1) What is Docker?

Docker is provide the facilitate to build and run applications quickly

2) Containerization vs virtualization

Container is lightweight, standalone package that contains everything needs application

Virtualization is running on BareMetal platform

3) Advantages of Containerization

Container can provide unique benefits for your application

- Faster delivery
- Efficiency
- Improved security
- Flexibility
- Faster app startup

4) Installing Docker

5) Understanding Docker Terminologies

   a. - Docker Images

   b. - Docker Containers

   c. - Docker Host

   d. - Docker Client

Docker image : docker image is file which contains the instructions and files to cerate the container

Docker container : Container is lightweight, standalone package that contains everything needs application.

Docker Host : docker host is terminal.

Docker Client : docker client is a command-line interface that allows users to interact with doacker daemon.

Downloading the image and creating container

Go the docker site and select the image and run the below commands to pull image and create container.

- docker pull tomcat
- docker run --name tomme -d -p 6060:8080 tomact
- docker image ls
- docker container ls

6) Stopping the container and removing the container

Docker stop <container Name>

7) Understanding detached mode and interactive mode

-D :Used for running the container in detached mode as a background process

- it : for opening an interactive terminal in a container

8) Using environment variable

Used for passing environment varaibles to the container

Ex:- docker run --name mydb -d -e MYSQL_ROOT_PASSWORD=sunil mysql:5

9) Creating multi-container architecture using –link

Multi container means create front end and backed containers and link eachother.

Ex:- docker run --name c10 -it busybox

docker run --name c20 -it –link c10:box busybox (box is name)

10) Implementing LAMP Architecture using Docker

L – linux

A – Apache

M – MySQL

P – PHP

docker run --name mydb -d -e MYSQL_ROOT_PASSWORD=koti mysql:5

docker run --name apache -d -p 6060:8080 --link mydb:mysql tome

To check if tomcat is linked with mysql

docker inspect apache (apache is the name of the container)

docker run –name php -d –link apache:tomcat –link mysql:db php

11) Creating CI-CD Environment using Docker

Docker run –name jenkin -d -p 7070:8080 jenkins/Jenkins

Docker run –name qa -d -p 6060:8080 –link jenkin:QA1 tomee

Docker run –name prod -d -p 5050:8080 –link jenkin:qa1 tomee

12) Creating testing environment using Docker

To start selenium/hub as container

docker run --name hub -d -p 4444:4444 selenium/hub

docker run --name chrome -d -p 5901:5900 --link hub:selenium selenium/node-chrome-debug

docker run --name firefox -d -p 5902:5900 --link hub:selenium selenium/node-firefox-debug

Note : mozilla and chrome container are related to GUI, to access gui container vnc viwer is required

13) Installing Docker compose

This is a feature of docker using which we can create multicontainer architecture using yaml files. This yaml file contains information about the containers that we want to launch and how they have to be linked with each other.Yaml is a file format. It is not a scripting language.

sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version

```
---
version: '3'
services:
 mydb:
  image: mysql:5
  environment:
   MYSQL_ROOT_PASSWORD: sunilsunil
apache:
  image: tomee
  ports:
   - 6060:8080
  links:
   - mydb:mysql
 php:
  image: php
  links:
   - mydb:mysql
   - apache:tomcat
```

14) Docker Volumes

- Simple volumes
- Shared volums

Simple docker volumes

These volumes are used only when we want to access the data,

even after the container is deleted.

But this data cannot be shared with other containers.

mkdir /data

docker run --name c1 -it -v /data centos (v option is used to attach volume)

Docker inspect centos – to collect the mount volume

Docker rm -f centos

Cd <mount volume>
/var/lib/docker/volumes/c5c85f87fdc3b46b57bb15f2473786fe7d49250227d1e9dc537bc594db001fc6/_data

15) Need for creating Docker Volumes (shared)

These volumes we can attach to another container.

mkdir /data

docker run --name c1 -it -v /data centos

docker run --name c2 -it --volumes-from c1 centos

docker run --name c3 -it --volumes-from c2 centos

16) Need for creating customized Images

17) Ways of creating customized Images

- Whenever docker container is deleted,
- all the softwares that we have installed within the container will also be deleted.
- If we can save the container as an image, then we can preserve the softwares.

