Docker

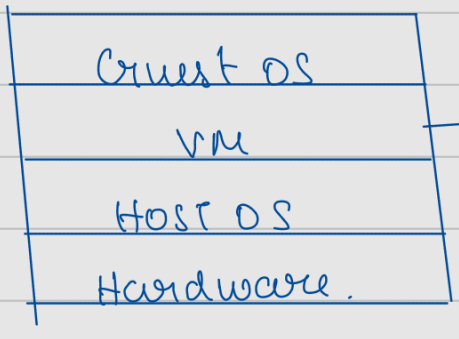
Docker: I'm solving this issue.

Docker is a container platform that allows you to build test and deploy applications quickly. A developer defines all the applications and it's dependencies in a Docker file which is then used to build Docker Image that defines a Docker container.

Doing this ensures that your application will run in any environment.

Before Docker

solution with virutalization,



(Hypervisor)

(Virutal Machine)

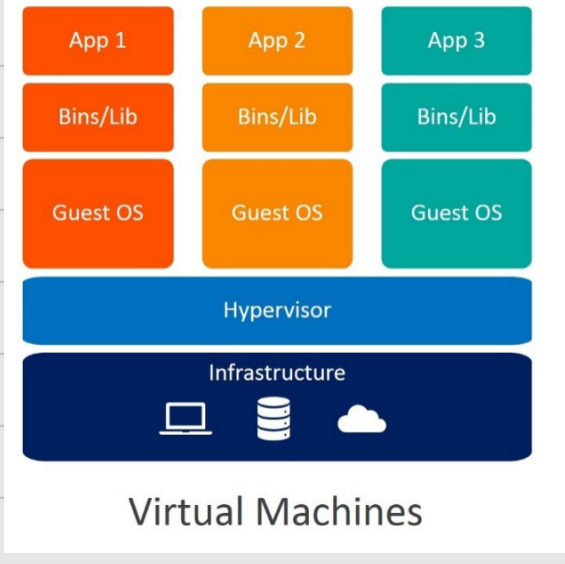
(where you're installing an OS on a virtual hardware)

with virtualization, you can run multiple OS on same system. So virtual machines needed their own operating system which tended to take up a lot of memory and storage.

Hypervisor, is used to create multiple machines on a host operating system and it manages virtual machines.

virtual machine have their own operating system and do not use the host's operating system.

Here, you given entire Guest OS to other teams to work on configuration setting as it depends on virtual hardware. (As image)



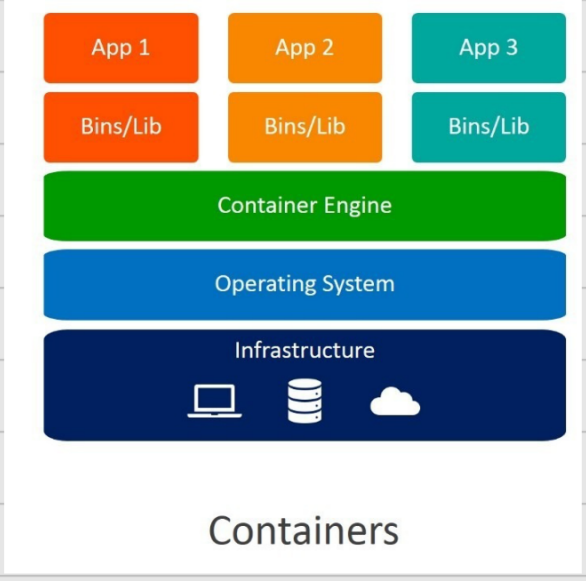
→ We're paying for different OS licences.

→ Wasting so much of RAM in running Guest OS

In earlier days, we used to run multiple apps on the same physical server. So it's risky if it's not isolated app | software.

Solution with containerization

A package of your software with all the configuration in a container and ship it to the other team System.



→ it's fast & safe and portable

In a software world, containerization is an efficient method for deploying application.

A container encapsulates an application with its own operating environment. It can be placed on any host machines without special configuration.

VM is hardware virtualization whereas containers is OS virtualization and it is light weight

Docker

Docker is a platform and a set of tools basically to achieve containerization.

→ portable
→ secure
→ scalable.

Docker Engine : It is responsible for the overall functioning of docker platform. It is a client-server based application

→ Creating containers
→ Managing containers

Docker Image (Immutable) : The file that contains source code, OS files to run the application along with its other dependencies inside the container is called Docker image

A containerized application is the running instances of a image.

Dockerfile: It contains the list of instructions to create Docker image.

Docker Hub: A container registry built for developers and open source contributions to find use and share their container image.

Command:

→ docker --version
→ docker run hello-world } keep Docker desktop open.
→ docker ps
↳ to see list container (running)
→ docker ps -a
↳ already ran container.

→ docker images (to see images)

More commands

→ docker help

→ docker rm <container ID> [to remove container]

→ docker rmi <image ID> [to remove image]

First remove container then image.

