Container orchestration

------------------------

This is the process of running docker containers in a distributed environment, on multiple docker host machines.

All these containers can have a single service running on them and they share the resources between eachother, even running on different host machines.

Docker swarm is the tool used for performing container orchestration

Advantages

--------------

1) Load balancing

2) scaling of containers

3) performing rolling updates

4) handling failover scenarios

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

Machine on which docker swarm is installed is called as manager.

Other machines are called as workers.

Lets create 3 machines

Name is as Manager, Worker1, Worker2

All the above machines should have docker installed in it.

Install docker using get.docker.com

( Optional step to change the prompt )

After installing docker in the 1st machine ( Manager ), Lets change the host name.

Host name will be available in the file hostname. We will change the hostname to manager.

# vim /etc/hostname

Manager

:wq

After changing the hostname, lets restart the machine

# init 6

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

Similary repeat the same in worker1 and worker2

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

Connect to Manager, install docker swarm in it.

$ sudo su -

Command to install docker swarm in manager machine

# docker swarm init --advertise-addr private\_ip\_of\_manager

# docker swarm init --advertise-addr 172.31.27.151

Please read the log messages

Now, we need to add workers to manager

Copy the docker swarm join command in the log and run in the worker1 and worker2

Open another gitbash terminal, connect to worker1

sudo su -

# docker swarm join --token SWMTKN-1-0etsmfa26vreeytq278q8ohhi73il7j1lpnrzzlowuld1r8yex-9x04pjmiq85jxjzjayzlglh1c 172.31.27.151:2377

Repeat for worker2

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

TO see the no of nodes from the manager

Manager # docker node ls ( we can see manager, worker1 and worker 2)

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

Load balancing:

Each docker container is designed to withstand a specific user load.

When the load increases, we can replica containers in docker swarm and distribute the load.

Ex: Start tomcat in docker swarm with 5 replicas and name it as webserver.

Manager# docker service create --name webserver -p 9090:8080 --replicas 5 tomee

( 5 conainers with the same service, distributed load in 3 machines)

How to see where thay are running?

Manager# docker service ps webserver

Lets take the note

Manager - 1 container

Worker1 - 2 container

Worker2 - 2 container

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

Note: Only one tomcat is running and load is shrared to 3 machines

Lets check

public\_ip\_manager:9090 ( Will show tomcat page )

public\_ip\_worker1:9090 ( Will show tomcat page )

public\_ip\_worker2:9090 ( Will show tomcat page )

13.235.76.249:9090

13.233.121.155:9090

13.126.47.212:9090

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

Ex 2: Start mysql in docker swarm with 3 replicas.

Manager# docker service create --name mydb --replicas 3 -e MYSQL\_ROOT\_PASSWORD=sunil mysql:5

How to see where thay are running?

Manager# docker service ps mydb

To know the total no of services running in docker swarm

Manager# docker service ls

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

If you delete a container, it will create another container.

Now,

Manager# docker service ps mydb

We can see one container is running in Manager machine

I want to delete the container which is running in manager

Manager# docker container ls

( we can see 1 mysql container, 1 tomcat container )

Take note of the container\_id of mysql

67238f47bc60

TO delete the container

# docker rm -f 67238f47bc60

Now lets check the mydb service

# docker service ps mydb ( we can see one service is failed, automatically 2nd service is started)

At anypoint of time, 3 container will be running.

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

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.

Ex 3: Start nginx with 5 replicas, later scale the services to 10.

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

# docker service ps appserver

Command to scale

# docker service scale appserver=10

To check

# docker service ps appserver

Now I want only two containers

# docker service scale appserver=2

To check

# docker service ps appserver

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

To remove a node from the docker swarm

Two ways

1) Manager can drain

2) Node can leave

To see the list of nodes

# docker node ls

# docker node update --availability drain Worker1

All the container running in Worker1 , will be migrated to Worker2 or manager.

# docker service ps mydb

# docker node ls

To add the node

# docker node update --availability active Worker1

# docker node ls

2nd Way ( Node can leave )

----------------

Lets Connect to worker2 from git bash

Worker2# docker swarm leave

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

TO see the list of services

# docker service ls

TO delete the services

Manager# docker service rm appserver mydb webserver

Rolling Updates

--------------------

The services running in docker swarm, can be updated to any other version

without any downtime.

This is perfomed by docker swarm by updating one replica after another. This is called as rolling update.

Ex: Create redis 3 service with 6 replicas. Update from redis 3 to redis 4 version.

docker service create --name myredis --replicas 6 redis:3

To check the replicas

docker service ps myredis

To update

# docker service update --image redis:4 myredis

docker service ps myredis

I want to display running containers not shutdown containers

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

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

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

Performing rolling rollback , to downgrade to redis:3 version

# docker service update --rollback myredis

To check redis:3 is running with 6 replicas and other version are shutdown.

# docker service ps myredis

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

TO add new nodes, in future, we need to docker swarm join command.

To generate the command

# docker swarm join-token worker ( We will get the command )

docker swarm join --token SWMTKN-1-0etsmfa26vreeytq278q8ohhi73il7j1lpnrzzlowuld1r8yex-9x04pjmiq85jxjzjayzlglh1c 172.31.27.151:2377

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

To add a new machine as a manager

# docker swarm join-token manager

docker swarm join --token SWMTKN-1-5wbamgr8x7gxabwtlm1j1i91bm5ilzotgna6bc0edubtwtjxi1-3jmzi67qdn5aawvielkcng2e4 172.31.34.112:2377

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

If there are two managers, one will be leader

# docker node ls ( we can see who is the leader )

Decision of which is machine should be leader is automatic.

If one manager goes down, other manager automatically become leader.

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

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

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