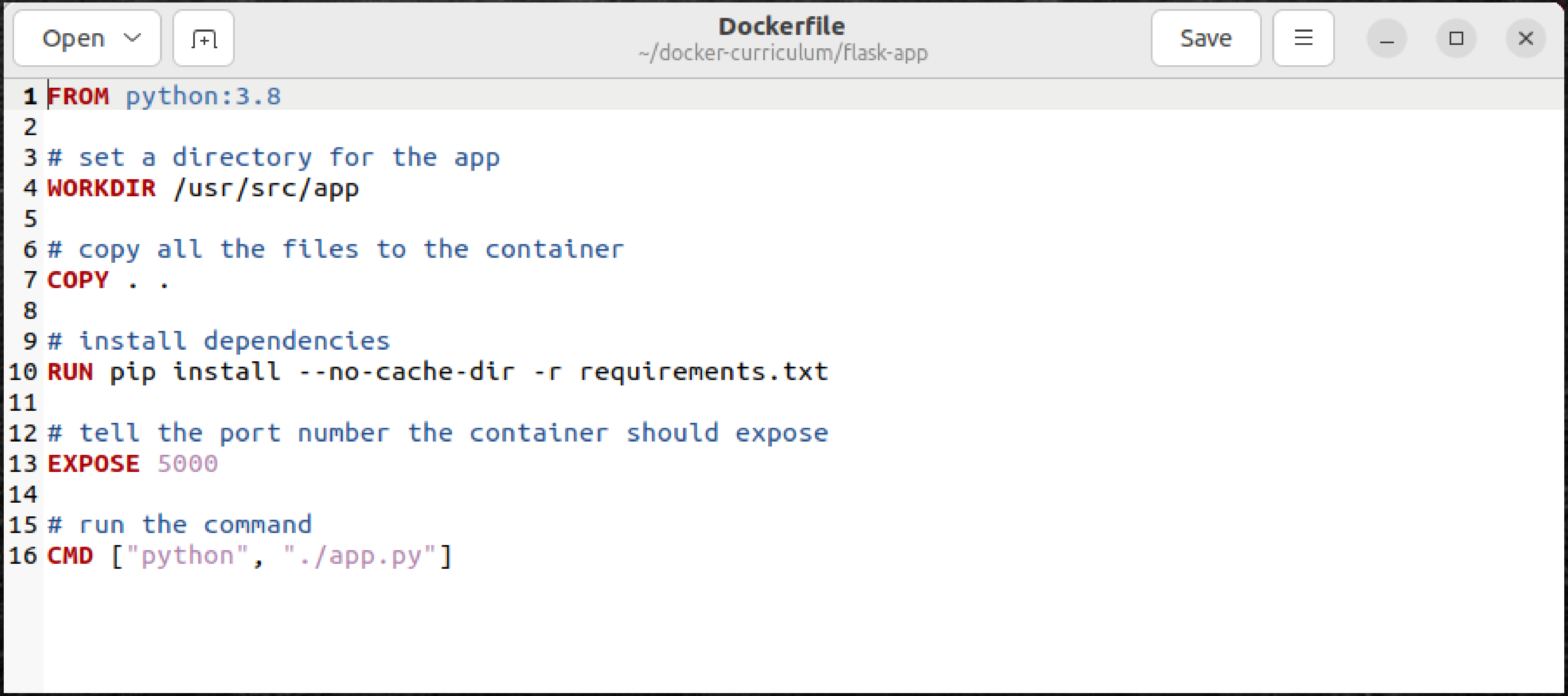
```
$ Ls
$ cd docker-curriculum/flask-app
```



```
Desktop  docker-curriculum  Documents  Downloads  Music  Pictures  Public  snap
bilalworker@bilalworker-virtual-machine:~$ cd docker-curriculum/flask-app
bilalworker@bilalworker-virtual-machine:~/docker-curriculum/flask-app$ ls
app.py  Dockerfile  Dockerrun.aws.json  requirements.txt  templates
bilalworker@bilalworker-virtual-machine:~/docker-curriculum/flask-app$ 
```

The next step now is to create an image with this web app. As mentioned above, all user images are based on a base image. Since our application is written in Python, the base image we're going to use will be [Python 3].

