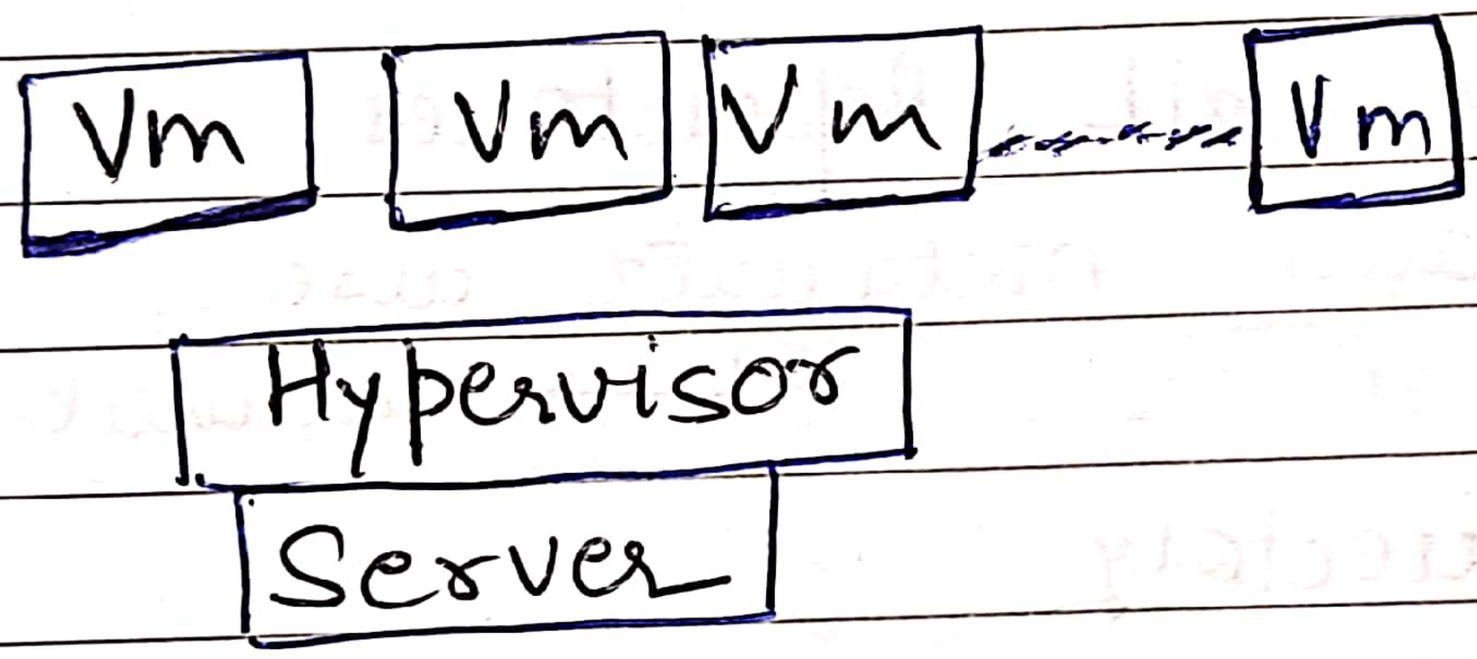
(1.)

Date :

Virtualisation

Technique of splitting a physical resource into as many logical resource as we want.
eg - CPU, memory (RAM)

- Transforms Hardware (Physical) into Software (logical)



Hypervisor :

