# Getting started with Docker

This lab is an introduction to Docker. We are going to see what Docker is, and how to manage and container images.

**What is a container?**

In the same way, the shipping industry uses physical containers to isolate different cargo to transport in trains and ships.  Software development technology uses a similar approach called containerization.  It is a standard package of software that bundles an application's code together with the related configuration file, and libraries with the dependencies required for the app to run.

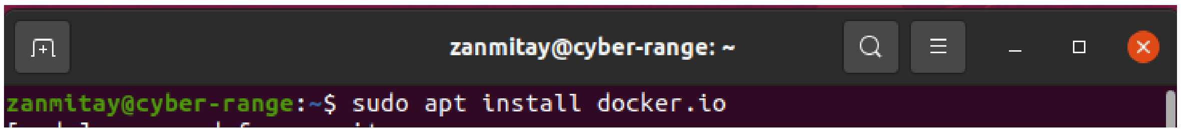
In other words, a container is a special type of process that is isolated from all other processes, operating system resources, and kernel. Containers are portable, multiplatform and they are widely used for software development.

**Container vs VMs**

VMs are an abstraction of physical hardware that includes a full copy of an operating system. Containers are abstractions at the app layer that packages code and its dependencies together.

We are going to install Docker on a Ubuntu desktop machine.  Assuming that your Ubuntu machine is up and running.

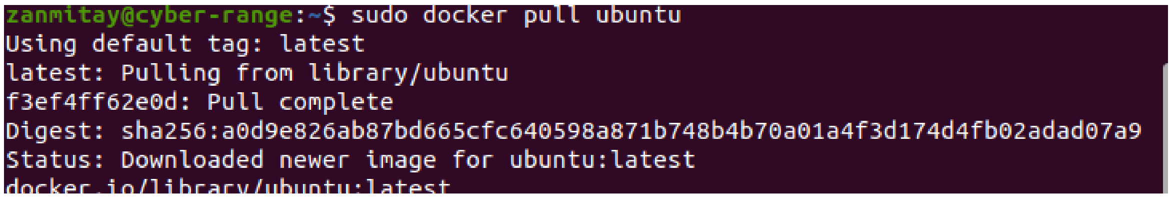
**1 installing Docker**



```
zanmitay@cyber-range:~$ sudo apt install docker.io
```

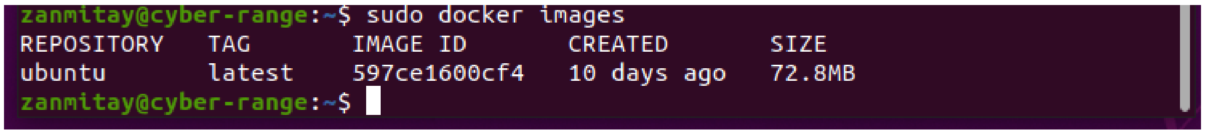
We can store and exchange containers in the Docker Registry. The most popular registry is the Docker Hub. Now let's check the hub to see if someone already has done that for us. We are going to test Docker Hub with a simple image of Ubuntu

Command: **sudo docker pull ubuntu**



```
zanmitay@cyber-range:~$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
f3ef4ff62e0d: Pull complete
Digest: sha256:a0d9e826ab87bd665cfc640598a871b748b4b70a01a4f3d174d4fb02adad07a9
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

Now let's list all Docker images on the machine.



