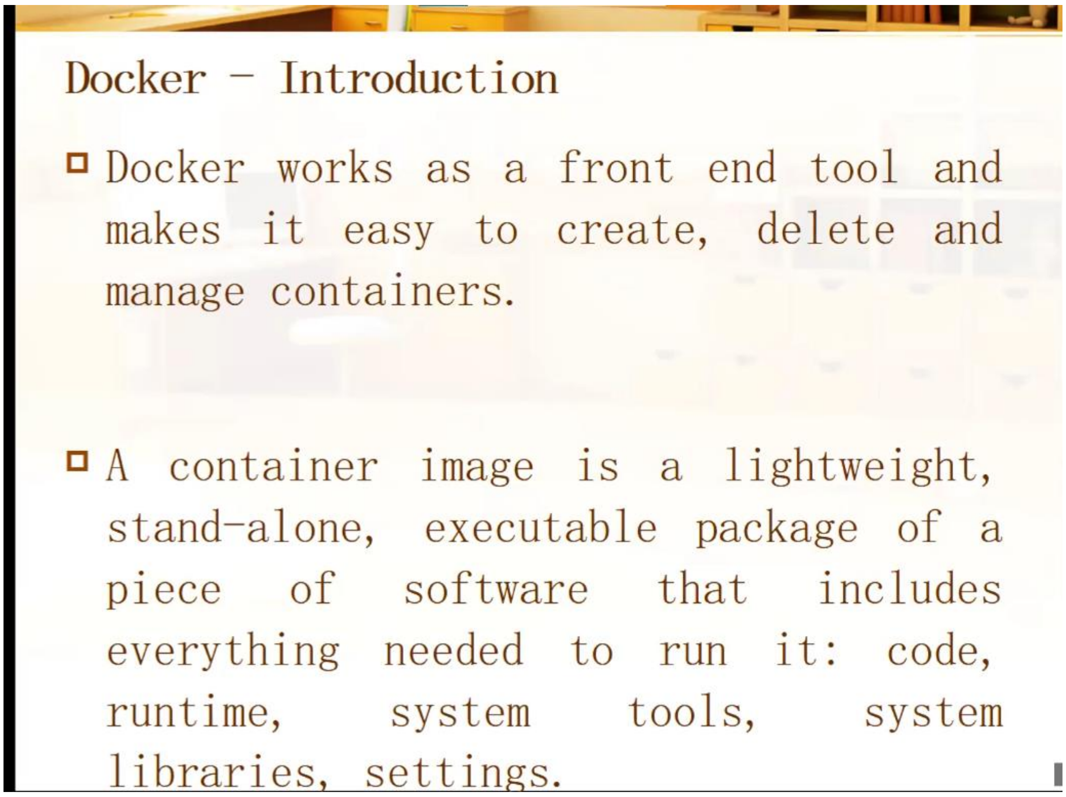
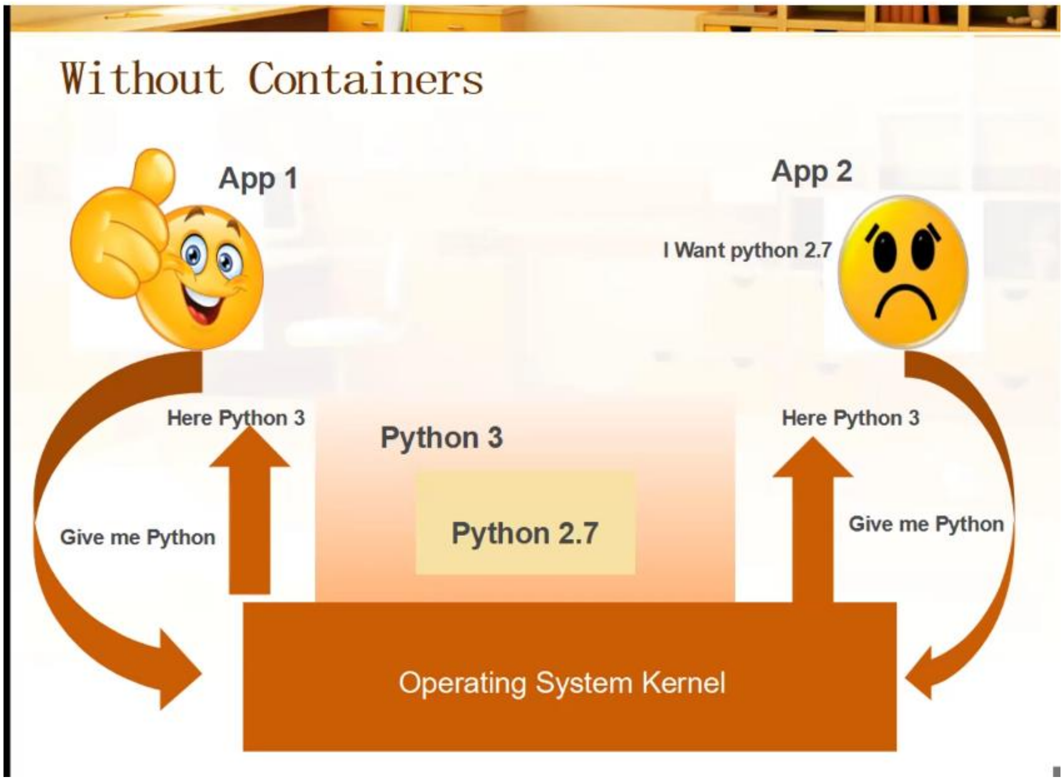
-------------------------------------Docker---------------------------------------



## Docker - Introduction

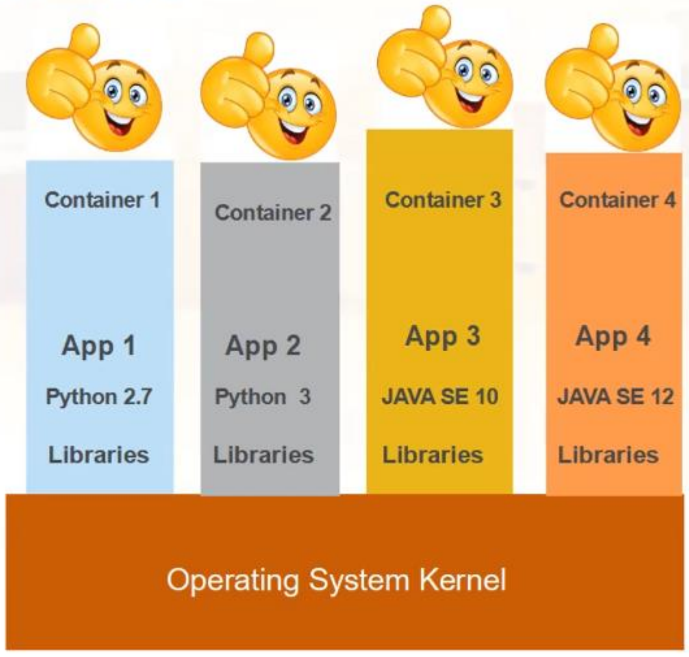
- Docker works as a front end tool and makes it easy to create, delete and manage containers.

- A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.



## Without Containers

## With Containers

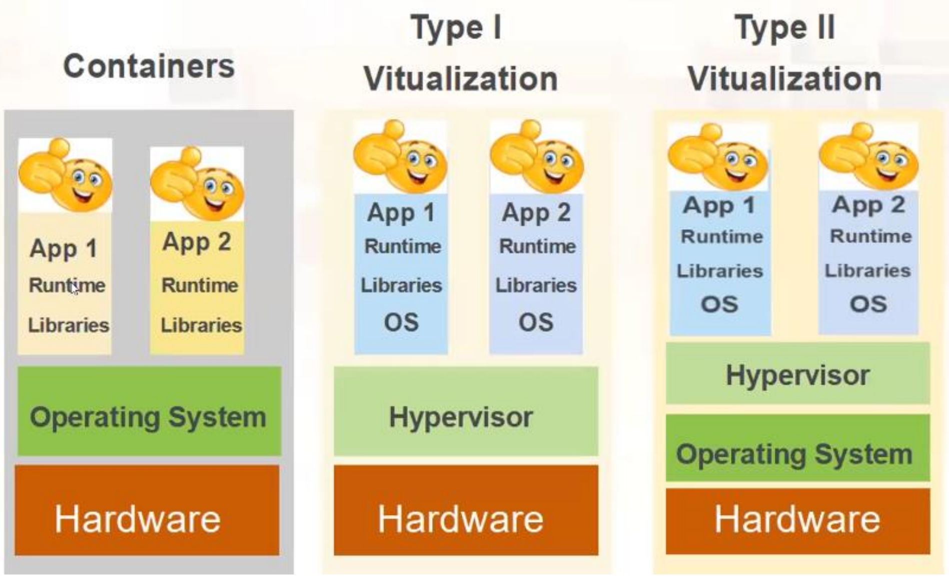


## Docker – Introduction

- Unlike VMs, containers do not bundle a full operating system – only libraries and settings required to make the software work are needed.

