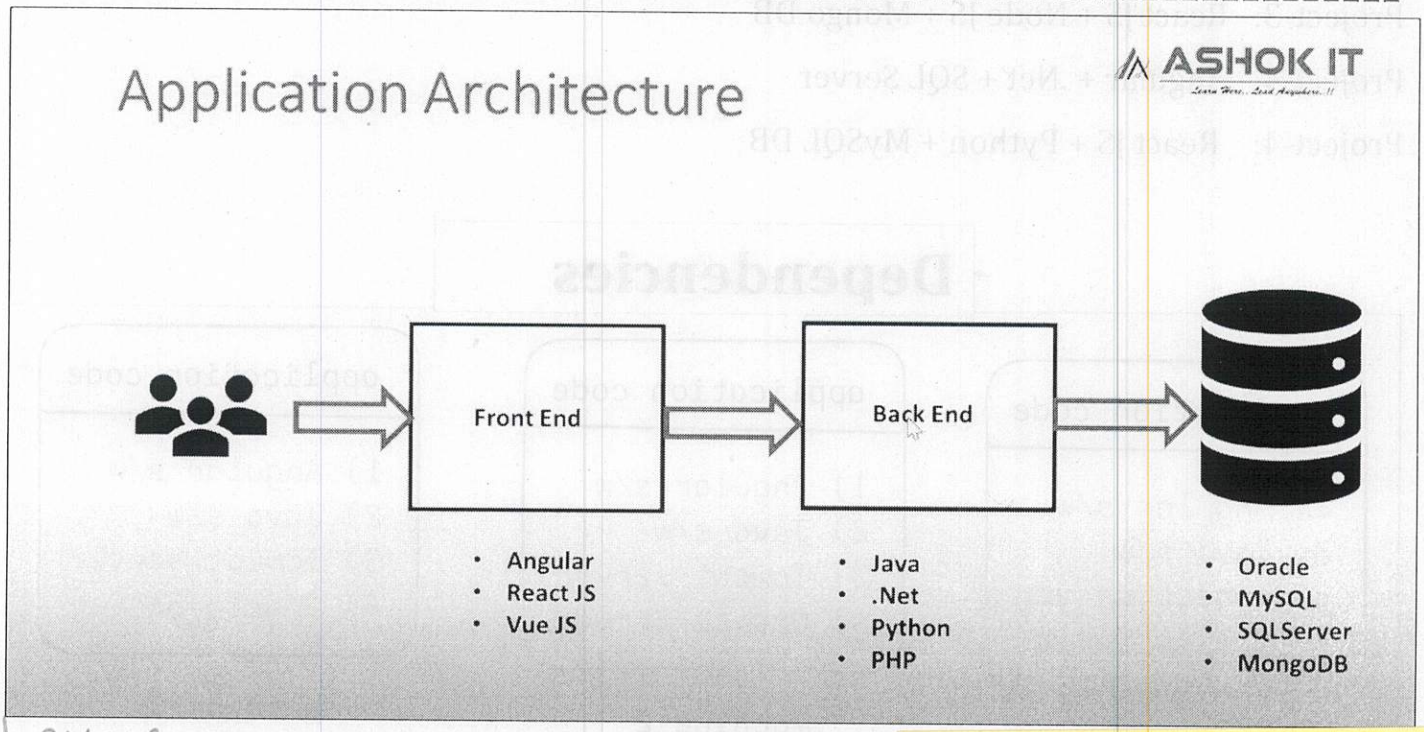
Docker Hub / Nexus / JFrog / AWS ECR
└ To Store docker Image

Docker (4)

01-28-Docker-11-Nov-24

Application Architecture



Layers —

1) Frontend : User Interface (UI)

2) Backend : Business logic

3) Database : Storage

docker pull hello-world
docker Image
↳ Artifact.

Tech Stack of Application

Frontend : Angular 16v

Backend : Java 17v

Database : MySQL DB Server 8.5

Webserver : Tomcat 9.0

Docker Image is a Artifact.