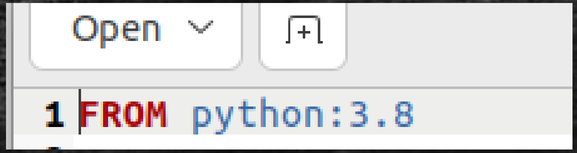
Docker file



```dockerfile
FROM python:3.8

# set a directory for the app
WORKDIR /usr/src/app

# copy all the files to the container
COPY . .

# install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# tell the port number the container should expose
EXPOSE 5000

# run the command
CMD ["python", "./app.py"]
```

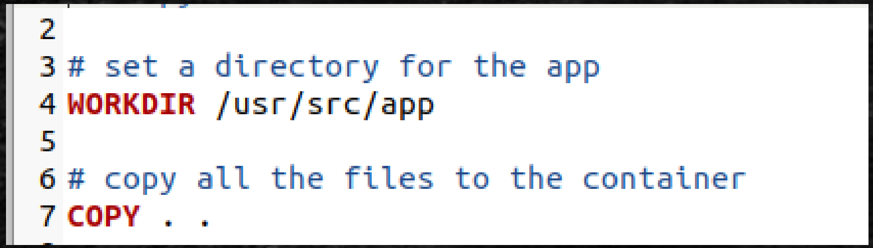
Docker Explain

We start with specifying our base image. Use the FROM keyword to do that -


Open

```
FROM python:3.8
```

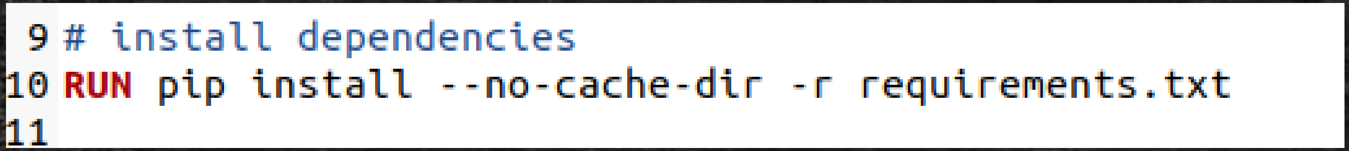
The next step usually is to write the commands of copying the files and installing the dependencies. First, we set a working directory and then copy all the files for our app.



```

# set a directory for the app
WORKDIR /usr/src/app

# copy all the files to the container
COPY . .
```

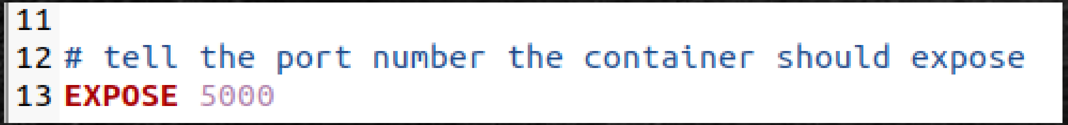
Docker Explain

Now, that we have the files, we can install the dependencies.



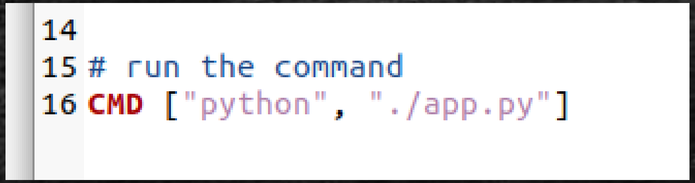
```
# install dependencies
RUN pip install --no-cache-dir -r requirements.txt

```

The next thing we need to specify is the port number that needs to be exposed. Since our flask app is running on port 5000, that's what we'll indicate.



```

# tell the port number the container should expose
EXPOSE 5000
```

he last step is to write the command for running the application, which is simply - python ./app.py. We use the CMD command to do that -



```
# run the command
CMD ["python", "./app.py"]
```

The primary purpose of CMD is to tell the container which command it should run when it is started. With that, our Docker file is now ready.

Docker Creation " Build "

Now that we have our Docker file, we can build our image. The docker build command does the heavy-lifting of creating a Docker image from a Docker file.