- This makes for efficient, lightweight, self-contained systems and guarantees that software will always run the same, regardless of where it's deployed.

## Containers Vs Virtual Mchines



## Docker – Introduction

- They start instantly and use less compute and RAM.

## Docker – Introduction

- A container is defined by its image as well as any configuration options you provide to it when

## Docker Editions

Docker is available in 2 editions

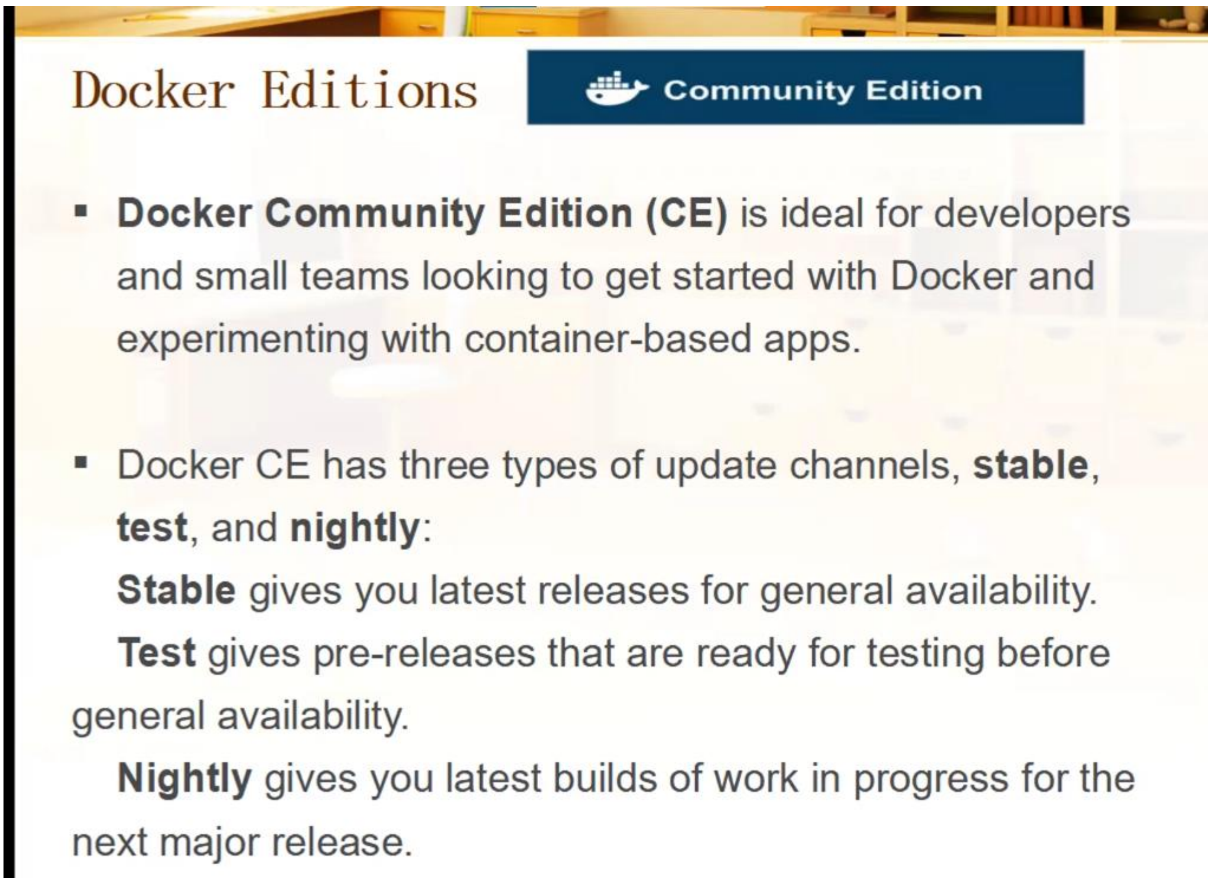
Enterprise Edition

Community Edition

its state
storage

in the
ined even

verything
container


## Docker Editions

Community Edition

- **Docker Community Edition (CE)** is ideal for developers and small teams looking to get started with Docker and experimenting with container-based apps.
- Docker CE has three types of update channels, **stable**, **test**, and **nightly**:
  **Stable** gives you latest releases for general availability.
  **Test** gives pre-releases that are ready for testing before general availability.
  **Nightly** gives you latest builds of work in progress for the next major release.

------------------**Docker Installation**------------------------------------------------------

Run the following command to uninstall all conflicting packages:

**for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done**

**# Add Docker's official GPG key:**

**sudo apt-get update**

**sudo apt-get install ca-certificates curl gnupg**

**sudo install -m 0755 -d /etc/apt/keyrings**

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg**

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg


# Add the repository to Apt sources:
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

To install the latest version, run:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

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

**To check Images present in local repo:-**

```
$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    9c7a54a9a43c   6 months ago   13.3kB
```

**To check Running Containers:-**

```
$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

**To download images from Docker hub:**

```
$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
aece8493d397: Pull complete
Digest: sha256:2b7412e6465c3c7fc5bb21d3e6f1917c167358449fecac8176c6e496e5c1f05f
```

Status: Downloaded newer image for ubuntu:latest

docker.io/library/ubuntu:latest

binay@binay:~$ sudo docker images

```
REPOSITORY   TAG      IMAGE ID       CREATED        SIZE
ubuntu       latest   e4c58958181a   5 weeks ago    77.8MB
hello-world  latest   9c7a54a9a43c   6 months ago   13.3kB
```

To create New Container we use, run command.

$ sudo docker run -ti Ubuntu -----------(-t Allocate a pseudo-TTY, i –interactive and Ubuntu is image name)

root@3b9a0b494599:/# pwd

On new Terminal:

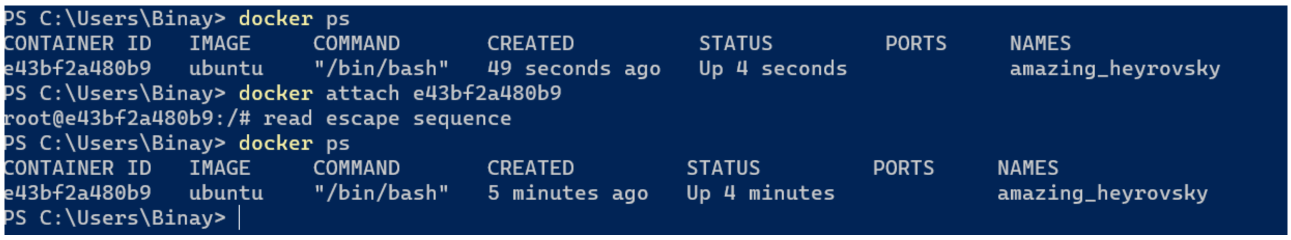
~$ sudo docker ps

