DOCKER CLASS-2

DOCKER BASIC COMMANDS:

- To install docker in Linux : yum install docker -y
- To see the docker version : docker --version
- To start the docker service : service docker start
- To check service is start or not : service docker status
- To check the docker information : docker info
- To see all images in local machine : docker images
- To find images in docker hub : docker search image name
- To download image from docker hub to local : docker pull image name
- To download and run image at a time : docker run -it image name /bin/bash
- To give names of a container : docker run -it --name raham img-name /bin/bash
- To start container : docker start container name
- To go inside the container : docker attach container name
- To see all the details inside container : cat /etc/os-release
- To get outside of the container : exit
- To see all containers : docker ps -a
- To see only running containers : docker ps (ps: process status)
- To see only exited containers: docker ps -q -f "status=exited"
- To stop the container : docker stop container name
- To delete container : docker rm container name
- To stop all the containers : docker stop $(docker ps -a -q)
- To delete all the stopped containers : docker rm $(docker ps -a -q)
- To delete all images : docker rmi -f $(docker images -q)

DOCKER RENAME: is used to rename the container.

To rename docker container: docker rename old_container new_container

ALTERNATE CONTAINER COMMANDS:

- To see list of containers : docker container ls
- To see all running containers: docker container ls -a
- To see latest 2 containers : docker container ls -n 2
- To see latest container : docker container ls --latest
- To see all container id's : docker ls -a -q
- To remove all containers : docker container rm -f $(docker container ls -aq)
- To see containers with sizes : docker container ls -a -s
- To stop container after some time: docker stop -t 60 cont_id

KILL VS STOP:

KILL: It passes SIGKILL signal to the container.

STOP: It passes SIGTERM signal to the container.

DOCKER EXEC: is a command that allows you to run commands inside a running Docker container. You can use docker exec to execute commands, check the container's file system, troubleshoot issues, or perform various tasks within the container without the need to start a new instance of the container.

syntax - docker exec cont_name command

ex-1: docker exec cont1 ls

ex-2: docker exec cont mkdir devops

to enter into container: docker exec -it cont_name /bin/bash **or** docker exec -it cont_name bash

CREATE IMAGE FROM CONTAINER:

- First it should have a base image - docker run nginx
- Now create a container from that image - docker run -it --name container_name image_name /bin/bash
- Now start and attach the container
  - go to tmp folder and create some files (if you want to see the what changes has made in that image - docker diff container_name)
- exit from the container
- now create a new image from the container - docker commit container_name new_image_name
- Now see the images list - docker images
- Now create a container using the new image
- start and attach that new container
- see the files in tmp folder that you created in first container.

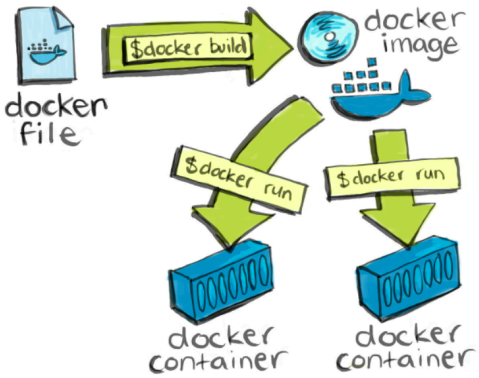
DOCKER FILE:

- It is basically a text file which contains some set of instructions.
- Automation of Docker image creation.
- Always D is capital letters on Docker file.

- And Start Components also be Capital letter.

HOW IT WORKS:

- First you need to create a Docker file
- Build it
- Create a container using the image



DOCKER FILE COMPONENTS:

- **FROM:** THIS COMPONENT IS USED TO DEFINE BASE IMAGES (ubuntu, centos, httpd, jenkins)
- **MAINTAINER:** THIS IS USED TO ADD AUTHOR DETAILS
- **LABEL:** IT IS USED TO ADD A DESCRIPTION FOR OUR IMAGE
- **COPY:** USED TO COPY THE FILES FROM HOST TO CONTAINER
- **ADD:** USED TO COPY THE FILES FROM HOST TO CONTAINER AND ALSO IT WILL DOWNLOAD THE FILES FROM IE AND SEND THOSE FILES TO CONTAINER.
- **RUN:** IS USED TO EXECUTE THE COMMANDS, WHILE CREATING AN IMAGE
- **CMD:** IS USED TO EXECUTE THE COMMANDS, WHILE CREATING A CONTAINER
- **ENTRYPOINT:** IS USED TO EXECUTE THE COMMANDS, WHILE CREATING A CONTAINER, IT HAS HIGH PRIORITY THAN CMD AND ALSO IT WILL OVERWRITE THE VALUES OF CMD
- **ENV:** IS USED TO DEFINE THE VARS, IT IS NOT POSSIBLE TO PASS THE VALUES DYNAMICALLY. WE CAN ACCESS THOSE VARS IN CONTAINER
- **ARG:** IS USED TO DEFINE THE VARS, IT IS POSSIBLE TO PASS THE VALUES. DYNAMICALLY. WE CAN'T ACCESS THOSE VARS IN CONTAINER.
- **WORKDIR:** USED TO SET A DEFAULT PATH IN CONTAINER
- **EXPOSE:** USED TO DEFINE CONTAINER PORTS