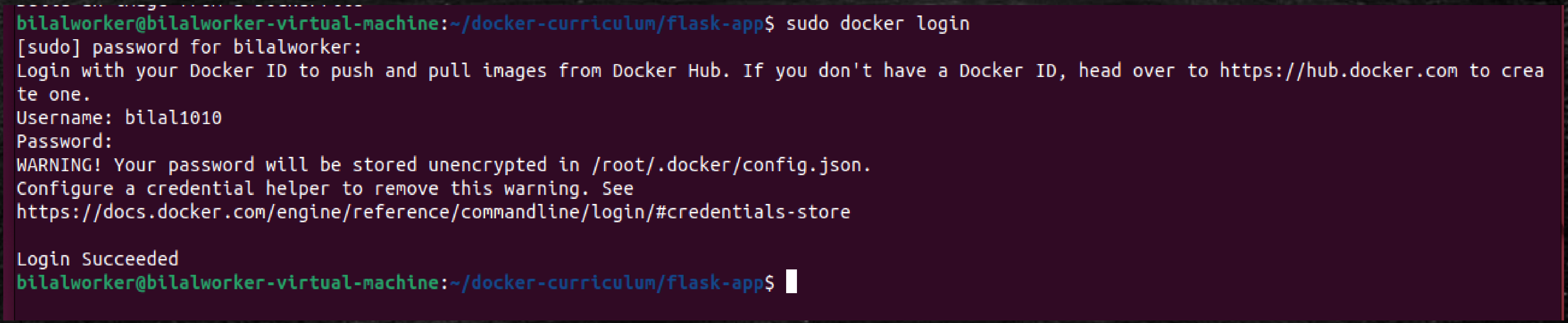
" docker build [OPTIONS] PATH "

- -t, --tag: Specifies the name and optional tag for the image. It allows you to provide a human-readable name and version for the image. For example, -t myapp:1.0 would tag the image as myapp with version 1.0.
- -f, --file: Specifies the path to the Docker file if it is not in the default location (i.e., ./Docker file in the build context).
- --build-arg: Allows you to set build-time variables used in the Docker file. You can pass key-value pairs to set values during the build process.
- --no-cache: Forces the build process to ignore the cached intermediate layers from previous builds, ensuring a clean build from scratch.
- --pull: Forces the command to always attempt to pull a newer version of the base image before starting the build.

Docker login

sudo docker login

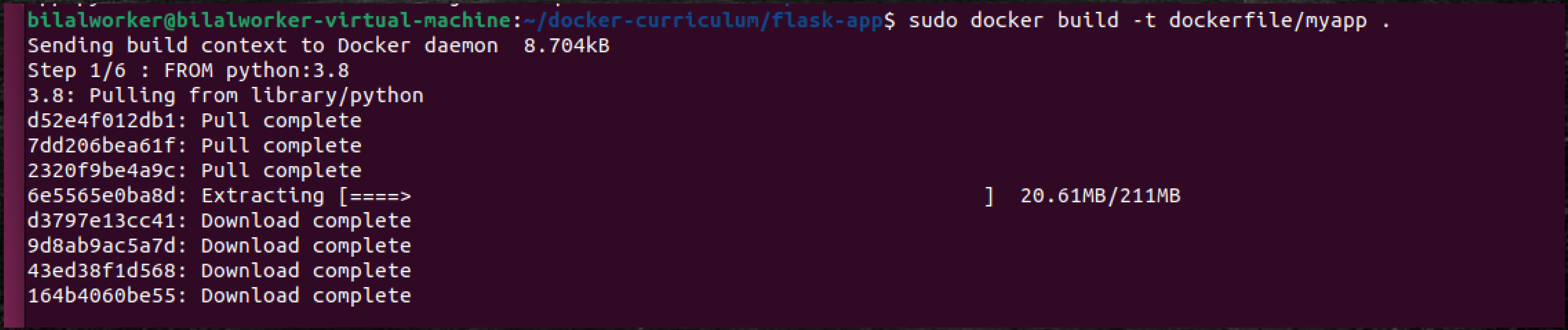
Enter Username and password (if not create docker hub login @ https://hub.docker.com/