```
CONTAINER ID   IMAGE    COMMAND       CREATED         STATUS         PORTS     NAMES
3b9a0b494599   ubuntu   "/bin/bash"   3 minutes ago   Up 3 minutes             strange_edison
```

root@3b9a0b494599:/# exit

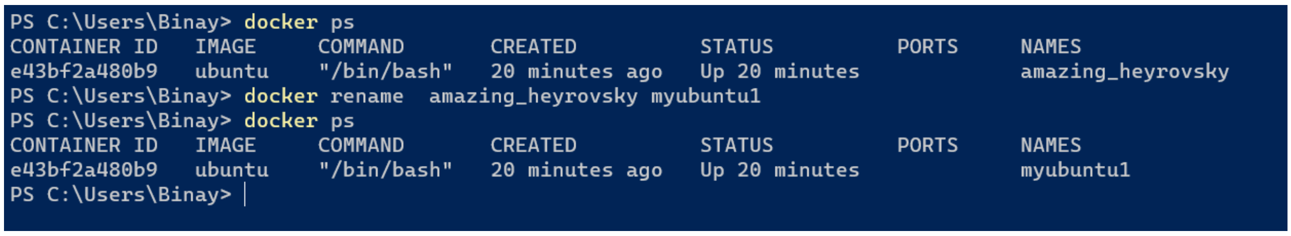
exit -----------(Container Stopped…….)

If you exit from above container, the container will stop automatic. So to avoid this(To Detach from container) press Ctrl+P followed by Ctrl+Q



```
PS C:\Users\Binay> docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS         PORTS     NAMES
e43bf2a480b9   ubuntu    "/bin/bash"   49 seconds ago   Up 4 seconds             amazing_heyrovsky
PS C:\Users\Binay> docker attach e43bf2a480b9
root@e43bf2a480b9:/# read escape sequence
PS C:\Users\Binay> docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS         PORTS     NAMES
e43bf2a480b9   ubuntu    "/bin/bash"   5 minutes ago   Up 4 minutes             amazing_heyrovsky
PS C:\Users\Binay>
```

To Rename a container:-



```
PS C:\Users\Binay> docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
e43bf2a480b9   ubuntu    "/bin/bash"   20 minutes ago   Up 20 minutes             amazing_heyrovsky
PS C:\Users\Binay> docker rename  amazing_heyrovsky myubuntu1
PS C:\Users\Binay> docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
e43bf2a480b9   ubuntu    "/bin/bash"   20 minutes ago   Up 20 minutes             myubuntu1
PS C:\Users\Binay>
```

```
$ sudo docker ps

CONTAINER ID   IMAGE    COMMAND   CREATED   STATUS    PORTS     NAMES


$ sudo docker ps -a

CONTAINER ID   IMAGE       COMMAND      CREATED       STATUS                    PORTS     NAMES

3b9a0b494599   ubuntu      "/bin/bash"   7 minutes ago   Exited (0) About a minute ago
strange_edison

9a30b4e65ce0   hello-world   "/hello"     4 days ago     Exited (0) 4 days ago                 epic_ishizaka

20fa8b400ef9   hello-world   "/hello"     4 days ago     Exited (0) 4 days ago                 funny_ramanujan
```

Now we create centos container without interactive mode.

So when we direct use run command it will look into your local repo for images but if image is not present it will download image.

```
$ sudo docker run centos

Unable to find image 'centos:latest' locally

latest: Pulling from library/centos

a1d0c7532777: Pull complete

Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177

Status: Downloaded newer image for centos:latest


$ sudo docker ps

CONTAINER ID   IMAGE    COMMAND   CREATED   STATUS    PORTS     NAMES

binay@binay:~$ sudo docker ps -a

CONTAINER ID   IMAGE       COMMAND      CREATED        STATUS                 PORTS     NAMES

43c36af3c8c7   centos      "/bin/bash"   34 seconds ago   Exited (0) 32 seconds ago          elated_nash

3b9a0b494599   ubuntu      "/bin/bash"   9 minutes ago    Exited (0) 3 minutes ago           strange_edison
```

Container is just executing. It is just a environment. It is the image which tells container to what executes. This images are configured to execute bash shell.

**To Delete a Container:- docker rm <container id>**

```
$ sudo docker ps -a
CONTAINER ID   IMAGE       COMMAND      CREATED      STATUS              PORTS     NAMES
43c36af3c8c7   centos      "/bin/bash"   2 hours ago   Exited (0) 2 hours ago           elated_nash
3b9a0b494599   ubuntu      "/bin/bash"   3 hours ago   Exited (0) 2 hours ago           strange_edison
9a30b4e65ce0   hello-world   "/hello"     4 days ago    Exited (0) 4 days ago            epic_ishizaka
20fa8b400ef9   hello-world   "/hello"     4 days ago    Exited (0) 4 days ago            funny_ramanujan
binay@binay:~$ sudo docker rm 43c36af3c8c7
43c36af3c8c7
binay@binay:~$ sudo docker ps -a
CONTAINER ID   IMAGE       COMMAND      CREATED      STATUS              PORTS     NAMES
3b9a0b494599   ubuntu      "/bin/bash"   3 hours ago   Exited (0) 2 hours ago           strange_edison
9a30b4e65ce0   hello-world   "/hello"     4 days ago    Exited (0) 4 days ago            epic_ishizaka
20fa8b400ef9   hello-world   "/hello"     4 days ago    Exited (0) 4 days ago            funny_ramanujan
```

**Images are read only.**

**To delete a Image, if any container is not using those image then we can delete those image.**

**docker rmi <image id>**

```
$ sudo docker images
REPOSITORY    TAG      IMAGE ID       CREATED       SIZE
ubuntu        latest    e4c58958181a   5 weeks ago    77.8MB
hello-world   latest    9c7a54a9a43c   6 months ago   13.3kB
centos        latest    5d0da3dc9764   2 years ago    231MB
```

```
binay@binay:~$ sudo docker rmi hello-world
```

Error response from daemon: conflict**: unable to remove repository reference "hello-world" (must** force) - container 20fa8b400ef9 is using its referenced image 9c7a54a9a43c

So first we have to delete those containers which are using hello-world image.

```
binay@binay:~$ sudo docker ps -a

CONTAINER ID   IMAGE        COMMAND      CREATED      STATUS                PORTS     NAMES

3b9a0b494599   ubuntu       "/bin/bash"   3 hours ago   Exited (0) 3 hours ago           strange_edison

9a30b4e65ce0   hello-world   "/hello"      4 days ago    Exited (0) 4 days ago            epic_ishizaka

20fa8b400ef9   hello-world   "/hello"      4 days ago    Exited (0) 4 days ago            funny_ramanujan

binay@binay:~$ sudo docker rm 9a30b4e65ce0

9a30b4e65ce0

binay@binay:~$ sudo docker rm 20fa8b400ef9

20fa8b400ef9

binay@binay:~$ sudo docker ps -a

CONTAINER ID   IMAGE     COMMAND      CREATED      STATUS                PORTS     NAMES

3b9a0b494599   ubuntu    "/bin/bash"   3 hours ago   Exited (0) 3 hours ago           strange_edison
```

**Now** We **will delete image**

```
binay@binay:~$ sudo docker rmi hello-world

Untagged: hello-world:latest

Untagged: hello-
world@sha256:88ec0acaa3ec199d3b7eaf73588f4518c25f9d34f58ce9a0df68429c5af48e8d

Deleted: sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f9e016fdc3384500c2823d

Deleted: sha256:01bb4fce3eb1b56b05adf99504dafd31907a5aadac736e36b27595c8b92f07f1
```

**To Start and End a container:-**

```
# docker ps -a

CONTAINER ID   IMAGE     COMMAND      CREATED      STATUS                PORTS     NAMES

3b9a0b494599   ubuntu    "/bin/bash"   3 hours ago   Exited (0) 3 hours ago           strange_edison

root@binay:~# docker start 3b9a0b494599

3b9a0b494599

root@binay:~# docker ps

CONTAINER ID   IMAGE     COMMAND      CREATED      STATUS       PORTS     NAMES

3b9a0b494599   ubuntu    "/bin/bash"   3 hours ago   Up 3 seconds           strange_edison
```

```
root@binay:~# docker stop 3b9a0b494599
3b9a0b494599
root@binay:~# docker ps
CONTAINER ID   IMAGE    COMMAND   CREATED   STATUS    PORTS     NAMES
root@binay:~#
```

**To get shell of running Container-**

```
#docker ps
CONTAINER ID   IMAGE     COMMAND      CREATED      STATUS        PORTS     NAMES
3b9a0b494599   ubuntu    "/bin/bash"   3 hours ago   Up 2 seconds            strange_edison
root@binay:~# docker attach 3b9a0b494599
root@3b9a0b494599:/#
```

**But when we exit from shell, the container will be stop.  (Default Behavior- Remember Very careful)**

