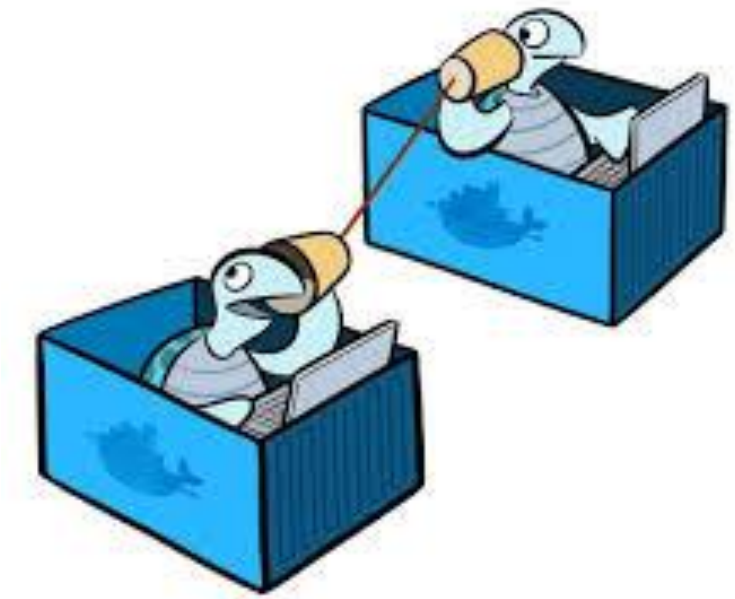

Docker Networking

Vikram Krishnamurthy

What is Docker Networking?

Lets lay down the context

- Containers provide services
- To be able to access these services from outside the container, one would need to understand how the container exposes its service
- Containers may also be distributed across many nodes. In such cases, knowledge of the available networks would allow you to choose the best network that suits your need

Types of Docker networks

What are the networks available to Docker containers?

Describe different types of networks available for docker and containers.

1. None
   - Used when your container doesn't provide a service over the network
   - The container isn't provided with an n/w interface and an IP address

```
docker run -d --network none --name testApp nginx
```

2. Host
   - Used when you would like to use the Docker host's network stack
   - Services running on any port within your container will be directly accessible through the host's IP

```
docker run -d --network host --name nginx_on_host nginx
```

Types of Docker networks

What are the networks available to Docker containers?

3. Bridge - default
   - Used when your container doesn't provide a service on the network
   - Needs "linking" for container DNS resolution

```
docker run -d --network bridge --name nginx_bridge nginx
docker run -d --network bridge -p 8080:80 --name nginx_on_bridge_with_port
nginx
```

4. Bridge – user defined
   - Used when you have a requirement for multiple networks on your Docker host, possibly because you'd like to isolate different deployments of containers on that host

```
docker network create --driver bridge isolated_nw01
docker run -d --network isolated_nw01 -p 8080:80 --name nginx_ud_bridge
nginx
```

Types of Docker networks

What are the networks available to Docker containers?

5. Overlay (swarm) - user defined
   - Multihost networking across Docker hosts that are part of the swarm

6. Overlay (external key-value mechanism) – user defined
   - Multihost networking across Docker hosts that are part of the swarm
   - In `/lib/systemd/system/docker.service`, add:

```
-H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock --cluster-
store=consul://<consul_IP>:8500 --cluster-advertise=<node_IP>:2385
```

   - Reload options:

```
systemctl daemon-reload
```

   - You can now create and use overlay networks:
     - `All nodes:  docker network create --driver overlay mhn01`
     - `Node 1:  docker run -itd --network mhn01 --name=container1 busybox`
     - `Node 2:  docker run -itd --network mhn01 --name=container2 busybox`

Types of Docker networks

What are the networks available to Docker containers?

7. Custom network plugin
   - In case you find the current networking options don't fit your need, you could write your own network plugin using the Docker plugin API
   - References:
     - https://docs.docker.com/engine/extend/plugins_network/
     - Sample "Weave" network plugin:
       - https://www.weave.works/docs/net/latest/introducing-weave/
     - Help with writing plugins:
       - https://docs.docker.com/engine/extend/legacy_plugins/#/writing-a-plugin

Using networks

What are the ways in which we configure containers to be accessible

1. --network
   - Ask Docker to connect your container to a specific network stack

2. "-p“
   - Ask Docker to “publish” specific ports on your container to the host’s ports

3. "-P“
   - Ask Docker to publish “All” the container’s active ports to the host

4. --link
   - Old way of establishing DNS registration for created containers in new containers. Still relevant for default bridge network.

Identifying container networks

Show how a user can identify which network mode is being used by a container

How can we identify which network mode is being used by a container?

1. docker inspect <container>


```
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "fe5dd0a77eb5f2189f4ed59625329cf201e959ec975942e9f089a292add04bbe",
    "HairpinMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "Ports": {},
    "SandboxKey": "/var/run/docker/netns/fe5dd0a77eb5",
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "c000a4c2ca0b6e47e22e03aec4f4b1a9eacb36ade8667b7a504969fd86eb1ec7",
    "Gateway": "172.17.0.1",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "172.17.0.3",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "MacAddress": "02:42:ac:11:00:03",
    "Networks": {
        "bridge": {
            "IPAMConfig": null,
            "Links": null,
            "Aliases": null,
            "NetworkID": "e5b194684a5fcac02cf1b87f2ea080b80ed47f7603ff1b396760ab1bb82b7548",
            "EndpointID": "c000a4c2ca0b6e47e22e03aec4f4b1a9eacb36ade8667b7a504969fd86eb1ec7",
```


2. docker network ls
3. docker network inspect <network_name>

General discussion

General Q and A

Thank you

Vikram.krishnamurthy@microfocus.com