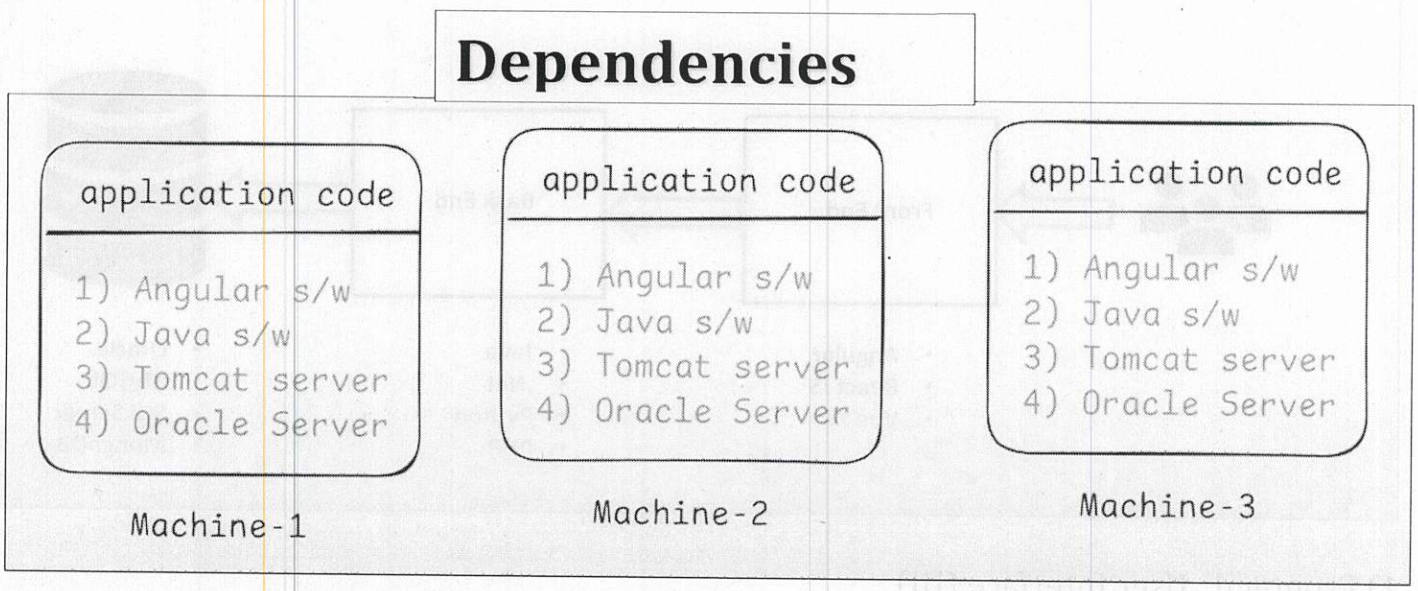
Project-1: Angular + Java + Oracle DB

Project-2: React JS + Java + MySQL DB

Project-3: React JS + Node JS + Mongo DB

Project-4: Angular + .Net + SQL Server

Project-4: React JS + Python + MySQL DB



Note: If we want to run our application code, then we need to setup all required dependencies in the machine.

Note: dependencies nothing but the softwares which are required to run our application.

Ex: java 17 + Angular 16 + MySQL 8.5 + Tomcat server 9.0

Note: If we want to run same application in 100 machines then it is hectic task to setup dependencies and there is a chance of human mistakes.

=> To overcome above problem we will use Docker tool.

If we want to run application in any machine then we have to install all the required software's with proper version compatibility.

Ex: Angular 17v, Java 11v, Tomcat 9.0v, MySQL 8.5v
Note: If we do any mistake in s/w installations then application can't execute.

Software Installation.

What is Docker ? Containerization

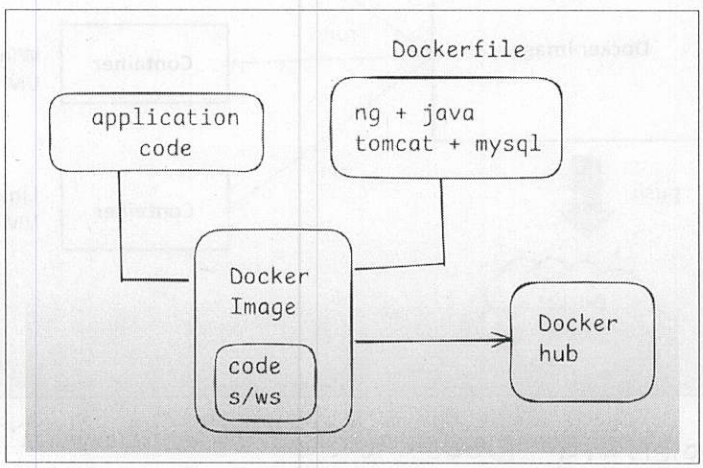
- Docker is a free & open source software.
- Docker is used for containerization.

Note: Containerization means packaging application code + application dependencies as single unit for execution.

- With the help of docker, we can run our application in any machine.
- Docker will take care of dependencies installation required for application execution.
- We can make our application portable using Docker.

Note: Docker is platform independent. We can use docker in windows, linux and mac also.

Docker Container = application code + application dependencies



Hey docker , in order to run our application we need angular,java,mysql installation what docker will do is it will take care of s/w installation and application code and it will create a docker image.

Once docker image is created that docker image we are going to store into one docker registry known as Docker Hub.