```
/# exit
exit
root@binay:~# docker ps
CONTAINER ID   IMAGE    COMMAND   CREATED   STATUS    PORTS     NAMES
root@binay:~#
```

**To run command inside a container without Getting Shell:-**

**# docker  exec  <conainer id or name>  <command>**

```
#docker ps
CONTAINER ID   IMAGE     COMMAND      CREATED      STATUS        PORTS     NAMES
3b9a0b494599   ubuntu    "/bin/bash"   4 hours ago   Up 4 seconds            strange_edison
root@binay:~# docker exec 3b9a0b494599 ls
bin
boot
dev
```

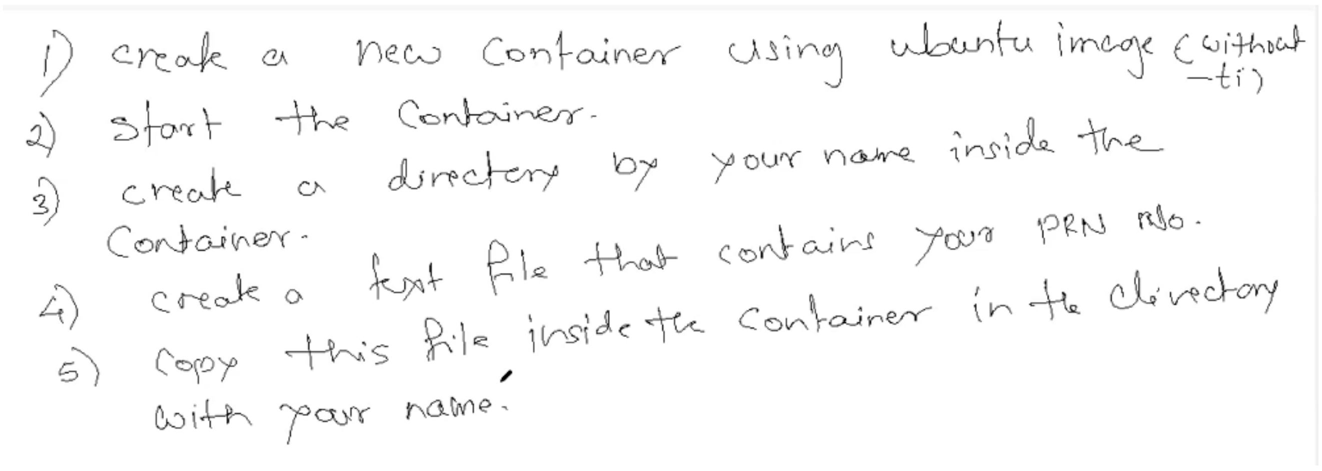
**To copy a file from local machine to Container:-**

```
vi one.txt

root@binay:~# cat one.txt

this file is inside containe.

root@binay:~# docker cp one.txt 3b9a0b494599:/test

Successfully copied 2.05kB to 3b9a0b494599:/test  (Here one.txt copied inside /test

root@binay:~# docker cp one.txt 3b9a0b494599:/data (here /data was not available so it created file
named data but it will conatin data of your local file one.txt

Successfully copied 2.05kB to 3b9a0b494599:/data

 oot@binay:~# docker exec 3b9a0b494599 cat /data

this file is inside containe.
```

**Container to local machine:-**

```
# docker cp myapp1:/container.txt .

Successfully copied 2.05kB to /root/.
```

**Assignment:-**



1) create a new Container using ubuntu image (without -ti)
2) start the Container.
3) create a directory by your name inside the Container.
4) create a text file that contains your PRN No.
5) copy this file inside the Container in the directory with your name.

1. **Creating new Container named myapp1 using Ubuntu image(without ti option):-**

```
docker run --name myapp1 ubuntu
root@binay:~# docker ps -a
CONTAINER ID   IMAGE    COMMAND       CREATED          STATUS                     PORTS    NAMES
802453ebefee   ubuntu   "/bin/bash"   3 seconds ago    Exited (0) 2 seconds ago            myapp1
```

2. **Starting container:-**

```
~# docker start myapp1

myapp1
```

```
root@binay:~# docker ps -a

CONTAINER ID   IMAGE    COMMAND     CREATED       STATUS                PORTS     NAMES

802453ebefee   ubuntu   "/bin/bash"   2 minutes ago   Exited (0) 1 second ago           myapp1
```

In first step, we created container without interactive option. Ubuntu or centos run bash command which requires interactive shell. That's why, when we start above container got exited. So delete above container and create new container with interactive option. Other container do not required interactive session.

```
~# docker run -ti --name myapp1 ubuntu

root@05b63540f2fb:/# exit

exit

root@binay:~# docker ps

CONTAINER ID   IMAGE    COMMAND   CREATED   STATUS    PORTS    NAMES

root@binay:~# docker ps -a

CONTAINER ID   IMAGE    COMMAND     CREATED       STATUS                PORTS     NAMES

05b63540f2fb   ubuntu   "/bin/bash"   32 seconds ago   Exited (0) 6 seconds ago          myapp1

root@binay:~#
```

**Starting the container:-**

```
docker start myapp1

myapp1

root@binay:~# docker ps

CONTAINER ID   IMAGE    COMMAND     CREATED          STATUS       PORTS    NAMES

05b63540f2fb   ubuntu   "/bin/bash"   About a minute ago   Up 6 seconds          myapp1

root@binay:~#
```

3. **Creating a directory inside a conatainer:-**

```
# docker exec myapp1 mkdir /binay
root@binay:~# docker exec myapp1 ls
bin
binay
```

4. **Creating a local file and copying to container:-**

```
# docker cp one.txt myapp1:/binay
Successfully copied 2.05kB to myapp1:/binay
root@binay:~# docker exec myapp1 cat /binay/one.txt
```

this file is inside container.

**HTTPD:**

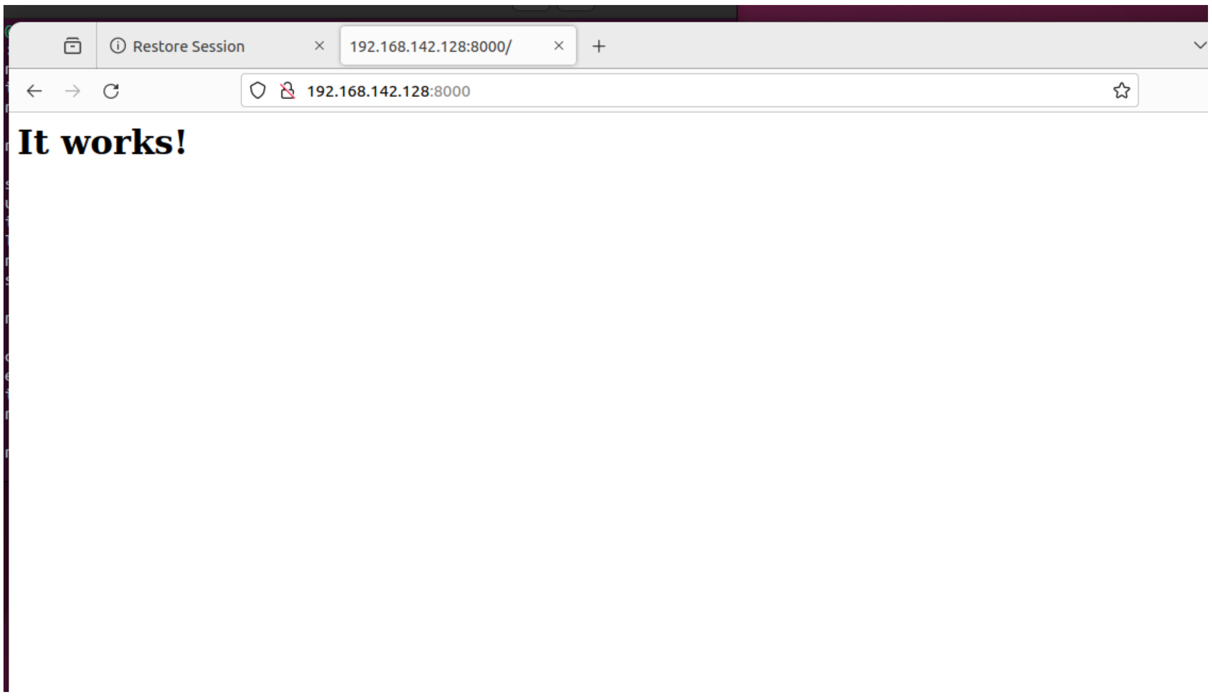
```
# docker pull httpd
Using default tag: latest
```

**When we host website inside container,  every container will have ip. So on those ip and port no 80 will be mapped but we can not access content through container ip. So we will map container port to any port of your machine. Ex.in container , webser is running on port no 80 so we can map on machine to any port like 8000 or 8080 (Own choice.)**

**Below 8000 is external port and 80 is conatiner's port.**

