Docker Networking

And

CNI/M (Container Networking Interface/Model)

Docker Networking Basics

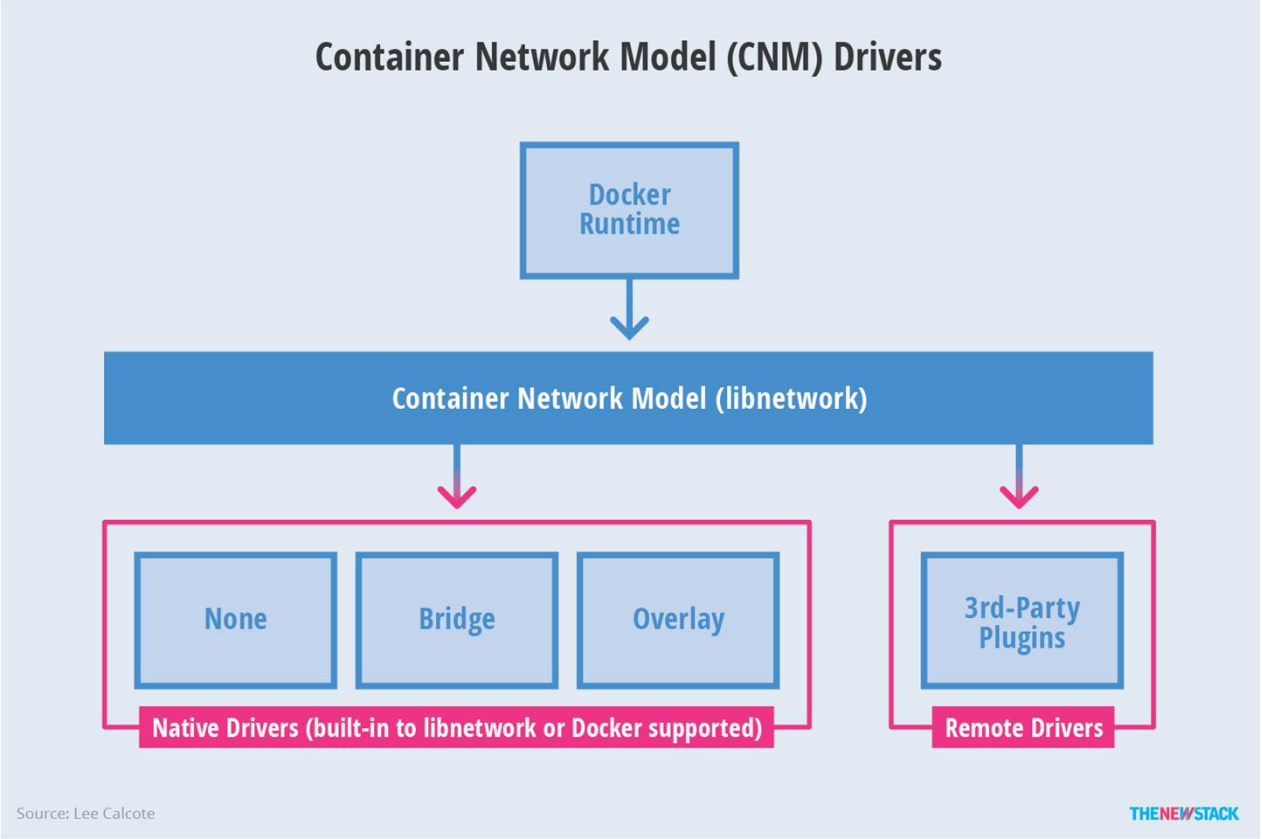
- Docker containers and services need not be aware that they are deployed on Docker or whether their peers are also on Docker and so on and so forth
- As long as each Docker Node or Container is able to discover other nodes and communicate to it, distributed architecture & high availability of Docker networking is achieved

Goals of Container Networking

- Flexibility
- Scalability
- User Friendly
- Cross Platform
- Decentralized
- Secure

Container Network Model (CNM)

- CNM is a spec proposed by Docker
- Formalizes steps required for container networking while providing abstraction to support multiple network drivers



Docker Network Drivers

Pluggable subsystem that can provide core network functionality

- Bridge : default driver, used when applications run in standalone containers that need to communicate
- Host : removes network isolation between containers and Docker host and uses hosts networking services directly. Swarm services uses this

Docker Network Drivers

- None : Disable all networking and usually used in conjunction with custom network driver
- Overlay : connect multiple docker daemons together and enable swarm service to communicate with other daemons
- Macvlan : Assign MAC address to a container and make it appear like a physical device on your network

Default Bridge Network Lab

- First check the available network drivers

>docker network ls

NETWORK ID	NAME	DRIVER	SCOPE
17e324f45964	bridge	bridge	local
6ed54d316334	host	host	local
7092879f2cc8	none	null	local

Default Bridge Network Lab

- Let us create two containers alpine1 and alpine2 by not specifying any network (default is bridge network)

>docker run -dit --name alpine1 alpine ash

>docker run -dit --name alpine2 alpine ash

>docker container ls

- Let us inspect the bridge network for IP addresses

>docker network inspect bridge

Default Bridge Network Lab

- Check the IP address of alpine1, alpine2, subnet and gateway
- Let us attach to the containers and check their IP address

>docker attach alpine1

(INSIDE THE CONTAINER)# ip addr show

- Let us check if we can reach google.com from inside the cont

#ping –c 4 google.com

Default Bridge Network Lab

- Let us now check if we can ping alpine2 by its ip address

(INSIDE THE CONTAINER)# ping –c 4 ALPINE2_IP_ADDRESS

- Let us check if we can ping alpine2 by container-name

(INSIDE THE CONTAINER)# ping –c 4 alpine2

ping: bad address 'alpine2‘

Please remove both containers

CONCLUSION : Both containers can ping each other by their IP, can reach external world but can not identify each other by container name

User defined Bridge Network Lab

- Let us create two containers alpine1 and alpine2 and attach them to a user defined network ‘alpine-net’

>docker network create --driver bridge alpine-net

>docker network inspect alpine-net

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

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

- Check IP address of both containers

>docker network inspect alpine-net

User defined Bridge Network Lab

- Let us create two more containers – alpine3 – which is ONLY on bridge network (default) and alpine4 which is on both alpine-net and default bridge network

>docker run -dit --name alpine3 alpine ash

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

>docker network connect bridge alpine4

- Inspect alpine-net and bridge network

>docker network inspect alpine-net

>docker network inspect bridge

User defined Bridge Network Lab

- Automated Service Discovery : On User defined network (alpine-net), containers can not only connect through IP, they can resolve container name to an IP Address
- Connect to alpine1 and ping alpine2 and alpine4 by their name

>docker attach alpine1

#ping –c 4 alpine2

#ping –c 4 alpine4

Can we ping alpine3 by name?? How about IP??

User defined Bridge Network Lab

- How about alpine4 which is both on alpine-net & default bridge?? Can it reach all containers by name?? IP??
- Attach to alpine4 and check it out
- Stop & remove all containers

>docker stop alpine1 alpine2 alpine3 alpine4

>docker rm alpine1 alpine2 alpine3 alpine4

- Remove network

>docker network rm alpine-net