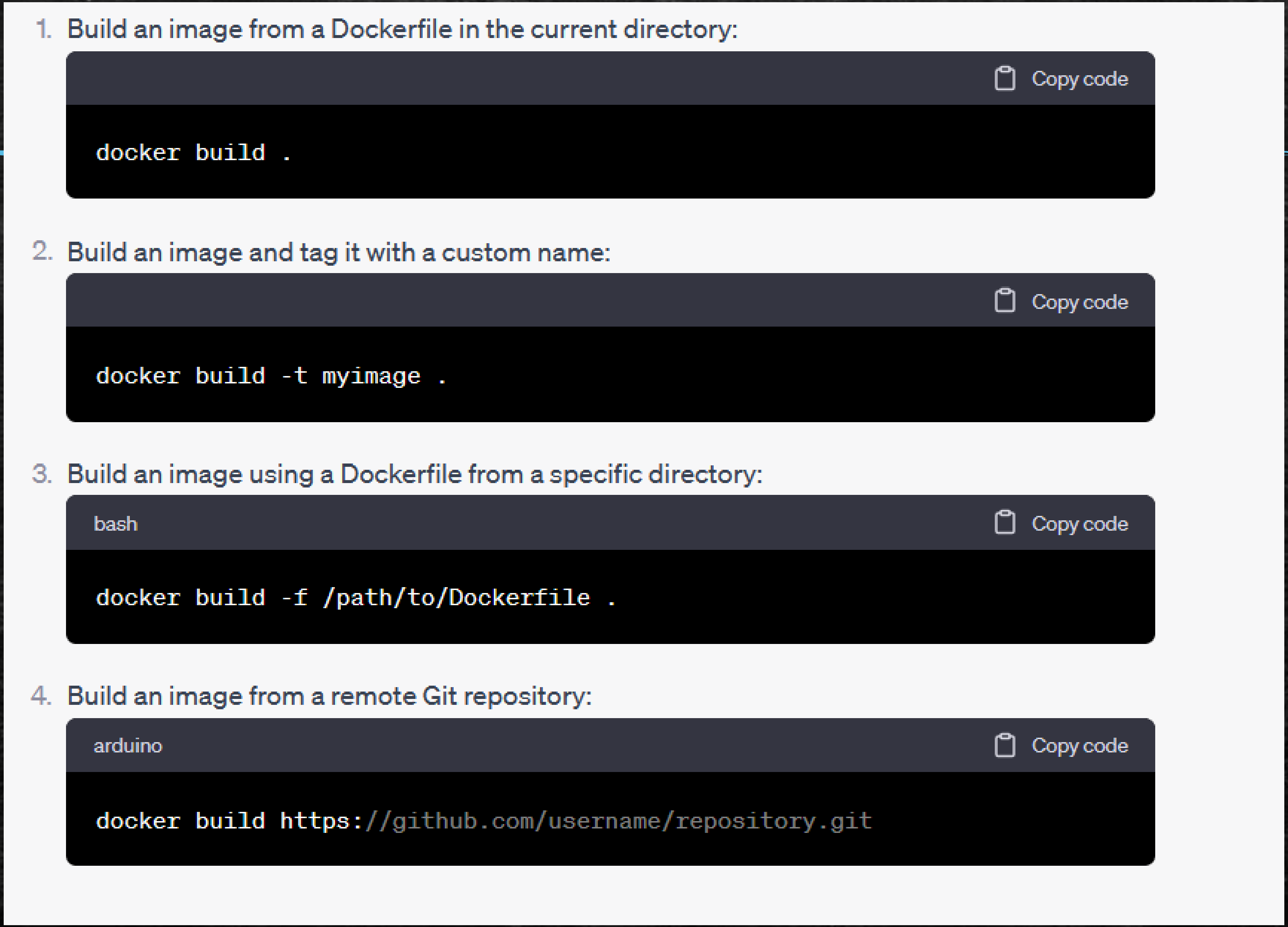
```
bilalworker@bilalworker-virtual-machine:~/docker-curriculum/flask-app$ sudo docker login
[sudo] password for bilalworker:
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to crea
te one.
Username: bilal1010
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
bilalworker@bilalworker-virtual-machine:~/docker-curriculum/flask-app$
```

Docker Creation " Build "

sudo docker build -t dockerfile/myapp .



```
bilalworker@bilalworker-virtual-machine:~/docker-curriculum/flask-app$ sudo docker build -t dockerfile/myapp .
Sending build context to Docker daemon  8.704kB
Step 1/6 : FROM python:3.8
3.8: Pulling from library/python
d52e4f012db1: Pull complete
7dd206bea61f: Pull complete
2320f9be4a9c: Pull complete
6e5565e0ba8d: Extracting [====>                                              ]  20.61MB/211MB
d3797e13cc41: Download complete
9d8ab9ac5a7d: Download complete
43ed38f1d568: Download complete
164b4060be55: Download complete
```



1. Build an image from a Dockerfile in the current directory:

Copy code

```
docker build .
```

2. Build an image and tag it with a custom name:

Copy code

```
docker build -t myimage .
```

3. Build an image using a Dockerfile from a specific directory:

bash Copy code

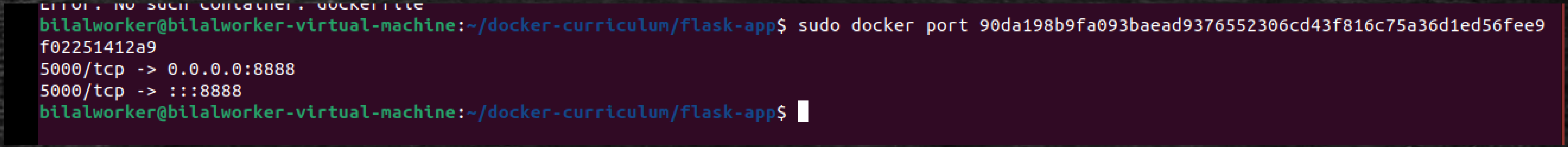
```bash
docker build -f /path/to/Dockerfile .
```