```
# docker run --name web1 -p 8000:80 httpd

AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message

AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message

[Fri Nov 17 17:30:34.541832 2023] [mpm_event:notice] [pid 1:tid 139622624925568] AH00489: Apache/2.4.58 (Unix) configured -- resuming normal operations

[Fri Nov 17 17:30:34.541987 2023] [core:notice] [pid 1:tid 139622624925568] AH00094: Command line: 'httpd -D FOREGROUND'
```

```
-- resuming normal operations

[Fri Nov 17 17:30:34.541987 2023] [core:notice] [pid 1:tid 139622624925568] AH00094: Command line: 'httpd -D FOREGROUND'

192.168.142.128 - - [17/Nov/2023:17:31:30 +0000] "GET / HTTP/1.1" 200 45

192.168.142.128 - - [17/Nov/2023:17:31:30 +0000] "GET /favicon.ico HTTP/1.1" 404 196
```



It works!

Location: /usr/local/apache2/htdocs -→ here apache looks for web pages.

**Press control + c to exit from container. Then container will stop. So we have to start manually container.**

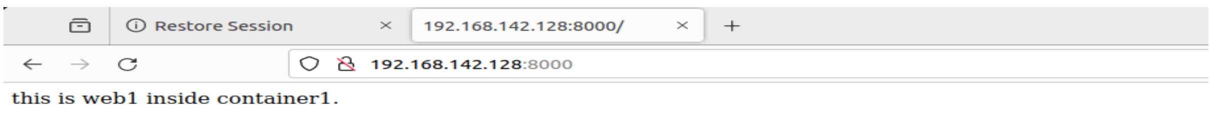
```
# docker ps -a
CONTAINER ID   IMAGE    COMMAND             CREATED          STATUS                 PORTS
NAMES
b025ae2d20b2   httpd    "httpd-foreground"  20 minutes ago   Up 3 seconds           0.0.0.0:8000->80/tcp,
:::8000->80/tcp   web1
05b63540f2fb   ubuntu   "/bin/bash"         23 hours ago     Exited (0) 5 hours ago
myapp1
```

```
# cat index.html
this is web1 inside container1.
```

```
# docker cp index.html web1:/usr/local/apache2/htdocs
```

Successfully copied 2.05kB to web1:/usr/local/apache2/htdocs

**So In**



this is web1 inside container1.

**above process, whenever we create container, we have to create index.html file and copy to /usr/local/apache2/htdocs. And if we want to modify contents then again we have to copy index.html to ALL containers. So to avoid this, we will mount /usr/local/apache2/htdocs to our own directory. So we will modify index.html which is inside our own directory, it will reflect to all containers.**

```
# mkdir /webdata

root@binay:~# cd /webdata/

root@binay:/webdata# vi index.html
```

```
root@binay:/webdata# docker run --name web2 -d -p 9000:80 -v /webdata/:/usr/local/apache2/htdocs/ httpd

befff73b82b06d65489365f508f47d177605057b5554b0d9ac7f419649815c23
```

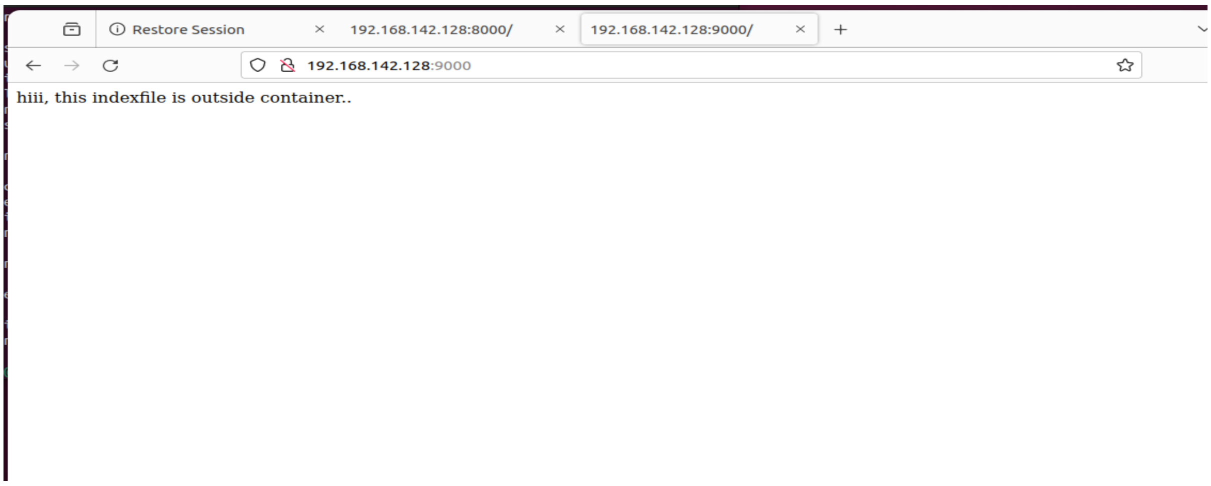
In above command:-

-d: it will run container in background not in forground

-p: port

-v : Volume (map /webdata directory to /usr/……./htdocs

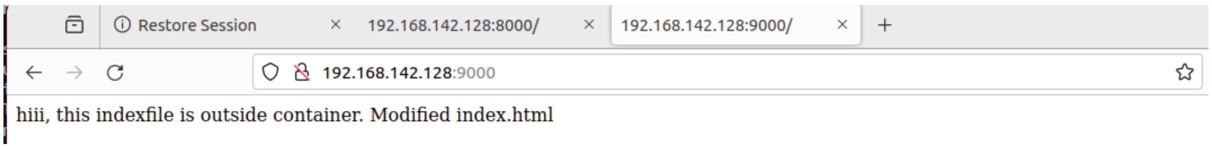
Httpd:- image name



hiii, this indexfile is outside container..

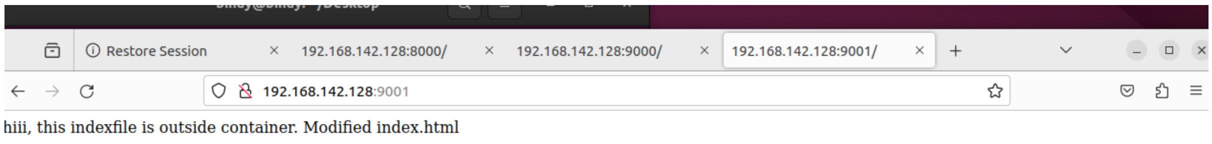
Now I will modify index.html

```
root@binay:/webdata# vi index.html
root@binay:/webdata#
```



hiii, this indexfile is outside container. Modified index.html

```
:/webdata# docker run --name web3 -d -p 9001:80 -v /webdata/:/usr/local/apache2/htdocs/ httpd
4f1162627452eac14297e575f156b166bd5a10ba580f4827892aed563fc831fd
```



hiii, this indexfile is outside container. Modified index.html

So similarly we can create multiple containers who will access machine’s index.html file.

**Script:**

```
#!/bin/bash
#To Create Multiple Httpd Container which uses one index.html file.
read -p "Please Enter No of containers you want to create:- " n
```

```
for (( i=1; i<=$n; i++ ))

do

docker run --name web$i -d -p 800$i:80 -v /webdata:/usr/local/apache2/htdocs/ httpd &> /dev/null

echo "Web$i Created.."

done

echo "No of Containers are running:-"

docker ps

~

bash multi_container.sh

Please Enter No of containers you want to create:- 5

Web1 Created..

Web2 Created..

Web3 Created..

Web4 Created..

Web5 Created..

