005-容器网络之MACVLAN

1. macvlan用于Docker网络

2. 相同macvlan网络之间的通信

3 不同 macvlan 网络之间的通信

4 总结

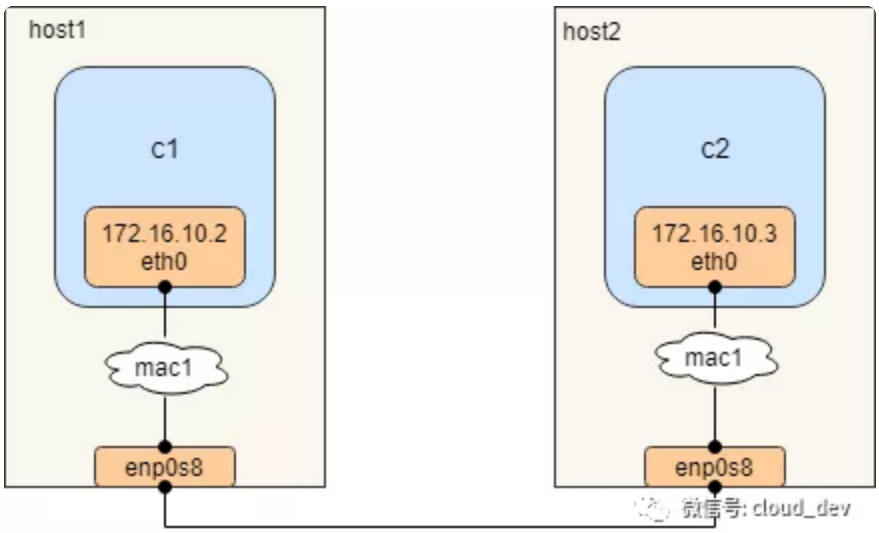
1. macvlan用于Docker网络

在 Docker 中，macvlan 是众多 Docker 网络模型中的一种，并且是一种跨主机的网络模型，作为一种驱动（driver）启用（-d 参数指定），**Docker macvlan 只支持 bridge 模式**。

下面我们做两个实验，分别验证相同 macvlan 网络和不同 macvlan 网络的连通性。

2. 相同macvlan网络之间的通信

首先准备两个主机节点的 Docker 环境，搭建如下拓扑图示：



1. 首先使用 docker network create 分别在两台主机上创建两个 macvlan 网络：

Shell | 复制代码

```
root@ubuntu:~# docker network create -d macvlan --subnet=172.16.10.0/24 --gateway=172.16.10.1 -o parent=enp0s8 mac
```

这条命令中，

- -d 指定 Docker 网络 driver
- --subnet 指定 macvlan 网络所在的网络
- --gateway 指定网关
- -o parent 指定用来分配 macvlan 网络的物理网卡

之后可以看到当前主机的网络环境，其中出现了 macvlan 网络：

```shell
root@ubuntu:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
128956db798a        bridge              bridge              local
19fb1af129e6        host                host                local
2509b3717813        mac1                macvlan             local
d5b0798e725e        none                null                local
```

2. 在 host1 运行容器 c1，并指定使用 macvlan 网络：

```shell
root@ubuntu:~# docker run -itd --name c1 --ip=172.16.10.2 --network mac1 bu
sybox
```

这条命令中，

- --ip 指定容器 c1 使用的 IP，这样做的目的是防止自动分配，造成 IP 冲突
- --network 指定 macvlan 网络

同样在 host2 中运行容器 c2：

```shell
root@ubuntu:~# docker run -itd --name c2 --ip=172.16.10.3 --network mac1 bu
sybox
```

