DOCKER

“To standardized way to package the application with its dependencies and deploy it on any environment.”

- Docker is a tool used to create, deploy,and run applications using containers.
- It makes it easy to deploy and run applications in a repeatable and secure
- Docker allows applications to run on separate machines with the same operating system kernel.
- Docker allows developers to create applications that can be quickly and easily shipped to different machines with different configuration

Docker is popular due to / advantages of docker

- Low system requirements
- Quick application portability
- Local development environment
- Continous integration and deployment
- Code isolation
- Increased efficiency

The Reason could be due to application not working fine : / why docker

- Dependencies
- Libraries and versions
- Framework
- OS Level features
- Microservices

DOCKER LIFE CYCLE

“Docker goes through complete life cycle from creation to deletion state”

1. **Create**: 1st stage is a creation of docker image .It can build from pull image from docker registry or by writing ‘Dockerfile’
2. **Run**: Running of docker container.It contains ports,volumes,environmental variables
3. **Manage**:After running container ,It is managed by various commands like docker ps ,docker start,docker stop,docker restart,etc
4. **Update**: Docker containers are updated by configuration,dependencies and restarting
5. **Backup**: Docker containers are backup by creating snapshots of filesystem & metadata by the tools are docker volume and docker images
6. **Restore**:Docker containers are restored by images or container filesystem & metadata.
7. **Delete**: Final stage is deletion of container by command ‘docker rm containername’

DOCKER ARCHITECTURE

- **Docker Daemon:** *It runs on host OS & resposible for running containers to manage docker service,

  * It communicating with other daemons

  *It offers various docker objects such images,containers,networking&storage

- **Docker Client**: It is a command line interface tool that allows user to interact with Docker Daemon

  * It send commands to Docker Daemon & receives response

- **Docker Host**: It provides environment to execute and run the application

  * It contains docker daemon,images,containers,networks & storage

- **Docker Registry**: It manages and stores the Docker images

  * 2 types of Docker Registry are

1) Public Registry: Docker hub
2) Private Registry: Images share within enterprise

- **Docker Network**: Docker provides build in network infrastructure that allows containers to communicate with each other & with the host machine.

* It can be created and configured using docker command line interface (CLI) or Docker compose

- **Docker Image**: It is a read only template that contain set of instructions for creating docker container
- **Docker Container**: It is a Run time environment to images with updating all its dependencies to run applications

Difference b/n Virtual machine and Docker container

Virtual machine	**Docker container**
1)occupies lot of memory space	1)occupies less memory space
2)Bootup time is long	2)Bootup time is less
3)Multiple running Virtual Machine leads to unstable	3)Multiple diff host OS containers run efficiently
4)Difficult to scale up	4)Easy to scale up
5)Low efficiency	5)High efficiency

DOCKER COMMANDS

$ docker version : To check detailed docker version

$ docker –v : To check short-info docker version

$ docker search imagename : To search image name

Ex: docker search nginx

$ docker pull imagename: To pull image from dockerhub

ex docker pull nginx

$ docker images: To list images

$ docker run –it - -name newcontainer imagename : To enter inside container

Ex : docker run –it - -name mycont nginx

ctrl+p+ctrl+q : exit container without stopping

$ docker ps : To check runnung container

$ docker run –it imagename: To go inside container with its random container name

Ex : docker run –it nginx,

exit : after by entering inside container to exit & container stops automatically

$ docker ps –a : To list Running and stopped container

$ docker attach runcontainername: To go inside running container name

$ docker stop containername: To stop running container

$ docker kill containername :- To stop container suddenly

$ docker rm containername: To delete stopped container

$ docker start containername: To start container

$ docker rmi imagename: To delete image name

$ docker run –t –d - -name newcontainername imagename: To run the container in detached mode means that without entering inside container.(if we try to docker attach containername to go inside it freezed due to it runs in detached mode)

$ docker exec –it containername /bin/bash : To enter inside container of detached mode

$ docker diff containername : To see any updates or logs in the image

Output shows: C-changed

A-appended(add)

D-Deleted

$ docker commit runcontainername newimagename: To create new image from running container

$ docker stop container1 container2 containeretc: To stop multiple container

CREATE AND BUILD DOCKER

Dockerfile is basically a **text file**. It contains some **set** of **instructions**.
Automation of **docker image** creation.

Dockerfile components:

FROM: for base image, this command must be on the top of the dockerfile.

WORKDIR: Specify the working directory inside the container.

RUN: to execute commands, it will create a layer in image

MAINTAINER: author/ owner/ description

COPY: copy files from local system (docker vm) we need to provide source, destination (we can't download file from internet and any remote repo.)

ADD: similar to copy but it provides a feature to download files from internet, also extract file at docker image side.

EXPOSE: to expose ports such as port 8080 for tomcat , port 80 for nginx etc.

CMD: execute commands but during container creation.

ENTRYPOINT: similar to CMD but has higher priority over CMD, first commands will be executed by ENTRYPOINT only.

ENV: environment variables

Create a file named Dockerfile

Add instructions in Dockerfile

Build dockerfile to create image

Run image to create container

EX (1)

$ vi Dockerfile

```
FROM nginx    #base image

RUN touch /tmp/custom     #to run commands
```

$ docker build –t newimagename . (#-t –tag,new image name, .-current dir)

: To build image from dockerfile

$ docker run –it --name newcontainername imagename : After build image created when we entered inside the container it directly take into directory mentioned as in the working dir

$ docker rm –f runcontainername : To delete running container forcefully

EX(2)

$ vi Dockerfile

```
FROM ubuntu
WORKDIR /tmp
RUN echo “welcome to docker” > /tmp/file1
ENV name docker
COPY testfile /home
ADD testfile.tar.gz/tmp
```

$ docker build –t newimagename .