No of Containers are running:-

CONTAINER ID   IMAGE     COMMAND              CREATED                  STATUS                  PORTS
NAMES

85f883fe03fc   httpd     "httpd-foreground"   Less than a second ago   Up Less than a second   0.0.0.0:8005->80/tcp, :::8005->80/tcp   web5

a953d59f0954   httpd     "httpd-foreground"   1 second ago             Up Less than a second   0.0.0.0:8004->80/tcp, :::8004->80/tcp   web4

461755843c67   httpd     "httpd-foreground"   1 second ago             Up Less than a second   0.0.0.0:8003->80/tcp, :::8003->80/tcp   web3

485c7c9a4bfd   httpd     "httpd-foreground"   2 seconds ago            Up 1 second             0.0.0.0:8002->80/tcp, :::8002->80/tcp   web2

eccfa65f1d90   httpd     "httpd-foreground"   2 seconds ago            Up 2 seconds            0.0.0.0:8001->80/tcp, :::8001->80/tcp   web1

root@binay:/webdata#
```

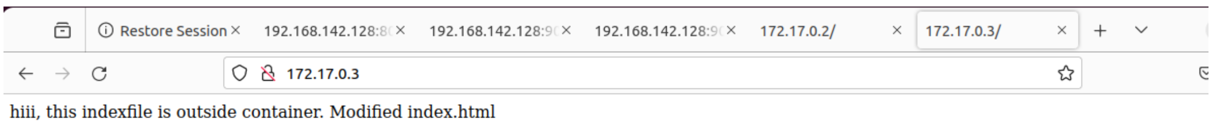
**To Delete all running container one by one:-**

```
for i in `docker ps | awk '{print $12}'`

> do
> docker stop $i

> docker rm $i

> done

web5

web5

web4

web4

web3

web3

web2

web2

web1

web1
```

**Docker inspect is a tool that enables you do get detailed information about your docker resources, such as containers, images, volumes, networks, tasks and services.**

```
# docker inspect web1

"SandboxKey": "/var/run/docker/netns/b8eb46afb3cf",

        "SecondaryIPAddresses": null,

        "SecondaryIPv6Addresses": null,

        "EndpointID": "cb03b0eb8d70efd5e37ceff5f25975d2d220bc7cf02b28837e379be0be60c677",

        "Gateway": "172.17.0.1",

        "GlobalIPv6Address": "",

        "GlobalIPv6PrefixLen": 0,

        "IPAddress": "172.17.0.2",  (ip add of container)

        "IPPrefixLen": 16,

        "IPv6Gateway": "",
            "MacAddress": "02:42:ac:11:00:02",
            "Networks": {
                "bridge": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "NetworkID": "22746b7ac00bd521623ea99006ca1f8fddea0952fce172246bd6a6cdc7c92a54",
                    "EndpointID": "cb03b0eb8d70efd5e37ceff5f25975d2d220bc7cf02b28837e379be0be60c677",
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:11:00:02",
                    "DriverOpts": null
                }
            }
        }
    }
]
```

Inside our linux machine, we can access container's web page using container's ip



hiii, this indexfile is outside container. Modified index.html

## Creating our Own image:-

To create our own image, we have to write a Dockerfile.

```
# vi Dockerfile
```

```
root@binay:/webdata# cat Dockerfile

From ubuntu

CMD [ "/bin/ls" ]


root@binay:/webdata# docker build -t myimage:1.0 .

[+] Building 1.3s (5/5) FINISHED
docker:default

 => [internal] load build definition from Dockerfile
0.5s

 => => transferring dockerfile: 67B
0.3s

 => [internal] load .dockerignore
0.5s

 => => transferring context: 2B
0.2s

 => [internal] load metadata for docker.io/library/ubuntu:latest
0.0s

 => [1/1] FROM docker.io/library/ubuntu
0.1s

 => exporting to image
0.1s

 => => exporting layers
0.0s

 => => writing image
sha256:96f0431bbafbabb500b3ee210da3c1aaa9cb96188c043643f96e6e789b167e2
4                                                  0.0s

 => => naming to docker.io/library/myimage:1.0
0.0s

root@binay:/webdata# docker images

REPOSITORY   TAG      IMAGE ID      CREATED      SIZE
```

```
 myimage      1.0       96f0431bbafb   6 weeks ago   77.8MB
```

Second Image:

```
# vi Dockerfile

# cat Dockerfile

From ubuntu

RUN mkdir /sc

COPY sc1.sh /sc

CMD [ "/sc/sc1.sh" ]

root@binay:/webdata# docker build -t script1 .

[+] Building 1.5s (8/8) FINISHED
docker:default

 => [internal] load build definition from Dockerfile
0.0s

 => => transferring dockerfile: 101B
0.0s

 => [internal] load .dockerignore
0.0s

 => => transferring context: 2B
0.0s

 => [internal] load metadata for docker.io/library/ubuntu:latest
0.0s

 => CACHED [1/3] FROM docker.io/library/ubuntu
0.0s

 => [internal] load build context
0.0s

 => => transferring context: 101B
0.0s
```

```
 => [2/3] RUN mkdir /sc
1.0s

 => [3/3] COPY sc1.sh /sc
0.1s

 => exporting to image
0.2s

 => => exporting layers
0.2s

 => => writing image
sha256:1497ffa72d786df84212048c94923d96a77acd48be6b9a9fee38d4ca7764c55
1                                                        0.0s

 => => naming to docker.io/library/script1
0.0s

root@binay:/webdata# docker ps

CONTAINER ID   IMAGE    COMMAND   CREATED   STATUS    PORTS     NAMES

root@binay:/webdata# docker images

REPOSITORY   TAG      IMAGE ID      CREATED         SIZE

script1     latest    1497ffa72d78   15 seconds ago   77.8MB
```

**# docker run --name myscript script1**

Hello .

But it’s not interactive . So we have to use –i option to create container.

Assignment:

-Write a python program that 2 numbers from user and display the sum.

-Create an image for above program and test image by running it.

Step1: **Creating Dockerfile:-**

```
#cat Dockerfile
FROM ubuntu
RUN mkdir /py
#WORKDIR /py
COPY sum.py /py
RUN apt update
RUN apt install python3 -y
CMD [ "/py/sum.py" ]
```

Step 2**: Docker Build:-**

```
# docker build -t pyscript1  .
```

```
# docker images
REPOSITORY   TAG       IMAGE ID      CREATED       SIZE
app1        latest    ab261c699135   18 hours ago   154MB
app3        latest    10f5af5427bc   19 hours ago   154MB
pyscript1    latest    10f5af5427bc   19 hours ago   154MB
```

Step3: **Creating Container:-**

```
# docker run -i --name conatiner1 pyscript1
Enter a number: 23
Enter a number: 12
Sum is 35
```

-------------------------------------------------------------------------------------------

Till now, we were creating images using Dockerfile. Now we will create images

Using container.

**Now Create a container using Ubuntu image:-**

```
# docker run --name ub1 -ti ubuntu
root@79fe71f8b62b:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
root@79fe71f8b62b:/# mkdir /script
# apt update
# apt install python3
```

Note: Now we got a shell of a container. But we cannot directly copy our script which is outside of container to container. So we will open new terminal and will copy to container.

**On new Terminal:-**

```
cd /webdata/
binay@binay:/webdata$ docker cp sum.py ub1:/script
```

**Copied to Container………**

```
root@79fe71f8b62b:/# ls /script/
sum.py
```

Now Exit From container and we have to save the container as image.

Data is inside the container, if we delete the container, data will be lost.

So to create an image form container we use **docker commit**

```
# docker commit
"docker commit" requires at least 1 and at most 2 arguments.
See 'docker commit --help'.
```

```
Usage:  docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
```

**Create a new image from a container's changes**

**Below ub1 is container name**

```
# docker commit ub1 pyscript2:01
sha256:dc98210801bff1a067c5ddd6936116961f34f92b6892c58eb11956d31d40725
7
root@binay:/webdata# docker images
REPOSITORY   TAG      IMAGE ID      CREATED        SIZE
pyscript2    01       dc98210801bf   9 seconds ago   154MB
```

Now we can remove container ub1

And we will create containers form image which we created.