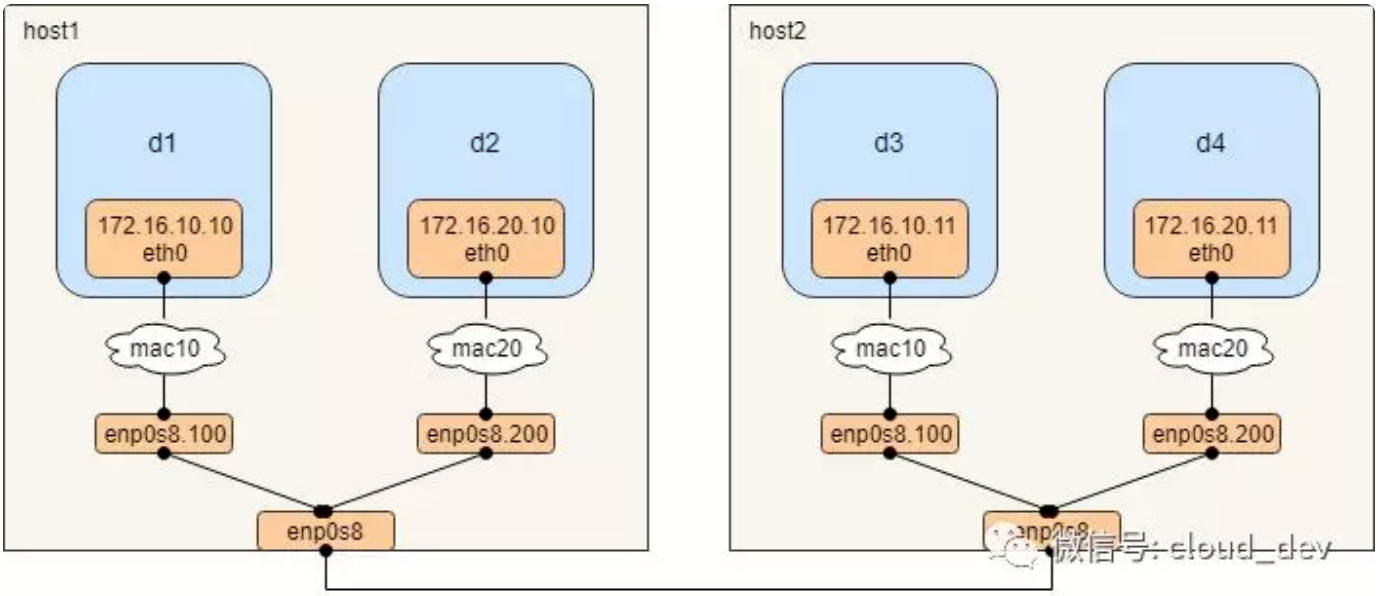
3. 在 host1 c1 中 ping host2 c2：

```shell
root@ubuntu:~# docker exec c1 ping -c 2 172.16.10.3
PING 172.16.10.3 (172.16.10.3): 56 data bytes
64 bytes from 172.16.10.3: seq=0 ttl=64 time=0.641 ms
64 bytes from 172.16.10.3: seq=1 ttl=64 time=0.393 ms

--- 172.16.10.3 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss

round-trip min/avg/max = 0.393/0.517/0.641 ms
```

> 注意：以上的实验都需要物理网卡 enp0s8 开启混杂模式，不然会 ping 不通。

3 不同 macvlan 网络之间的通信

接下来，我们来看看不同 macvlan 网络之间的连通性，搭建以下的拓扑环境：



由于 macvlan 网络会独占物理网卡，也就是说一张物理网卡只能创建一个 macvlan 网络，如果我们想创建多个 macvlan 网络就得用多张网卡，但主机的物理网卡是有限的，怎么办呢?

macvlan 网络也是支持 VLAN 子接口的，所以，我们可以通过 VLAN 技术将一个网口划分出多个子网口，这样就可以基于子网口来创建 macvlan 网络了，下面是具体的创建过程。

1. 首先分别在两台主机上将物理网口 enp0s8 创建出两个 VLAN 子接口。

```shell
# 使用 vconfig 命令在 eth0 配置两个 VLAN
root@ubuntu:~# vconfig add enp0s8 100
root@ubuntu:~# vconfig add enp0s8 200

# 设置 VLAN 的 REORDER_HDR 参数，默认就行了
root@ubuntu:~# vconfig set_flag enp0s8.100 1 1
root@ubuntu:~# vconfig set_flag enp0s8.200 1 1

# 启用接口
root@ubuntu:~# ifconfig enp0s8.100 up
root@ubuntu:~# ifconfig enp0s8.200 up
```

2. 分别在 host1 和 host2 上基于两个 VLAN 子接口创建 2 个 macvlan 网络，mac10 和 mac20。

```shell
root@ubuntu:~# docker network create -d macvlan --subnet=172.16.10.0/24 --gateway=172.16.10.1 -o parent=enp0s8.100 mac10
root@ubuntu:~# docker network create -d macvlan --subnet=172.16.20.0/24 --gateway=172.16.20.1 -o parent=enp0s8.200 mac20
```

