Date

Docker

Initially: One application was running on one server.

Note: you want more servers to run application; For new application, you needed to buy new servers, which was not feasible.

VM ware solved this problem by "virtual machine".

After this you could run multiple application on one server, but there was still a problem;

The problem: virtual machine required their own operating system.

∴ better than before, but not perfect. (licsencing problem, etc)

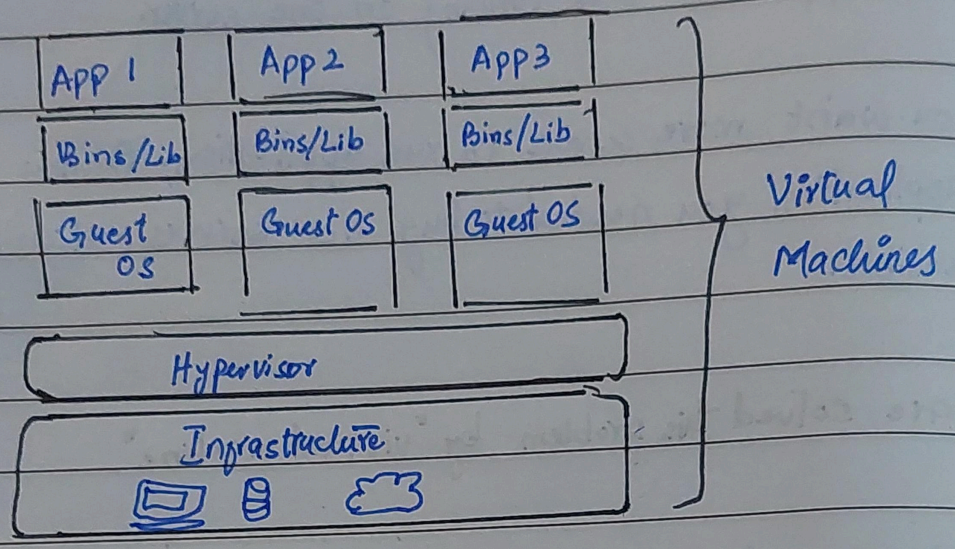
Multiple OS were required to use multiple applications. (required storage, part of CPU was allocated to vm,)

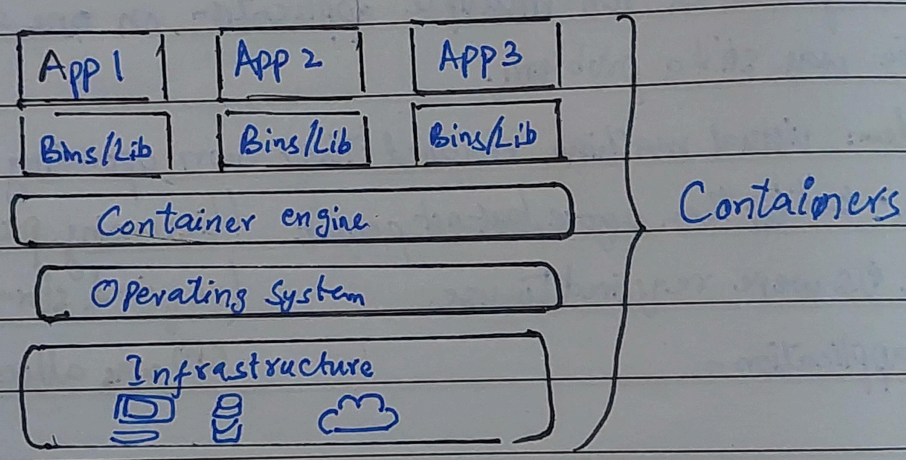
Note: In simple terms, the OS is the software that runs on the server (the computer), and the server is the physical or virtual machine that provides services or resources.

We wanted to run multiple applications on one OS.

~~This~~ This problem was solved by "containers".

Date





Hypervisor: It is used To create multiple machines on a host operating system and it manages virtual machines.

Date

What is Docker?

