容器网络

前言

- 本章主要介绍Docker容器的3种原生网络驱动：none、host、bridge。学习容器如何实现通信。

目标

- 学完本课程后，您将能够：
  - 描述容器网络模型
  - 描述容器间通信、容器与外部通信原理

目录

Docker Native Network drivers

- Docker提供如下5种原生的Network drivers。

模型	说明
None	none网络中的容器，不能与外部通信。
Host	容器加入到宿主机的Network namespace，容器直接使用宿主机网络。
Bridge	默认网络驱动程序。主要用于多个容器在同一个Docker宿主机上进行通信。
Overlay	Overlay网络可基于Linux网桥和Vxlan，实现跨主机的容器通信。
Macvlan	Macvlan用于跨主机通信场景。

- Docker安装时，自动在host上创建了如下3个网络。

```
[root@localhost ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
83dbc070d5c5        bridge              bridge              local
fa44bc39bef0        host                host                local
800a87290229        none                null                local
```

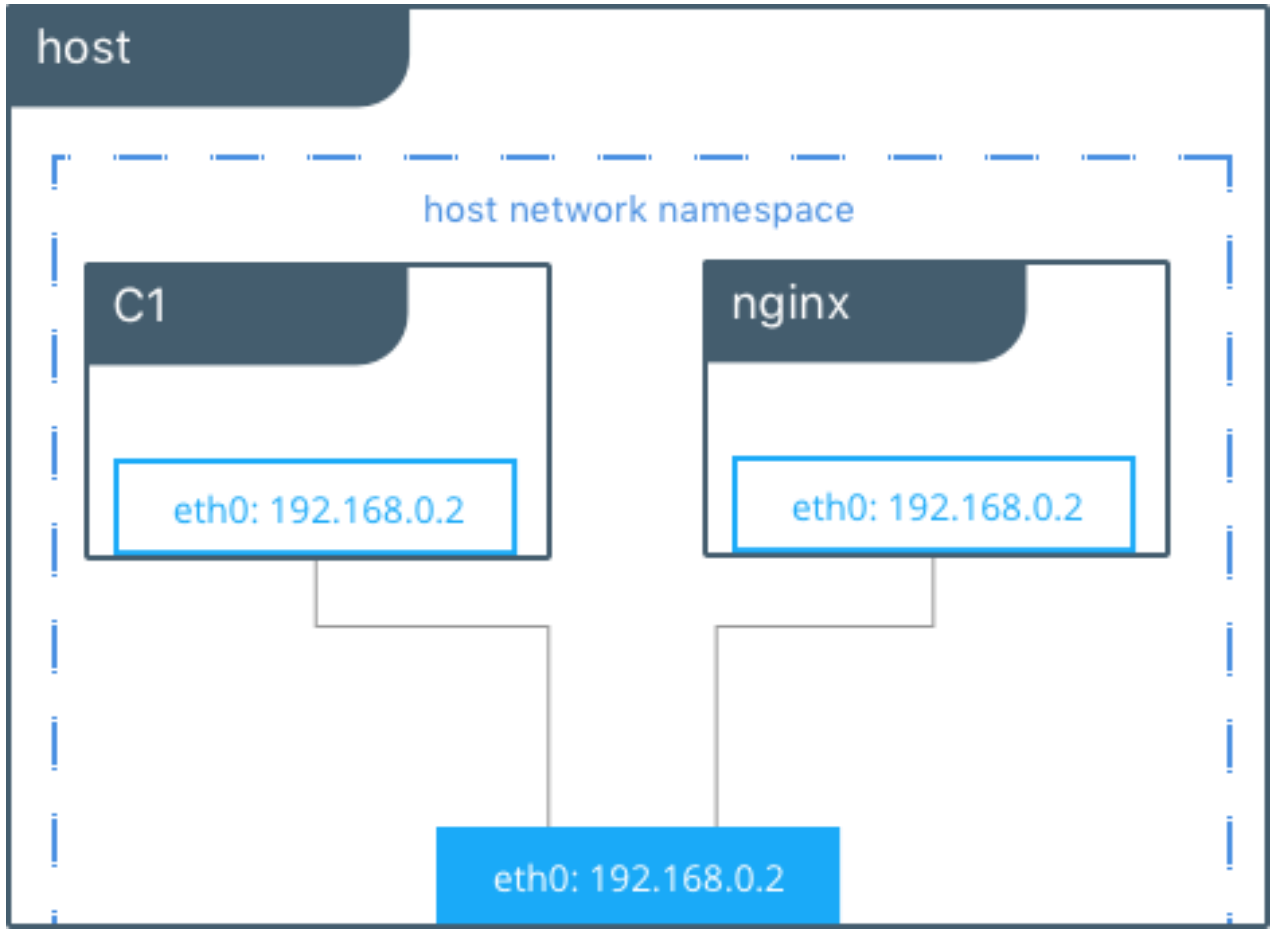
none网络

- none网络的driver类型是null，IPAM字段为空。挂在none网络上的容器只有lo，无法与外界通信。

```
[root@localhost ~]# docker network inspect none
[
    {
        "Name": "none",
