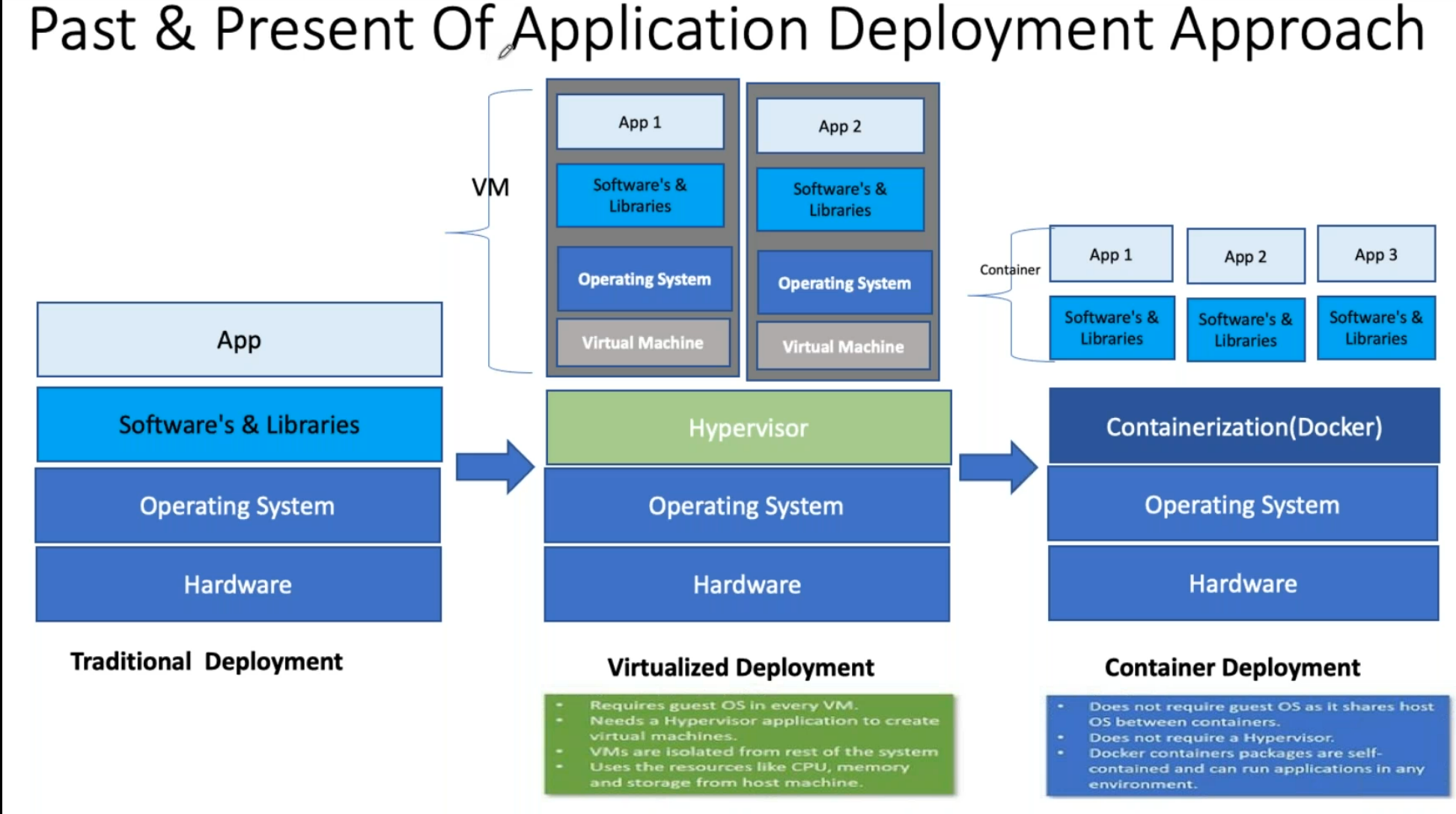
**Docker**

Docker is a software development platform for virtualization with multiple Operating systems running on the same

host. It helps to separate infrastructure and applications to deliver software quickly. Unlike Hypervisors, which are

used for creating VM (Virtual machines), virtualization in Docker is performed on system-level, also called Docker

containers.