Then go to

$ cd /tmp,**$ ls** :files of working dir available here

$ cd /home, $ ls :testfile mentioned in copy is available here

$ echo $name: to excute env name

$ docker run –td - -name newcontainername –p portrange imagename : To run image on port,its available internet & localhost

Ex:$ docker run –td - -name mycontainer –p 88:8080 ubuntu

$ docker run –td - -name newcontainername –expose portrange imagename : To expose image on internally means on docker environment not on locally

Ex: $ docker run –td - -name mycontainer –expose 88 ubuntu

$ docker port containername : To check port map for container

$ docker run –it –p portrange imagename : To run container by giving port without mentioning container name

DOCKER VOLUMES

Docker volume is a persistent data storage mechanism that can be used to store data outside of a container's filesystem.

2 types of docker volume

1.Normal volume

2. shared volume

* container to container

* host to container

$ docker volume ls : To list docker volumes

$ docker volume create volumename : To create docker volume

$ docker run –it –v volumename:/mountpointname --name containername imagename : To run volume as container & enter inside container

Ex:$ docker run –it –v vol1: /ub --name ubun ubuntu

$ touch f1 f2 : create some files inside container

$ docker rm –f container : delete container but data still available on docker vol

$cd /var/lib/docker/volumes/volumename/data :docker volumes avilable here even deletion of container

$ docker inspect containername : To see detailed about container,files created inside container identified by “mount” path

$docker run –it –volume-from containername –name newcontainername imagename :use the volume from previously used volume in the container by specifieng container name

$ docker volume prune : To delete all docker volumes

$ mkdir volume

$ cd volume

$ Docker run –it –v /home/contname/volname:/vol --name contname imagename : To attach volume to container

DOCKER NETWORK

Docker provides build in network infrastructure that allows containers to communicate with each other & with the host machine

Three types of networks are

1.Bridge

2.Host

3.None

$ docker netwok ls : To list docker networks

$ docker run –it imagename : To run container

$ ping google.com : To ensure that ip is up and running or not

$ docker inspect bridge : To see details about bridge network

$ docker network create --driver bridge mynetwork :To create custom bridge network

$ docker run –it --network mynetwork --name containername imagename :To create container on network

$ docker network connect mynetwork container : To connect container with network

$ docker inspect mynet : To inspect network ,which are the containers connected to it

$ docker inspect bridge:To inspect bridge network

$ docker attach container

$ ping container

Ex:$ docker run –it --network host --name web nginx : To create docker network on local host

DOCKER COMPOSE

Docker Compose is a tool that allows you to define and run multi-container Docker applications. It provides a way to define the services that make up an application in a YAML file, and then start and stop those services using a single command.

$ apt install docker compose : To install docker compose

Ex-1

$ vi docker-compose.yml

```
---
Services:
 tomcat:
    image:tomee
    ports:
        - “888:8080”
    network_mode: bridge
```

$ docker-compose up –d : To run image by yamal file as container

$ docker-compose stop : To stop docker-compose container

$ docker-compose start : to start container again

$ docker-compose down : to delete container

(yamal lint web page helpful for checking syntax of yaml file)

$ vi docker-compose.yml

```
---
Networks:
```

```
  Mynet:
      Driver: bridge
services :
  c1:
    Container_name: c1
    Image: centos
    Networks:
      - mynet
     Tty: true
      Volumes:
        -  “voll:/volume1”
   c2:
    Container_name: c2
    Image: centos
    Networks:
      - mynet
     Tty: true
      Volumes:
        -  “voll:/volume2”
   c3:
    Container_name: c3
    Image: centos
    Network_mode: bridge
     Tty: true
Version: “3.9”
Volumes:
   Vol1:
      Name: -  Vol1
```

$ sudo su

$ docker-compose up –d

$ docker ps

$ docker inspect c3

$ docketr inspect c1

$docker volumel

$ docker volume ls

$ docker network ls

$ docker exec –it c1 /bin/bash

$ docker exec –it c2 /bin/bash

$ docker-compose stop

$ docker-compose start

$ docker-compose down

DOCKER HUB

Docker Hub is a cloud-based registry service that allows you to store and manage Docker images. It provides a centralized location for developers to share and distribute their Docker images, making it easier to collaborate on projects and distribute software.

Login docker hub

Goto hub.docker.com

Sign up using name ,email & paswd

Two types of docker hub

1.public repository

2.private repository

(1)Push and (2)pull image from docker hub

Open CLI

(1)

$ docker login

$ docker run –it –name mycont imagename

$ vi /etc/hostname : To change host name

$ exit

$ docker ps –a

$docker commit runcontainer newimagename

Ex: $docker commit mycont myubuntu

$ docker tag imagename dockerid/reponame

Ex: $docker tag myubuntu raa/myimage

$ docker push tagedimagename

Ex: $docker push raa/myimage

(2)

$ docker pull imagename : pull image from dockerhub

$docker pull **raa/myimage**

$ docker stop $(docker ps –a -q): To stop all container

$ docker rm -f $(docker ps –a -q):To delete all container

$ docker images prune –a :To delete all images

$ docker system prune –a : To delete stopped container,network not used by atleast one container,images not used by one container,build cache

$ docker status containername: To see details about container