→ docker images (now verify)

→ docker create hello-world

[to create a container]

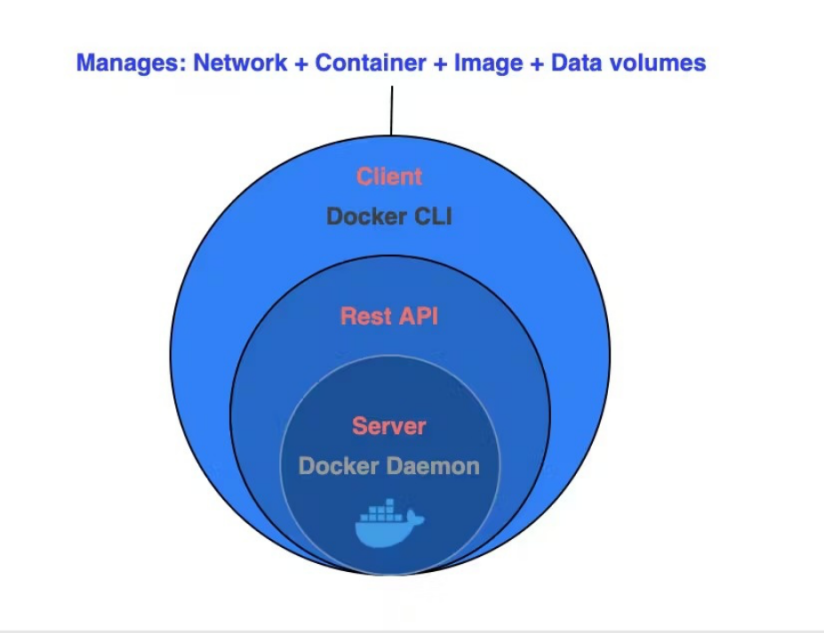
1. docker search hello-world [list of images]
2. docker pull hello-world [pull from registry]
3. docker create hello-world
4. docker start <container ID> [to run container]
5. docker stop <container ID> [to stop container]
6. docker pause <container ID>

↳ only in running state.

docker run -it <image>

↳ for interactive.

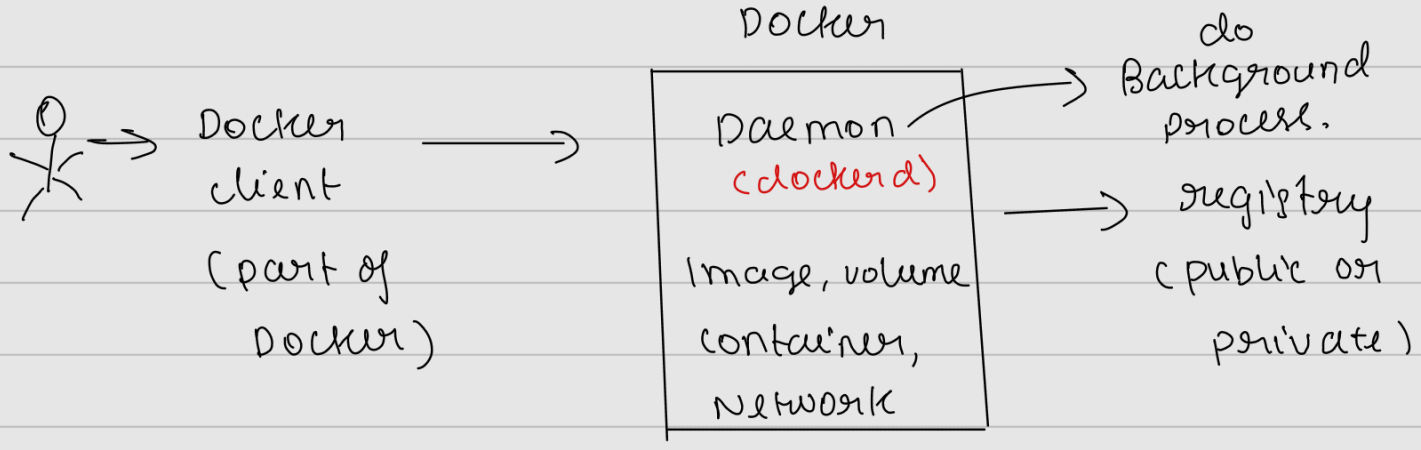
Docker Architecture



Docker Daemon → It is the heart of the docker architecture, which does the crucial work of building, running and distributing the containers

↓

It also manages the Docker Images and the Containers



Client - Server architecture

1. Server → (which runs the daemon)
2. Rest API → (deals with the interaction of application with their server)

3. client → Interface (CLI)

Docker Registries

↳ Docker images are stored. Dockerhub is the official online public repository of docker where you can find all the images of popular applications.

docker exec <name> ls a

↳ List all the folder of particular image.

Dockerfile

→ docker run -it openjdk:22-jdk

→ docker cp <jar file> < container name>
(in case java spring)

→ Create image.

docker commit --change='CMD ["java", "-jar", "..."]'
<container>

problem: repetative (tough)

Need to Automate: Dockerfile.

So, create a Dockerfile

FROM

ADD

ENTRYPOINT

docker build -t <name> .

-t → for tag

. → directory path.

Docker compose

Docker compose is a tool for running Multi-container applications on docker defined using the compose file format.

A compose file (YML) is used to define how one or more containers that make up your application are configured once you have a compose file, you can create & start your application with a

→ docker-compose up. --build.

Example:

one container for application and another container for database.

Docker itself have own network and storage and make you connect your network which is in

the computer and outside Network with docker network.

2 then two different containers and they're running in their own Network and need to communicate.

Running Multiple container

Configure the network in docker compose file.

→ docker network ls.

But you have to also make sure that both these images are running on the same network.

Docker volume

→ docker-compose down (to stop the multiple container)

Data is lost, when we reload the container again so, we need to store data for that we need volumes. (Storage on system)