Docker is an open platform for developing , shipping and running applications.

(or)

Docker is a platform which packages an application and all its dependencies together in the form of containers.

What problems docker solve ?

Let us say we want to install two versions of Java like , Java 8 and Java 11.

It does not allow us to set up both of them in the one OS due to version conflicts.

One possible solution is the virtualization of OS , like we can install multiple operating systems and install different versions on the each.

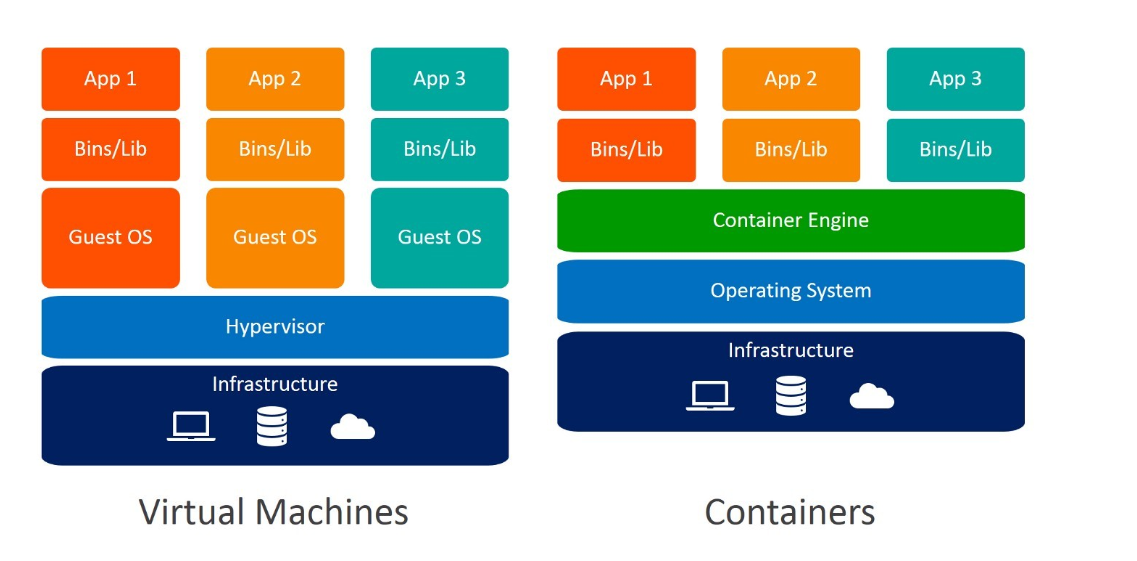
The problem with the above approach is that , each OS takes us exclusive resources like CPU , harddisk , memory which needs to be alloted/reserved for that. We need to partition the resources for them.

Docker basically uses the virtualization concept only.

We can have 2 containers one can have Java 8 and other can have Java 11.

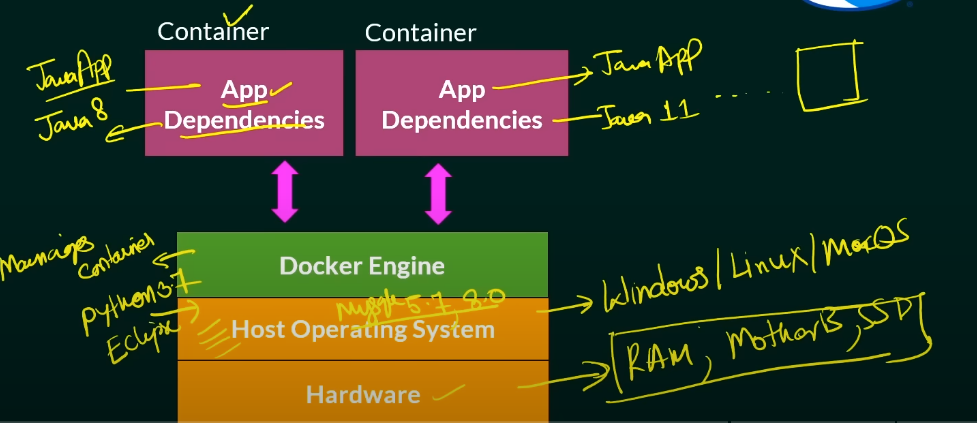
Docker engine manages our containers like providing necessarily amounts of resources like Memory , CPU etc.It uses the same underlying host operating system.