Docker Engine

Docker is the client-server type of application which means we have clients who relay to the server. So, the Docker daemon called: dockerd is the Docker engine which represents the server. The docker daemon and the clients can berun on the same or remote host, and they communicate through command line client binary, as well as a full RESTful API to interact with the daemon: dockerd.

Docker Images

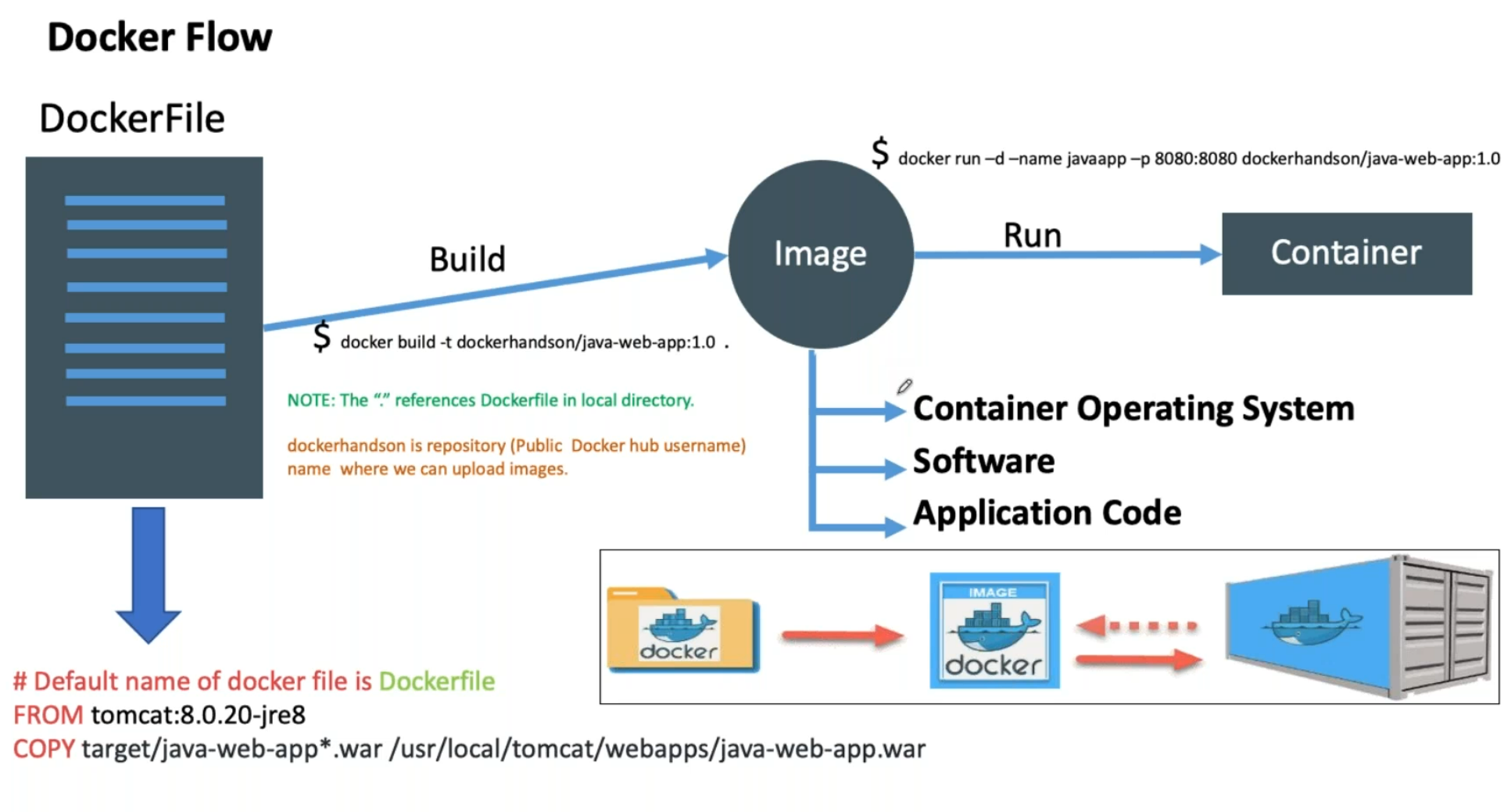
Docker images are the “source code” for our containers; we use them to build containers. They can have software pre-installed data which speeds up deployment. They are portable, and we can use existing images or build our own Image by using Docker file(Joker file is a text file which contains some instructions to create image).