```
# docker run pyscript2:01
root@binay:/webdata# docker ps
CONTAINER ID   IMAGE    COMMAND   CREATED   STATUS    PORTS     NAMES
root@binay:/webdata# docker ps -a
CONTAINER ID   IMAGE        COMMAND            CREATED          STATUS
PORTS     NAMES
ddd67d29fa69   pyscript2:01   "/bin/bash"         7 seconds ago      Exited (0) 6
seconds ago              wonderful_snyder
```

In docker ps command’s output, there is no running container running pyscript2 because there was no way to specify a command what to execute.

So by default Ubuntu executing /bin/bash.

So to run my container, I have to provide manually command like this:

```
root@binay:/webdata# docker run -i pyscript2:01 ./script/sum.py
Enter a number: 34
Enter a number: 23
Sum is 57
```

So this is the way we can create images from container but we cannot mark them to execute our command. We have to provide command manually.

That's why we use Dockerfile. Because In dockerfile we can specify CMD

And whenever we run our image that application will always execute.

**Docker Network :**

In docker, Each Container gets different ip address. There is an adapter gets created that is like virtual switch which provide ips to containers. The default network adapter contains dhcp (like nat) which provide ips. So let's assume we have three containers. Assigned ips are …3 , …4 and …5 to container1,2&3 respectively. Now Container 1 & 3 Goes down. And we start container3 so ip of container3 …3 will be assigned to container 5 and then we start container 1 . SO previpous ip of container3 will be assigned. So its quite difficult to communicate between containers. So we use dns to communicate . But Default Adapter doesn't have DNS. So we have to create our own adapter.

```
# docker network create ditis1
8d3ded103b59b841c8a8f4488412807873a7ffa9803f6a9abb328617f6ae6483
root@binay:/webdata# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
22746b7ac00b   bridge    bridge    local
```

```
8d3ded103b59   ditis1    bridge    local
```

Now **Creating New Container with new Network:-**

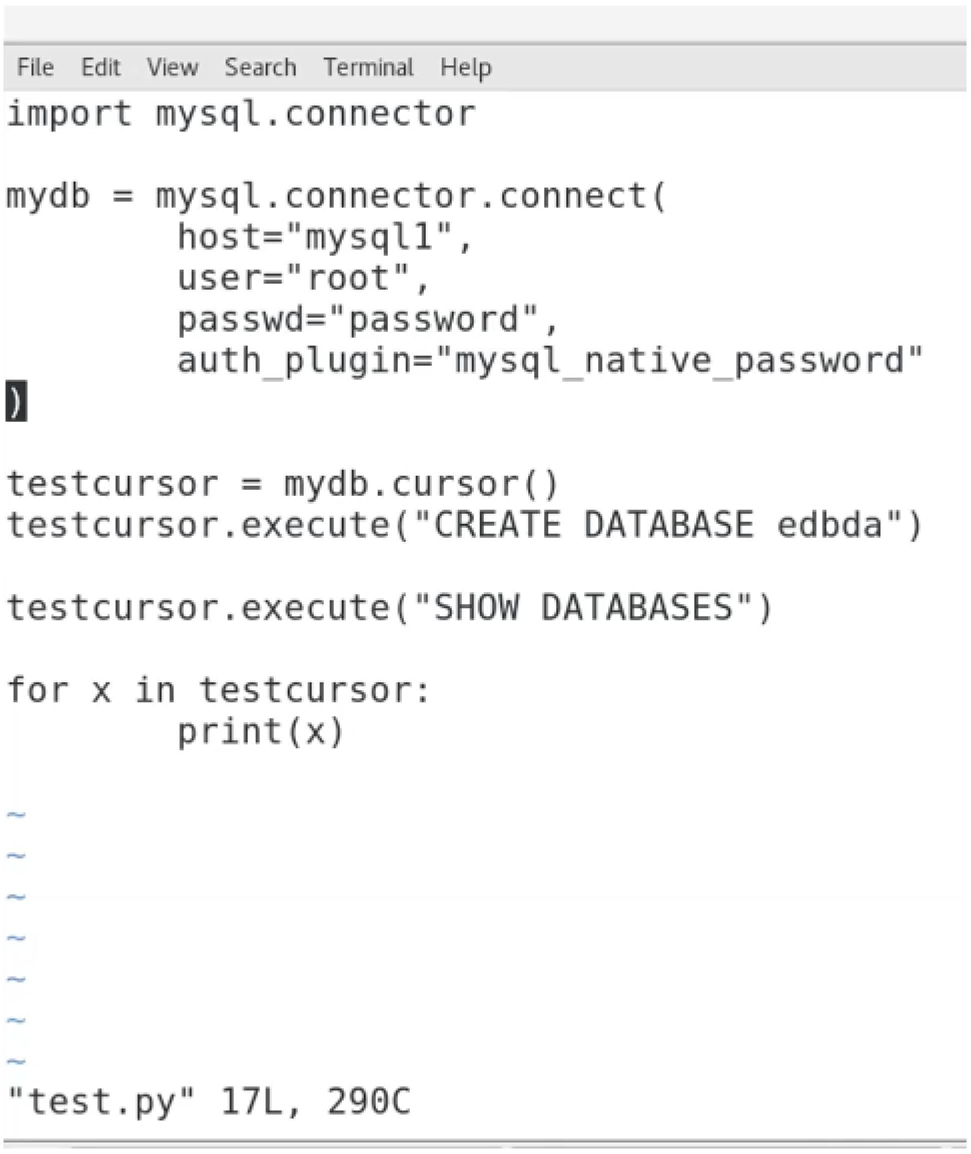
**# docker run --name ub2 --network ditis1 -ti ubuntu**

```
root@f786772d4cf6:/#
```

Now we will see ip of container ub2 on different terminal using inspect command:-

```
# docker inspect ub2
"Gateway": "172.18.0.1",
                "IPAddress": "172.18.0.2",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "MacAddress": "02:42:ac:12:00:02",
                "DriverOpts": null
```

=============================================================



```
import mysql.connector

mydb = mysql.connector.connect(
        host="mysql1",
        user="root",
        passwd="password",
        auth_plugin="mysql_native_password"
)

testcursor = mydb.cursor()
testcursor.execute("CREATE DATABASE edbda")

testcursor.execute("SHOW DATABASES")

for x in testcursor:
        print(x)

~
~
~
~
~
~
~
"test.py" 17L, 290C
```

#cat test.py

import mysql.connector

mydb = mysql.connector.connect(

```
        host="mysqldb",
        user="root",
        passwd="password",
        auth_plugin="mysql_native_password"
)

testcursor = mydb.cursor()
testcursor.execute("CREATE DATABASE ditiss")
testcursor.execute("SHOW DATABASES")

for x in testcursor:
   print(x)
```

```
#cat Dockerfile
FROM python:3.7-buster
RUN pip install mysql-connector-python
RUN mkdir /test
COPY test.py /test
CMD ["python", "/test/test.py" ]
```

Creating Image:

```
#docker build -t py-mysql .
[+] Building 101.9s (9/9) FINISHED

#docker images
REPOSITORY   TAG      IMAGE ID      CREATED           SIZE
```

```
py-mysql     latest    b0ce81c83460   About a minute ago   1.01GB
```

```
# docker pull mysql
# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
22746b7ac00b   bridge    bridge    local
8d3ded103b59   ditis1    bridge    local
3e305900a9d0   host      host      local
82222d793c1f   none      null      local
```

Creating Container:

In below command,

-d:- Detach

-e:- Environment