Virtualization vs Containerization



These virtual machines have their own operating system and do not use the host’s operating system. They have some space allocated.

A container encapsulates an application with its own operating environment. It can be placed on any host machine without special configuration, removing the issue of dependencies.



DockerFile

It is a text document which contains all the comands that an user can call on the command line to assemble an image.

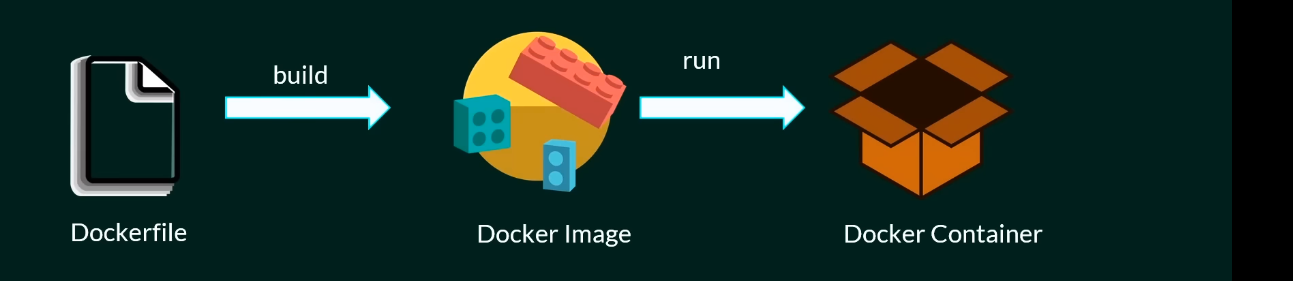
Like what is the parent image to be used like Java / Ubuntu etc.

Docker Image

It is a template to create docker container.

Docker Container

A container is a runnable instance of an image. This is where your application is running.



Docker Architecture

Docker uses a client-server architecture. The Docker client talks to the Docker daemon.

Docker daemon

It listens to the API requests being made through the Docker client and manages Docker objects such as images, containers, networks, and volumes.

Docker client

This is what you use to interact with Docker. When you run a command using docker, the client sends the command to the daemon, which carries them out. The Docker client can communicate with more than one daemon.

Docker registries

This is where Docker images are stored. Docker Hub is a public registry that anyone can use. When you pull an image, Docker by default looks for it in the public registry and saves the image on your local system.

You can also store images on your local machine or push them to the public registry. <https://hub.docker.com/>

Useful Commands

PS C:\Users\SESH\Desktop> **docker -v**

Docker version 24.0.5, build ced0996

PS C:\Users\SESH\Desktop> **docker pull hello-world**

Using default tag: latest

latest: Pulling from library/hello-world

719385e32844: Pull complete

Digest: sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c

Status: Downloaded newer image for hello-world:latest

docker.io/library/hello-world:latest

PS C:\Users\SESH\Desktop> **docker images**

REPOSITORY TAG IMAGE ID CREATED SIZE

hello-world latest 9c7a54a9a43c 3 months ago 13.3kB

PS C:\Users\SESH\Desktop> **docker search openjdk**

NAME DESCRIPTION STARS OFFICIAL AUTOMATED

openjdk Pre-release / non-production builds of OpenJ… 3708 [OK]

eclipse-temurin Official Images for OpenJDK binaries built b… 398 [OK]

sapmachine Official SapMachine Docker Image, SAP's buil… 45 [OK]

ibm-semeru-runtimes IBM Semeru Runtimes Official Images for Open… 31 [OK]

circleci/openjdk CircleCI images for OpenJDK 11 [OK]

