

Contents

环境准备 Docker (/tags/#Docker) Kubernetes (/tags/#Kubernetes)

使用Macvlan构建Docker网络

通过 MacVLAN 实现 Docker 跨宿主机互联

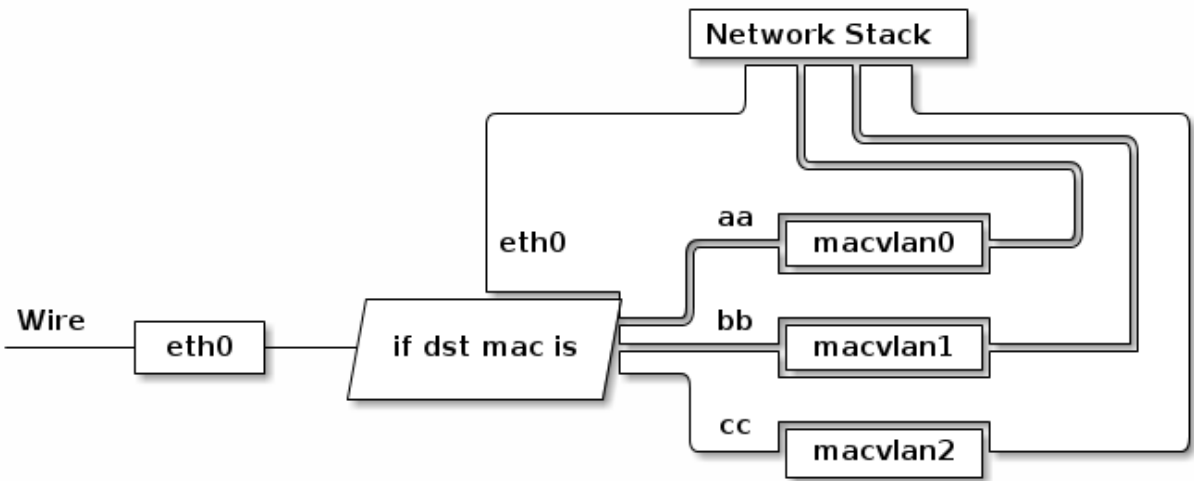
一些问题

作者: Mike on 2017-04-26

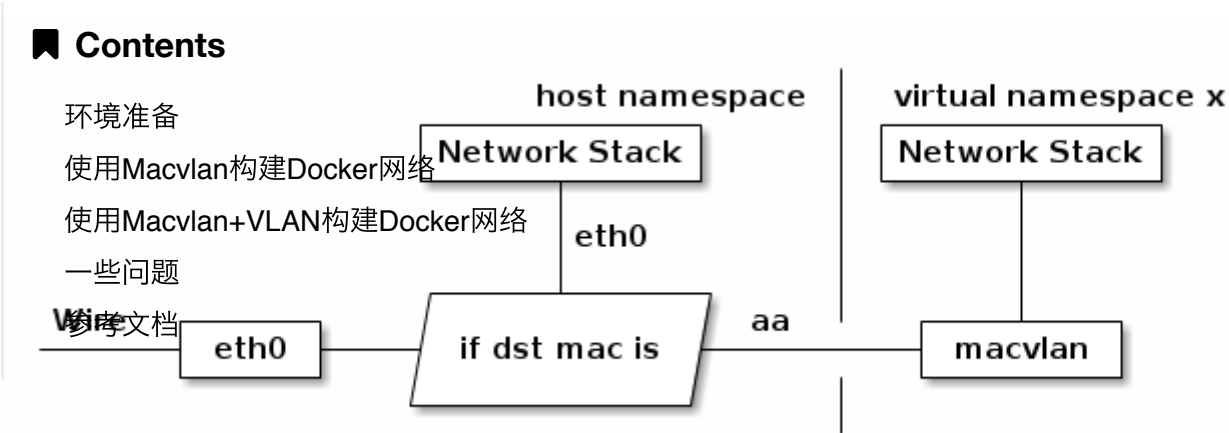
Docker以前的版本不支持直接配置宿主机所在网段ip并跟其直接互通的功能，当然也可以借助一些第三方工具，如pipework把这些琐碎的过程封装起来。Docker从1.12开始支持了overlay和macvlan网络，macvlan已经可以直接支持了使用宿主机所在网段资源。