- Type 1 → Bare metal & Native
- Type 2 → Hosted
  - Enterprises
  - testing

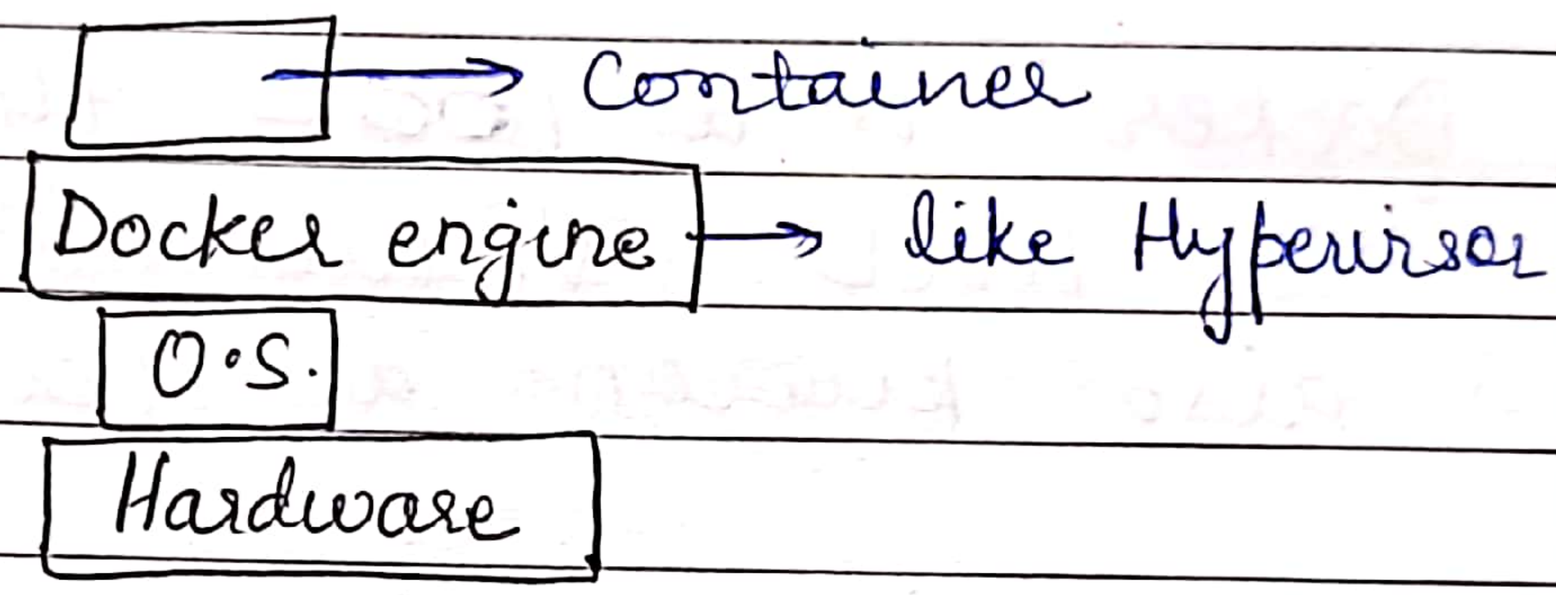
	Vm
	Hypervisor
Hypervisor	Host O.S.
Hardware	Hardware
T1	T2

Date: Container is like a Virtual machine

Docker is a tool which creates this virtual machine.

Docker is a advance version of Virtualisation.

Docker ⟶ Containerisation



- Container uses the resources while working on task after completion of task it returns the resources used.
- Docker is an open source centralised platform designed to create, deploy and run applications.
- Docker uses container on the host O.S. to run applications. It allows app. to use the same linux kernel as a system on the host computer, rather than creating a whole Virtual O.S.

Scanned with OKEN Scanner

③

Date: We can install docker on any O.S. but Docker engine runs natively on Linux distribution.

- Docker written in 'go' language.
- Docker is a ~~tool~~ that performs O.S. level Virtualisation, also known as Containerization.
- Docker is a set of platform as a Service (**PaaS**) that uses O.S. level Virtualization whereas VMware uses Hardware level Virtualisation.

Advantages of Docker :

(i) No pre-allocation of RAM
(ii) Less cost
(iii) light in weight
(iv) CI efficiency -
(v) You can re-use image.

Date : Disadvantages :

- Docker does not provide Cross-platform Compatibility means if an application is designed to run in docker Container on Window, then it can't run Linux or Vice Versa.
- Docker is suitable when the development O.S. and testing O.S. are same, if the O.S. is different, we should use VM.

Components :

Docker Daemon –

- It runs on the host O.S.
- It is responsible for running Containers to manages docker services.
- It can Communicate with other daemons.

Docker Client –

Docker users can interact with docker daemon through a client (CLI).

5.

