Docker

↳ Slove the problem of working on my machine not on yours

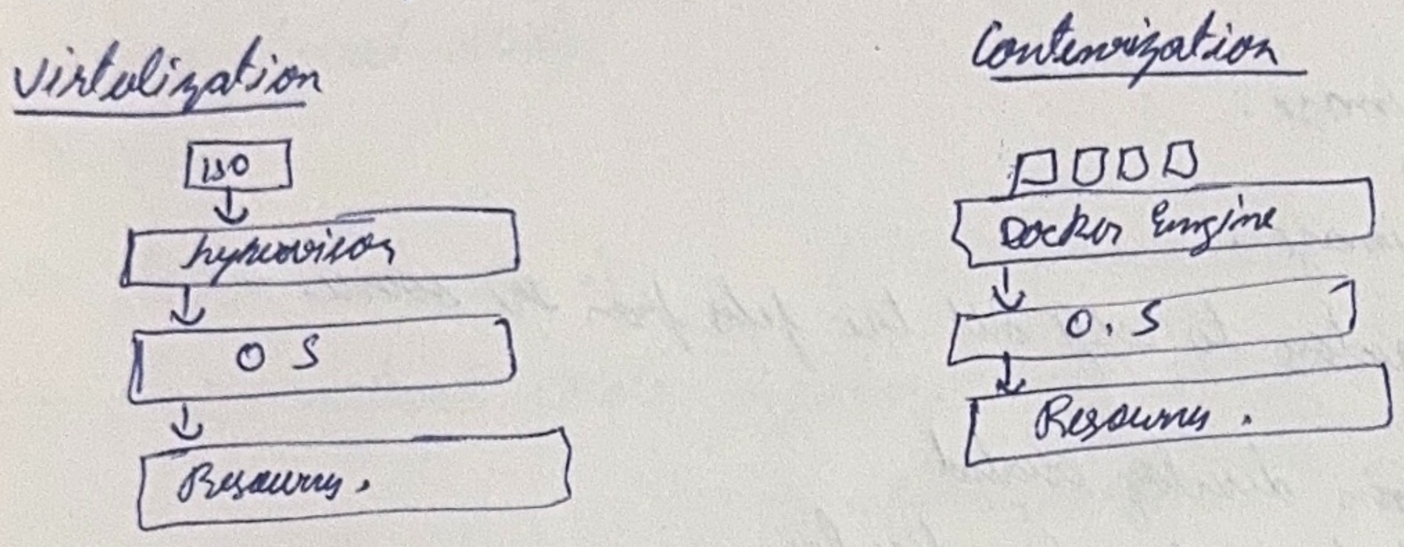
↳ Virtualization v/s Containarization

- VM
- hypervisor

- containr, docker
- : will run on docker engine & will use the host O.S

(will allocate resource from the O.S like hardware, storage and will run on top of it)

⇒ will have to get our own O.S due to which it will be heavy



→ Architecture

Docker Engine :- Core software that runs & manages dockers.

Docker daemon :- will manage all the contains

↳ Containerd :- It is a tool which manage contains

Docker CLI :- To execute all the docker commands.

Docker client :- with the help of this will talk to docker engine using API like docks desktop.

c→ sudo apt-get install docker.io (Install docker).

c→ sudo system-ctl status docker (To check if docker is up & running).

c→ sudo usermod -aG docker $USER (Add the current user to the group docker)

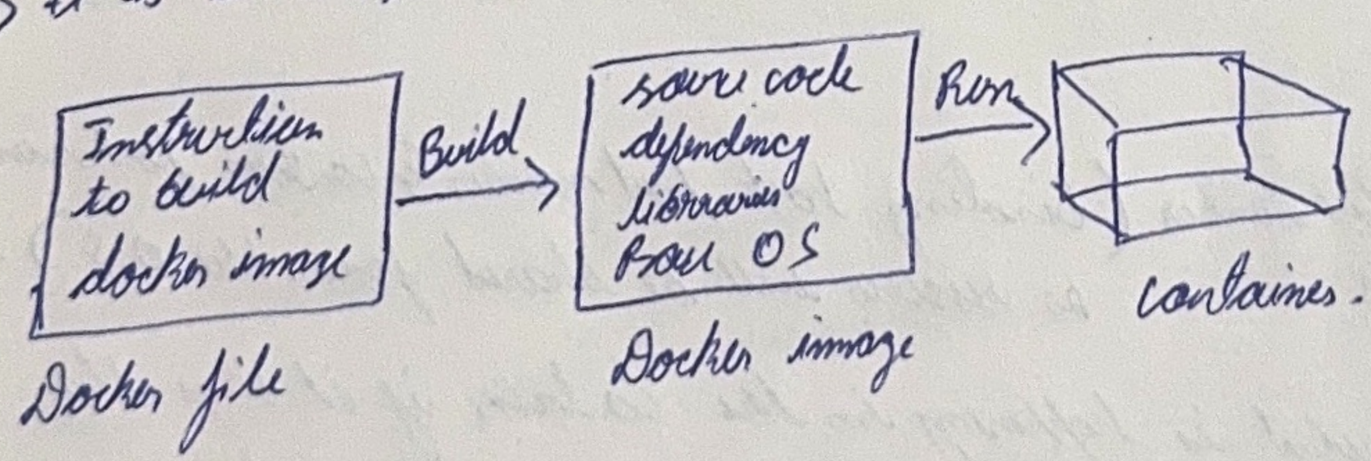
c→ newgrp docker (To refresh the group docker).

