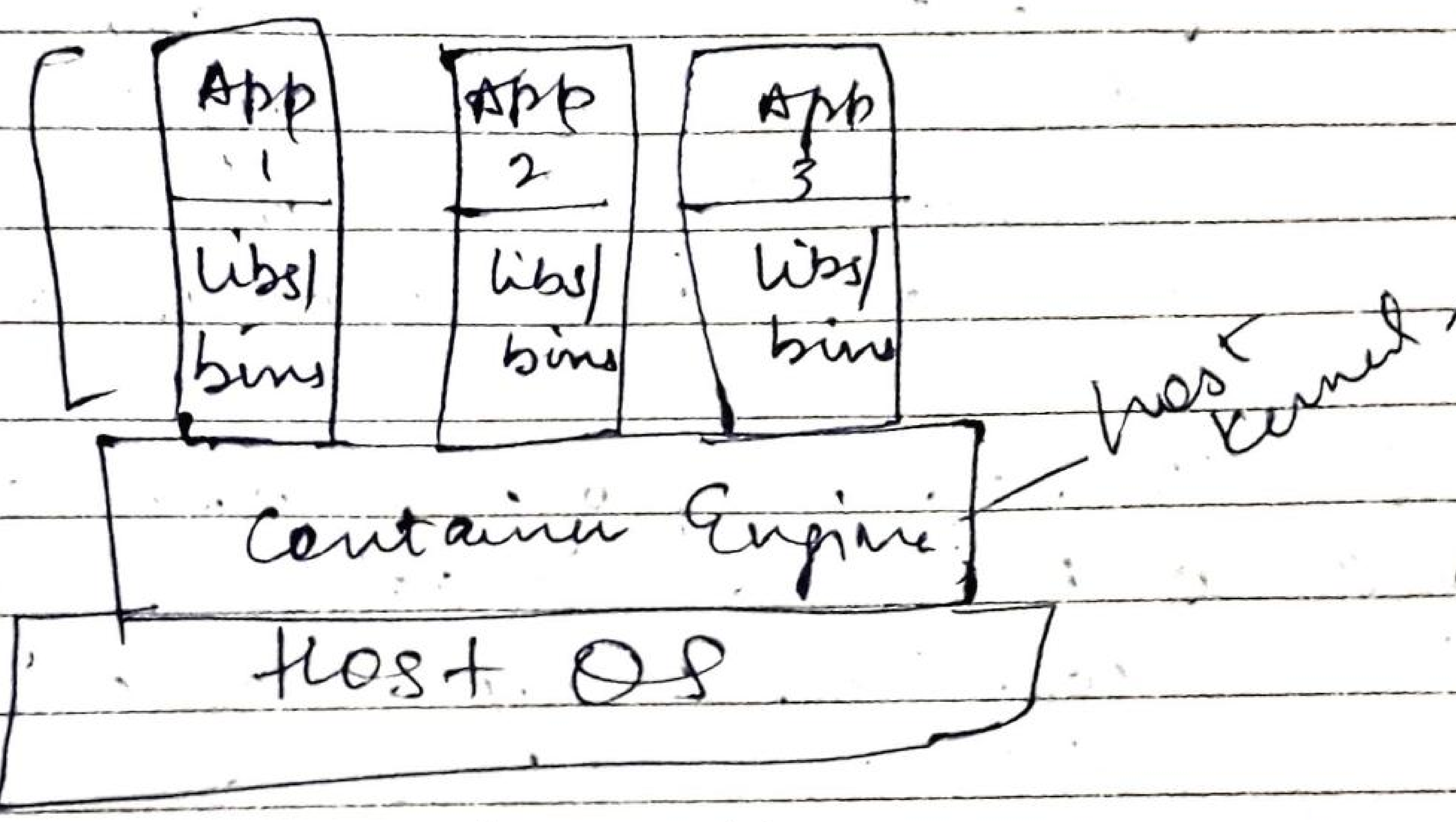
Q2. Define Containerisation:-

Ans2. Containerisation is OS-based virtualisation which creates multiple virtual units in userspace known as containers. Containers share the same host kernel but are isolated from each other through private namespaces & resource control mechanisms at OS level.



Containerisation.

Q3. What are docker Images?