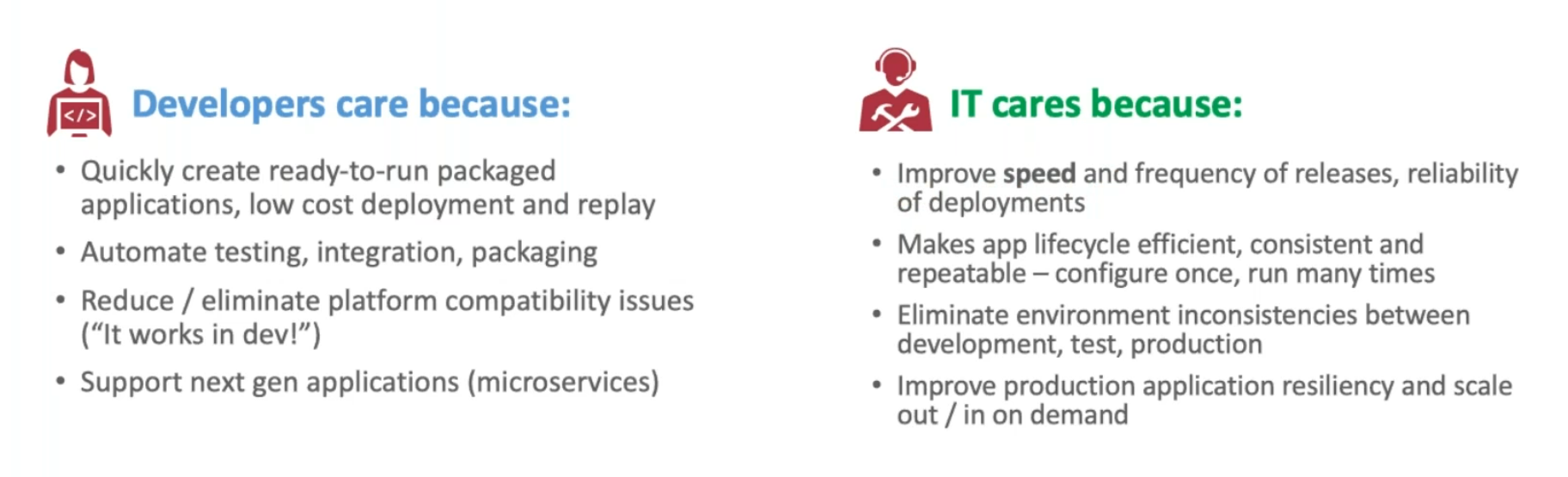
Docker Registries

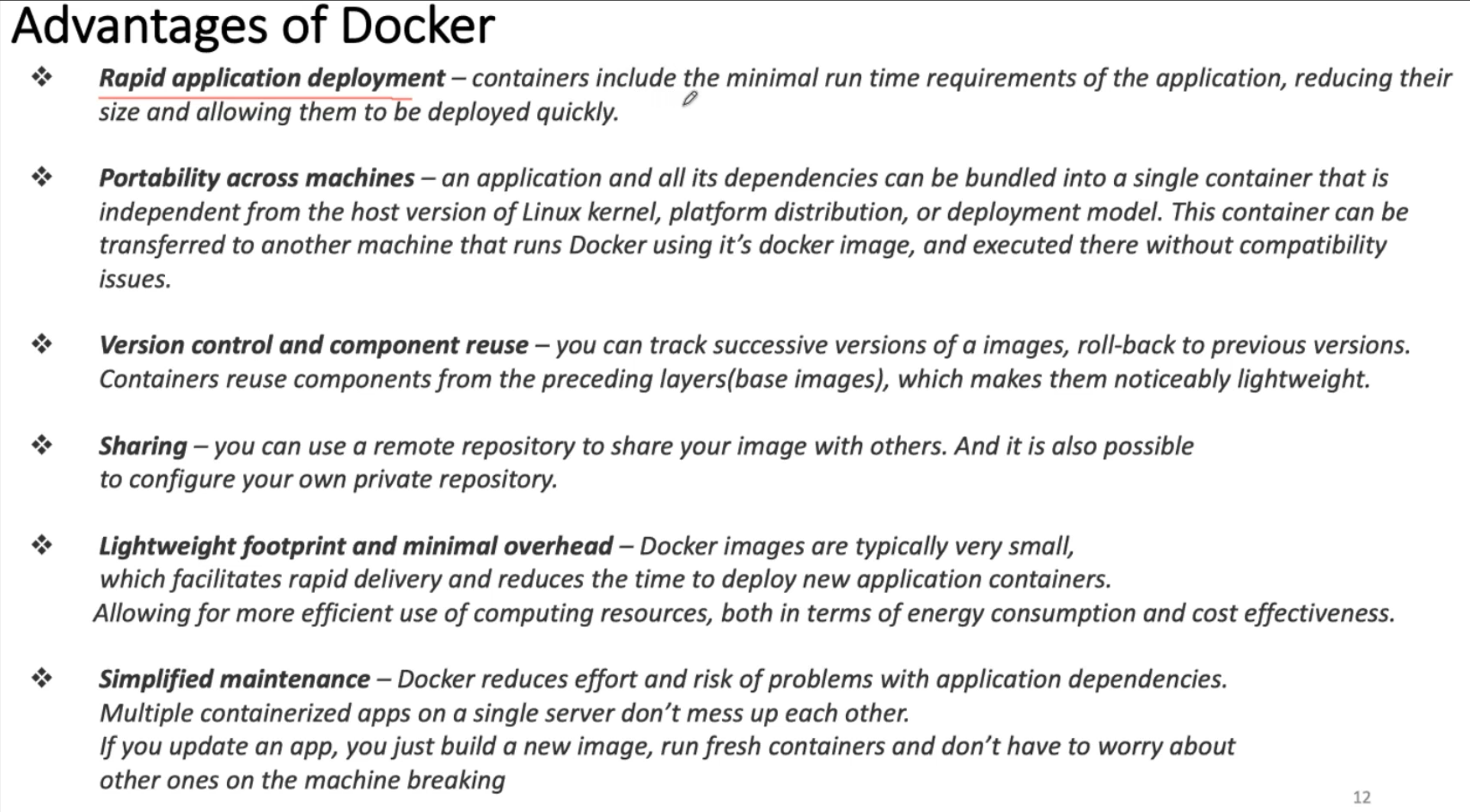
Docker stores the images we build in registries. There are public and private registries. Docker company has public registry called Docker hub, where you can also store images privately. Docker hub has millions of images, which you can start using now.