18) Using Docker commit command

docker run --name c11 -it ubuntu

apt-get install git

docker commit c11 myubuntu

docker run --name c22 -it myubuntu

git --version (git is pre installed)

19) Using docker file and Keywords in Docker file

FROM – use d to specify the base image from which the docker file has to be created

MAINTAINER – name of organization

CMD – this is used to specify the initial command that should be used be exeuted when the container strats

RUN – used for running linux command within the container

USER – used to specify default user who should login into the container

Workdir – used to specify default working directory

COPY – copying the files from host machine to the container

ADD – used for copying files from host to container, it can also be used for downloading files from remote servers.

ENV – used for specifying the environment varibles that should be passed tpo the container

EXPOSE -- Used to specify the internal port of the container

VOLUME -- used to specify the default volume that should be attached to the container.

LABEL -- used for giving label to the container

STOPSIGNAL -- Used to specify the key sequences that have to be passed in order to stop the container.

Ex:

```
# cd docker
# vim dockerfile
Lets just add one more instruction
FROM ubuntu
MAINTAINER logiclabs
RUN apt-get update
RUN apt-get install -y git
RUN apt-get install -y tree
docker build -t myubuntu .
```

20) **Changing the default process of the container**

Every docker image come with default process.

As long as default process is running, the container will be running condition.

Observer the command section.

It tells you the default process that gets executed, when we start the container.

Container	Default process
tomcat	catalina.sh
jenkins	/bin/tini
ubuntu	/bin/bash

21) **Working with Registry**

a. **Public Registry**

b. **Private Registry**

public registry is hub.docker.com

Images uploaded here are available for everyone.

Usecase: Create a customized ubuntu image, by installing tree in it.

Save this container as an image, and upload this image in docker hub.

Step 1: Create a new account in hub.docker.com

Step 2: Creating our own container

docker run --name c5 -it ubuntu

Lets install tree package in this container

/# apt-get update

/# apt-get install tree

/# exit

Step 3: Save the above container as an image

docker commit c5 kotidevops77/ubuntu_img26

(kotidevops77/ubuntu_img15 -- is the image name)

Note: Image name should start with docker_id/

To see the list of images

docker image ls (we can see the new image)

TO upload the image to hub.docker.com (docker login command is used)

 # docker login (provide docker_id and password)

To upload the image

 # docker push <image_name>

 # docker push kotidevops77/ubuntu_img26

22) **Introduction to Container Orchestration**

Container running on distributed environment and it will provide the below featues.

- **Load balncing**
- **Rolling updates**
- **Scaling of containers**
- **Handling failover scenario.**

Docker swarm service is mandatory to maintain docker orchestration

We need to install docker swarm using the below command

- **We required 3 docker nodes**
- **Docker manger**
- **Docker worker1**
- **Docker worker 2**

Install docker swarm in manger node using below command

docker swarm init --advertise-addr private_ip_of_manager

docker swarm init --advertise-addr 172.31.42.135

After installed doccker swarm it will generate the token code to join worker nodes into manger.

- Login each worker node and execute the command.

docker swarm join --token SWMTKN-1-27lf3n7xxqy2u3gb61mvfybk51uqjq9hj4m5uwdd4lcgtgafth-0hmrtwzrcyqv4h7cl3euekq68 172.31.44.50:2377

Create the container on docker swarm

Docker service create –name tome -p 6060:8080 –replicas 5 tomee

Docker service ps tome

docker service create --name mydb --replicas 3 -e MYSQL_ROOT_PASSWORD=sunil mysql:5

Scaling of containers : When business requirement increases, we should be able to increase the no of replicas.