Date : Docker Host -

Host H/w

It contains the docker daemon, images, container, networks and storages.

Docker Hub / Registory -

It manages and stores the docker images.

(i) Public Registory is called docker hub.

(ii) Private Registory is used to share images within the enterprise.

Docker images -

- It is read only binary templats used to create docker container.

File with all dependencies and configuration required to run a programme.

Date: To create an Image:-

- Take image from Docker hub.
- Create image from docker file
- Create image from existing docker Containers.

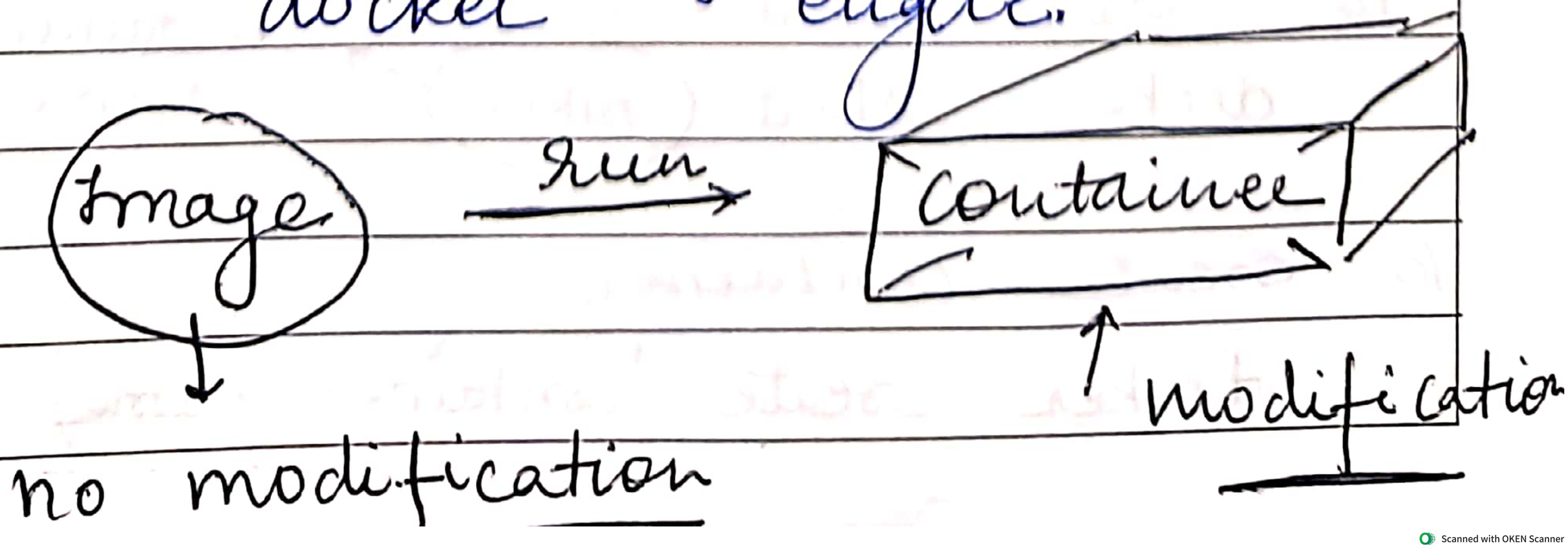
Docker Container

- Container hold the entire packages that is needed to run the application.

Or

the image is a template and the container is a copy of that template.

- Container is like a Virtual machine.
- Images becomes container when they run on docker engine.



7.

Date :

To see all images present in your local machine

[ ] docker images

To findout images in docker hub

docker search __

To download image from docker hub to local machine

docker pull (centos) → image name

To give name to container

docker run -it --name nikhil (ubuntu)

-it → interactive mode; terminal

/bin/bash

To check service is start or not

service docker status

To start container

docker start (nikhil) → container name

To create container

docker create [container name]