```
zanmitay@cyber-range:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    597ce1600cf4   10 days ago    72.8MB
zanmitay@cyber-range:~$
```

Here are few terms to keep in mind when pulling Docker images

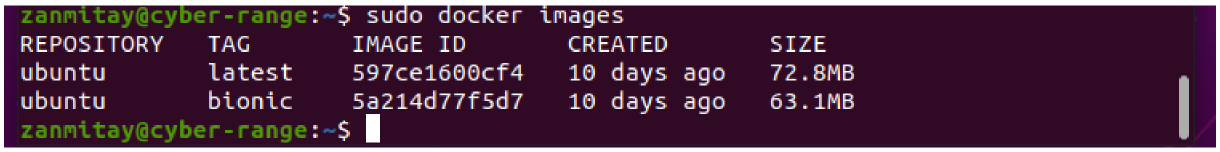
**Repository**: this is the name of the repository where you pulled the image from.  It is a unique and case sensitive

**Tag**: this version of the image pulled. If now tag is not specified Docker automatically uses the latest tag as default.

**Image ID**: this is locally generated and assigned image identification

Next, let pull the Ubuntu image with the bionic tag  and print the list of images again

- sudo docker pull Ubuntu:bionic
- Sudo docker images



```
zanmitay@cyber-range:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    597ce1600cf4   10 days ago    72.8MB
ubuntu        bionic    5a214d77f5d7   10 days ago    63.1MB
zanmitay@cyber-range:~$
```

**Docker image**: a set of layers combined by the Dockerfile or the pattern to run containers.

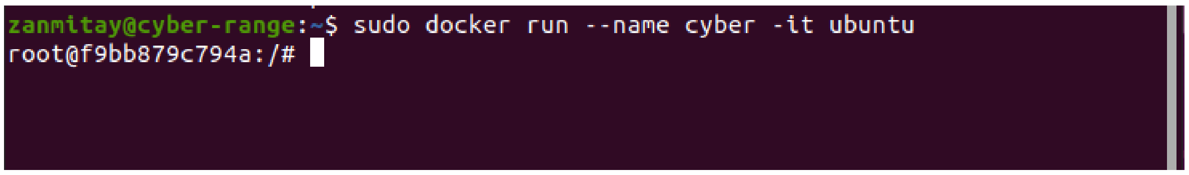
**Docker Container**: this is the run time instance of images.

**Container ID**: is a unique locally assigned name for identifying the Docker container.

**Names**: this is the human-readable format of the Docker container identification.

Now, Run the Ubuntu Docker Image with the name cyber to create a Docker Container.

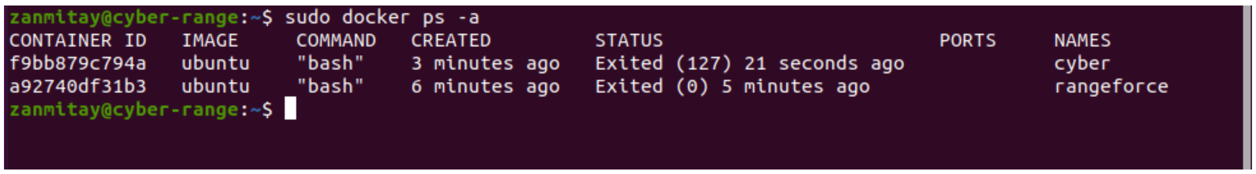
- **sudo docker run –name cyber –it Ubuntu**



```
zanmitay@cyber-range:~$ sudo docker run --name cyber -it ubuntu
root@f9bb879c794a:/# 
```

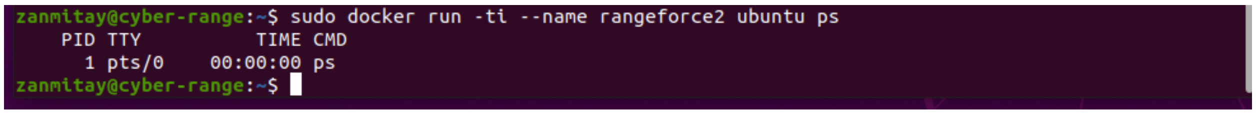
**List all the containers**

Command: **sudo docker ps –a**