        "Id": "800a872902294685a1530c30c85fdf06f7fb517367026ffeec967504b518c482",
        "Created": "2019-07-16T04:48:40.074248621-04:00",
        "Scope": "local",
        "Driver": "null",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": []
        },
```

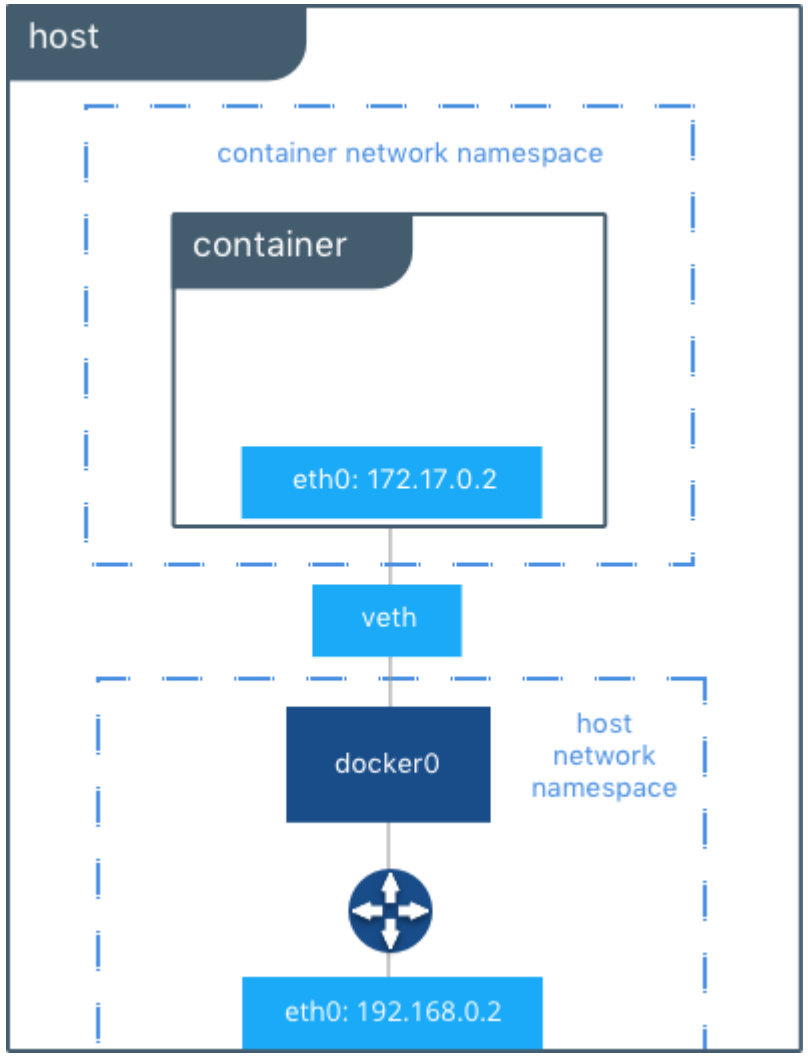
host网络

- 挂在host网络上的容器共享宿主机的network namespace。即容器的网络配置与host网络配置完全一样。



docker0网络 (1)

- docker0网络
  - 容器创建时，默认挂载在docker0上。
  - docker0是一个linux bridge。
  - docker0网络创建时已默认配置了Subnet。



docker0网络 (2)

- 在宿主机上查看docker0。

```
[root@localhost ~]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:d0ff:fefe:bdfb  prefixlen 64  scopeid 0x20<link>
        ether 02:42:d0:fe:bd:fb  txqueuelen 0  (Ethernet)
        RX packets 57041  bytes 3632480 (3.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 71965  bytes 271063262 (258.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- 查看docker0网络配置。