Ans3. A docker image is a file used to execute code in a Docker container. Docker images act as a set of instructions to build a Docker container, like a template. Docker images also act as the starting point when using Docker. An image is comparable to snapshot in virtual machine environments. Docker images have multiple layers,

Scanned by TapScanner

Date
Page No.

each one originates from previous layer but is diff from it.

Q4 How to create container?
Ans4: To create a container, we use 'docker create <containername> command. This command creates a layer over original image which is editable & ready to run specific commands. This allows full manipulation of Docker images without running them, although once the user is satisfied with their amendments the image can be run so that it becomes a container.

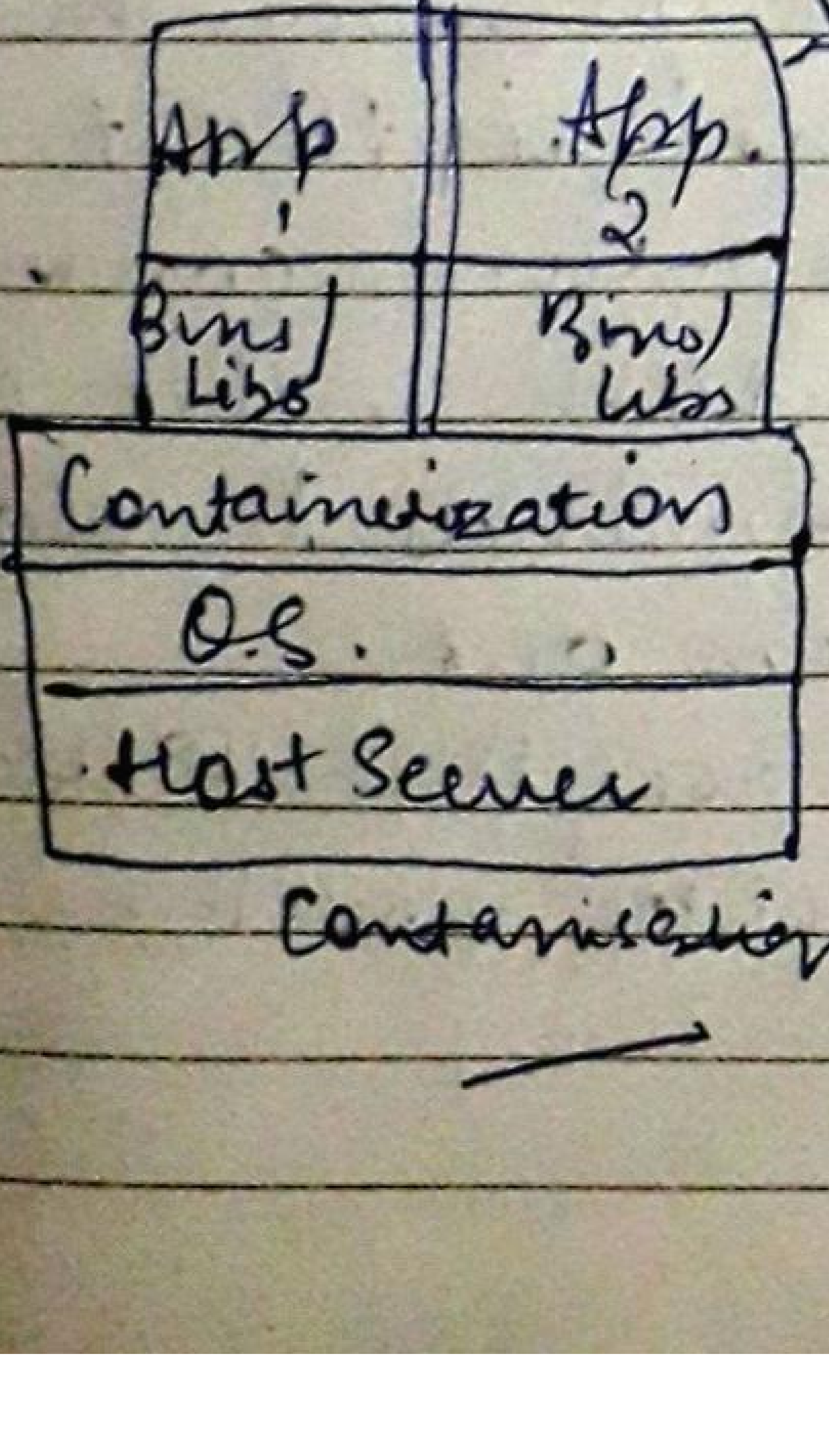
Q5. How to deploy docker container?
Ans5: ~~Q2~~ Build a container image for your container. Every container you deploy will be based on an image pulled from the docker hub. You can pull down a single image & use it as often as you like. Eg. Nginx, Apache, Alpine etc.