cimg/openjdk The CircleCI OpenJDK (Java) Docker Convenien… 7

adoptopenjdk/openjdk11 Docker Images for OpenJDK Version 11 binarie… 222

adoptopenjdk/openjdk8 Docker Images for OpenJDK Version 8 binaries… 135

adoptopenjdk/openjdk16 Docker Images for OpenJDK Version 16 binarie… 18

adoptopenjdk/openjdk14 Docker Images for OpenJDK Version 14 binarie… 11

adoptopenjdk/openjdk15 Docker Images for OpenJDK Version 15 binarie… 10

adoptopenjdk/openjdk13 Docker Images for OpenJDK Version 13 binarie… 12

adoptopenjdk/openjdk12 Docker Images for OpenJDK Version 12 binarie… 17

adoptopenjdk/openjdk11-openj9 Docker Images for Eclipse OpenJ9 Version 11 … 56

adoptopenjdk/openjdk8-openj9 Docker Images for Eclipse OpenJ9 Version 8 b… 53

adoptopenjdk/openjdk10 Docker Images for OpenJDK Version 10 binarie… 3

adoptopenjdk/openjdk9 Docker Images for OpenJDK Version 9 binaries… 3

adoptopenjdk/openjdk15-openj9 Docker Images for Eclipse OpenJ9 Version 15 … 2

adoptopenjdk/openjdk16-openj9 Docker Images for Eclipse OpenJ9 Version 16 … 2

adoptopenjdk/openjdk14-openj9 Docker Images for Eclipse OpenJ9 Version 14 … 1

adoptopenjdk/openjdk13-openj9 Docker Images for Eclipse OpenJ9 Version 13 … 2

clearlinux/openjdk OpenJDK implementation of the Java Platform … 3

adoptopenjdk/openjdk12-openj9 Docker Images for Eclipse OpenJ9 Version 12 … 4

noenv/openjdk OpenJDK Docker Image 1

adoptopenjdk/openjdk9-openj9 Docker Images for Eclipse OpenJ9 Version 9 b… 3

PS C:\Users\SESH\Desktop> **docker run hello-world**

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.

2. The Docker daemon pulled the "hello-world" image from the Docker Hub.

(amd64)

3. The Docker daemon created a new container from that image which runs the

executable that produces the output you are currently reading.

4. The Docker daemon streamed that output to the Docker client, which sent it

to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:

https://hub.docker.com/

For more examples and ideas, visit:

<https://docs.docker.com/get-started/>

**docker ps**

This will list down all running containers only.

PS C:\Users\SESH\Desktop> **docker ps -a**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d88340f86bfc hello-world "/hello" 2 minutes ago Exited (0) 30 seconds ago great\_elion

This command lists all containers, including stopped ones.

PS C:\Users\SESH\Desktop> **docker run --name javaContainer -d openjdk:8**

f7524d3400ed64f66c45681a4749e0b103a83ea56041b979b440b0e4806fcce2

The -d flag in the docker run -d openjdk:8 command stands for "detached mode." It is used to run the container in the background, allowing you to continue using the terminal or shell from which the command was executed without being attached to the container's console output

When you run a container in detached mode, you will receive a container ID as output, indicating that the container has started running in the background.