```
[root@localhost ~]# docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "83dbc070d5c5879351d5998fd9c895d1ddc4d8b55cdd37d83f1e9c71a12d8222",
        "Created": "2019-08-06T23:20:32.5497092-04:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
```

docker0网络 (3)

- 在后台运行一个名为httpd1的httpd容器。

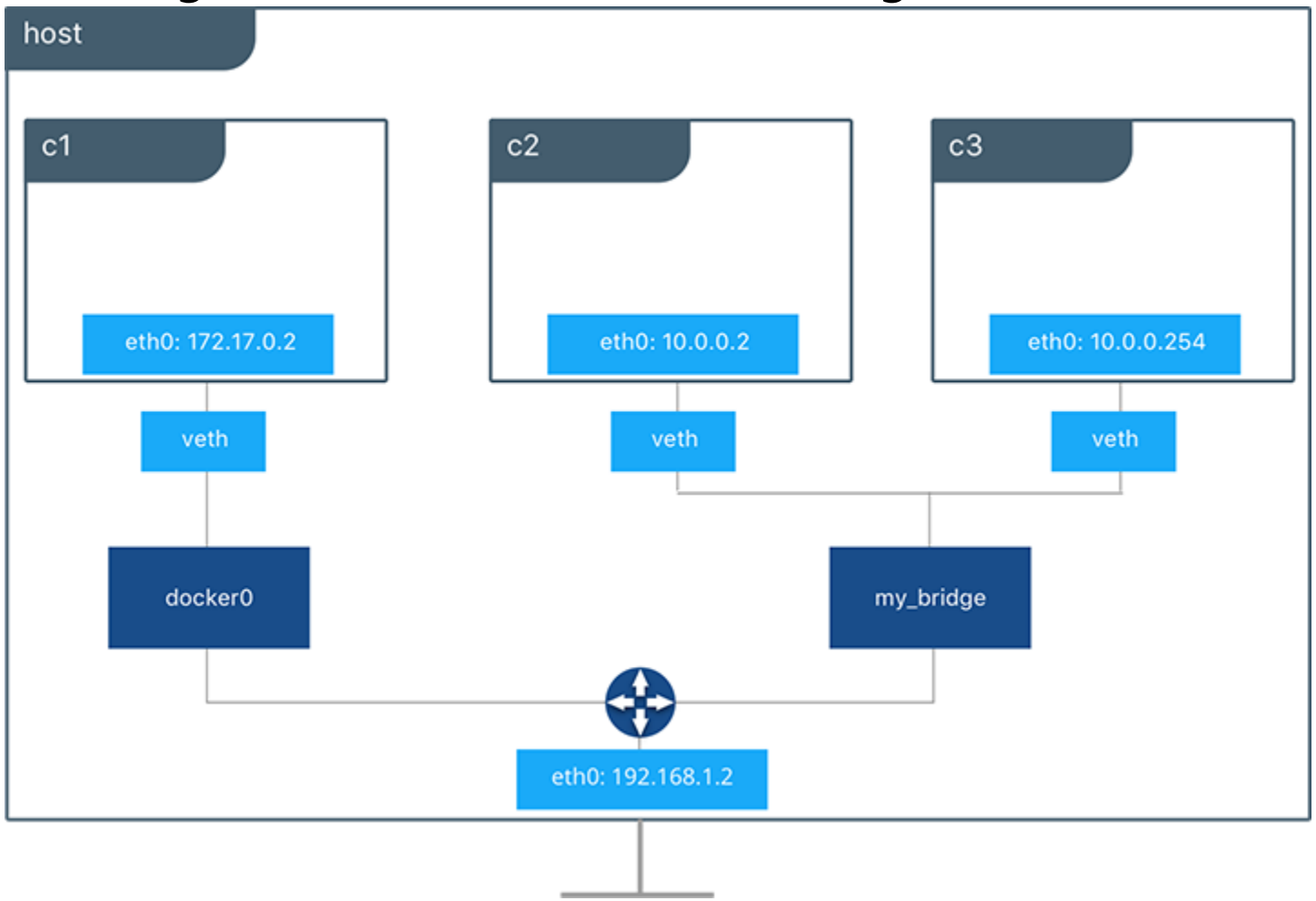
```
[root@localhost ~]# docker run --name httpd1 -dit httpd
b3073023e7a5068e84eb5cb4e2637aacf903fc3b69aa22f7e348ce6a9f570ef6
```

- 查看该容器的网络配置：
  - 该容器挂载的NetworkID=docker0网络的ID；
  - 该容器的网卡ip为172.17.0.2/16，Gateway指向docker0。