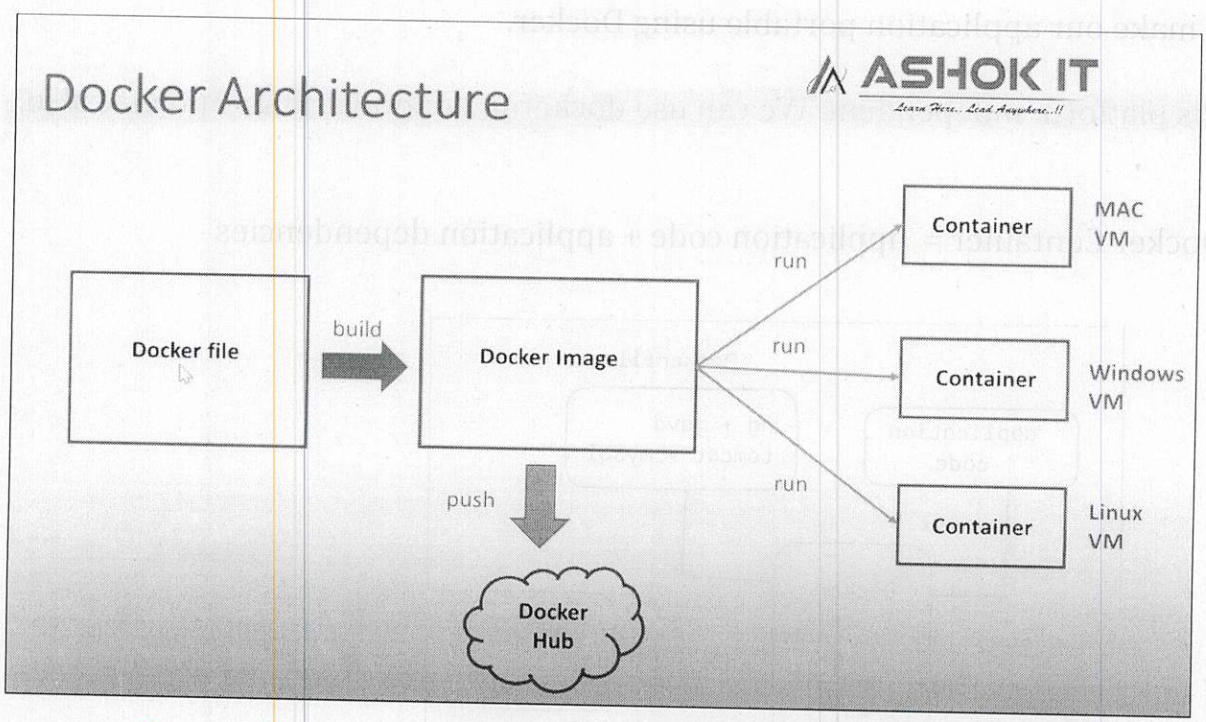
Docker Hub used to store docker images.
Once docker image is stored in docker hub we can download that image from docker hub and we can run that docker image.

After running docker image docker container will be created.

Advantage Of Docker

- Application can be run on any machine without bothering about which software is available on which machine we no need to check that.
- We can run application on any machine if the docker file is available in that machine.

==========



- Container Are Nothing but the Virtual Machines.

==========

Docker Architecture

==========

1) Docker file
2) Docker Image
3) Docker Registry /nexus /jfrog / AWS ECR
4) Docker Container

1) Docker file

- Docker file Contains Set of instructions to build docker image.
- Dockerfile is used to specify where is app code and what dependencies are required for our application execution.

Note: Using dockerfile we will build docker image.

- Developer and Devops engineer both are write docker file.
- Docker file is a text file , will specify what softwares required for application
- Docker image will download that softwares and keep that.

2) Docker Image (Public / Private)

- Docker image is a package which contains app code and app dependencies.
- Docker Registry is used to store docker images.
- Docker images are public and private , it depends on us.
- We cant see the docker image .
- Docker image is called as artifact that is store in nexus / Jfrog.

==========

Docker Container

==========

- Note: When we run docker image then docker container will be created.
- Every Docker container is a linux virtual machine.
- Every different container should be mapped with different port number.
- Docker Conatiner is a Linux machine.
- We can run docker image in any machine , we no need to install softwares.

Download docker s/w → Download docker image → run docker image in any machine → Application will run in any machine. This process is called containerization.

We are packaging our code and dependencies for easy execution.

- Inside Docker Container our application will be executed.
- Inside the docker container our application wil be executed.

-we Can't Delete running Containers , For that we first Need to Stop the Container .

==========

Install Docker in Linux VM

==========

Step-1 : Create EC2 VM (amazon linux) & connect with that vm using ssh client

Step-2 : Execute below commands

```
# Install Docker
sudo yum update -y
sudo yum install docker -y
sudo service docker start

# Add ec2-user user to docker group
sudo usermod -aG docker ec2-user

# Exit from terminal and Connect again
exit

# Verify Docker installation
docker -v
```

==========

Create account in Docker Hub
Url - https://hub.docker.com/

And download Docker hub.

Ashokit Docker repositories
Url-https://hub.docker.com/search?q=ashokit

Download repo from ashokit
url -docker pull ashokit/spring-boot-rest-api

Always we are going to run docker container in detached mode to execute other commands in terminal otherwise terminal will be blocked.

Command – docker rum –d < Image Id >

Docker System prune -a
↳ Delete all Stopped Containers.
+ unused Images

03-28-Docker-13-Nov-24

==========

Docker commands

==========

docker images : To display docker images available in our system.

- Default docker image is Hello-world
- Command – docker pull hello-world

docker pull <image-id/name> : To download docker image from docker hub.

docker rmi <image-id/name> : To delete docker image.

- If the container is running we cant delete images directly first we need to delete container then we can delete the images.

docker run <image-id/name> : TO create/run docker container.

docker ps : To display running docker containers.

docker ps -a : To display running + stopped containers.

docker stop <container-id> : To stop running docker container.

docker start <container-id> : To start docker container which is in stopped state.

docker rm <container-id> : To delete docker container.

- Only container will be deleted not the images.

delete stopped containers + unused images + build cache
docker system prune -a

docker build -t <tag-name> . : To build docker image , -t represent tag name

docker login : To login into docker hub account

docker push <img-name> : To push docker img into docker hub

docker logs < Container-id > : To see container logs.

Date: 13-Nov-2024 Topic : Docker Containers

==========

QUE - 1) What is detached mode ?

ANS - It is used to created in background. It will allow us to run commands in terminal.

Ex : docker run -d ashokit/spring-boot-rest-api

QUE -2) What is port mapping ?

