## Overview

Docker has a number of ways to provide network connectivity for containers. The capabilities vary based on what is chosen. Here we will explore a couple options via Docker tutorials.

<https://docs.docker.com/network/network-tutorial-standalone>

<https://docs.docker.com/network/network-tutorial-host/>

**1.** Steps completed:

docker run -dit --name alpine1 alpine ash

docker run -dit --name alpine2 alpine ash

(Hashes returned)

**2.** docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

bbff153bddfa alpine "ash" 55 seconds ago Up 54 seconds alpine2

dd4d85438bff alpine "ash" About a minute ago Up About a minute alpine1

**3.** Inspect network info

CORP\croy@a-2yuewitruhybn containerize]$ docker network inspect bridge

[

{

"Name": "bridge",

"Id": "2df9a79dedc4d9197ee0a6b784e8050556fd79896163c5dd3d206bd5c1d87126",

"Created": "2022-10-24T16:53:34.98479577-07:00",

"Scope": "local",

"Driver": "bridge",

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": null,

"Config": [

{

"Subnet": "172.17.0.0/16",

"Gateway": "172.17.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

"Containers": {

"bbff153bddfabe18d5efb4319eab1286dc93056bd2ad5c1fe5b3c2977967eaee": {

"Name": "alpine2",

"EndpointID": "1f5bc4c2e38d4e68a8c15617106c9b3111e9b928589ccf43ecfbb02b3a04bd2b",

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

"IPv4Address": "172.17.0.3/16",

"IPv6Address": ""

},

"dd4d85438bff393cd0a2e9f5bcad491f7f3c2e03bb8c41b702c95eda88c74515": {

"Name": "alpine1",

"EndpointID": "0d469594d43bd0e5d057f0d9aa6779fbdb3f89cdae985886f3def1c6beb9cdd9",

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

"IPv4Address": "172.17.0.2/16",

"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": {}

}

]

**4.** Attach to container

Docker attach alpine1

**5.** and check connectivity

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

45: eth0@if46: <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

In alpine1:

ping -c 2 google.com

PING google.com (172.253.62.113): 56 data bytes

64 bytes from 172.253.62.113: seq=0 ttl=95 time=2.228 ms

64 bytes from 172.253.62.113: seq=1 ttl=95 time=2.675 ms

--- google.com ping statistics ---

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

round-trip min/avg/max = 2.228/2.451/2.675 ms

<<Tested pinging alpine2 by IP, and that worked. However, you can't ping alpine2 from alpine1 by name or vice versa.>>

**7 and 8:** disconnect from and stop / rm alpine1 and alpine1

**~~DONE~~** ~~with standalone portion of the lab – moving on.~~

\* Missed that I was supposed to have done the next section

**Use User-defined bridge networks**

1. Create the alpine-net network.

docker network create --driver bridge alpine-net

2. Result: (docker network ls)

[CORP\croy@a-2yuewitruhybn spring-boot-reactive]$ docker network ls

NETWORK ID NAME DRIVER SCOPE

bbecdbfd7a1f alpine-net bridge local

c16de87c5ccc bridge bridge local

9e43b0e09d85 host host local

652bae6325ad none null local

docker network inspect alpine-net

[CORP\croy@a-2yuewitruhybn spring-boot-reactive]$ docker network inspect alpine-net

[

{

"Name": "alpine-net",

"Id": "bbecdbfd7a1fd67751c040a6f5d1e4d1454a1cccc5aee0182c08b8bb62f08956",

"Created": "2022-10-27T08:56:15.990473818-07:00",

"Scope": "local",

"Driver": "bridge",

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": {},

"Config": [

{

"Subnet": "172.18.0.0/16",

"Gateway": "172.18.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

"Containers": {},

"Options": {},

"Labels": {}

}

]

3. Create 4 containers

docker run -dit --name alpine1 --network alpine-net alpine ash

docker run -dit --name alpine2 --network alpine-net alpine ash

docker run -dit --name alpine3 alpine ash

docker run -dit --name alpine4 --network alpine-net alpine ash

docker network connect bridge alpine4

Confirm they’re running

[CORP\croy@a-2yuewitruhybn spring-boot-reactive]$ docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