```
"Networks": {
    "bridge": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": null,
        "NetworkID": "83dbc070d5c5879351d5998fd9c895d1ddc4d8b55cdd37d83f1e9c71a12d8222",
        "EndpointID": "19ed858de3b404d9ef4ec8eb943605204957337c8dde5481fab9f95602a4b3ba",
        "Gateway": "172.17.0.1",
        "IPAddress": "172.17.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:11:00:02",
        "DriverOpts": null
    }
```

user-defined Bridge网络

- 用户可按需创建bridge网桥，称为user-defined bridge。



user-defined Bridge (1)

- 创建一个user-defined Bridge，命名为net1。

```
[root@localhost ~]# docker network create --driver bridge net1
575cce6c6f9c0f1c45a244458c248b4f1c1b2e1d98efedbf7f3c9e64a8e846cb
```

- 查看net1网桥信息，已自动配置subnet和gateway。

```
[root@localhost ~]# docker network inspect net1
[
    {
        "Name": "net1",
        "Id": "575cce6c6f9c0f1c45a244458c248b4f1c1b2e1d98efedbf7f3c9e64a8e846cb",
        "Created": "2019-08-13T22:01:38.485257766-04:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
```

user-defined Bridge (2)

- 创建第二个网桥，指定IP网段，命名为net2。

```
[root@localhost ~]# docker network create --driver bridge --subnet 172.10.10.0/24 --gateway
172.10.10.1 net2
cef5892af33eb7c62bf64562fbe402b4b158f02192fc542cff530072e2bcfd1a
```

- 启动3个centos容器，分别命名为centos1、centos2、centos3。其中centos1加入到net1，centos2加入net2，centos3加入net2并配置静态IP。

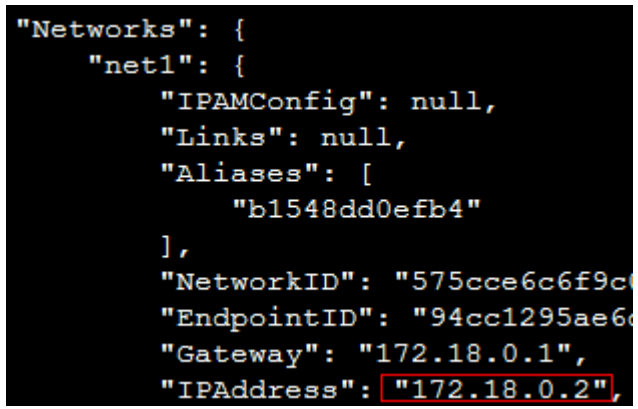
```
[root@localhost ~]# docker run --name centos1 -dit --network=net1 centos
b1548dd0efb4e2d2a4001531a1db203fc9d7c19a1fed327523f8bee9917c3377
```