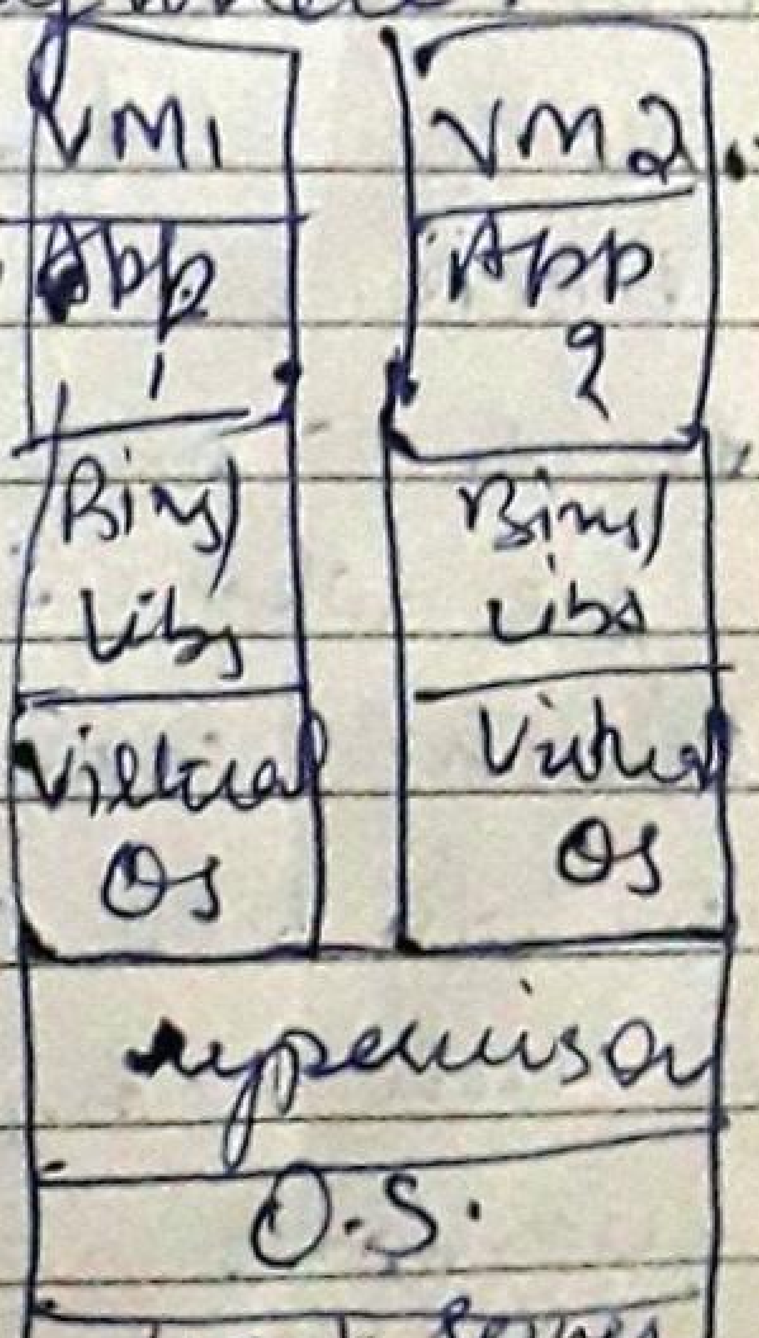
Q6. How ~~can~~ to you deploy an image?
Ans6. (1) Install Docker on your system.
(2) Download image from dockerhub.
(3.) Image will be Present in cache memory.
(4) ~~At~~ Craft Image will be in docker server, then client will access

