DOCKER

1. what is Docker?

**Ans:** Docker is a tool.This used for Package the application in side this package everything is their tools,libraries, and files.so run anywhere with out any problem.

1. what is image?

**Ans:**  Image is contains everything is needs for run the app.

This contains App code, programming language ,libraries,settings,

Os files. This is not running -it is only ready made package.

1. what is container?

Ans: The container is a running version of image.

Run the image docker create a container.

it’s a isolated Environment.

1. How docker internally works?

Ans: Docker help run the application inside the container.

Docker file -> Docker Engine -> Docker image -> Docker Container

Docker file:

Docker file is create own our. It’s just set of instruction for

Docker build an image automatically.

Step 1:use a image as a base

From <image>

Step 2: Set the working directory inside the container

WORKDIR /app

Step 3:Copy the Java source file to the container

Copy< filename>

Step4:Compile the source file

Run compile <filename>

Step5:run complie class

CMD [“filetype”,”filename”]

**Docker Engine:**

**docker build -t any name** [this for image name ] [run

command prompt]

Docker Engine reads the Docker file and creates an **image.**

**Docker image:**

In this image contains

App code,system tool,dependecies.

**Docker container:**

**docker run <image name>**  [run command prompt]

It’s create a new container.

1. **How it is different from virtual machine?**

**Ans:** In virtuval machine -it has a own full os,size is very high(GBs),speed slow,startup time -minutes,performance slow.

docker shares host os kernel,size is very light(MBs),startpu time-seconds,performance high.

1. **can we see the logs of our program running inside the container?**

**Ans:**

To debug errors

To see console output

To confirm your app is running correctly

Command:

**docker logs <container name or id>**

Program is still running command:

**docker logs -f <container\_name\_or\_id>**

1. **what is the purpose of DockerFile and docker-compose.yaml?**

**Ans:**

**Docker file:**

A **Docker file** is a **set of instructions** to build a Docker image. Includes base image, dependencies, code, commands

**Docker-compose.yaml:**

This file is used to **run multiple containers and manage easily.**

**Purpose:**

1. **Run multiple services like my app,frontend,data base.**
2. Connect containers with networking

8)**can I run the docker-compose.yaml without DockerFile, if yes how and why?**

****Ans:****

**Yes ,we have use offical image or ready made image we have only use docker-compose.yaml file without Dockerfile.**

**But we build a own image we have must create a docker file.**

**9)**How docker internally works?****

****Ans:****

**Once we download the docker desktop and install.**

****Docker componet in my machine:****

**Docker Engine: main software-create and run container.**

**Docker CLI: Command line tool-this interact with engine.**

**Docker Runtime: Actually run container-part of engine.**

**Docker Daemon: Backround service-this manage images,containers,**

**Networks,storage.**

**Docker image:read-only templates.**

**Docker container:run the image create the container.**

**Docker uses ower OS kernal feature isolate the container.**

****Dacker Architecture:****

**We are run docker command in terminal**

**Docker CLI send command to the Docker Daemon.**

**Docker uses ower OS kernal feature isolate the container.**

****Docker use Operating System:****

**Docker container share the host OS kernal but run**

**Isolated processes.**

**This uses linux kernel features:**

****Namespaces:**isolated container-they have have own process,networks ,filesystems.**

****Controlgroups(cgroups):**Limit and manage resource used by container (CPU,Memory)**

****Union file system(Overlay FS):** Efficient layer of image.**

****In linux machine:****

**Run the image->Daemon check that image locally exists or not if not**

**Download that image->create container-setup container isolated**

**environment using namespaces and cgroups->start the process inside**

**the container->container run the process on machine->isolated but lightweight.**

****In Windows/Mac:****

**Docker use as a lightweight virtual machine(VM) running small linux os**

**Docker Destop install this VM with linux kernal.**

**Container run inside the VM.**

**Docker CLI communicate with VM’s docker Daemon.**

**10)**How does we know that docker can run a particular image properly or not?****

****Ans:****

****1)check image compatibility with docker -it’ all true****

**Image format-must valid docker image.**

**Cpu Architecher-x86\_64-most intel/AMD destop,laptop**

**Arm64-newer Apple M1/M2 Macs,Raspberry pi**

**Check system architecture-in your system**

**Linux/mac-> uname -m**

**Windows -> echo %PROCESSOR\_ARCHITECTURE%**

**Ckeck Docker image architecture**

**docker inspect <image-name> | grep Architecture (or)**

**docker buildx imagetools inspect <image-name>**

**Base os compatibility-most docker images are linux based**

**On Windows/Mac-docker use linux VM**

****Run image:****

****Commamd-> Docker run <image name> - expected output -workproperly****

**It’s fail:**

****Command**->**Docker log <container-id> ->detail error****

**Check image details:**

****Command->** **Docker inspect <image name>****

**Interactive mode of debugging:**

****Command->docker run -it <image name> /bin/bash** [manually]**

**11)**How our programs runs docker image inside container?****

****Ans:****

****Build docker image:****

**Write a docker file and build the image**

**Command-> **docker build -t <name> . ->create docker image****

**Run image create a container**

**Command->**docker run -d --name <name> -p 8080:8080 <image name>****

**Docker run->create new container**

**-d -> Detached mode-run the container in background**

**--name <name>-give your container name**

**-p 8080:8080 ->port mapping**

**<image name> ->which image you run**

**Check that is run :**

**Command->**docker ps****

****12)How to create own docker image in scratch?****

****Ans:****

**Scratch docker image is the most minimal and advanced way to build the docker image.**

**Scratch is not a operating system-it is empty.**

****Static binary- no os (or) no libraries****

****No shell,no bash -can’t run bin/bush (or) use run****

****No package manager- all needed files copy own****