```
[root@localhost ~]# docker run --name centos2 -dit --network=net2 centos
c5ef65d4569ea8589b07848ded201734d72a84eafe2b5f88c056093447a544ba
```

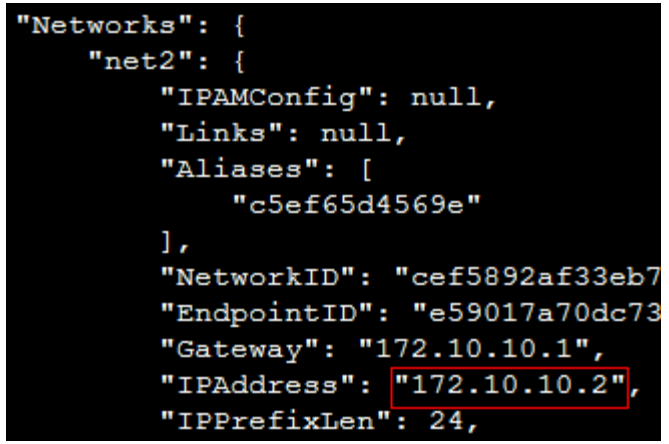
```
[root@localhost ~]# docker run --name centos3 -dit --network=net2 --ip 172.10.10.10 centos
355520d8aba2d1f0241d6e49525e59fe99582a1f9927474f24f014992b8e53a0
```

user-defined Bridge (3)

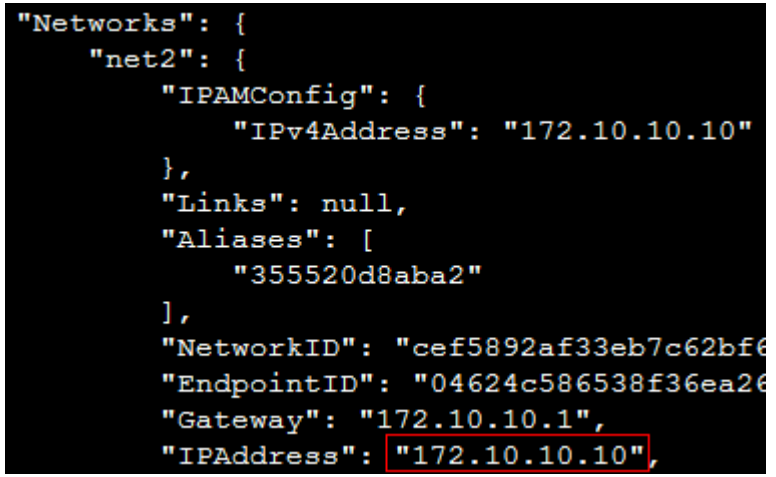
- 查看三个centos容器的IP地址信息。



CentOS1



CentOS2



CentOS3

user-defined Bridge (4)

- 进入容器centos3，进行连通性测试。centos3与centos2可以通信，但centos1不能通信。

```
[root@localhost ~]# docker exec -it centos3 bash
[root@355520d8aba2 /]# ping 172.10.10.2
PING 172.10.10.2 (172.10.10.2) 56(84) bytes of data.
64 bytes from 172.10.10.2: icmp_seq=1 ttl=64 time=0.134 ms
64 bytes from 172.10.10.2: icmp_seq=2 ttl=64 time=0.084 ms
64 bytes from 172.10.10.2: icmp_seq=3 ttl=64 time=0.089 ms
64 bytes from 172.10.10.2: icmp_seq=4 ttl=64 time=0.114 ms
64 bytes from 172.10.10.2: icmp_seq=5 ttl=64 time=0.089 ms
^C
--- 172.10.10.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.084/0.102/0.134/0.019 ms
[root@355520d8aba2 /]# ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2) 56(84) bytes of data.
```

user-defined Bridge (5)

- 为centos1添加一块网卡，加入到net2网络。

```
[root@localhost ~]# docker network connect net2 centos1
```

- 进入centos1，验证连通性。

```
[root@localhost ~]# docker exec -it centos1 bash
[root@b1548dd0efb4 /]# ping 172.10.10.2
PING 172.10.10.2 (172.10.10.2) 56(84) bytes of data.
64 bytes from 172.10.10.2: icmp_seq=1 ttl=64 time=0.127 ms
64 bytes from 172.10.10.2: icmp_seq=2 ttl=64 time=0.118 ms
64 bytes from 172.10.10.2: icmp_seq=3 ttl=64 time=0.101 ms
^C
--- 172.10.10.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.101/0.115/0.127/0.013 ms
[root@b1548dd0efb4 /]# ping 172.10.10.10
PING 172.10.10.10 (172.10.10.10) 56(84) bytes of data.
64 bytes from 172.10.10.10: icmp_seq=1 ttl=64 time=0.128 ms
64 bytes from 172.10.10.10: icmp_seq=2 ttl=64 time=0.108 ms
64 bytes from 172.10.10.10: icmp_seq=3 ttl=64 time=0.084 ms
^C
--- 172.10.10.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.084/0.106/0.128/0.021 ms
```

实验任务

- 实验任务
  - 请按照实验手册1.4部分完成容器网络部分实验。

思考题

1. 使用Host网络时，容器可使用宿主机上已使用的端口对外提供服务。T or F
2. 下列哪一项不是Docker native network drivers类型？（ ）

A. bridge

B. overlay

C. host

D. flannel

本章总结

- none网络
- host网络
- docker0网络
- user-defined bridge网络

学习推荐

- https://docs.docker.com/network/
- https://docs.docker.com/network/iptables/
- https://success.docker.com/article/networking


谢 谢
www.huawei.com