- It is used to map container port number to host machine port number
- Once we perform port mapping then we can access our application which is running inside the container by using host machine public address.

Syntax : docker run -d -p <host-port:container-port> <image-name>

Port no Host id

Ex : docker run -d -p 9090:9090 ashokit/spring-boot-rest-api

first - 9090 → Host port
second - 9090 → Container port

Note: To access our application we need to enable HOST PoRT in security group inbound rules.

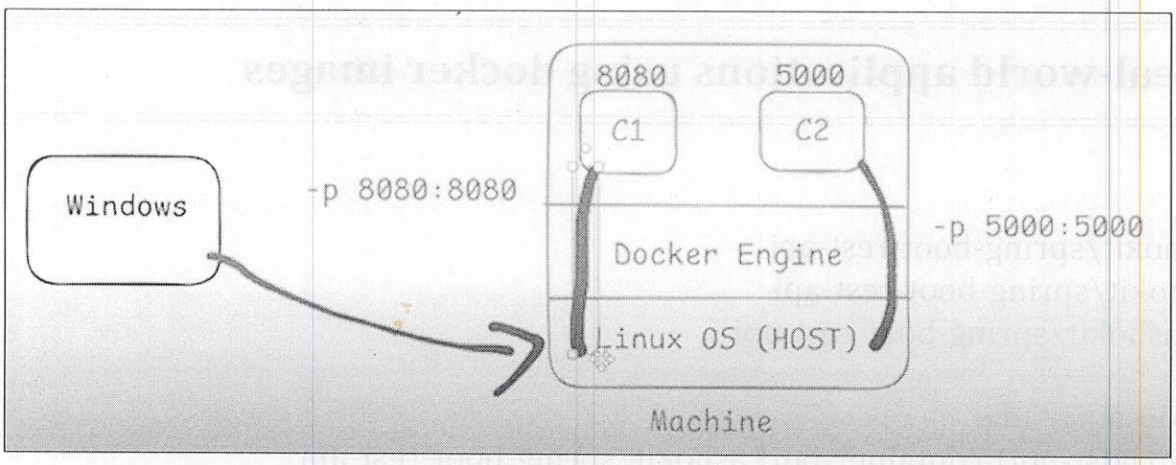
App URL : http://public-ip:host-port/welcome/raja

Note: If we are using docker in windows machine instead of public we can use localhost in the url.

Que - Why we do port mapping ?

Ans - ~~In Order to Access the Container which is Appli~~

Ans - In Order to Access the Application which is Running in docker Container, we do port Mapping.



- If you want to access the application which is running in the coantainer the you need to do port mapping .
- for that container port number we need to map with the host post number ,Then only then we can access the application.

Host is nothing but the machine in which we have installed the docker.

↳ We Can install Multiple Containers Inside Host Machine

Interview Question

QUE -Can run two containers on same port number ?

Ans -Yes , you can run two containers on same port number but when you map port number with host id then host id should be different otherwise we wont able to run two application on same port

```
[ec2-user@ip-172-31-32-178 ~]$ ll
total 8
-rw-r--r--. 1 ec2-user ec2-user 106 Nov 19 04:02 Dockerfile
-rwxrwxrwx. 1 ec2-user ec2-user 108 Nov 19 03:28 dockerimg
```

Issue – how to delete all the dockerimg files ?

Solution – rm <file-name>

==========

Running Real-world applications using docker images

==========

docker pull ashokit/spring-boot-rest-api
docker run ashokit/spring-boot-rest-api
docker run -d ashokit/spring-boot-rest-api
docker ps
docker logs <container-id>
docker run -d -p host-port:container-port ashokit/spring-boot-rest-api

Ex: docker run -d -p 9090:9090 ashokit/spring-boot-rest-api

########## Java App URL : http://public-ip:host-port/welcome/{name}

docker pull ashokit/python-flask-app
docker run -d ashokit/python-flask-app
docker run -d -p 5000:5000 ashokit/python-flask-app

########## Python App URL : http://public-ip:host-port/

Note: Here -d represents detached mode.
Note: Here -p represents port mapping. (host-port:container-port)

Note: host port and container port no need to be same.

Note: Host port number we need to enable in ec2-vm security group inbound rules to allow the traffic.

How to create a dockerfile and docker img ?

- command – docker build –t < img-name > . (Dot refers the current working directory in which our docker file is going to execute)
- You can specify other directories instead of . if your Dockerfile or build context is located elsewhere. For example:

- docker build -t image-name /path/to/context
  Here, /path/to/context is used as the build contex

- Depends on the Project either Developers will Create / Devops team

==========

Dockerfile

==========

=> Dockerfile contains set of instructions to build docker image.

file name : Dockerfile

(1) Application Dependencies
(2) Application Execution Process.

Note: We will keep Dockerfile inside project directory.

=> To write dockerfile we will use below keywords

1) FROM
2) MAINTAINER
3) RUN
4) CMD
5) COPY
6) ADD
7) WORKDIR
8) EXPOSE
9) ENTRYPOINT
10) USER

QUE -How to create a dockerfile ?

