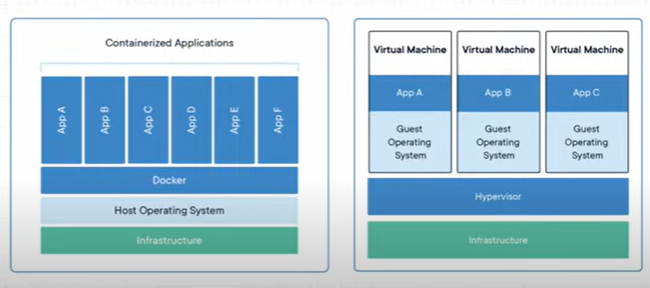
**Docker**

**VM vs Docker**

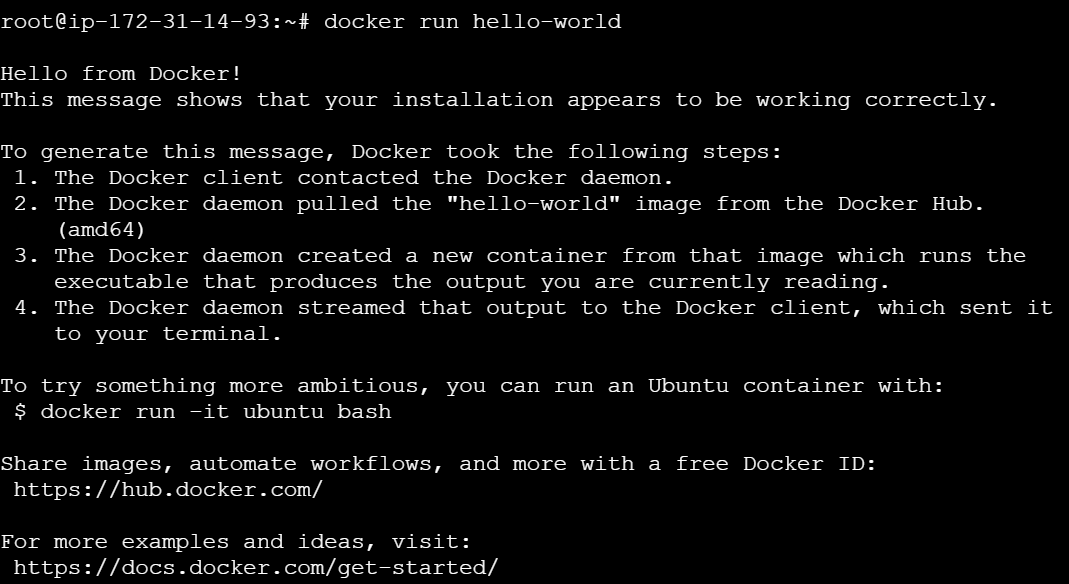
****

**Installation in Linux Ubuntu AWS EC2**

* curl -fsSL https://get.docker.com -o get-docker.sh
* sudo sh get-docker.sh

**To get the access for a particular user to work on docker**

* sudo usermod –aG docker user name

****



if we want to directly run a image we can use,

* docker run ‘image’

or else we can pull the image by mentioning image tag and then we can run

* docker image pull ubuntu:latest # here latest is a tag
* docker run -it 4e5021d210f6 /bin/bash # see above example in screenshot

# here ‘-it’ is used in the command because to make the current terminal as console for the ubuntu

image which we are using

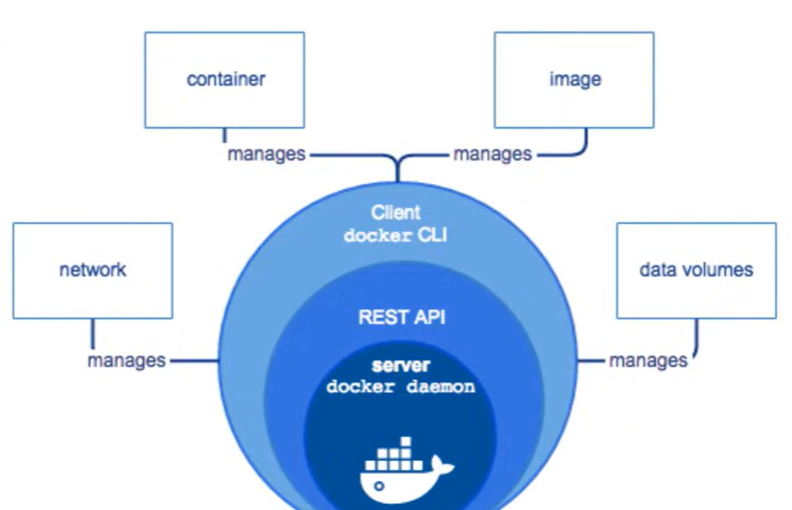
**To list all images**

* docker image ls

ex:

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.  
  
