## 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 debt portion of the lab