that image through command :-
docker run <imgname>

Q7 Diff btw Virtualisation & Containerization
Ans

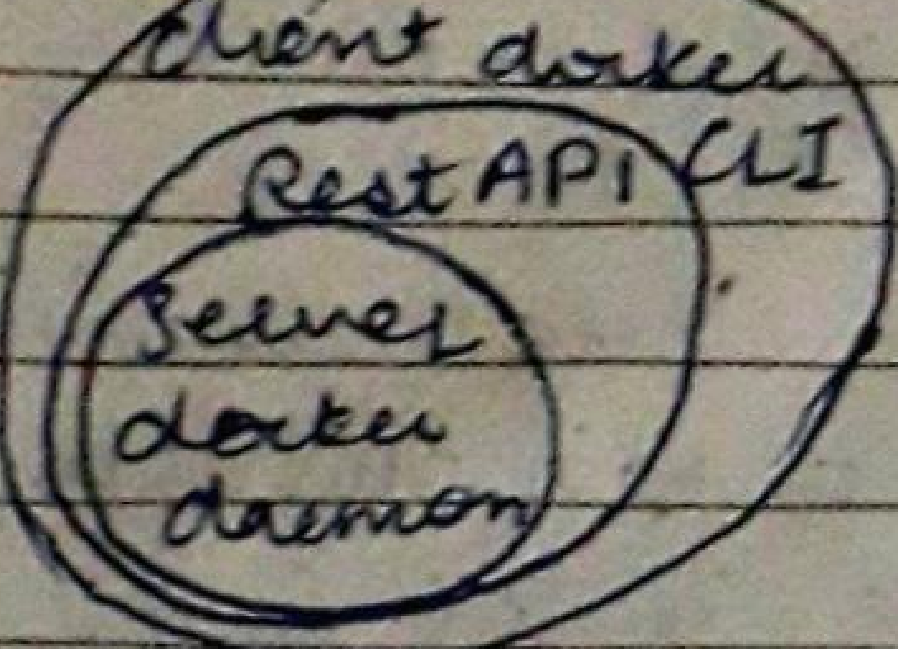
Virtualisation	Containerisation
- It is a technology which can simulate your physical hardware (such as CPU cores, memory, disk etc). & represent it as separate machine.	- It is OS-level virtualization. It doesn't simulate the entire physical machine
- It uses hypervisor to detach the physical machine.	- It uses docker engine to detach.
- It has hardware level isolation so it is fully secured.	- It has process level isolation.
- It is heavyweight	- It is very lightweight
- It is not portable.	- It is very portable. We can build, ship & run anywhere.





Date
Page No.

Q8. Explain docker client & server. / Docker Architecture
~~Imp~~



- Docker uses client server architecture
- Docker client talks to the Docker Daemon, which does the heavy lifting of building, running & distributing Docker Containers.
- Docker client & daemon can run on same system or you can connect a Docker client to remote Docker daemon.
- Docker Client & daemon ~~using~~ communicate using a REST API.

- Docker daemon listens for Docker API req. & manages Docker objects such as images, containers, networks & volumes.
- A daemon can also communicate with other daemons to manage Docker services [dockerd]

- Docker Client is the primary way that many docker users interact with Docker. when we run commands such as docker run, the client sends these commands to dockerd, which carries them out.
- The docker command uses the docker API
- It can communicate with more than one daemon.

Scanned by TapScanner

Creating first container:-

(1) Install docker on your machine.

(2) Create your ~~f~~project.
- Create a folder on your computer which should contain 2 files:-

(i) 'main.py' → python file that contain code to be executed.

(ii) 'dockerfile' → file that contain the necessary instructions to create environment.

(3) (i) Edit python file - print ("Docker is magic").

(ii) Edit the Docker file:-

```
from python: latest
copy main.py/
CMD [" python", "./main.py"]
```

(4.) Create docker image:-

```
$ docker build -t python-test
```

↳ // it allows to define
// name of the image

(5.) Run docker Image:-

Once image is created, code is ready to be launched.

```
$ docker run python-test
```

Output:- "Docker is magic!".

Stop container:-

(1) Stop a specific container:-

```
$ docker stop [container name]
        or
$ docker kill ~~stops~~ [container name]
```

(2) Stop all running containers:-

```
$ docker stop $(docker ps -a -q)
```

Remove Container:-

To remove container, we need to stop it otherwise it will give error without stopping command,

rm command is used for removing one or more container.

=> Delete a specific container (only if stopped)

```
$ docker rm [container name]
```

=> Forcefully deleting a container

```
$docker rm -t [container name]
```