==========
Software Application Architecture
==========

1) Frontend : User Interface (UI)

2) Backend : Business logic

3) Database : Storage

==========
Tech Stack of Application
==========

Frontend : Angular 18v

Backend : Java 17v

Database : MySQL

Webserver : Tomcat

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

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

==========
Application Environments
==========

=> In realtime we will use several environments to test our application.

1) DEV => 10 machines

2) SIT => 20 machines

3) UAT => 50 machines

4) PILOT => 100 machines

5) PROD (final delivery) => 200 machines

=> DEV env used by developers for code integration testing.

=> SIT env used by Testers for system integration testing.

=> UAT env used by client side team for acceptance testing (Go or No Go)

=> Pilot env used for pre-production testing

=> Prod env used for live deployment (end users can access our app)

=> DevOps team is responsible to setup infrastructure to run our application.

=> We need to install all required softwares (dependencies) to run our application

Note: We need to setup dependencies in all environments to run our application.

Note: There is a chance of doing mistakes in dependencies installation process (version compatability issues can occur)

==========
Life without Docker
==========

(1) "It works on my machine" problems"

- Developers would build and test apps on their laptops.

- When they move the app to servers, it often breaks because of differences in OS, libraries, versions, configurations, etc.

Without Docker, teams would spend hours or days debugging these environment differences.

(2) Complex Software Installations

- installing something like a database (e.g., PostgreSQL) required manually:

- download right version of s/w
- install dependencies
- configure services properly

(3) Heavy Virtual Machines (VMs)

- Before Docker, developers used Virtual Machines like VirtualBox, VMware, or Hyper-V to simulate servers.

- VMs are large, slow to start, and resource-hungry (each VM has a full OS inside).

******** To resolve above probelms we can use Docker **********

==========
What is Docker ?
==========

=> Docker is a free & open source software

=> Docker is used for containerization

Container = Execute application as a package (code + required s/w)

=> With the help of docker, we can run our application in any machine very easily.

=> Docker will take care of dependencies installation required for app execution.

=> We can make our application portable using Docker.

==========
Docker Architecture
==========

1) Dockerfile

2) Docker Image

3) Docker Registry / Docker Hub

4) Docker Container

=> Dockerfile is used to specify where is our app-code and what dependencies (softwares) are required for our application execution.

=> Docker Image is a package which contains (app_code + dependencies)

Note: Dockerfile is required to build docker image.

=> Docker Registry is used to store Docker Images.

Note: When we run docker image then Docker container will be created. Docker container is a linux virtual machine.

=> Docker Container is used to run our application.

```
============================
Docker Setup in Linux VM
============================

*Reference Video* : https://www.youtube.com/watch?v=FjUjUw_O6iE

Step-1 : Create EC2 VM (Amazon linux ami with t2.micro)

Step-2 : Connect with that vm using ssh client (git bash)

Step-3 : 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

=================
Docker Commands
=================

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

docker ps : To display only running docker containers

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

docker pull : To download docker image from docker hub

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

docker run : To create docker container

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

docker rm : To delete stopped docker container

                $ docker rm <container-id>

docker stop : To stop docker container

                $ docker stop <container-id>
```

```
docker start : To start docker container

                $ docker start <container-id>

docker rmi : To delete docker image

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

docker logs : To display container logs

                $ docker logs <container-id>

=======================================================
Running Real-world applications using docker images
=======================================================

### public docker image name (java springboot app) : ashokit/spring-boot-rest-api

$ docker pull ashokit/spring-boot-rest-api

$ docker run ashokit/spring-boot-rest-api

Note: Create container with detached mode and port mapping options

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

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

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

Note: To access application running in the container we will use below URL

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


### public docker image name (python app) : ashokit/python-flask-app

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

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

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

========================
What is Port Mapping ?
========================

Note: By default, services running inside a Docker container are isolated and not accessible from
outside.

=> Docker port mapping is the process of linking container port to host machine port.

=> It is used to  allow external access to applications running inside the container.

Syntax : docker run -p <host_port>:<container_port> image_name

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

=============
Dockerfile
=============

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

Filename : Dockerfile

=> 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) ENTRYPONT

====
FROM
=====

=> It is used to specify base image required to run our application.

Ex:

FROM openjdk:17

FROM python:3.3

FROM node:19.5

FROM mysql:8.5

============
MAINTAINER
============

=> MAINTAINER is used to specify who is author of this Dockerfile.

=> This is Optional in Dockerfile.

Ex : MAINTAINER Ashok <ashok.b@oracle.com>

=====
RUN
=====

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

Ex :

RUN 'git clone <repo-url>'

RUN 'mvn clean package'

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

========

CMD
========

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

Ex:

CMD 'java -jar app.jar'

CMD 'python app.py'

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

=====
COPY
=====

=> 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.

EX:

COPY <source> <destination>

COPY target/app.jar /usr/app/

COPY target/app.war /usr/bin/tomcat/webapps/

=====
ADD
=====

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

Ex :

ADD <source> <destination>

ADD <URL> <destination>

========
WORKDIR
========

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

Ex:

COPY target/sbapp.jar /usr/app/

WORKDIR /usr/app/

CMD 'java -jar sbapp.jar'

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

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

```
Ex :

EXPOSE 8080

Note: By using EXPOSE keyword we can't change application port number. It is just to provide
information the people who are reading our Dockerfile.

===========
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"]


Note-1 :  CMD instructions we can override while creating docker container using command line args.

Note-2 :  ENTRYPOINT instructions we can't override.


========================================
Dockerizing Java Spring Boot Application
========================================

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

Note: To package java application we will use 'Maven' as a build tool.

=> To run spring boot application we need to execute jar file.

        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 to run SpringBoot App ############

FROM openjdk:17

COPY target/app.jar /usr/app/

WORKDIR /usr/app/

EXPOSE 8080

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

=====================================================

==========
Lab Task
==========

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

Note: Connect with Docker VM using SSH client and execute below commands

# install git client s/w
$ sudo yum install git -y
```

```
# install maven to build app
$ sudo yum install maven -y

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

# build project

$ cd spring-boot-docker-app

$ mvn clean package

$ ls -l target

# build docker image
$ docker build -t <image-name> .

# check images
$ docker images

# Create docker container
$ docker run -d -p 8080:8080 <image-name>

# check containers running
$ docker ps

# check container logs
$ docker logs <container-id>

=> Enable host port number in security group inbound rules and access our application.

                URL :: http://public-ip:8080/

============================================================
Q) How to push Docker Image to Docker Hub / Docker Registry ?
============================================================

$ docker images

$ docker tag <image-name> <docker-hub-username>/<image-name>:<tag-name>

Ex: docker tag sbapp ashokit/sbapp:v1

$ docker login

$ docker push <tagged-image-name>



==================
Docker summary
==================

1) What is Virtualization

2) What is Containerization

3) Life without Docker

4) What is Docker

5) Docker Architecture

6) Dockerfile
```

7) Docker Image

8) Docker Registry

9) Docker Container

10) Docker Commands

11) Dockerfile KEYWORDS

12) Writing Dockerfile

13) Creating Docker Images

14) Creating Docker Containers

15) Pushing images to Docker Hub

==========

@@ Kubernetes Crash Course : https://www.youtube.com/watch?v=ljqLf1s5l3w&

@@ jenkins crash course : https://www.youtube.com/watch?v=4cG7dWKbrC8

@@@ Jenkins + Docker + Kubernetes Integration : https://www.youtube.com/watch?v=llBvl_iSLDw

==========