Similarly, we should also be able to decrease the replica count based on business requirement. This scaling should be done without any downtime.

docker service create --name appserver -p 8080:80 --replicas 5 nginx

docker service scale appserver=10

docker service scale appserver=2

how to remove the node form swarm service :

docker node update --availability drain Worker1

docker node update --availability active Worker1

Lets Connect to worker2 from git bash

Worker2# docker swarm leave

Rolling Updates: for example redish running with 3 version and required to update the 4 version without down time.

docker service update --image redis:4 myredis

docker service ps myredis | grep Shutdown (We get shutdown container)

docker service ps myredis | grep -v Shutdown (-v used for inverse operation)

Rolling Updates rollback:

docker service update --rollback myredis.

To add a new machine as a manager: # docker swarm join-token manager

To promote worker1 as a manager node :

docker node promote Worker1

To demote Worker1 and make him back as a worker

docker node demote Worker1

23) Understading Docker Networking:

When we install the docker by default network created in the host machine called as docker 0 netwrok.

- Docker inspect <container id>

Two container eachother cannot communicate in docker 0 network.

- docker run -d --name tester1 busybox:1.28 sleep 3600
- docker exec 7216021a4778 ping a780f294a2c5 – not pinging
- docker exec 7216021a4778 ping hopeful_greider – not pinging
  docker exec 7216021a4778 ping 172.17.0.2 - pinging
-

24) Drawback to Docker0 Network

By using docker0 network, one container can communicate to another container by using IP address.

Ip address also frequently changing and not permenet.

25) Creating our custom network

- docker network create koti
- docker network ls
- docker inspect koti
- docker run -d -P --name app --net koti nginx
- docker run -d -P --name mybox --net koti busybox:1.28 sleep 3600

docker network connect koti 62fb696ce79e (container id)

To see the list of container in a network:# docker inspect koti

Types Networks in Docker

1)Bridge

2) host

3) none

4) overlay

Bridge : by default docker0 network is related Bridge and bridge network confined to one machine only.

Host : if container Ip running as docker host ip we can use host network.

- Docker run -d -p –net host ngnix

Take the public IP of the Dockerhost and access using default port.

13.233.156.196:80

13.127.218.160:80

We get the nginx page

Null Network

Refers to no network.

If we do not want the container to get exposed to the world.

docker run -d -P --net none redis

Overlay network :

In realtime, we know, container will be running on multiple machines (docker swarm).

When container present in one machine, want to communicate to container in another machine

docker info

We can see - Swarm in inactive

Swarm: inactive

I want to initialize swarm

docker swarm init

Now, lets see the list of networks

docker network ls

Observer, a new network is created of type "overlay"

Name of the network is ingress

One machine container can communicate to another machine container using overlay network.

Lets inspect ingress network.

docker inspect ingress

Observe the IP address, Lets take a note of it.

"IPv4Address": "10.0.0.2

Now, when we create container using service concept,

The container uses overlay network.

Docker swarm uses overlay network.

docker service create --name s1 --replicas 5 -p 1234:80 nginx

As we have one machine right now.

All the container will be running in the same machine (Manager)

docker ps

I want to know the IP addres of the 1st container.

Take note of the 1st container ID

0ac84d7cdd37

docker inspect 0ac84d7cdd37

Observer the IP address is

"IPAddress": "10.0.0.7"

It has taken overlay network series.

So, swarm uses existing overlay network for communication between containers on different machines.

++++++++++++++++++++++++++

Can we create our own overlay network?

Yes!!

Lets create

docker network create ol1 --driver overlay

docker network ls (to see the list of networks)

We can see ol1 network is created of type overlay

docker inspect ol1 (To know the series it is taking)

"Subnet": 10.0.1.0

It has taken 1.0 series

++++++++++++++++++++++

Lets delete the existing service

docker service rm s1

Lets remove all the existing containers

docker rm -f `docker ps -aq`

Lets create service on ol1 network

docker service create --name mynginx --replicas 5 --network ol1 -p 1224:80   nginx

docker ps

Take note of the 1st container id

814d5db02f47

docker inspect 814d5db02f47