Types of networks in Docker

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

Actually, they are not networks. They are network drivers.

Docker follows CNM ( Container Network Model ), By using which container capabilities come to docker by defualt.

There are Following network drivers in-built

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

1) Bridge

2) Host

3) None

4) Overlay

Last session, we have experienced two types of networks.

1. Docker0

2. sunil ( Custom network )

By default docker0 is of type bridge network.

In bridge networks, container on one machine can communicate to another container, in the same machine.

By default type is bridge.

# docker network ls

We can see "sunil" network is of type bridge.

Bridge network is confined to one machine only.

Host Network

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

Assume, I am running a database.

I want to use machine IP as the container IP

It is possible by using Host network.

We already have host network available.

# docker network ls

We can see one host network

# docker run -d -P --net host nginx

# docker container ps

Observe: There is no port no

Eventhough we have mentioned -P, is has not published port.

We can access directly using IP without port, to connect to the container.

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

13.233.156.196:80

13.233.149.155:80

We get the nginx page

Ok. fine.

But, can we create the same container again?

No!

# docker run -d -P --net host nginx

As, 80 port number is already published

When we create another container,It will not be in running.

# docker ps

We can see only one container.

# docker ps -a

We can see the latest container is exited.

To know, Why the container is exited?

Take note of the container id, which is exited

f70267b40b4d

# docker logs f70267b40b4d

We can see clearly as " Address already in use"

So, with host network, one type of image we can use only once.

None 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

# docker ps

Take note of container id

fadebe0ad3cf

# docker inspect fadebe0ad3cf

We do not see any IP for the container

"IPAddress": ""

When none network is used?

When we use orchestration tools like Kubernetes or openshift.

If you want orchestration tools to take of networking, in such case we do not want docker networking.In such case we use none network of docker.

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

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

eaa1e645ae48

# docker inspect 96631cf17237

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 10 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

We can see the network as ol1. And the Ip address series.

"ol1": {

"IPAMConfig": {

"IPv4Address": "10.0.1.7"

We generally used bridge and overlay networks.

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