Docker Commands

Installation

- **sudo apt update**
- **sudo apt install docker.io**
- **sudo service docker start**
- **sudo systemctl enable docker**
- **sudo systemctl status docker**

Docker Hub

- **docker pull img_name**: download an image from Docker Hub

Running Containers

- **docker run image-name/image-id**: create a container from an image
- **docker run img_name**: create a container
- **docker run --detach/-d img_name**: pull, start, and create a container
- **docker run -d --name cont_name container_id**: create a container with a name

Listing Containers and Images

- **docker ps**: show running containers
- **docker ps -a/--all**: show all containers (running and stopped)
- **docker container ls**: show all containers (running and stopped)
- **docker images**: show a list of images

Deleting Containers and Images

- **docker rmi img_name**: delete an image
- **docker rmi $(docker images -a) / docker image prune / docker rmi $(docker image ls)**: remove all images
- **docker stop container_id**: stop a container
- **docker rm container_id**: delete a stopped container
- **docker rm $(docker ps -a / docker container ls -a) / docker container prune**: delete all stopped containers

Working with Containers

- **docker exec -it container_id /bin/bash**: start working in a container
- **docker inspect container_id**: show all information about a container
- **docker run -d --name name nginx**: create a container with a name
- **docker history img_id**: show all layers of an image

Port Mapping

- **docker run -d -p80(ec2-port):80(container-port) img_id**: map a port for an existing image
- **docker run -d --name name -p80(ec2-port):(container-port) nginx**: create a container with port mapping

Creating Images

- **docker commit container_id name**: create an image from a container
- **docker save > img_name.tar**: create a tar file of an image
- **docker export container_id > file_name.tar**: create a tar file of a container

Volumes

- **docker run -d --name name -p80(ec2-port):(container-port) -v path of directory:project directory path in container img_name**: create a container with a volume
- **docker volume create vol_name**: create a named volume

Environment Variables

- **docker run -d --name merabaladb1 -e MYSQL_ROOT_PASSWORD='Pass@123' MySQL**: create a MySQL container with environment variables
- **docker run -d --name sqlvol -v /home/ubuntu/sqlvol:/var/lib/mysql -e MYSQL_ROOT_PASSWORD='pass@123' -e MYSQL_DATABASE="facebook" MySQL**: create a MySQL container with a database name and mount with a bind volume

Linking Containers

- **docker run -d --name wordpress -p80:80 -e WORDPRESS_DB_HOST=mydb -e WORDPRESS_DB_USER=root -e WORDPRESS_DB_PASSWORD=pass123 -e WORDPRESS_DB_NAME=db1 --link mydb:mysql wordpress**: link a MySQL container to a WordPress container

Dockerfile

Components of Dockerfile

- **FROM**: use for base images or to pull an image (can be used multiple times)
- **RUN**: for installing software or running Linux commands (can be used multiple times)
- **EXPOSE**: to open a port number
- **COPY**: to copy files and directories from the host to the image
- **ENV**: to set environment variables
- **CMD**: specifies the command to run when a container is run from the image (can only be used once)
- **ENTRYPOINT**: specifies the command to run when a container is run from the image, but allows additional arguments to be passed in (can only be used once)

- **ADD**: copies files from the host to the image, downloads zip or tar files from a given link, and extracts them automatically
- **ARG**: defines a variable that is passed to the container while building the image
- **VOLUME**: creates a volume, sets a volume
- **WORKDIR**: sets the working directory
- **MAINTAINER**: sets the name and email of the author/user
- **LABEL**: adds metadata (data about data)
- **USER**: sets the user (root, ec2-user, docker, etc.)
- **HEALTHCHECK**: specifies the path for a health check or checks the health of a mentioned URL
- **SHELL**: specifies the shell to be used to run commands
- **STOPSIGNAL**: specifies the signal to be sent to the container to stop it gracefully
- **ONBUILD**: specifies the instruction to be used when the

Docker Build

- **docker build -f file_name .**: run a file (if file name is different)
- docker build -t tag_name -f file_name . --> run file with tag
- docker system df --> for check container space

Docker Network

- **docker network ls**: show a list of networks
- **docker network create name**: create a network
- **docker network inspect name**: show all information about a network
- **docker network rm name**: delete a network
- **docker run -d --name cont_name --network network_name image_id**: create a container in a network
- **docker network connect network_name cont_name**: add a container to a network
- **docker network disconnect network_name cont_name**: remove a container from a network

Docker Compose

- **docker-compose up -d**: run a compose file (if the file name is **docker-compose.yml**)
- **docker-compose -f file_name up -d**: run a compose file with a different name
- **docker-compose down**: delete containers

Docker System

- **docker system df**: check container space