4. Build an image from a remote Git repository:

arduino Copy code

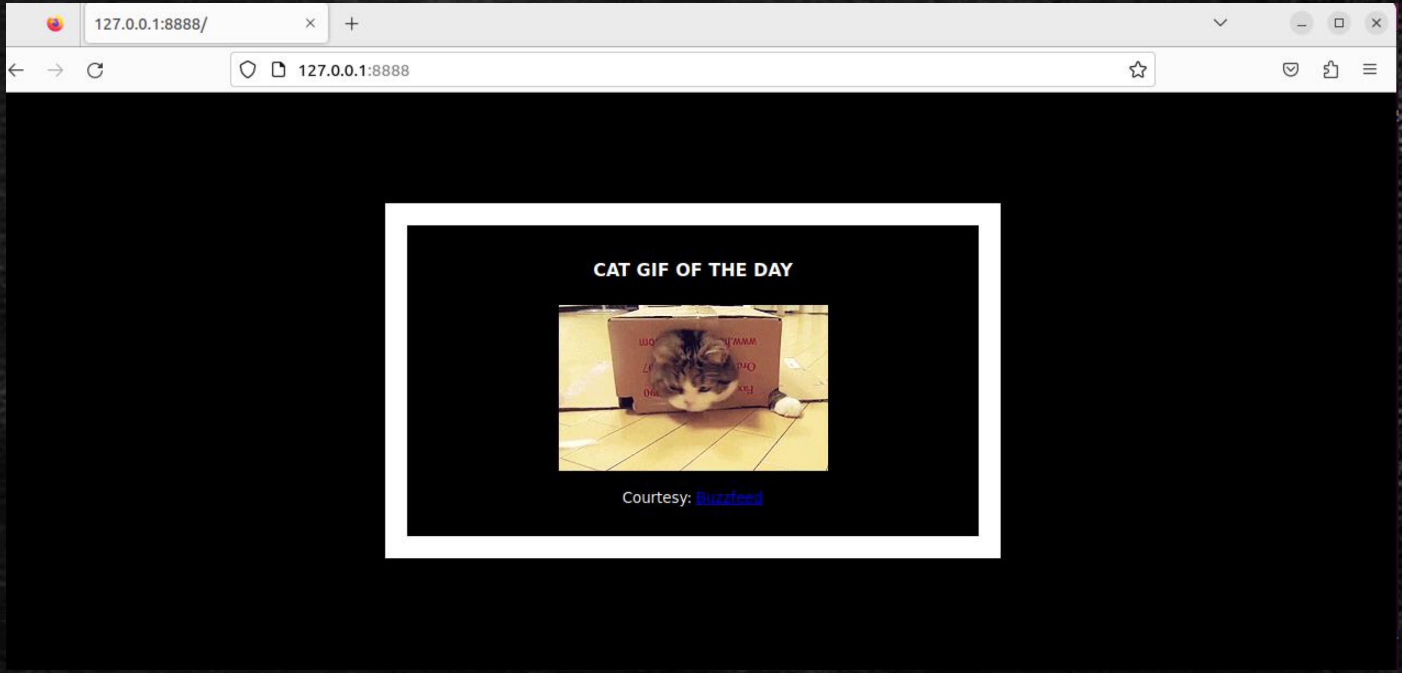
```arduino
docker build https://github.com/username/repository.git
```

```
sudo docker run -d -p  8888:5000 dockerfile/myapp
```



```
Error: No such container: dockerfile
bilalworker@bilalworker-virtual-machine:~/docker-curriculum/flask-app$ sudo docker port 90da198b9fa093baead9376552306cd43f816c75a36d1ed56fee9
f02251412a9
5000/tcp -> 0.0.0.0:8888
5000/tcp -> :::8888
bilalworker@bilalworker-virtual-machine:~/docker-curriculum/flask-app$ 
```

http://127.0.0.1:8888/



Congratulations! You have successfully created your first docker image.

Reference

- https://www.aquasec.com/cloud-native-academy/docker-container/100-best-docker-tutorials/
- https://docker-curriculum.com/#hello-world
- https://www.educba.com/docker-commands/
- https://docker-curriculum.com/#our-first-image