Docker training course :-

① Introduction :-

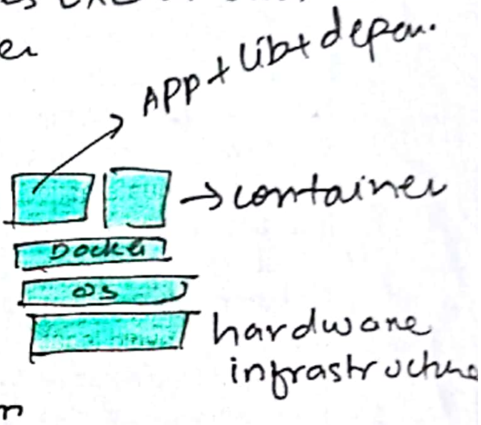
Docker → ensure compatibility
→ allows you to run each comp. in separate container

containe → isolated environment
→ shares same OS kernel

traditional dep.

- Compatibity / Dependencies
- long set up time
- Diffren Dev | Test | Pro. env.

- Docker uses LXC enviroment container



VM Vs Docker

Docker run nodejs
Docker run nodejs
docker run mongodb
docker run redis

If you run multiple instance.

Images Vs Container

VM Vs Docker:

- images are like template use to create one or more two containers

- containers are running instances of images use ~~to create~~ have isolated processes and env.

2) Getting started with Docker:

when you run any container of ubuntu image it stops immediately because ubuntu is just a base image of OS use for base imago of other APP.

Docker commands.

to check version:

- sudo docker vesion

- Docker run nginx
  (use to run a container from image) If image not exists it will download it from docker hub.

- docker ps → to list all containers.
- docker ps -a → to see all running and previously stopped contained.
- docker ~~ps~~ stop cnt_name → to stop container

- docker rm cont-name → to remove container permamely

- docker images → to list all existing images.
- docker rmi imag_name → to remove image (make sure to delete all dependent cont.)
- docker pull imagename → to pull image directly
- dockerrun ubuntu sleep5 → append a command
- docker exec distracted_mcclintock cat /etc/hosts
  container name
  ↳ to print content of /etc/hos
- docker run kode/simple_webapp (shows output of webservice on screen)
- docker run -d " " " " (to run webapp in bg mode)
- docker attach a043d → (attach container)
  (id of container)

- docker run -it centos bash. (If want to run container alive) login to container directely
  name

- docker run -d centos sleep 20 (-d to run task in background)

- Cat /etc/*release* (to check the operating systems we are on)

- docker ps, docker ps -a.

- rmi -remove images

- docker run image → it first looks for an image locally then if not found than pull it if we don't want to run ~~and~~ image immedialtely than use pull command.

- docker pull imagename

- docker run redis
- docker run redis:4.0
  tag

} If tag is given the specified version will run otherwise docker will suppose it as (latest)

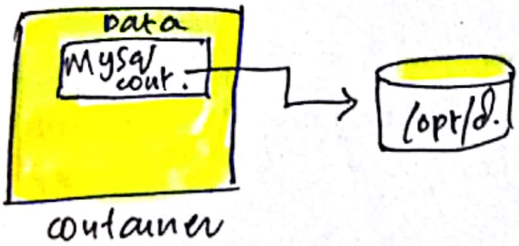
- docker run -i wepapp

-i is for interaction mode
-it will show promp.

- docker run -p 80:5000 kodecloud.
- docker run -p 8306:3306 mysql

} Port mapping.

- docker run -v /opt/datdir:/var/lib/mysql mysql



Value mapping for persisting data.

- docker inspect cont_name → to inspect data in json form

- docker logs cont-name → to show logs of container

How to create my own Image ?

- create dockerfile.
- create image.

```
docker build Dockerfile -t amna/myapp
docker push amna/myapp (topush in docker
                         registry)
```

Dockerfile :

contains instructions and Arguments.

FROM Ubuntu

each line of instruction creates a new Layer in docker image.

follows layered Archictecture

- installs all dependencies
- defines Base OS.
- Copy source code
- Specifyentrypoint.

to see file detail run:

```
docker history amna/myapp.
```

-> docker build ~~output~~ ⟶ to build output of docker

⟶ FROM Ubontu → docker run ubuntu-sleeper sleep10.
CMD sleep 5 ⟶ CMD ["sleep","5"] } command

but if we change time of sleep?

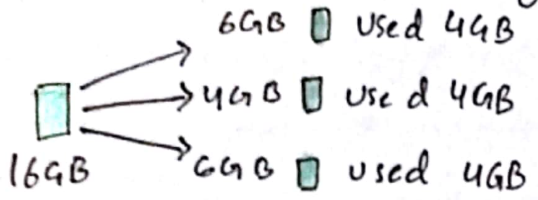
⟶ FROM UBUNTU
ENTRYPOINT ["sleep"] → at terminal
(sleep 10)

⟶ Docker Service create -replicas= 100 nodejs.