→ Docker images

↳ It is like a cheat-code for example

↳ we write it using docker file (Instructions will be written in docker file).

↳ It is a blueprint which contain all the steps, dependency, requirements to run the application



↳ docker pull hello-world (Pulling an image called hello-word from docks repo)

↳ docker images (list all the docker images)

↳ docker run image-name (to run the image)

↳ docker ps (list all the running containers)

↳ docker stop (contain_id) (to stop the containers running).

↳ docker run -d image (to run contains in detach mode).

Docker file

↳ steps of instruction to build docker image.

FROM openjdk:17 → base image. → version of the image.

WORKDIR /app → creating a directory to copy all the files from the source.

copy . . → copy to contains directory created
→ copy from host current working directory

RUN javac src/main.java → For compiling the java file
→ which file to run which is present in the contains directory.

CMD ["java", "main"]
↳ Contains will be created even without this but this will run the contain after created

CMD → can be overridden
ENTRYPOINT → cannot be overridden.

↳ docker build -t java-app . → location of where the docker file is located
→ image name
→ cmd to build docker image.

↳ docker run java-app.

If we have any requirements then ↳ RUN pip install -r requirents.txt.

↳ docker run -p 80:80 flask-app
→ contain port number
→ host port number
(binding host port number & contains port number as resources will be shared from the O.S).

run docker logs contains-ID, to check what is happening in the contains if it says its running but you can't see anything on port 80 then in security group of the contains we need to allow the inbound traffic on port 80.

↳ docker attach contain_ID (live logs on the contains).

any container is running and you wanted to execute it then the following cmd can be run.

↳ docker exec -it contain-id bash
↳ execute ↳ interactive terminal.

Docker networking

↳ Host
↳ Bridge (default)
↳ User defined bridge
↳ None

↳ docker network ls (list all network)

↳ docker network create mynetwork -d bridge → type of network.
↳ network name ↳ driver

Two tier app

① Create a network
↳ docker network create two-tier -d bridge

② Create mysql contains & attach two-tier network
↳ docker run -d --name mysql --network two-tier -e Mysql-root-password=root -e Mysql-database=logs
↳ contain name ↳ environment name

dockps mysql
image name

③ Run flask app on same network.
↳ docker run -d -p 5000:5000 --network two tier two-tier-backend:latest
image name.

↳ docker network inspect two-tier
↳ You will find two containers running in this network.

⇒ If we restart the sql server the data wont be lost, but if we delete it the data will be gone.

Docker volume

↳ docker volume ls (list all the volumes)
↳ docker volume create mysql-data (creating a new volume)
↳ docker volume inspect mysql-data (to find the location where it is binded with)
↳ docker run -d --name mysql --network two tier -v mysql-data:/var/lib/mysql
this is named volume ← host volume container's data.

we can also give specify folder location if we want & that works too.

Docker Compose

version : "3.8" → Version of docker compose.

services: → All the containers comes here

mysql:
  container_name : mysql → name of the first container
  image : mysql → which image is it using to build the container
  environment: → environment variables.
    mysql_database : "always"
    mysql_root_password : "root"
  ports: → which ports is it using
    - "3306 : 3306"
  volumes: → Mapping volume of host to container.
    - mysql-data : /var/lib/mysql
  networks: → which network is it using
    - two-tier
  restart : always → Restart until healthchecks are passed
  healthcheck:
    test : ["CMD", "mysqladmin", "ping", "-h", "localhost", "-uroot", "-proot"]
      ↳ It is running a command to ping mysql
    interval : 10s
    timeout : 5s
    retries : 5
    start_period : 60s

flask:
  build:
  context : . } → building this contains from the image which is mentioned at the context.
  container_name : two-tier-flask
  ports:
    - "5000":"5000"
  networks:
    - two-tier
  depends_on: } → Until mysql is created this contains will not be created
    - mysql

volumes:
  mysql-data: } → which volumes are we using

networks:
  two-tier: } → which networks are we using.

c→ docker compose up (to run the docker compose file).