```
zanmitay@cyber-range:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED         STATUS                        PORTS     NAMES
f9bb879c794a   ubuntu    "bash"    3 minutes ago   Exited (127) 21 seconds ago             cyber
a92740df31b3   ubuntu    "bash"    6 minutes ago   Exited (0) 5 minutes ago                rangeforce
zanmitay@cyber-range:~$ 
```

How to find the docker container process ID.



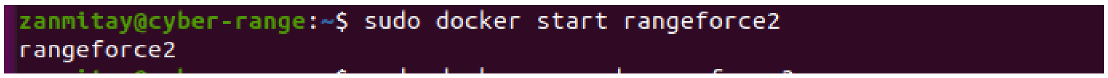
```
zanmitay@cyber-range:~$ sudo docker run -ti --name rangeforce2 ubuntu ps
    PID TTY          TIME CMD
      1 pts/0    00:00:00 ps
zanmitay@cyber-range:~$ 
```

Process ID = 1

As you can notice every time we run the container.  it stops when we exit the shell. Now let's see how we can make the container run in the background.

Option 1: starting previously stopped container (rangeforce2)

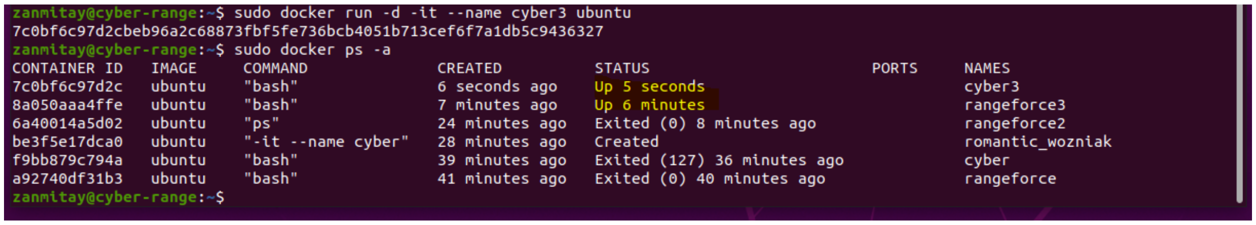
Command: sudo docker start rangeforce2



```
zanmitay@cyber-range:~$ sudo docker start rangeforce2
rangeforce2
```

Option 2: starting a detached container name **cyber3**

Command: **sudo docker run –d –it --name cyber3 ubuntu**



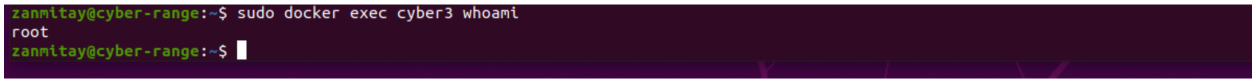
```
zanmitay@cyber-range:~$ sudo docker run -d -it --name cyber3 ubuntu
7c0bf6c97d2cbeb96a2c68873fbf5fe736bcb4051b713cef6f7a1db5c9436327
zanmitay@cyber-range:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND             CREATED          STATUS                        PORTS     NAMES
7c0bf6c97d2c   ubuntu    "bash"              6 seconds ago    Up 5 seconds                            cyber3
8a050aaa4ffe   ubuntu    "bash"              7 minutes ago    Up 6 minutes                            rangeforce3
6a40014a5d02   ubuntu    "ps"                24 minutes ago   Exited (0) 8 minutes ago                rangeforce2
be3f5e17dca0   ubuntu    "-it --name cyber"  28 minutes ago   Created                                 romantic_wozniak
f9bb879c794a   ubuntu    "bash"              39 minutes ago   Exited (127) 36 minutes ago             cyber
a92740df31b3   ubuntu    "bash"              41 minutes ago   Exited (0) 40 minutes ago               rangeforce
zanmitay@cyber-range:~$
```

## Command execution

We have learned how to create containers and how to run them in the background. How can we execute commands in a running container?

**exec** command is used for command execution in running containers. Let's check who is the running user in the cyber3 container.

Command: sudo docker exec cyber3 whoami



```
zanmitay@cyber-range:~$ sudo docker exec cyber3 whoami
root
zanmitay@cyber-range:~$
```