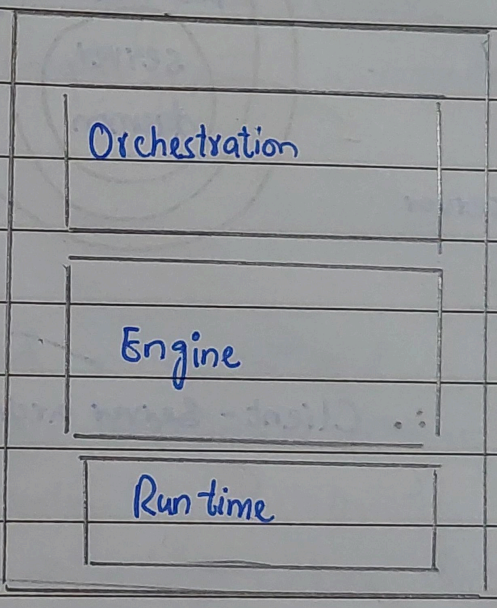
Docker is a container platform - that allows you to build, test and deploy applications quickly. A developer defines all the applications and its dependencies in a Dockerfile which is then used to build Docker images that defines a Docker container.

1. Run time

- It basically starts and stops the containers.

- Lower level run-time

• runc : the role of this run-time is to work with the operating system and starts & stops the containers.



• containerd : It lies on the higher level of hierarchy; it is used to manage runc.
It does stuff like; how to interact with your containers with the network (internet).
e.g pulling the images.

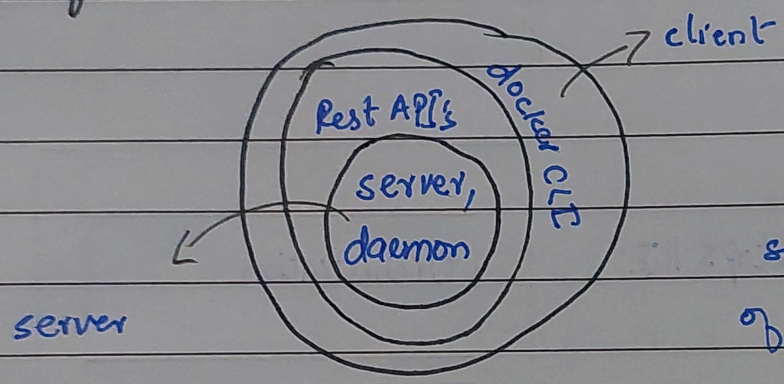
It is also used in kubernetes run-time.

Date

2. Engine

Dockerengine is something that we used to interact with docker.

e.g: "Docker Daemon" ⇐ server



∴ Via API you can communicate with server, daemon, by the help of docker CLI.

∴ Client - Server architecture.

Date

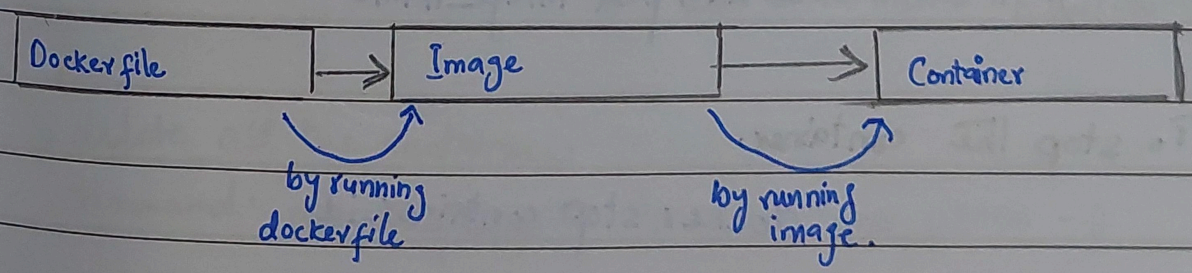
3. Orchestration: It helps in managing containers.

What is Orchestration?