1. vi < filename>
2. Write script
3. docker build -t sachinm2460/demo1 .
- Sachinm2460/demo1 – is for to store docker image in docker-hub account .
4. $docker images
5. $docker login
6. $docker run sachinm2460/demo1
7. $docker push sachinm2460/demo1

==========

FROM

==========

- It is used specify base image to create our docker image.
- Which software is required to run your application that will specify in dockerfile.
- Base images we can find in docker hub only

Ex:

FROM tomcat:9.0

└) To Run Application Java Software is Required then will Specify Java Version

FROM openjdk:17

FROM python:3.3

FROM node:19

FROM mysql:8.5

==========

MAINTAINER ⟶ Optional

==========

=> To specify author of Dockerfile (who created/modifed Dockerfile)

Ex:

MAINTAINER Ashok<ashok.b@oracle.com>

Note: It is optional.

========
RUN Image Creation
========

=> RUN keyword is used to specify instructions (commands) which are required to execute at the time of docker image creation.

Ex:

RUN 'git clone <repo-url>' ——→ cloning Repository

RUN 'mvn clean package' ——→ Packaging application.

Note: We can specify multiple RUN instructions in Dockerfile and all those will execute in sequential manner.

========
CMD Container Creation.
========

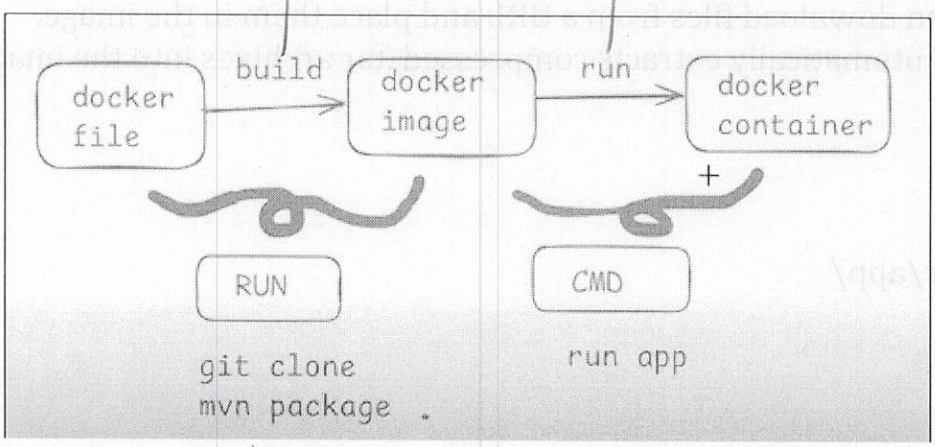
=> CMD keyword is used to specify instructions (commands) which are required to execute at the time of docker container creation.

Ex:

CMD "java -jar <jar-file-name>"

CMD "python app.py"

Note: If we write multiple CMD instructions in dockerfile, docker will execute only last CMD instruction.



CMD will Configure Application Execution Process.

- Copy war files from project target directory to tomcat Server webapps folder for Deployment.

=====
COPY
=====

One Location to Another Location.

=> COPY instruction is used to copy the files from source to destination.

Note: It is used to copy application code from host machine to container machine.

Source : HOST Machine

Destination : Container machine

EX: War File will be Executed from Container Machine.

COPY target/app.jar /usr/app/

COPY target/webapp.war /usr/app/

COPY app.py /usr/app/

- Copy a war file from our container machine to tomcat machine.
- Hey docker my project war file is available in the project target folder , you copy that file to the tomcat container which is available in the docker .

=====
ADD
=====

=> ADD instruction is used to copy the files from source to destination.

- **File copying:** Similar to COPY.
- **Remote URLs:** Can download files from a URL and place them in the image.
- **Tar extraction:** Automatically extracts compressed .tar archives into the image.

EX:

ADD target/app.jar /usr/app/

ADD <file-url> /usr/app/

========
WORKDIR — used to change the Path
========

=> WORKDIR instruction is used to set / change working directory in container machine.

while executing dockerfile

Ex:

```
COPY target/app.jar /usr/app/

WORKDIR /usr/app/

CMD "java -jar app.jar"
```

↳ Configure to run our Application.

========
EXPOSE
========

=> EXPOSE instruction is used to specify application is running on which PORT number.

Ex:

```
EXPOSE 8080
```

==========
ENTRYPOINT
==========

=> It is used to execute instruction when container is getting created.

Note: ENTRYPOINT is used as alternate for 'CMD' instructions.

```
CMD "java -jar app.jar"

ENTRYPOINT ["java", "jar", "app.jar"]
```

==========

What is the diff between 'CMD' & 'ENTRYPOINT' ?

==========

- CMD instructions we can override.
- ENTRYPOINT instructions we can't override.

==========

USER

==========

USER : It is used to specify with which user account we want to use to execute dockerfile instructions.

Ex:

```
USER devopsuser
```

==========

```
FROM ubuntu

MAINTAINER Ashok

RUN echo 'hi - run-1'

RUN echo 'hello - run-2'

CMD echo 'Hi - cmd1'

CMD echo 'hello - cmd2'
```

==========

docker build -t ashokit-img1 .
— t Represent tag name (Image Name)
— dot represents current working directory.

$ docker images

$ docker system prune -a

$ docker build -t <image-name> .

Ex : docker build -t <dockerhub-uname/demo1> .

$ docker images

$ docker run <image-name/image-id>

$ docker login —> Connect from our machine to docker hub Account.

Note: Need to enter docker hub account credentials.

$ docker push <dockerhub-uname/demo1>

==========

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Date: 15-Nov-2024
Topic : Writing Dockerfile
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

— Write Dockerfile for without Springboot Application?