PS C:\Users\SESH\Desktop> **docker run --name javaContainer -it -d openjdk:8**

550ab81e60cd2ff35acc53bfd77a30056980e09df0c599a5aba5d6d47da98fb8

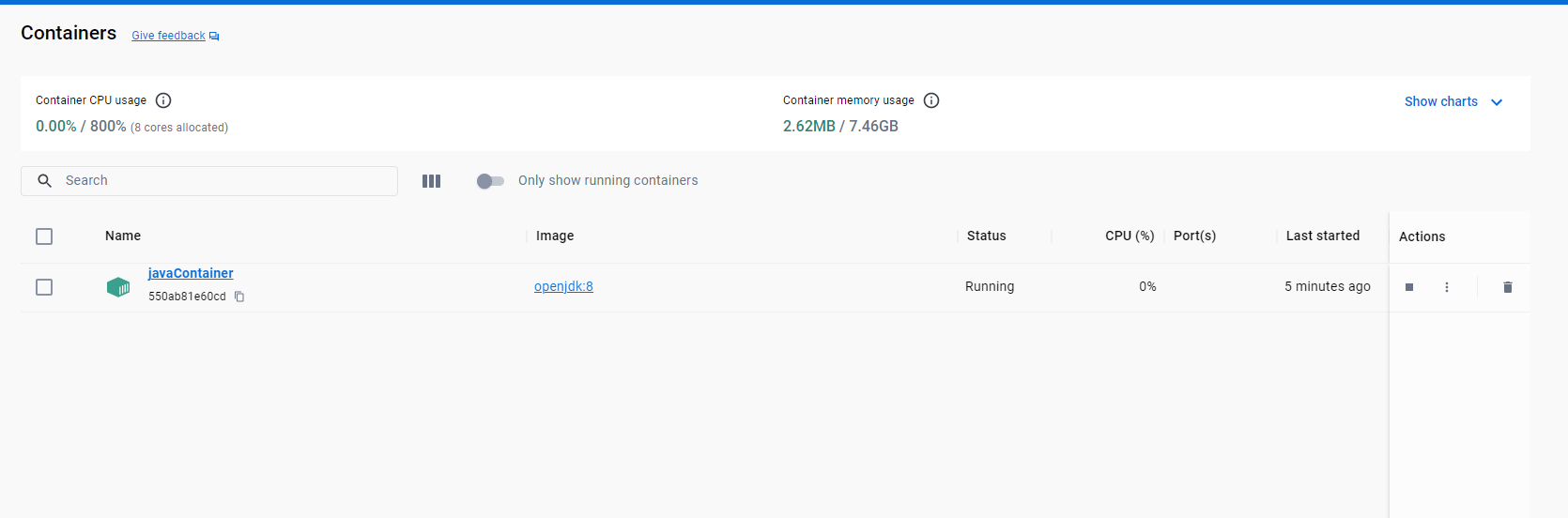
PS C:\Users\SESH\Desktop> docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

550ab81e60cd openjdk:8 "bash" 20 seconds ago **Up** 19 seconds javaContainer

The -it command in the docker run --name javaContainer -it -d openjdk:8 command stands for "interactive mode with a TTY." It is used to run the container in the foreground and attach a TTY (teletype)

to the container's console output, allowing you to interact with the container's console output from the terminal or shell from which the command was executed.



PS C:\Users\SESH\Desktop> **docker exec -it javaContainer bash**

root@550ab81e60cd:/# java -v

Unrecognized option: -v

Error: Could not create the Java Virtual Machine.

Error: A fatal exception has occurred. Program will exit.

root@550ab81e60cd:/# java -version

openjdk version "1.8.0\_342"

OpenJDK Runtime Environment (build 1.8.0\_342-b07)

OpenJDK 64-Bit Server VM (build 25.342-b07, mixed mode)

root@550ab81e60cd:/#

The above command allows you to enter into an interactive shell session inside a running container named javaContainer that was started with the OpenJDK:8 image. The -it flag is used to start an interactive session with a TTY, while the bash command is used to start a Bash shell inside the container.

To stop a running container

PS C:\Users\SESH\Desktop> **docker stop javaContainer**

javaContainer

PS C:\Users\SESH\Desktop> docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

550ab81e60cd openjdk:8 "bash" 31 minutes ago Exited (137) 30 seconds ago javaContainer

PS C:\Users\SESH\Desktop> **docker rm javaContainer**

javaContainer

PS C:\Users\SESH\Desktop> docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

PS C:\Users\SESH\Desktop> docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

hello-world latest 9c7a54a9a43c 3 months ago 13.3kB

openjdk 8 b273004037cc 12 months ago 526MB

PS C:\Users\SESH\Desktop> **docker rmi hello-world**

Untagged: hello-world:latest

Untagged: hello-world@sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c

Deleted: sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f9e016fdc3384500c2823d