3. 分别在 host1 和 host2 上运行容器，并指定不同的 macvlan 网络。

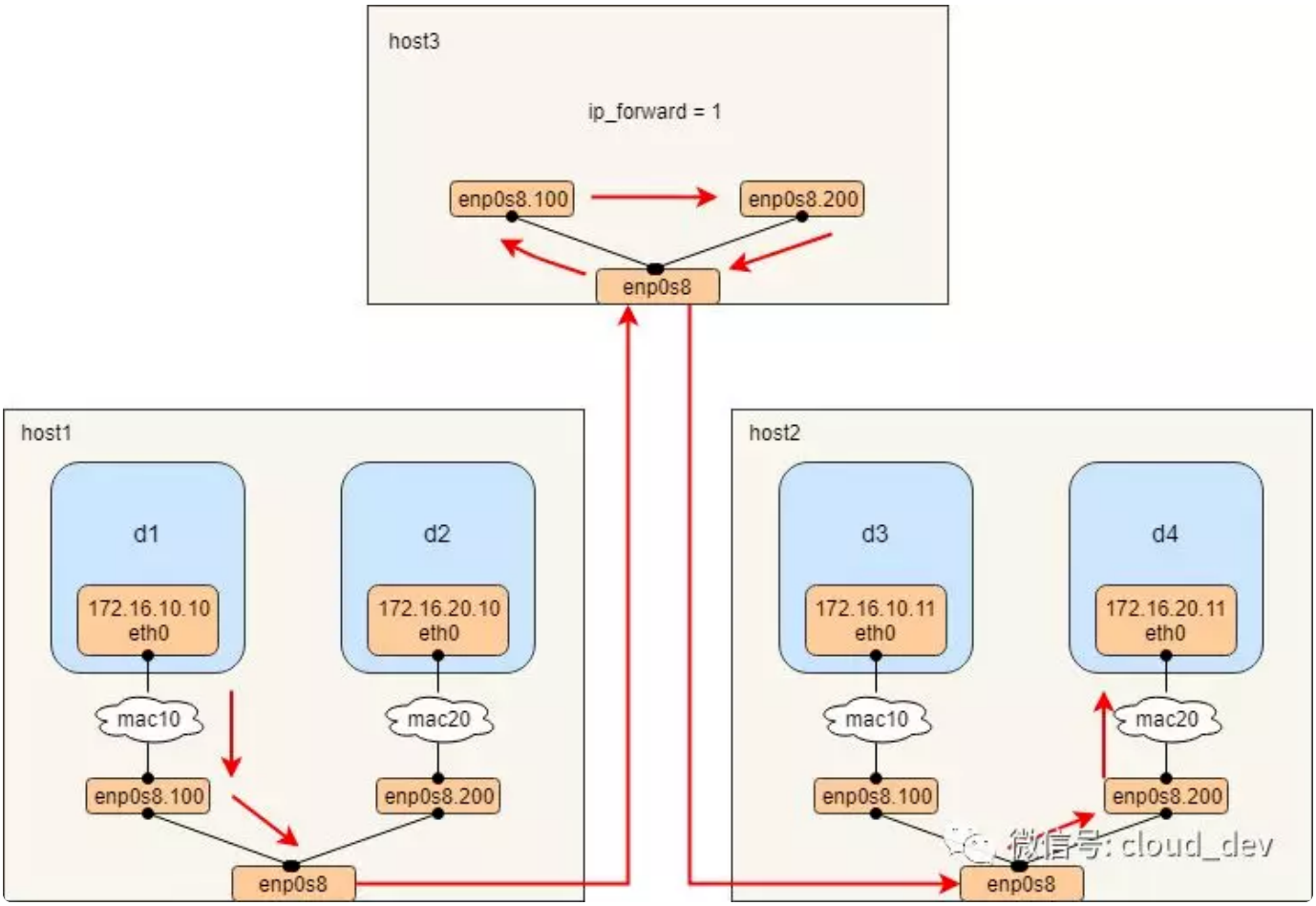
```shell
# host1
root@ubuntu:~# docker run -itd --name d1 --ip=172.16.10.10 --network mac10 busybox
root@ubuntu:~# docker run -itd --name d2 --ip=172.16.20.10 --network mac20 busybox

# host2
root@ubuntu:~# docker run -itd --name d3 --ip=172.16.10.11 --network mac10 busybox
root@ubuntu:~# docker run -itd --name d4 --ip=172.16.20.11 --network mac20 busybox
```

通过验证，d1 和 d3，d2 和 d4 在同一 macvlan 网络下，互相可以 ping 通，d1 和 d2，d1 和 d4 在不同的 macvlan 网络下，互相 ping 不通。