==========

Dockerizing Java web application (without springboot)

==========

=> Java web applications (without springboot) will be packaged as war file.

=> To run war file we need a webserver like apache Tomcat.

=> Inside tomcat server "webapps" folder will be available to deploy war files.

Note: We need to copy project war file to tomcat server webapps folder for execution.

— To run war file we Need a Server (Tomcat Server)

Dockerfile

==========

- FROM tomcat:latest

  - I'm telling to the docker to download latest docker image of tomcat server to run my application.

- EXPOSE 8080

- COPY target/maven-web-app.war /usr/local/tomcat/webapps/

  - Copy a war file from our container machine to tomcat machine.
  - Yes, you can change the deployment directory for a .war file in Tomcat, but it requires some additional configuration.

==========

@@ Maven Web App Git Repo : https://github.com/ashokitschool/maven-web-app.git

Note: If you are using docker in linux vm then install git and maven softwares

$ sudo yum install git -y
$ sudo yum install maven -y

clone git repo
git clone https://github.com/ashokitschool/maven-web-app.git

Perform maven build
cd maven-web-app
mvn clean package

Note: After maven package it will generate war file inside target directory.

check targer directory content
ls -l target

create docker image
docker build -t app1 . ——→ Is the Image Name

Check docker images
docker images

Run container

docker run -d -p 8080:8080 app1

Note: Enable 8080 port number in security group inbound rules.

Access application in browser
URL : http://public-ip:8080/maven-web-app/

==========

Dockerizing Java Spring Boot Application

==========

- Springboot is a Java framework to Develop Java framework fasily.

=> Every SpringBoot application will be packaged as jar file only

=> To run spring boot application we need to execute jar file.
- java application without spring-boot will be packaged as war , war file requires tomcat
- java application with spring-boot will be packaged as jar , jar file we can run directly

Syntax : java -jar <jar-file-name>

Note: When we run springboot application jar file then springboot will start tomcat server with 8080 port number (embedded tomcat server).

========== **Dockerfile for Spring Boot Application** ==========

```
FROM openjdk:17

MAINTAINER Ashok

COPY target/app.jar /usr/app/

WORKDIR /usr/app/

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar"]
```

==========

Java Spring Boot App Git Repo : https://github.com/ashokitschool/spring-boot-docker-app.git

$ git clone https://github.com/ashokitschool/spring-boot-docker-app.git

$ cd spring-boot-docker-app

$ mvn clean package

$ ls -l target

$ docker build -t sb-app .

$ docker run -d -p 8080:8080 sb-app

Note: Once container created check logs of container

$ docker logs <container-id>

Note: Access our application using host-vm public and host port

URL : http://localhost:host-port/

- for Springboot app we don't Need to talk with the tomcat Server, Springboot will take Care of that.

==========

Dockerize Python Flask Application

==========

=> Python is a scripting language

=> We don't need any build tool for python app

=> Directley we can run python programs

Ex : python app.py

=> Flask is a python library which is to develop rest apis in python.

=> To download flask library we will use 'python pip software'

Note: We will configure dependencies in "requirements.txt"

========== **Dockerfile for Python Flask App** ==========

FROM python:3.6

MAINTAINER Ashok

COPY . /usr/app/ → here (.) means pwd we are copying all file to the /usr/app directory.

➔ Copy from source to destination .

WORKDIR /usr/app/

RUN pip install -r requirements.txt

EXPOSE 5000 → 5000 is container port no , python application run on 5000 port

ENTRYPOINT ["python", "app.py"]

==========

Python App Git Repo : https://github.com/ashokitschool/python-flask-docker-app.git

$ git clone https://github.com/ashokitschool/python-flask-docker-app.git

$ cd python-flask-docker-app

$ docker build -t <img-name> .

$ docker run -d -p 5000:5000 <img-name>

$ docker ps

Note: Enable 5000 port in security group inbound rules.

=> Access application with URL

URL : http://public-ip:host-port/

==========

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Date: 19-Nov-2024

Topic : Docker Projects

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

==========

Dockerizing Angular application

==========

=> Angular is a frontend framework

=> Angular is used to develop frontend of the applications

Note: Angular developed by google company.

=> We will use Node Package Manager(NPM) to build angular applications

Note: When we build angular app it will generate "dist" folder for deployment

=> To run angular app we can copy "dist" folder data to "Nginx" server.

```
FROM node:18 AS build
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build --prod
FROM nginx:alpine
COPY --from=build /app/dist/angular_docker_app /usr/share/nginx/html
EXPOSE 80
```

for building application

for running application .

```
git clone https://github.com/ashokitschool/angular_docker_app.git
cd angular_docker_app
docker build -t ngapp .
docker images
docker run -d -p 80:80 ngapp
```

Note: Enable 80 port number in security group inbound rules.

=> Access application in browser using docker-server public ip address

URL : http://13.200.252.152/

==========

Dockerizing React JS application

==========

=> React JS is a Java Script library it is used to develop frontend of the applications.

Note: React JS developed by facebook.

=> We will use Node Package Manager(NPM) to build react applications

Note: When we build react app it will generate "build" folder for deployment.

=> To run angular app we can copy "build" folder data to "Nginx" server.

FROM node:18 AS build

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build --prod

FROM nginx:alpine

COPY --from=build app/build /usr/share/nginx/html

EXPOSE 80

---

git clone https://github.com/ashokitschool/ReactJS_Docker_App.git

cd ReactJS_Docker_App

docker build -t reactapp .

docker images

docker run -d -p 80:80 reactapp

Note: Enable 80 port number in security group inbound rules.