VM vs Docker :?
virtualization vs cont..?

Virtualization

- relies on hypervisors and OS instances
- provides complete isolation and compatibility with diverse OS.
- remaining memory cannot be used by any other VM.
- no reusability of memory.

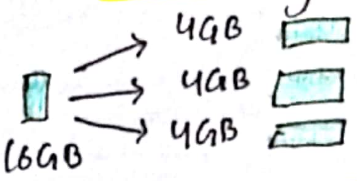


remaining 4 cannot be reallocated.

- uses Guest OS

containerization (Docker)

- leverages lighweight containers shared OS kernels.
- offers increase agility, portability and scalibility.
- better in speed, size, integration.
- remaining memory can be allocate to other containers
- reusability



4gb remaing can be allocated again.

- uses host OS

Docker
- uses client server Architecture
- easy to deploy Create and run applications
- light weight
- no preallocation required.

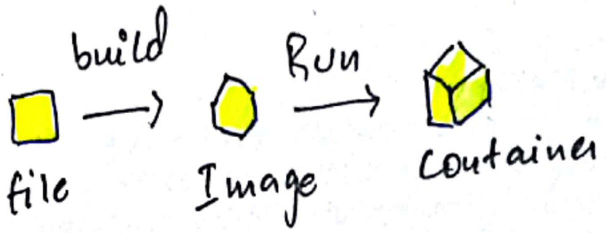
how docker works :

client: Dock.CLI → REST API → server: Docker Daemon

- Dock.CLI: installed on your host machine.
- REST API: communication using rest API
- Docker Daemon: daemon checks the request to manage container.

Components of Docker:

- Dockerfile
- DockerImage
- Docker Container
- Docker Registry



Dockerfile (text doc)

- contain all command
- executes commandline inst.
- to buid an image from dockerfile.

  execute

  ```
  docker build . -t imagename:tag
  ```

Docker Registry

- storage component for docker images.
- images can be save in pub or private repos.
- Use to control where images are being stored.
- integrates images with in-house dev workflow.

Docker Compose:

- Tool for defining and running multi-containers.
- .yml is used to configure app services.

  ```
  docker-compose.yml
  ```

- Single command create and starts all services

  ```
  docker-compose up
  ```

Docker Hub: App store for docker images.

Dockercompose

1) docker-compose up

2) --link (uses to link containers together)

↳ docker run -d --name = worker-p --link redis:redis votingapp

Demo - Voting App

1) cat Dockerfile

2) docker build . -t voting-app

3) docker images

4) docker run -p 5000:80 voting-app (The container app runs at 80 and will map that too at my host on 5000)

//failed to establish connectivity to reddis error.

first run a reddis container

5) docker run -d --name =redis redis (-d to run in bg)

6) docker rm reddis → to clear delete container.

7) again -d wali command.

8) docker ps

9) docker run -p 5000:80 --link redis :redis votingapp

8) docker run -d --name =db postgress:9.4

9) /worker → ls

10) /worker → cat Dockerfile

11) docker built . -t worker-app

12) docker images.

13) docker run --link redis:redis --link db:db worker-app

Deploying resulting app:

14) cat Dockerfile 15) docker build . -t result-app 16) docker images

17) docker run -p 5001:80 --link db:db result-app

Example Voting App. with Docker Compose.

⇒ `docker stop 69 54 5b 2f 0b`   first 2 digits of all running containers id to stop them

⇒ `cat > docker-compose.yml`   (create file)

```
redis:          worker:
db:             result:
vote:
```

} define images.

⇒ `vi docker-compose.yml` → to edit file.

```
redis:                   db:                      vote: image: voting app
image : redis            image: Postgres:9.4            Ports: -5000:80
worker:                                                 links: -redis
 image: worker-app       result:
 links:                   image: result-app
   -db                     Ports: - 5001: 80
   -redis                  links:
                            -db  ~~postgres~~
                           ~~-bod~~
```

save file.

⇒ `docker-compose up`   (in which directory you will be it will make it as prefix)

every container will reach out to each other easily. so need to specify links.

```
.yml
version: "3"
services:
  redis:                          worker:
   image: redis                    image: worker-app.
 d db:
   image: Postgress9.4
   environ...:
     Postgres_user: postgres
     Postgress-Password: postgress.
  vote:   images: voting-app
  result:  Ports: -5000:80
   images: result-app
   Ports:
    -5001:80
```