这个原因也很明确，不同 macvlan 网络处于不同的网络，而且通过 VLAN 隔离，自然 ping 不了。

但这也只是在二层上通不了，通过三层的路由是可以通的，我们这就来验证下。

重新找一台主机 host3，通过打开 ip_forward 把它改造成一台路由器（至于为什么可以这样，可以参考我之前的一篇文章），用来打通两个 macvlan 网络，大概的图示如下所示：



1. 首先对 host3 执行 sysctl –w net.ipv4.ip_forward=1 打开路由开关。
2. 然后创建两个 VLAN 子接口，一个作为 macvlan 网络 mac10 的网关，一个作为 mac20 的网关。

Shell | 复制代码

```
[root@localhost ~]# vconfig add enp0s8 100
[root@localhost ~]# vconfig add enp0s8 200
[root@localhost ~]# vconfig set_flag enp0s8.100 1 1
[root@localhost ~]# vconfig set_flag enp0s8.200 1 1

 # 对 vlan 子接口配置网关 IP 并启用
[root@localhost ~]# ifconfig enp0s8.100 172.16.10.1 netmask 255.255.255.0 up
[root@localhost ~]# ifconfig enp0s8.200 172.16.20.1 netmask 255.255.255.0 up
```

3. 这样之后再从 d1 ping d2 和 d4，就可以 ping 通了。

```shell
root@ubuntu:~# docker exec d1 ping -c 2 172.16.20.10
PING 172.16.20.10 (172.16.20.10): 56 data bytes
64 bytes from 172.16.20.10: seq=0 ttl=63 time=0.661 ms
64 bytes from 172.16.20.10: seq=1 ttl=63 time=0.717 ms

--- 172.16.20.10 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.661/0.689/0.717 ms
```

```shell
root@ubuntu:~# docker exec d1 ping -c 2 172.16.20.11
PING 172.16.20.11 (172.16.20.11): 56 data bytes
64 bytes from 172.16.20.11: seq=0 ttl=63 time=0.548 ms
64 bytes from 172.16.20.11: seq=1 ttl=63 time=0.529 ms

--- 172.16.20.11 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.529/0.538/0.548 ms
```

PS：可能有些系统做了安全限制，可能 ping 不通，这时候可以添加以下 iptables 规则，目的是让系统能够转发不通 VLAN 的数据包。

```shell
iptables -t nat -A POSTROUTING -o enp0s8.100 -j MASQUERADE

iptables -t nat -A POSTROUTING -oenp0s8.200 -j MASQUERADE

iptables -A FORWARD -i enp0s8.100 -o enp0s8.200 -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -A FORWARD -i enp0s8.200 -o enp0s8.100 -m state --state RELATED,ESTABLISHED -j ACCEPT


iptables -A FORWARD -i enp0s8.100 -o enp0s8.200 -j ACCEPT

iptables -A FORWARD -i enp0s8.200 -o enp0s8.100 -j ACCEPT
```

为什么配置 VLAN 子接口，配上 IP 就可以通了，我们可以看下路由表就知道了。

首先看容器 d1 的路由：

```shell
root@ubuntu:~# docker exec d1 ip route
default via 172.16.10.1 dev eth0
172.16.10.0/24 dev eth0 scope link  src 172.16.10.10
```

我们在创建容器的时候指定了网关 172.16.10.1，所以数据包自然会被路由到 host3 的接口。再来看下 host3 的路由：

```shell
[root@localhost ~]# ip route
default via 192.168.108.1 dev enp0s3 proto dhcp metric 100
172.16.10.0/24 dev enp0s8.100 proto kernel scope link src 172.16.10.1
172.16.20.0/24 dev enp0s8.200 proto kernel scope link src 172.16.20.1
192.168.56.0/24 dev enp0s8 proto kernel scope link src 192.168.56.122 metric 101
192.168.108.0/24 dev enp0s3 proto kernel scope link src 192.168.108.2 metric 100
```

可以看到，去往 172.16.10.0/24 网段的数据包会从 enp0s8.100 出去，同理 172.16.20.0/24 网段也是，再加上 host3 的 ip_forward 打开，这就打通了两个 macvlan 网络之间的通路。

4 总结

macvlan 是一种网卡虚拟化技术，能够将一张网卡虚拟出多张网卡。

macvlan 的四种通信模式，常用模式是 bridge。

在 Docker 中，macvlan 只支持 bridge 模式。

相同 macvlan 可以通信，不同 macvlan 二层无法通信，可以借助三层路由完成通信。

思考一下：

1. macvlan bridge 和 bridge 的异同点
2. 还有一种类似的技术，多张虚拟网卡共享相同 MAC 地址，但有独立的 IP 地址，这是什么技术?