=> Access application in browser using docker-server public ip address

URL : http://13.200.252.152/

==========

Revision

==========

Download and Install Required Softwares

1) What are the challenges in app deployment process

2) Containerization - Packaging code & Dependencies as Single unit

3) Docker Introduction

4) Docker Architecture

5) Docker Setup in windows & linux

6) Docker Registry (docker hub) - Store Docker Images

7) Docker Images - It will take care of Software Installations

8) Docker Containers –

9) Dockerfile –

10) Dockerize java web app with tomcat

11) Dockerize java springboot app

12) Dockerize python flask app

13) Dockerize angular app

14) Dockerize react app

==========

Docker Network

==========

=> Network is all about communication

=> Docker network is used to provide isolated network for containers.

=> If we want to access one docker container from another docker container then those 2 containers should run on the same network.

=> By default we have 3 networks in the docker

1) bridge
2) host
3) none

=> Bridge network is used to run standalone containers. It will assign one IP address for the container. It is the default network used by docker container.

=> Host network is used to run standalone containers. This will not assign any ip address for the container.

=> None means no network will be provided.

 # display docker networks

```
docker network ls
```

 # create docker network

```
docker network create ashokit-nw
```

 # inspect docker network

```
docker network inspect ashokit-nw
```

 # run docker container with custom network

```
docker run -d -p 8080:8080 --network ashokit-nw ashokit/java-web-app
```

 # delete network

```
docker network rm ashokit-nw
```

==========

Docker Compose **07-28-Docker-20-Nov-24**

==========

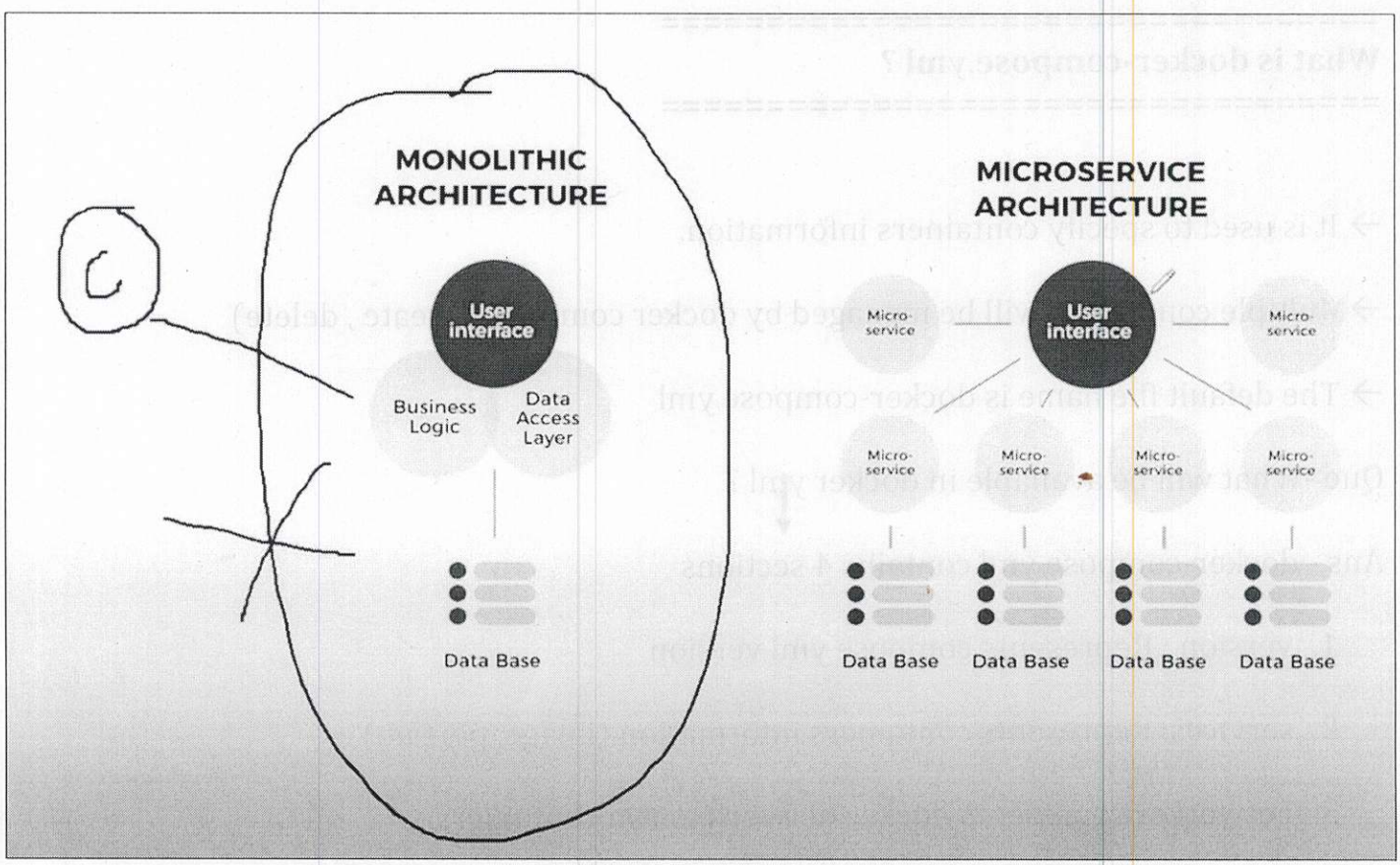
- Alternative is k8s

=> Earlier companies used to develop applications using Monolithic architecture (everything in single application).
- one docker image is created and easily we can manage the one container .

