DOCKER INSTALLATION

-> First go to aws and create an ec2 instance
-> now go to docker documentation [Install Docker Engine | Docker Documentation]
-> select the os we using in our ec2 instance
-> sudo -i
-> apt-get update
-> follow the documentation
-> paste everything in it

docker images -> used to see docker images
sudo vim /etc/group -> you can go to here and add user to docker group
usermod -aG docker ubuntu
docker run hello-world -> it will run image named hello-world if not there it download
docker ps -a -> is used to see all containers even dead one

DOCKER HUB

Docker hub is registry for docker images

DOCKER IMAGE

-> A stopped Container is like vm image
-> Consist of multiple layers
-> An app will be bundled in an image
-> Containers run from the images
    Images become containers when they run on docker engine

DOCKER REGISTRY

-> cloud based registries are
-> docker hub
-> GCR (google container registry)
-> Amazon ECR

-> inhouse or local registry
-> Nexus 3+
-> Jfrog artifactory
-> DTR (docker trusted registry)

Docker Commands

docker images -> list images locally
docker run -> command creates a new container
docker ps -> list's running containers
docker ps -a -> lists all the containers
docker exec -> execute the commands on container
docker start/stop/restart/rm
docker rmi -> used to remove the image
docker rm container id -> used to remove dead processes
docker inspect -> used to see details of container and image
docker images --filter=reference=alpine
docker save nginx > nginx.tar -> to make docker image to tar file
docker load < nginx.tar -> is used to create image from tar file

To run containers

docker run --name httpd service -p 7090:80 -d httpd
docker run --name myweb -p 7090:80 -d nginx

You can run more than one container with single image

cd /var/lib/docker -> here you can see the containers
du -sh container id -> is used to see the size of container

docker exec myweb ls/ -> used to run commands in container
docker exec -it myweb /bin/bash -> used to go into container interactive mode
docker exec -it myweb /bin/sh -> some containers will have this sh shell

When you try to remove some images sometimes you will get an error it is because the image is being used by container so you have to first stop the container and then remove the image

docker stop container name -> used to stop the container
docker log container id -> used to see details of container logs
docker inspect container id -> used to see details of container too
docker inspect image:tag -> used to see details of image

Container volume

Container data will be gone once the container is no longer exists it is difficult to get the data out if another processor needs it

The container writable layer is tightly coupled to the host machine it is hard to move the data to somewhere else

Docker has two options for

1) volumes

The container will be connected to a directory of host machine all the data from that container will be stored in that directory (var/lib/docker/volumes/) managed by docker

docker volume create nameofdb -> used to create a volume
docker volume ls -> is used to list the volumes
docker run --name vprodb -d -e MYSQL_ROOT_PASSWORD=secretpass -p 3030:3306 -v nameofdb:/var/lib/mysql mysql:latest

ls /var/lib/docker/volumes/mydb/_data -> here is our data is saved

2) Bind mount

It's similar to vagrant synch directory in bind mount we set volume location it is used when you want to make changes in container you can first make changes in host machine and that will reflect for container

docker run --name vprodb -d -e MYSQL_ROOT_PASSWORD=secretpass -p 3030:3306 -v /root/vprodb:/var/lib/mysql mysql:latest

By doing inspect to container you can get the ip of it by using that ip you can connect to container like mysql

mysql -h ip -u root -p secretpass -> to connect to that container via host machine

DOCKERFILE INSTRUCTIONS

FROM => base image (service or os)
LABELS => adds metadata to image (like tags in aws)
RUN => Execute commands in new layer and commit the results(execute commands like install packages etc)
ADD/COPY => add files and folders into image (add will un tar file and paste copy just paste)
CMD => Runs binaries and commands on docker run
ENTRYPOINT => Allow you to configure a container that will run as a executable (we can give command at entrypoint and use cmd to pass argument to the command)
VOLUMES =>Creates a mount point
EXPOSE => container listen on specific networks port at runtime
ENV => Sets the environment variable
USER => sets the username
WORKDIR => sets the working directory
ARG => Defines variable that user can pass at the build time
ONBUILD => used to use this image as a base image to built other image

CREATING A DOCKER FILE

First create a ec2 instance ubuntu
Then install docker
Go to toolplate and get the file using wget and link
Mkdir images inside that put a nano directory
Now unzip it
Then go to unzipped folder and archive it using -> tar czvf nano.tar.gz *
mv nano.tar.gz ../ -> used to move file 1 step down to current folder
Move this file to nano directory
Now in side nano folder
vim Dockerfile
Inside the Docker file

```
FROM ubuntu:latest
LABEL "AUTHOR"="TARUN"
LABEL "PROJECT"="dockerfile"
ENV DEBIAN_FRONTEND=noninteractive
RUN apt update && apt install git apache2 -y
CMD ["/usr/sbin/apache2ctl","-D","FOREGROUND"]
EXPOSE 80
WORKDIR /var/www/html/
VOLUME /var/log/apache2
ADD nano.tar.gz /var/www/html/
```

docker build -t name_image and dockerfile path -> if current path use . at path
docker build -t name_image:tag . -> we can also give tag

REGISTERING DOCKER IMAGE IN DOCKER HUB

-> go to the docker hub and sign in
-> click on repository and click on create a repository
-> can set public or private

But we can directly push the image to the repository and create it same time

-> while building the image
-> docker build -t kandulatarun/nanoimg:v1 .
-> docker login (in instance)
-> docker push kandulatarun/nanoimg:v1 (to push this image to docker repo)

docker pull kandulatarun/nanoimg:v1 -> used to pull images from our repo or public one
docker run --name nano -p 8000:80 -d kandulatarun/nanoimg:v1 -> directly pull and run the image from our repo or public one