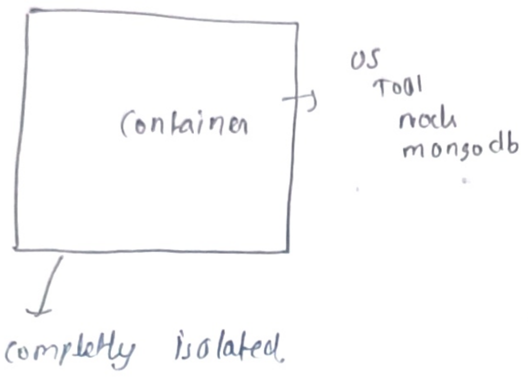
# // Docker

using docker, you can quickly deploy and scale application into any environment and reliable, low cost way to build, ship and run distributed application at any scale.



Daemon

## cmd

→ docker → To see all docker commands

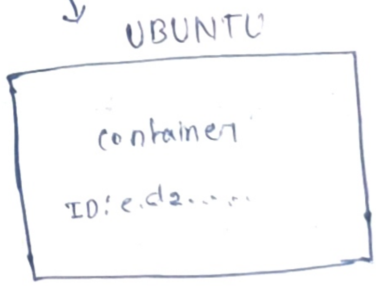
→ docker -V → To check version of docker

## // First docker image

→ docker run -it ubuntu

run → command
-it → interactive mode
ubuntu → image name

At first it will check that at your local computer whether you have ubuntu or not if not the it will downlode the ubuntu image from hub.docker.com and then it will create a container and this image will be run in that container which is completely isolated from outside of that container.

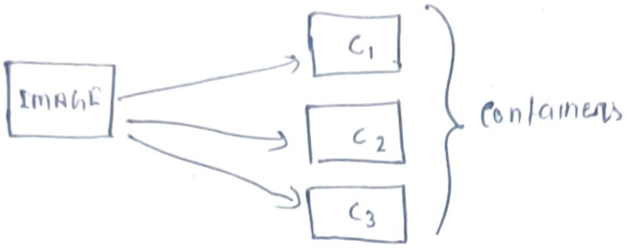


[ctrl+D] → To exit from container.

# use --name to give name to your container

→ docker run -it --name <name> <image_name>

A single image can be use. to run in multiple containers. and each contain is isolated from each other.



⇛ `docker container ls` → This will show your running container on your local machine.

→ `docker container ls -a` → This will show all container on your machine (exited ~~or~~ running)

-a → all

# every container on your machine have unique name. you can start or stop a container. by.

→ `docker start <name>` → this will start an exited container

→ `docker stop <name>` → This will stop an running container

# If you want to execute any command in a container, (make sure it's running) . you can do by this!

→ `docker exec <name> <command>`

name of container

command you want to run inside the container.

This will run that command in that container, show the result and get back to "cmd".

// But if you want to keep running this container in "cmd", you can do by this :

-> docker exec -it <name> bash

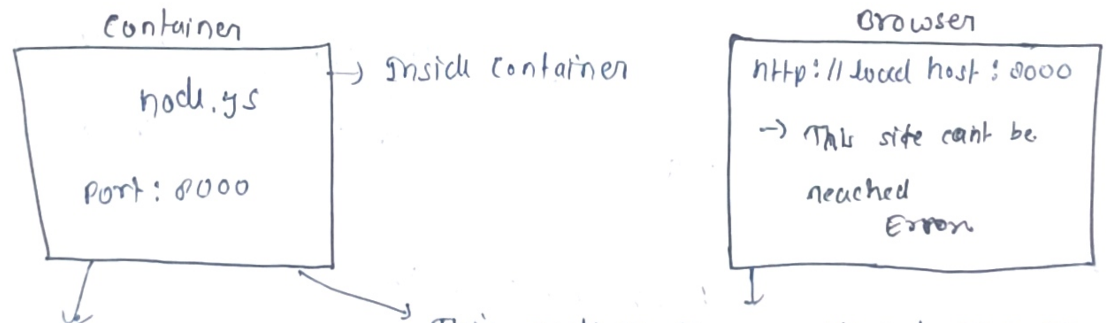
By adding this flag called "Interactive" keep running the container

shell name of ubuntu.

-> docker images ]-> By this we can see all images in our machine

// To explore images you can go to "hub.docker.com"

## // Port mapping



We need to expose this port to our local machine.