Orchestration is the automated coordination and management of multiple tasks or components to achieve a specific goal or workflow efficiently. It involves organizing and controlling various elements to work together seamlessly, often in a complex system or environment.

What container orchestration?

Container orchestration is the automated management and coordination of containerized applications, like Docker containers, to ensure they run reliably and efficiently in a distributed environment. It automates tasks such as deploying, scaling, load balancing and monitoring containers, simplifying the management of these applications at scale.



Date

Command:

$ docker run hello-world

docker CLI

It's saying to run an image to create a container

image-name.

Command:

docker run -it ubuntu // it will run ubuntu in an // ~~interact~~ interactive environment.

To show live containers :

command: docker ps or command: docker container ls

To show downloaded images:

command: docker images.

If you want to download image and don't want to run it.

command:

docker pull image_name

To stop the container:

docker stop container_id

Date

To ~~stative~~ show live and stopped containers:

command: docker ps -a

To remove container:

Command: docker rm container_id

To get info about container:

command: docker inspect container_id

To attach bash into a running container; it would happen via using two terminals; one for running container in interactive version and another for attaching bash.

command: docker run -it ubuntu //in one terminal

docker exec -it container_id bash //in other terminal

Command: docker logs container_id //it will show all the outputs

// that were inside of this container

To delete all the containers:

Command: ~~docker prune -f~~ docker container prune -f

Date

To run the image in the background:

command: docker run -d image_name command

e.g: docker run -d alpine ping www.civo.com

To show the last 5 seconds of this command:

command: docker logs --since 5s container_id

// you can type only few starting numbers of
// container_id

To remove image:

command: docker rmi image_name -f

e.g. docker rmi alpine -f

To stop container:

command: docker stop container_id

To removes container:

Command: docker rm container_id

Date

nginx = Port 80 and Port 443

http = Port 80

https = Port 443

Let's say I want to access nginx ~~say~~ on my local port (8080)

Command: run -d -p 8080:80 nginx //-d represents that it is running ~~on~~ in background; we are forwarding nginx port to our local port.

now, browse: "localhost:8080"

-x-x-x-

for example you created a file in a container (e.g: ubuntu)

Comands:

1. docker run -it ubuntu //in one terminal
2. ~~docker exec~~ -it ubuntu bash //~~in other terminal; this command~~ is adding bash
   container_id
3. ~~touch~~ touch names.txt
4. echo "my name is ahmed" >> names.txt
4. exit //it will close the container

Now you want this file to send it to your friend; you can do this by ~~committing message as container as an~~ using image.

5: docker start container_id

6. docker exec -it container_id bash //names.txt is available in it.

Date

. Now create image of it containing names.txt using commit message.

version/optional
image_name:↑

1. docker commit -m "added names.txt file" container_id ↑

e.g: docker commit -m "added names.txt file" 099... names_updated

To show only the ids of images:

command: docker images -q

To remove all images all at once

command: docker rmi $(docker images -q)

if forced: command:

docker rmi $(docker images -q) -f

Note: It can't delete image; if that image is run by live container; first, you have to stop that container.

Date

Layers:

Images are built in layers. Each layer is an immutable file, but is a collection of files and directories. The last layer can be used to write out data to.

To List hash value of docker images:

command: docker images -q --no-trunc ~~--no-trunc~~

Docker file

Describes steps to create a Docker image. It's like a recipe with all ingredients and steps necessary in making your dish. This file can be used to create Docker image. These images can be pulled to create containers in any environment. These images can also be store online at docker hubs. When you run docker image you get docker containers. The containers will have the application with all its dependencies.

Date

Creating image:

1. touch Dockerfile // it will create a file "Dockerfile"
2. vi Dockerfile // vi editor

3.

FROM ubuntu // "ubuntu" is the base image.
MAINTAINER kunal Kushwaha <kunal@gmail.com> // maintainer
RUN apt-get update // whilst the container is being created; it will run this command
CMD ["echo", "Hello World"] // After container is created, run this command.

