**Docker notes**

**Dockerfile:**

* A Dockerfile is a text file that contains instructions on how to build the image.
* It specifies the base image to use, sets up the environment, copies application code and dependencies into the image, and defines commands to run when the container is launched.
* First we define the Dockerfile, then we can build the Docker image using the **docker build** command, and then we can use the **docker run** command to create a container from that image. This container will then run your application, encapsulated with all its dependencies and configurations.
* Define base image, Working directory ,Files to copy, Commands to run ,Which location to go ,Expose to which port

**Docker file is built -> image is formed -> image run will create container.**

docker build -t imageName->it will create image

**Removing image**:

docker rmi imgid

Image->file with all dependencies and library to run the program-> This image can be transferred to different teams or on different server

For example, if we have a Python application, the image would include the Python runtime, any libraries/modules the application depends on, and the application code itself.

Instance created after running image->called **container**

If Multiple times run image->multiple container instance can be created

**Image:**

Suppose if I need redis running so my application can connect to it.First,I will have to create container for that redis image

**Commands:**

**docker pull redis**:pulls image from repo to local environment

**Create a container with image:**

**docker run redis**: this will start image in a container

**docker ps:**list all running docker containers

**docker run -d redis**:runs container in detached mode:will get id of container as output and container will start running

Running container in detached mode:

Docker starts the container and returns control to the terminal immediately, allowing us to continue using the terminal for other tasks.

Detached mode is commonly used in scenarios where we want a container to run in the background without blocking the terminal

**docker stop id**:will stop container with given id

To restart a container:

**docker start id:**will start again

**docker ps -a**:shows running and stopped container

**docker run redis:4.0 ->this will pull the image and start the container**

**Logs of container:**

docker logs container\_id or docker logs container\_name

**Port binding**

docker run -p 6000:6379 redis:

maps port 6000 on the host machine to port 6379 inside the Redis container.

Port 6379 is the default port used by Redis, so this command allows us to access the Redis service running inside the container via port 6000 on the host machine.

We can interact with the Redis service running inside the Docker container by connecting to port 6000 on the host machine.

**Giving name to a container:**

docker run -p 6000:6379 –name redis-older

**docker exec**:get terminal for running container

docker exec -it container-id /bin/bash can also use container name instead of id

it : interactive terminal

**docker start:we are working with container** not images

**Container run in interactive mode**

docker run --name containername -it -d imgname

**Go inside container**

docker exec -it containername [command]

Example**: To run python interpreter**

docker exec -it containerid python

**docker inspect id**:will give all info regarding container

**docker stop id**:still shown in docker ps -a

**docker rm id**:will not be shown in docker ps-a

docker restart containerId1

**Container is running environment of image**

Container:isolated environment in which application run

App1 will not see app2

Deployment and delivery easy

If in one system->2 application with same nodejs but different version,we can use virtualization

Same can be done with container->use different version of same software in same machine

Docker container->takes very less resource

Virtual image:takes a lot of source

**Docker registry:**

central repo for storing and distributing docker images.ex:docker hub.

Image examples:postgres,mongo,redis

In docker hub:all the things are image

**for putting our created image on dockerhub:**Use docker login

docker commit->used to save an image in edited container in localhost

docker push->to push image on repo

docker copy:copy file from docker to local system

docker volume:create volume to store the data,we use it when we want docker container to store the data

docker logout