16df3ea8aa10 alpine "ash" 2 minutes ago Up 2 minutes alpine2

64e484af876c alpine "ash" 2 minutes ago Up 2 minutes alpine1

3c30bae494bf alpine "ash" 5 minutes ago Up 5 minutes alpine4

61cb500824fa alpine "ash" 5 minutes ago Up 5 minutes alpine3

4. Inspect the networks:

[CORP\croy@a-2yuewitruhybn spring-boot-reactive]$ docker network inspect bridge

[

{

"Name": "bridge",

"Id": "c16de87c5cccf0a18853c6b48fc69577b3c2fa7219f0b75fc4abc1e3bfcb98ec",

"Created": "2022-10-26T16:29:53.897989052-07:00",

"Scope": "local",

"Driver": "bridge",

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": null,

"Config": [

{

"Subnet": "172.17.0.0/16",

"Gateway": "172.17.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

"Containers": {

"3c30bae494bf06e5e73e202f720aa0f3c5f2cb3a446fd7cff81baca8eedc3ff8": {

"Name": "alpine4",

"EndpointID": "dc33f700d0aac34ae12a4531565d954fa6eeeaeaa934a023db129845b0bebd7a",

"MacAddress": "02:42:ac:11:00:04",

"IPv4Address": "172.17.0.4/16",

"IPv6Address": ""

},

"61cb500824fa916008be68d53e22548972618bdbc6e87b5d4bbac23b704d3f4c": {

"Name": "alpine3",

"EndpointID": "5bd5887f9bb556bfddab5e3e91935d172431c28dedeec5c2e996a77ca97bdbd8",

"MacAddress": "02:42:ac:11:00:06",

"IPv4Address": "172.17.0.6/16",

"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": {}

}

]

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

[CORP\croy@a-2yuewitruhybn spring-boot-reactive]$ docker network inspect alpine-net

[

{

"Name": "alpine-net",

"Id": "bbecdbfd7a1fd67751c040a6f5d1e4d1454a1cccc5aee0182c08b8bb62f08956",

"Created": "2022-10-27T08:56:15.990473818-07:00",

"Scope": "local",

"Driver": "bridge",

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": {},

"Config": [

{

"Subnet": "172.18.0.0/16",

"Gateway": "172.18.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

"Containers": {

"16df3ea8aa10a0cad20bc01d5b3d4e01a3fd327b26cf968dbd871fb16d9524ac": {

"Name": "alpine2",

"EndpointID": "8bcf733b5c6834aa5d59932780157d9f3aff1af51541d3c64e74431713b7ae1e",

"MacAddress": "02:42:ac:12:00:04",

"IPv4Address": "172.18.0.4/16",

"IPv6Address": ""

},

"3c30bae494bf06e5e73e202f720aa0f3c5f2cb3a446fd7cff81baca8eedc3ff8": {

"Name": "alpine4",

"EndpointID": "41bde80b8aeb740c4f1ce15a2cb43ab903db96e96e69cf4b0799b83672ed21d4",

"MacAddress": "02:42:ac:12:00:02",

"IPv4Address": "172.18.0.2/16",

"IPv6Address": ""

},

"64e484af876c811c17de3cfb0caf8929ca60c6c261555c75e92fe8642af5a073": {

"Name": "alpine1",

"EndpointID": "bef93ee502f4926b46245b2a2131ef44b27f0cbb2be219cf0c775e69e09e8032",

"MacAddress": "02:42:ac:12:00:03",

"IPv4Address": "172.18.0.3/16",

"IPv6Address": ""

}

},

"Options": {},

"Labels": {}

}

]