This node.js server will not run on browser, because container are complety isolated from our local machine.

But, to run this server on our local machine we use port mapping.

→ docker run -it -p 3000:3000 <image_name>

image name that you want to run

This flag is used for port-mapping.

3000:9000

your local machine server port

your container server port

This line means 8000 ( container server running port will be exposed to 3000 port on your machine.

we can now run container's server on our local machine brower on 3000 port.

"http://localhost:3000"

→ we can also do multiple port mapping, because some image run on two or more ports. e.g - mailhog

e.g

→ docker run -p 8025:8025 -p 1025:1025 mailhog/mailhog

①

②

image-name

## // Environment variable

→ docker run -it -p 1025:2025 -e key=value -e key=value <image-name>.

# 11. Dockerization of Node.js Application.

Inside your node js application create a file named "Dockerfile" → Be carefull with name.

node.js
index.js
package.json
package-lock

Dockerfile

First we have to chose a base image.
↓
[OS] e.g (ubuntu, windows, mackos).

~~testate~~

① Lets go with ubuntu
↓
to run node js we need to install node on ubuntu.

Ubuntu

② To install node js on ubuntu we need follow these steeps
Refer to any documentation

③ Then we need to copy our source file to the ~~container~~ image.

→ From ubuntu

→ Run apt-get ubdate
→ Run apt-get install -y curl
→ ~~Run~~ Run curl -SL < Link >
→ Run apt-get upgrade-y
→ Run apt-get install -y node.js

→ Copy package.json package.json

our local machine file get copyied to image

→ copy package.lock.json pack . . . . .

→ COPY main.js index.js

local machine file name

image copyied file name

→ RUN npm install. (

→ ENTRYPOINT [ "node", "index.js" ]

#
all those underline command should be capital.
eg RUN, FROM, COPY

→ ENTRYPOINT [ "node" , "index.js" ]

after writing these code in "dockerfile".

go to cmd inside of that folder location and

Run :

```
→ docker build -t <name_of_image> .
```

path of the file

• means it exist in same folder.

Deamon will perform some step and create your image as per your instructions.

you can run your docker image by.

```
→ docker run -it <name_of_image> -p 8080:8080
```

dont forget to map the port.

and you can go inside your running container by.

```
→ docker exec -it <Id_of_container> bash
```

You will enter inside the container and now you can see the files by 'ls' command.

you can even see the code inside your index.js by command.

```
→ cat index.js
```

// code

we can also choose "node" as our base so, we need not install ubuntu and then inside the ubuntu node js.

# Caching Layers.

if we make some change in our index.js and re build the docker image, in above step where no changes are made, it run fast and when it will find some change in this step then after that step it will run every command again.

that why the ordering of the docker command inside the dockerfile should be carefully.

so the common command should be above and where we are going to make change should be below, so we don't need to run every command every time

# Publishing to hub.

go to "hub.docker.com" sign in there and create a repository.

you will get a name like

<username>/<repo-name>.

e.g [ addi4491/my-1st-docker-img ] -> copy this

and create an image with the same name.

```
-> docker build -t addi4491/my-1st-docker-img .
```

after that go to cmd. and run

→ docker push <name_of_newly-created_image>

↓

But make sure you are logged In Inside your local machine.

To login your docker locally, run command.

→ docker login

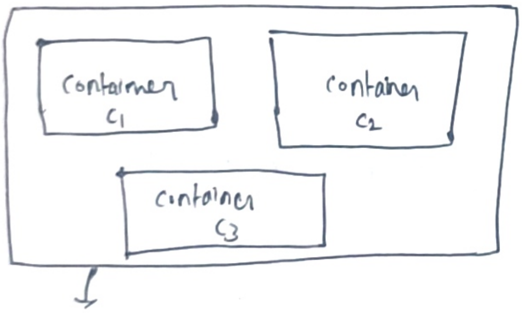
username : adii4451

password : .........

and enter.

// docker compose

- services
- Port mapping
- Env variable



↓

docker compose

docker-compose.yml

extension

```
version: "3'8"

services:
  postgres:
    image: postgress.
    ports:
      - '4000:4000'
    environment:
      key = value
      key = value
  redis:
    image: redis
    ports:
      -
    environment:
      key = value
```

container name ← postgres

image from hub.docker.com ← image

It will run an stack of containers.

To run this

→ docker compose up.

To close this

→ docker compose down