```
root@binay:/webdata# docker run --name mysqldb -d -e
MYSQL_ROOT_PASSWORD=password --network ditis1 mysql
46933777c5c87d81afba707b934d04a3aa32abdd918599b2a57a47f506399028
root@binay:/webdata#
# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS
PORTS                 NAMES
46933777c5c8   mysql     "docker-entrypoint.s…"   About a minute ago   Up About a
minute   3306/tcp, 33060/tcp   mysqldb
root@binay:/webdata#
```

Creating Container:

# **docker run --name p1 --network ditis1 py-mysql**

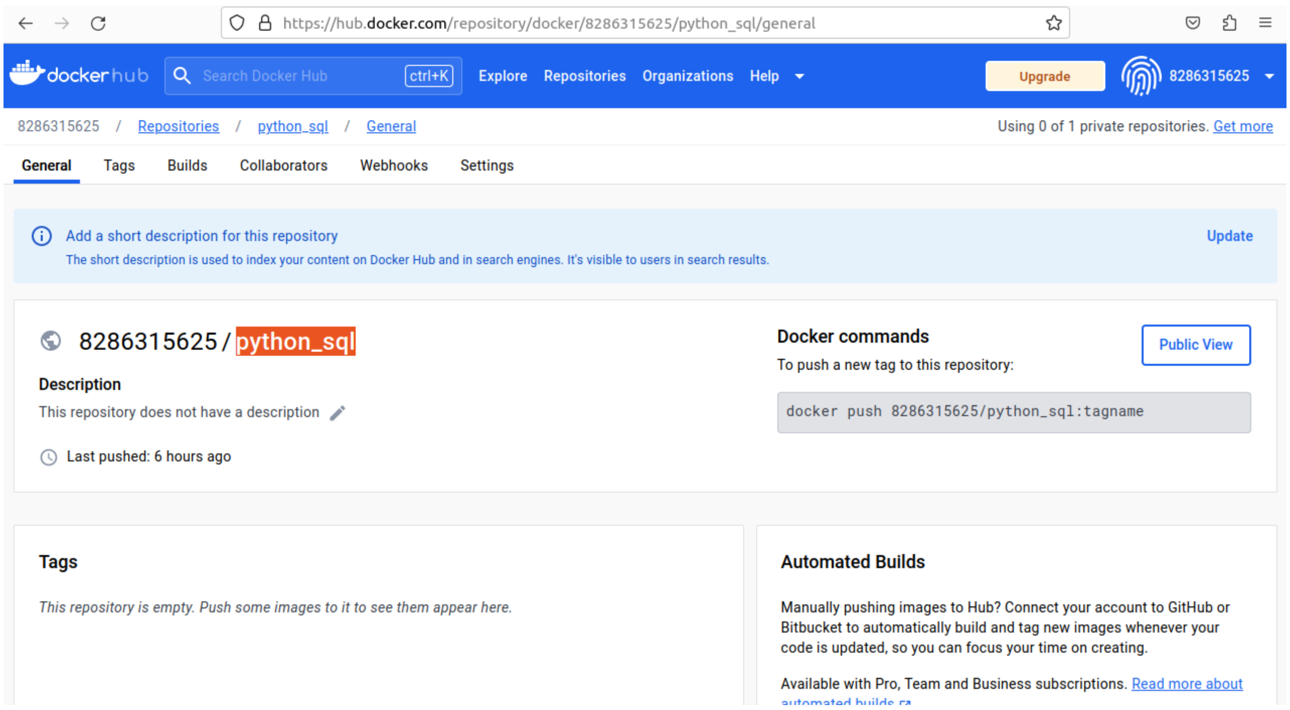
```
('ditiss',)
```

```
('information_schema',)

('mysql',)

('performance_schema',)

('sys',)
```

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

To push docker image to Dockerhub:-

Step1: Create a repo on Dockurhub



https://hub.docker.com/repository/docker/8286315625/python_sql/general

8286315625 / Repositories / python_sql / General

Using 0 of 1 private repositories. Get more

Add a short description for this repository

The short description is used to index your content on Docker Hub and in search engines. It's visible to users in search results.

8286315625 / python_sql

Description

This repository does not have a description

Last pushed: 6 hours ago

Docker commands

To push a new tag to this repository:

```
docker push 8286315625/python_sql:tagname
```

Tags

This repository is empty. Push some images to it to see them appear here.

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. Read more about

Step2: Rename your image same as repo name

```
# docker images

REPOSITORY   TAG      IMAGE ID       CREATED        SIZE

py-mysql     latest   0ac60838db91   16 hours ago   1.01GB
```

**# docker tag py-mysql 8286315625/python_sql**

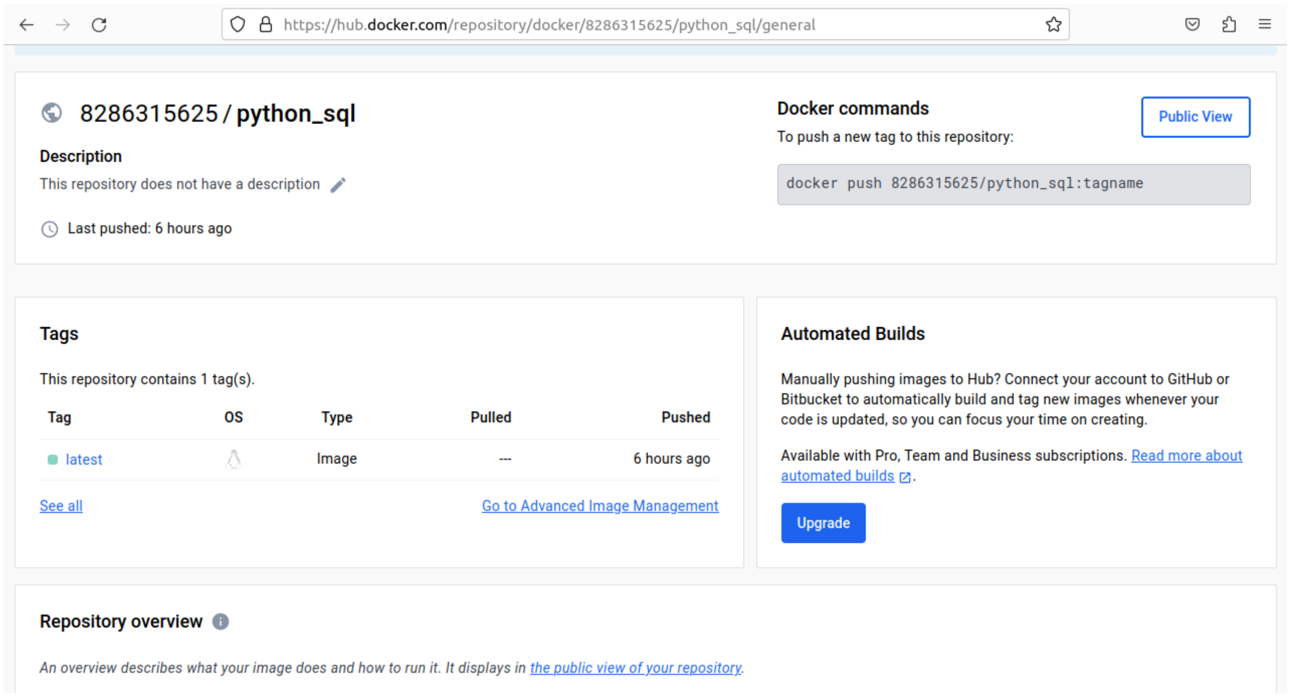
```
#docker images
REPOSITORY           TAG      IMAGE ID       CREATED        SIZE
8286315625/python_sql   latest    0ac60838db91   16 hours ago   1.01GB
```

**# docker push 8286315625/python_sql**

```
Using default tag: latest
The push refers to repository [docker.io/8286315625/python_sql]
```

Step3: login to your Dockerhub Account:-

```
# docker login
# docker push 8286315625/python_sql
```



https://hub.docker.com/repository/docker/8286315625/python_sql/general

8286315625 / **python_sql**

**Description**

This repository does not have a description

Last pushed: 6 hours ago

**Docker commands**

To push a new tag to this repository:

```
docker push 8286315625/python_sql:tagname
```

**Tags**

This repository contains 1 tag(s).

| Tag | OS | Type | Pulled | Pushed |
|---|---|---|---|---|
| latest | | Image | --- | 6 hours ago |

**Automated Builds**

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. Read more about automated builds.

**Repository overview**

*An overview describes what your image does and how to run it. It displays in the public view of your repository.*

DOCKER COMPOSE:---

```
]# cat myfirst.yaml
services:
  web:
    image: "nginx"
    container_name: apache2


    ports:
      - "80:80"
    volumes:
      - /root:/usr/share/nginx/html
  web2:
    image: "httpd"
    container_name: nginx
    ports:
      - "81:80"
    volumes:
      - /root:/usr/local/apache2/htdocs

#docker compose -f myfirst.yaml up -d
#chmod 755 /root
```

Interview Questions:

What is command to create a container with bash shell?

```
docker run -it --name my-ubuntu-container ubuntu /bin/bash (For ubuntu image)
```

```
docker run -it --name my-ubuntu-container ubuntu /bin/bash
```