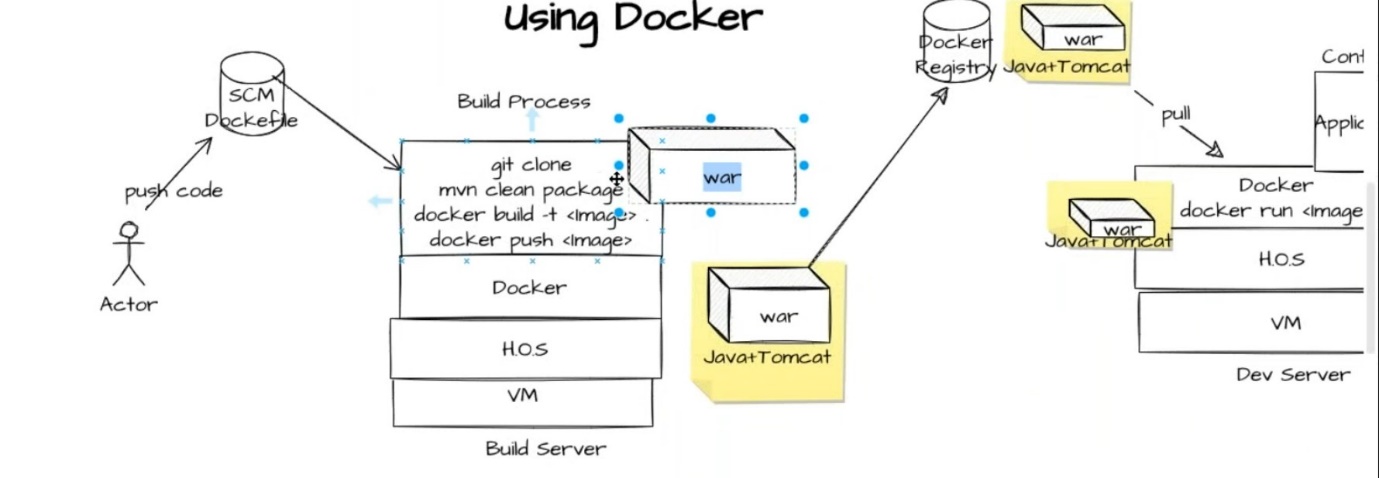
Docker Containers

Containers are the organizational units and one of the Docker basics concepts. When we build an image and start running it; we are running in a container. The container analogy is used because of the portability of the software we have running in our container. We can move it, in other words, “ship” the software, modify, manage, create, or get rid of it, destroy it, just as cargo ships can do with real containers.In simple terms, an image is a template, and a container is a copy of that template. You can have multiple containers(copies) of the same image.









Take an ubuntu server in aws

­----------------------­-- ------------------ -------------------

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

------------------------ ------------------------ ---------------------- -----------

sudo docker info

sudo service docker start

sudo service docker status

sudo usermod -aG docker ec2-user (giving permissions for the docker or adding user to the group)

(or)

sudo usermod -aG docker $user

exit and login again into the server

git --version (check the git version)

git clone <https://github.com/rajnaveen788/maven-web-application.git>

cd maven-web-application

sudo apt install maven -y

mvn clean package (.war file contains only compiled source code )

vi Dockerfile (create one docker file)

--------------------- -------------- ---------- ------------

# Use the official Tomcat latest base image

FROM tomcat:latest

# Copy the WAR file from the build context to the webapps directory

COPY /home/ubuntu/maven-web-application/target/maven-web-application.war /usr/local/tomcat/webapps/maven-web-application.war

---------------- -------------------- ------------- -----------

Syntax:

docker build -t <Image-name> <build-context>

or

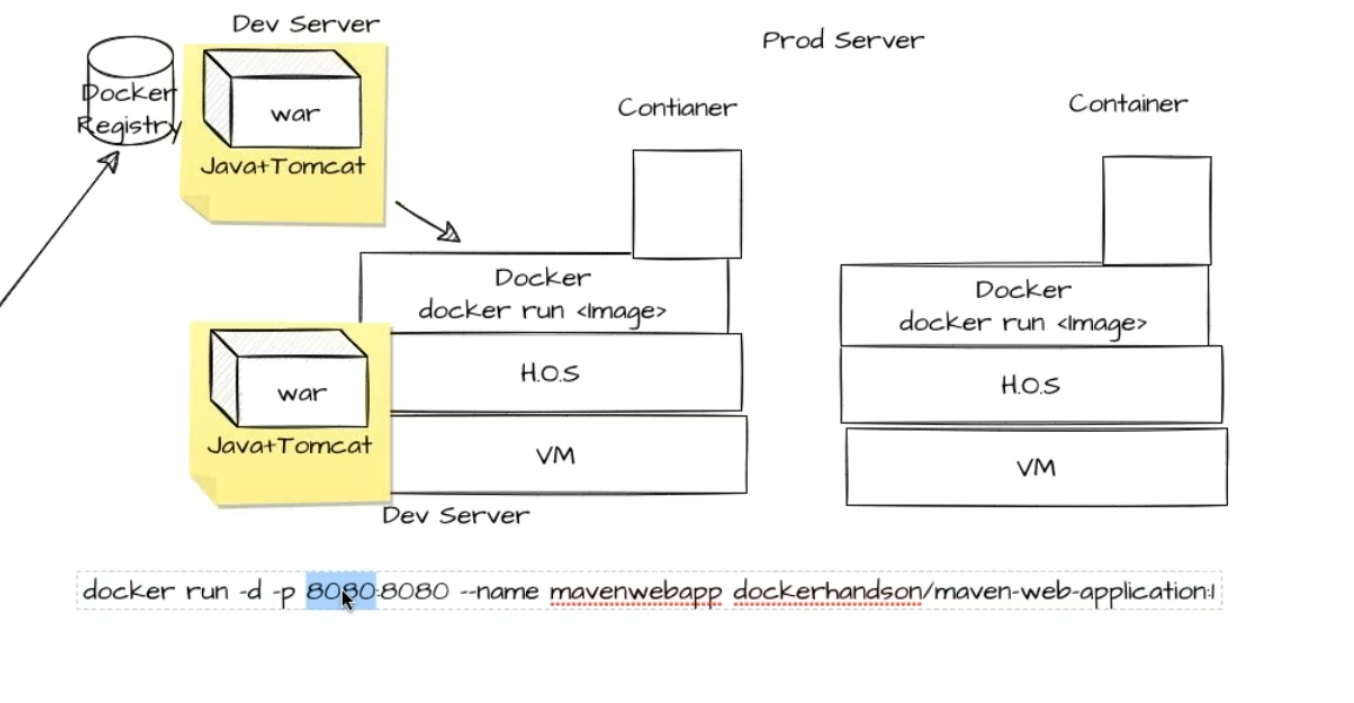
docker build -t naveen001bond/maven-web-application .

Login to push the image to the registory

docker login -u naveen001bond

password:\*\*\*\*\*\*\*

$ docker push naveen001bond/maven-web-application



Build docker image once deploy any-where

---------------- -------------------- ---------------------- -

Create another Docker engine in new ec2-server

docker will create and start the container in detached mode -d with port mapping -p(host port: container port)

docker run -d -p 8080:8080 --name maven-web-app-1 naveen001bond/maven-web-application

docker ps

docker ps -a

punlic-ip:8080/maven-web-application (search in google)

docker inspect \*\*\*\*\*\*(container name) to see complete details of container

docker rmi -f \*\*\*\*\*(image name or ID)

docker create --name<container name> -p <host port:container port> <image ID /name>

docker start \*\*\*\*\*(container-ID/name)

docker stop \*\*\*\*\*\*(container-Id/name)

docker restart \*\*\*\*\*(container-ID/name)

docker rename <oldname> <newname>

docker rm <container>

docker rm -rf <container>

docker images prune ---🡪 To delete all Images at single Time

**Dockerfile**

FROM 🡪Using FROM instructions we can define the base image. On top of that base image we can execute some instructions to create Our own image.

Syntax: FROM tomcat:latest

MAINTAINER 🡪 using maintainer we can define who is the maintainer in the brackets author of Docker file image .

Syntax: MAINTAINER <Name & Email>

COPY 🡪Using COPY we can copy files/folders from local build context ( from server where we are creating image) to the image.

Syntax: COPY <source> <Destination>

ADD 🡪Using ADD also we can copy files to the image. ADD can copy local files and files from remote locations or repository (http/https endpoints URLs).

Syntax: ADD <URL> <Destination>

By using RUN, CMD, ENTRYPOINT we can execute commands

RUN 🡪Using RUN command we can run some commands while creating image

Ex: RUN mkdir tomcat

RUN wget <URL> (RUN Can be used to install the packages)

Note: All the instructions can be executed from top to bottom order

CMD 🡪Cmd is used to start the process(application) inside a container While starting the container

Note: Catalina.sh will run the CMD command in background

Even though if we have more than one CMD in a docker file. But Docker will promises only the recent or latest CMD

WORKDIR

ENV

EXPOSE