Deleted: sha256:01bb4fce3eb1b56b05adf99504dafd31907a5aadac736e36b27595c8b92f07f1

PS C:\Users\SESH\Desktop> **docker rmi openjdk**

Error response from daemon: No such image: openjdk:latest

PS C:\Users\SESH\Desktop> docker rmi openjdk:8

Untagged: openjdk:8

Untagged: openjdk@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8

Deleted: sha256:b273004037cc3af245d8e08cfbfa672b93ee7dcb289736c82d0b58936fb71702

Deleted: sha256:db8b8ac27f8bb637f587d8dbbb61c5eb80e71f6f598041d573eff4f44295f1e6

Deleted: sha256:d4a32253866810ba357e383be1df02371bca6f91ebd4c96ae70d2777395ac41e

Deleted: sha256:e7dd29f3ffa73e8d8d9ccb88f1507bb35fe14749d2f5a7c1a44b33ea62a13a1b

Deleted: sha256:1376fe23991c7bd9ac29c2469f6489e5e68b2311f78191e87c47acd67e846372

Deleted: sha256:935ab298b59cf4955c8a62f40960766ceedee432fde87f22a71d557be7e05d0a

Deleted: sha256:6fa094ba2e6e15e0fab64e7d1372945f05e70ed3bdf6fd90409153d7ec19d160

Deleted: sha256:9c742cd6c7a5752ee36be8ecb14be45c0885e10e6dd34f26a9ae3eb096c5d492

PS C:\Users\SESH\Desktop\sde\_preparation\Spring Interview\Docker\java-docker> **docker build -t myjavaimage .**

[+] Building 33.6s (9/9) FINISHED docker:default

=> [internal] load build definition from Dockerfile 0.1s

=> => transferring dockerfile: 214B 0.0s

=> [internal] load .dockerignore 0.1s

=> => transferring context: 2B 0.0s

=> [internal] load metadata for docker.io/library/openjdk:8 4.7s

=> [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8 24.5s

=> => resolve docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8 0.0s

=> => sha256:2068746827ec1b043b571e4788693eab7e9b2a95301176512791f8c317a2816a 10.88MB / 10.88MB 1.3s

=> => sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8 1.04kB / 1.04kB 0.0s

=> => sha256:001c52e26ad57e3b25b439ee0052f6692e5c0f2d5d982a00a8819ace5e521452 55.00MB / 55.00MB 7.5s

=> => sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165 5.16MB / 5.16MB 2.3s

=> => sha256:3af2ac94130765b73fc8f1b42ffc04f77996ed8210c297fcfa28ca880ff0a217 1.79kB / 1.79kB 0.0s

=> => sha256:b273004037cc3af245d8e08cfbfa672b93ee7dcb289736c82d0b58936fb71702 7.81kB / 7.81kB 0.0s

=> => sha256:9daef329d35093868ef75ac8b7c6eb407fa53abbcb3a264c218c2ec7bca716e6 54.58MB / 54.58MB 7.1s

=> => sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5 5.42MB / 5.42MB 3.5s

=> => sha256:52a8c426d30b691c4f7e8c4b438901ddeb82ff80d4540d5bbd49986376d85cc9 210B / 210B 3.8s

=> => sha256:8754a66e005039a091c5ad0319f055be393c7123717b1f6fee8647c338ff3ceb 105.92MB / 105.92MB 13.3s

=> => extracting sha256:001c52e26ad57e3b25b439ee0052f6692e5c0f2d5d982a00a8819ace5e521452 5.8s

=> => extracting sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165 0.8s

=> => extracting sha256:2068746827ec1b043b571e4788693eab7e9b2a95301176512791f8c317a2816a 0.4s

=> => extracting sha256:9daef329d35093868ef75ac8b7c6eb407fa53abbcb3a264c218c2ec7bca716e6 5.3s

=> => extracting sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5 0.4s

=> => extracting sha256:52a8c426d30b691c4f7e8c4b438901ddeb82ff80d4540d5bbd49986376d85cc9 0.0s

=> => extracting sha256:8754a66e005039a091c5ad0319f055be393c7123717b1f6fee8647c338ff3ceb 3.1s

=> [internal] load build context 0.0s

=> => transferring context: 424B 0.0s

=> [2/4] WORKDIR /usr/src/myapp 2.9s

=> [3/4] COPY ./src . 0.1s

=> [4/4] RUN javac Test.java 1.1s

=> exporting to image 0.1s

=> => exporting layers 0.1s

=> => writing image sha256:de8733800e42095c0f7ac1877d7b36e6046e0f2f27e24b5338d5669d96c711c4 0.0s

=> => naming to docker.io/library/myjavaimage