**Bridge networking - using default bridge network-(not recommended for production) see user defined bridge networks**

**(Connect two containers to the bridge network)**

1.

Docker sets up bridge network by default.

It is connected to the docker0 interface of the host

2.

Can use this bridge network to connect stand alone containers

3.

This network is not good for production systems

**Tutorial**

1.

List current networks

sarala@sarala-pc:~/SARALA/BL/spark-2.4docker/docker-spark-master$ sudo docker network ls

NETWORK ID NAME DRIVER SCOPE

1f8ada3b2cce bridge bridge local

1ba4df7d51a1 host host local

adbff08e1d50 none null local

2.

Host & none networks are not fully fledged networks

But used to start the following containers

1. Containers directly bind to the docker daemon host’s networking stack
2. Containers with no network devices

3.

Start 2 alphine containers running ash

Ash - alphine’s default shell rather than bash

sarala@sarala-pc:~/SARALA/BL/spark-2.4docker/docker-spark-master$ sudo docker run -dit --name alpine1 alpine ash

2891ed7dd03c8090abb29035da66ecf341f102512cc8fdc01eb79934da377cdc

sarala@sarala-pc:~/SARALA/BL/spark-2.4docker/docker-spark-master$ sudo docker run -dit --name alpine2 alpine ash

6e6517f51c0d8f01f2c5c6912ceda3d2255651bfc942e5de662bf57e6726899c

-d

Start the container detached (in the background)

Since you are starting it detached you won’t be connected to the container right away

Instead container’s id will be printed

Because you have not specified the network flags containers got connected to the default bridge network

-i

Interactive

With the ability to type in to it

-t

With TTY

Can see the input and output

4.

Check the running containers

sarala@sarala-pc:~/SARALA/BL/spark-2.4docker/docker-spark-master$ sudo docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

6e6517f51c0d alpine "ash" 6 minutes ago Up 5 minutes alpine2

2891ed7dd03c alpine "ash" 7 minutes ago Up 7 minutes alpine1

5.

Inspect the bridge network to see what containers are connected to it.

sarala@sarala-pc:~/SARALA/BL/spark-2.4docker/docker-spark-master$ sudo docker network inspect bridge

[

{

"Name": "bridge",

"Id": "1f8ada3b2ccefcc2fed6351d87a5453545725377fbabbd57daf620ca732413a7",

"Created": "2019-01-12T12:42:49.223610422+05:30",

"Scope": "local",

"Driver": "bridge",

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": null,

"Config": [

{

"Subnet": "172.17.0.0/16",

"Gateway": "172.17.0.1" → ip address of the gateway between the docker host & the bridge network

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

"Containers": {

"2891ed7dd03c8090abb29035da66ecf341f102512cc8fdc01eb79934da377cdc": {

"Name": "alpine1",

"EndpointID": "2685b30a5cb3c7b9ed22d05fdd885aaf007b17ec1e6973704d151f33a904e7e5",

"MacAddress": "02:42:ac:11:00:02",

"IPv4Address": "172.17.0.2/16", → alphine 1 ip address

"IPv6Address": ""

},

"6e6517f51c0d8f01f2c5c6912ceda3d2255651bfc942e5de662bf57e6726899c": {

"Name": "alpine2",

"EndpointID": "a3e153053aaef1d1a155cf08956b525b73d704655e473fc3db8ff56e922a4914",

"MacAddress": "02:42:ac:11:00:03",

"IPv4Address": "172.17.0.3/16", → alphine 2 ip address

"IPv6Address": ""

}

},

"Options": {

"com.docker.network.bridge.default\_bridge": "true",

"com.docker.network.bridge.enable\_icc": "true",

"com.docker.network.bridge.enable\_ip\_masquerade": "true",

"com.docker.network.bridge.host\_binding\_ipv4": "0.0.0.0",

"com.docker.network.bridge.name": "docker0",

"com.docker.network.driver.mtu": "1500"

},

"Labels": {}

}

]

sarala@sarala-pc:~/SARALA/BL/spark-2.4docker/docker-spark-master$

6.

Both the alphine1 & alphine2 are running in the background

Use docker attach command to connect to the container

Connect to alphine1 container

sarala@sarala-pc:~/SARALA/BL/spark-2.4docker/docker-spark-master$ sudo docker attach alpine1

/ # ls

bin etc lib mnt root sbin sys usr

dev home media proc run srv tmp var

/ #

Now you are the root user with in the container

# indicates it.

7.

Use the following command to see the network interfaces from with in the container

/ # ip addr show

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

28: eth0@if29: <BROADCAST,MULTICAST,UP,LOWER\_UP,M-DOWN> mtu 1500 qdisc noqueue state UP

link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff

inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0

valid\_lft forever preferred\_lft forever

/ #

There are 2 network interfaces listed.

I.

Loop back device

II.

Second ip address= alphine1 ip address

8.

From alphine1 connect to google.com

/ # ping -c 2 google.com

PING google.com (216.58.196.78): 56 data bytes

64 bytes from 216.58.196.78: seq=0 ttl=54 time=69.983 ms

64 bytes from 216.58.196.78: seq=1 ttl=54 time=77.898 ms

--- google.com ping statistics ---

2 packets transmitted, 2 packets received, 0% packet loss

round-trip min/avg/max = 69.983/73.940/77.898 ms

-c 2

Limits the command to 2 ping attempts

Otherwise it will ping infinitely until we stop

9.

Alphine2 container’s ip address → 172.17.0.3

Now connect to aphine2 container from alphine1 container

/ # ping -c 2 172.17.0.3

PING 172.17.0.3 (172.17.0.3): 56 data bytes

64 bytes from 172.17.0.3: seq=0 ttl=64 time=0.125 ms

64 bytes from 172.17.0.3: seq=1 ttl=64 time=0.075 ms

--- 172.17.0.3 ping statistics ---

2 packets transmitted, 2 packets received, 0% packet loss

round-trip min/avg/max = 0.075/0.100/0.125 ms

/ #

10.

Can’t ping using the container name

/ # ping -c 2 alpine2

ping: bad address 'alpine2'

/ #

11.

Detach from alphine1 without stopping it.

CTRL p,q

Hold down CTRL , type p followed by q

12.

Stop & remove both the containers

sarala@sarala-pc:~/SARALA/BL/spark-2.4docker/docker-spark-master$ sudo docker container stop alpine1 alpine2

[sudo] password for sarala:

alpine1

alpine2

sarala@sarala-pc:~/SARALA/BL/spark-2.4docker/docker-spark-master$ sudo docker container rm alpine1 alpine2

alpine1

alpine2