=> Now a days companies are using Microservices architecture to develop the applications.

=> Microservices means we will have multiple backend apis and every backend api is a separate project .

1. Hotels-api
2. Flight-api
3. Train-api
4. Cab-api



→For every backend api seperate dockerfile will be available and we need to create seperate docker image for every api.

→ In order to deploy our application we need to manage multiple containers.

→ When we have multiple containers then managing them will be difficult (create/start/stop/re-start).

→ To overcome these management problems we will use Docker Compose.

→ Docker Compose is used to manage multi-container based applications only .

→ using docker-compose with single command we can "create/stop/start/ delete" multiple containers.

→ which containers should be managed by docker-compose we will configure that information in docker-compose.yml file.

==========

What is docker-compose.yml ?

==========

→ It is used to specify containers information.

→Multiple containers will be managed by docker compose (create , delete)

→ The default file name is docker-compose.yml

Que -What will be available in docker yml ?

Ans - docker-compose yml contains 4 sections

1. version : Represents compose yml version
2. services: Represents containers information (image, port no)
3. networks: Represents docker network to run containers
4. volumes : Represents storage location for containers .

Devops Compose Setup : -

sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

docker-compose –version

==========
Spring Boot with MySQL DB using Docker-Compose
==========

```
version: "3"

services:
        application:
          image: spring-boot-mysql-app
 ports:
    -   8080: 8080

networks:
    -   springboot-db-net

depends_on:
    -   mysqldb

volumes:
    -   /data/springboot-app

mysqldb:
   image: mysql:5.7

networks:
    -   springboot-db-net

environment:
    -   MYSQL_ROOT_PASSWORD=root
    -   MYSQL_DATABASE=sbms

volumes:
    -   /data/mysql

networks: springboot-db-net:
```

Difference between Dockerfile & Docker-Compose ?

- Dockerfile is used to create docker image and Docker-Compose is used to create a container.

==========

Application Execution Process

==========

clone git repo
git clone https://github.com/ashokitschool/spring-boot-mysql-docker-compose.git

#go inside project directory cd spring-boot-mysql-docker-compose

#build project mvn clean package

create docker image
docker build -t spring-boot-mysql-app .

check docker images
docker images

create containers using docker-compose
docker-compose up –d

stop containers using docker-compose
docker-compose stop

start containers using docker-compose
docker-compose start

delete containers using docker-compose
docker-compose down

Docker-compose ps

```
            Name                          Command               State                Ports
-------------------------------------------------------------------------------------------------------
spring-boot-mysql-docker-       java -jar /spring-boot-mys    Up      0.0.0.0:8080->8080/tcp,:::80
compose_application_1           ...                                   80->8080/tcp
spring-boot-mysql-docker-       docker-entrypoint.sh mysqld   Up      0.0.0.0:3306->3306/tcp,:::33
compose_mysqldb_1                                                     06->3306/tcp, 33060/tcp
[ec2-user@ip-172-31-46-8 spring-boot-mysql-docker-compose]$ |
```

Docker-swarm Link :

https://youtu.be/sxBodGYHv_s

Alternative for docker swam is K8s

spring-boot-mysql-docker-compose_mysqldb_1
spring-boot-mysql-docker-compose_application_1

Connect with the database using below command:-

docker exec-it spring-boot-mysql-docker-compose_mysqldb_1 mysql -h mysqldb -u root -proot -P 3306

docker exec-it spring-boot-mysql-docker-compose_application_1 ping mysqldb

==========

Stateless Containers vs Statefull Containers

==========

Stateless Container: Data will be deleted after container deletion
Statefull Container: Data will be available permanentley.

Note: Docker containers are by default stateless.

Note: In above example when we delete and re-create containers we lost data that we inserted in application. This is not accepted in the real-time.

In order to maintain containers data we will use Docker Volumes concept.

→ Using Docker Volumes concept we can make docker container as state full container.

==========

Making docker container as stateful

==========

=> Map ".app" directory to database container as a bind mount volume.

```
version: "3"
services:
 application:
  image: spring-boot-mysql-app
  ports:
   - "8080:8080"
  networks:
   - springboot-db-net
  depends_on:
   - mysqldb
  volumes:
   - /data/springboot-app

 mysqldb:
  image: mysql:5.7
  networks:
   - springboot-db-net
  environment:
   - MYSQL_ROOT_PASSWORD=root
   - MYSQL_DATABASE=sbms
  volumes:
   - .app:/var/lib/mysql
networks:
 springboot-db-net:
```

==========
Summary
==========

1) What are the challenges in app deployment process

2) Containerization

3) Docker Introduction

4) Docker Architecture

5) Docker Setup in windows & linux

6) Docker Registry (docker hub)

7) Docker Images

8) Docker Containers

9) Dockerfile

10) Dockerize java web app with tomcat

11) Dockerize java springboot app

12) Dockerize python flask app

13) Dockerize angular app

14) Dockerize react app

15) Docker Network

16) Docker Compose

17) Docker Volumes

18) Docker Swarm

We can skip these 4