Date: To go inside container

docker attach nikhil

To see all containers

docker ps -a

To see only running containers

docker ps → Process states

To stop container

docker stop nikhil → Container name

To delete container

docker rm nikhil

docker diff container name

→ Create container from our own image

Step 1

Create Container

→ docker run -it --name nikcontainer ubuntu /bin/bash

→ cd tmp/

Scanned with OKEN Scanner

Date: Step 2 Create one file inside the tmp directory

→ touch myfile

Now, if you want to see the difference the base image & changes on it then

→ docker diff nikcontainer updateimage

O/p — C /root
A /root/.bash_history
C /tmp
A /tmp/myfile

Step 3 Now, Create ~~container from this~~ image of container

→ docker commit newcontainer updateimage
→ docker images

Step 4 Now, Create container from this image

docker run -it --name miscontainer updateimage /bin/bash

(10.) root @cid # ls

Date: root @cid # cd tmp/

tmp # ls

O/p → myfile [All files will be back]

DOCKER FILE

Docker is basically a text file.
It contains some set of instruction.

Docker Components

FROM – for base image. It must be on top of the dockerfile.

RUN – To execute commands. It well create a layer in image.

MAINTAINER – Author/owner/Description

COPY – Copy file from local system (docker VM) We need to provide source, destination.
(We can't download file from internet and any remote repositary)

ARG –

(11.)

Date :

ADD – Similar to COPY but it provides a feature to download files from internet, also we extract file at docker image side.

EXPOSE – To expose ports such as port 8080 for tomcat, port 80 for

WORKDIR – To set working directory for a Container.

CMD – Execute commands but during container creation.

ENTRY POINT – Similar to CMD. but has higher priority over CMD, first commands will be executed by ENTRYPOINT only.

ENV – Environments Variable

Scanned with OKEN Scanner

(12)

Date : Steps: **Dockefile**

(I) Create a file named Dockerfile
(II) Add instructions in Dockerfile
(III) Build dockerfile to create image
(IV) Run image to Create container.

```
Vi Dockerfile
FROM ubuntu
RUN echo "Technical guftgu" >
/tmp/testfile
```

To create image out of dockerfile

```
docker build -t myimg .
```

```
docker ps -a
docker images
```

Now, create container from the above image.

```
docker run -it --name
mycontainer myimg /bin/bash
```

→ cat /tmp/testfile

13.

Date : #

Vi Dockerfile

```
FROM ubuntu
WORKDIR /tmp
RUN echo "Nikhil" >/tmp/testfile

ENV myname Nikhil

COPY testfile1 /tmp

ADD test.tar.gz /tmp
```

Date :

Docker Volume

→ Volume is simply a directory inside our container.

→ You can declare a directory as a Volume only while creating container.

→ You can't create Volume from existing container.

→ You can share one Volume across any number of containers.

→ Firstly, we have to declare this directory as a Volume and then share Volume.

→ Even if we stop container, still we can access volume.

→ Volume will be created in one container.

→ Volume will not be included when you update an image.

You can mapped Volume in two ways :

(i) Container ⟷ container

(ii) Host ⟷ container

(15)

Date: Benefits of Volume:

→ Decoupling container from storage.
→ Share Volume among different containers.
→ Attach Volume to container.
→ On deleting container Volume does not get deleted.

LAB

(i) Create a Dockerfile and write

```
vi Dockerfile
FROM ubuntu
VOLUME ["/myvolume1"]
```

(ii) Then create image from this dockerfile

→ docker build -t myimage.

(iii) Now create a container from this image & Run

```
docker run -it --name container1
           myimage /bin/bash
```

— ls

O/P — myvolume1

Date :

Now, share Volume with another Container

Container ⟷ Container

- docker run -it --name Container2 (New)
  --privileged=true --Volume from Container1
  ubuntu /bin/bash

Now after creating Container 2, myVolume1 is visible whatever you do in one Volume, can see from other Volume.

→ touch /myVolume1/ Samplefile

→ docker start Container1

→ docker attach Container1

- ls — myVolume1

  exit

17

Date: Docker Port Expose

- docker run -td --name techserver -p 80:80 ubuntu

- docker ps
- docker port techserver
- docker exec -it techserver /bin/bash

Stop all running containers :

docker stop $(docker ps -a -q)

Delete all stopped container :

docker rm $(docker ps -a -q)

Delete all images :

docker rmi -f $(docker images -q)

Scanned with OKEN Scanner