Note: Congrats!, ~~Docker~~ now save the Dockerfile contents.

4. cat Dockerfile // Display contents of Dockerfile.

5. docker build -t image_name : version ← optional // it will // build docker image.

You have to push your image onto docker hub; for that you have to create account on docker hub.

on CMD

6. docker login
7. Give username & passwd.

To build the image on docker hub now

~~8. docker build~~ 8. docker build. // "." show the current path.

Date

in step 8, you can also give tag name for the image.

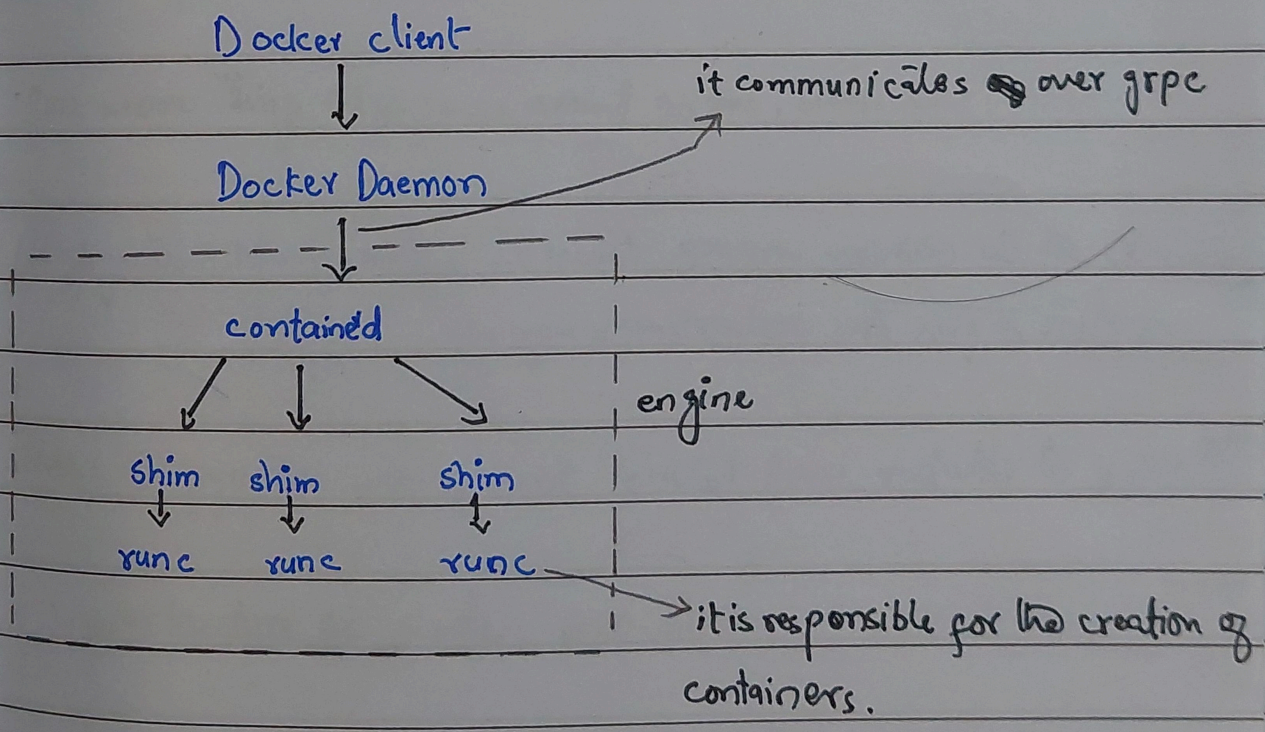
8. docker build -t image_name .

∴ Image has been created, you can check it by "docker images".

You can run the image by:

9. docker run image_name

Architecture of Docker engine.



Date

client $\xrightarrow{\text{HTTP (2375)}}$ server

For suitable connections, docker only allows connections that are secure. for example by TLS.

TLS: Transport Layer Security.