A diagram of containers with text

AI-generated content may be incorrect.

# With one image we can create multiple container and we can run, let say we can run 100 containers

also with that one image.

# Let say a application is running in one container with particular image, in another container we can

store the data of this application also.

But if we delete the container where application data is present, then data also we be lost.

# so in order to not to loose the data we use **Volumes**.

# now we if we want to store the data in the containers we have different storage drivers

basically. One of the famous driver is **Overlay2**

**Docker Commands**

# to list the running containers

* docker ps

# to list all containers which are running and exited

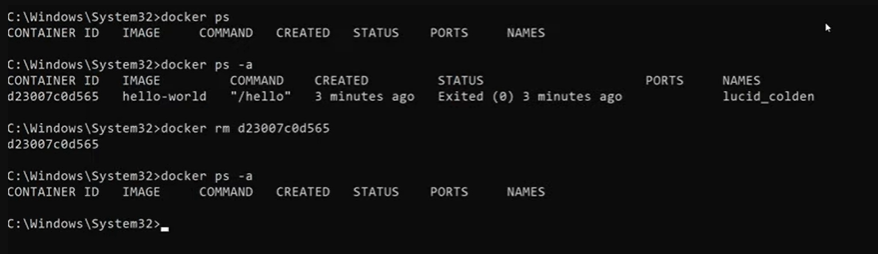
* docker ps -a

# to stop the container

* docker stop ‘container id’

# to remove the container

* docker rm ‘container id’



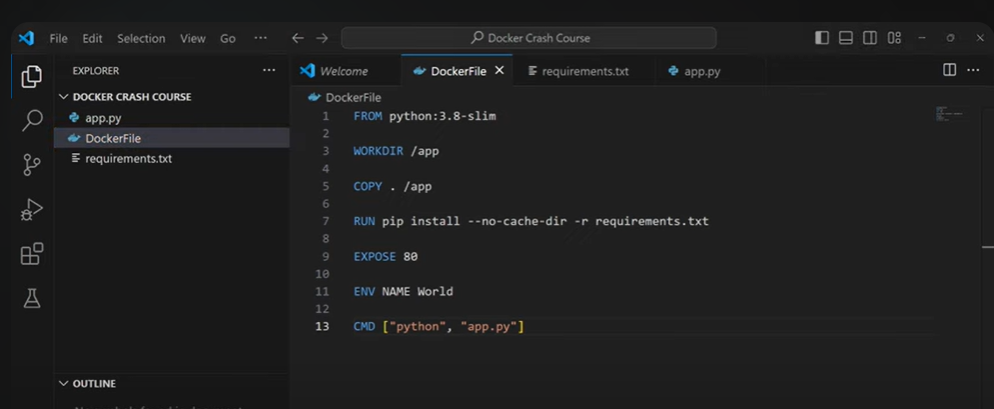
# to list all images

* docker image ls

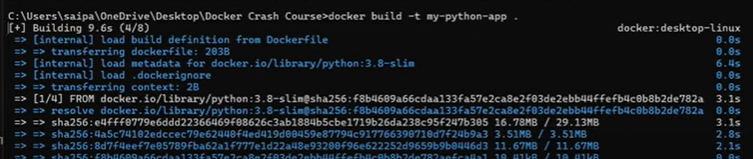
# to remove image

* docker rmi ‘image id’

Creating a Dockerfile



# to build the Dockerfile



Here ‘.’ Is context or the current directory



# to run this image

A screen shot of a computer program

AI-generated content may be incorrect.

Here -p is for port,

here 4000 is your host OS or local machine port number, that which you want to run on.

80 is the port number that you exposed in Dockerfile

Docker Compose

Whenever a larger enterprise application if we are running in docker then, multiple of services will be

there in the application. We tend to use multiple containers. So at this point we use docker compose.

Docker compose config is return in docker-compose.yml file

Example:

A screenshot of a computer

AI-generated content may be incorrect.

Now if you want to execute something on the existing container,

A screenshot of a computer

AI-generated content may be incorrect.

Here ‘exec’ means execute

* Two ways to Dockerize the application

1. Creating and running a app through docker commands, docker compose, Dockerfile
2. First after app is created, then creating a Dockerfile and building an image and run an image.

Notes Prepared From below youtube videos

* <https://www.youtube.com/watch?v=ITYYOTK0WE8>
* <https://youtu.be/GGaDSAMeopo?si=55AvDftMkvqYhLFd>
* <https://www.youtube.com/watch?v=QskegXV-90U>

some commands

Dockerfile, index.html

* docker build -t simpleapp:latest .
* docker run -it -p 8080:80 simpleapp:latest

if you are using windows pc, here 8080 is localhost port number, 80 is application port number, to test it

hit “localhost:8080” in the browser.

if you are using EC2 add 8080 in security group and to test it hit “Public IP of EC2:8080” in the browser.