5. Test **automatic service discovery**

[CORP\croy@a-2yuewitruhybn labs]$ docker container attach alpine1

/ # ping -c 2 alpine2

PING alpine2 (172.18.0.4): 56 data bytes

64 bytes from 172.18.0.4: seq=0 ttl=64 time=0.126 ms

64 bytes from 172.18.0.4: seq=1 ttl=64 time=0.121 ms

--- alpine2 ping statistics ---

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

round-trip min/avg/max = 0.121/0.123/0.126 ms

/ # ping -c 2 alpine4

PING alpine4 (172.18.0.2): 56 data bytes

64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.111 ms

64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.080 ms

--- alpine4 ping statistics ---

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

round-trip min/avg/max = 0.080/0.095/0.111 ms

/ # ping -c 2 alpine1

PING alpine1 (172.18.0.3): 56 data bytes

64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.045 ms

64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.060 ms

--- alpine1 ping statistics ---

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

round-trip min/avg/max = 0.045/0.052/0.060 ms

6. Can you ping/connect alpine3 across these networks? Nope.

/ # ping -c 2 alpine3

ping: bad address 'alpine3'

/ # ping -c 2 172.17.0.6

PING 172.17.0.6 (172.17.0.6): 56 data bytes

--- 172.17.0.6 ping statistics ---

2 packets transmitted, 0 packets received, 100% packet loss

(lab says it should be 172.17.0.2 but I’ve been messing around a lot so I ended up with a different address for alpine3)

7. Connect to alpine4 and see where else you can get to from there:

[CORP\croy@a-2yuewitruhybn labs]$ docker container attach alpine4

/ # ping -c 2 alpine1

PING alpine1 (172.18.0.3): 56 data bytes

64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.424 ms

64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.078 ms

--- alpine1 ping statistics ---

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

round-trip min/avg/max = 0.078/0.251/0.424 ms

/ # ping -c 2 alpine2

PING alpine2 (172.18.0.4): 56 data bytes

64 bytes from 172.18.0.4: seq=0 ttl=64 time=0.118 ms

64 bytes from 172.18.0.4: seq=1 ttl=64 time=0.076 ms

--- alpine2 ping statistics ---

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

round-trip min/avg/max = 0.076/0.097/0.118 ms

/ # ping -c 2 alpine3

ping: bad address 'alpine3'

/ # ping -c 2 172.17.0.6

PING 172.17.0.6 (172.17.0.6): 56 data bytes

64 bytes from 172.17.0.6: seq=0 ttl=64 time=0.141 ms

64 bytes from 172.17.0.6: seq=1 ttl=64 time=0.381 ms

--- 172.17.0.6 ping statistics ---

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

round-trip min/avg/max = 0.141/0.261/0.381 ms

/ # ping -c 2 alpine4

PING alpine4 (172.18.0.2): 56 data bytes

64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.037 ms

64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.056 ms

--- alpine4 ping statistics ---

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

round-trip min/avg/max = 0.037/0.046/0.056 ms

/ #

7. Can we ping google from here?

From alpine4: nope

From alpine3: yes

From alpine2: nope

From alpine1: nope

*(I gathered from discussion in class that this was not the expected result. JR thinks it has something to do with the configuration of our WorkSpace. Not going to worry about it.)*

**Next lab: Networking using the host network**

1. Create process as a detached process

* docker run --rm -d --network host --name my\_nginx nginx

1. “Welcome to nginx!” page displayed
2. Verify process bound to port 80

[CORP\croy@a-2yuewitruhybn spring-boot-reactive]$ sudo netstat -tulpn | grep :80

[sudo] password for CORP\croy:

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 650/nginx: master p

tcp 0 0 0.0.0.0:8080 0.0.0.0:\* LISTEN 24402/docker-proxy

tcp 0 0 0.0.0.0:8082 0.0.0.0:\* LISTEN 23379/docker-proxy

tcp6 0 0 127.0.0.1:8007 :::\* LISTEN 4822/java

tcp6 0 0 :::8080 :::\* LISTEN 24407/docker-proxy

tcp6 0 0 :::8082 :::\* LISTEN 23384/docker-proxy

tcp6 0 0 :::8085 :::\* LISTEN 4822/java

**DONE** with host part of the lab