Macvlan工作原理

- Macvlan是Linux内核支持的网络接口。要求的Linux内部版本是v3.9–3.19和4.0+。
- 通过为物理网卡创建Macvlan子接口，允许一块物理网卡拥有多个独立的MAC地址和IP地址。虚拟出来的子接口将直接暴露在底层物理网络中。从外界看来，就像是把网线分成多股，分别接到了不同的主机上一样。
- Macvlan有四种工作模式：Private、VEPA、Bridge和Passthru。最常用和默认的模式是Bridge模式。
- 物理网卡收到包后，会根据收到包的目的MAC地址判断这个包需要交给哪个虚拟网卡。



- 如果配合Network Namespace 使用，可以构建这样的网络：



MacVLAN可以工作在四种模式下：

- VEPA(Virtual Ethernet Port Aggregator)mode： default模式
- Brideg mode
- Private mode
- Passthru mode

VEPA需要接入交换机支持 hairpin mode 。相对而言 Bridge mode 更加常用。

macvlan和overlay网络不同，overlay是global scope类型的网络，而macvlan是local scope。scope指的是网络作用的范围。global类型的网络其作用于一组docker daemon集群，local类型的网络只作用于单一主机。

每台主机创建的macvlan网络是独立的，A机器上创建的macvlan网络并不影响B机器上的网络，但两台主机在网卡配置混杂模式、两台主机上macvlan存在overlap、两个macvlan网络没有分配过同样IP，这三个条件满足时，同样可以实现跨主机通信。

## 环境准备

一共两台机器：两台机器都安装Docker且内核版本>3.9。

主机名	IP地址	软件环境
dev-master-01	192.168.2.210	kernel>3.9,Docker
dev-node-01	192.168.2.211	kernel>3.9,Docker

## 启用网卡混杂模式

### 📖 Contents

两台机器上使用桥接模式,网卡混杂模式开启全部允许。

使用Macvlan构建Docker网络

主机上使用MacvlanVLAN构建Docker网络,主机上配置的enp0s5网口或者创建的vlan网口,都需要开启混杂模式。如果不开启混杂模式会导致macvlan网络无法访问外界,具体在不使用vlan时,表现为无法ping通路由,无法ping通同一网络内其他主机。  
[一些问题](#)  
[参考文档](#)

设置混杂模式可以通过ip或ifconfig指令实现。

- 通过ip指令

设置enp0s5为混杂模式

```
1 $ ip link set enp0s5 promisc on
```

取消enp0s5的混杂模式

```
1 $ ip link set enp0s5 promisc off
```

- 通过ifconfig指令

设置enp0s5为混杂模式

```
1 $ ifconfig enp0s5 promisc
```

取消enp0s5的混杂模式

```
1 $ ifconfig enp0s5 -promisc
```

验证网卡混杂模式是否设置成功

```

1  $ ifconfig enp0s5
2  enp0s5      Link encap:Ethernet  HWaddr 00:1c:42:97:53:2a
3              inet addr:192.168.2.210  Bcast:192.168.2.255  Mask:255.255.255.0
4  Contents    inet6 addr: fe80::21c:42ff:fe97:532a/64  Scope:Link
5              UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
6  环境准备    RX packets:1059321 errors:0 dropped:145 overruns:0 frame:0
7              TX packets:15030 errors:0 dropped:0 overruns:0 carrier:0
8  使用Macvlan构建Docker网络  collisions:0 txqueuelen:1000
9  使用MacvlanVLAN构建Docker网络(785.3 MB)  TX bytes:1039141 (1.0 MB)
    一些问题
    参考文档

```

其中 UP BROADCAST RUNNING PROMISC MULTICAST的PROMISC 说明网卡enp0s5已经设置成混杂模式。

## 使用Macvlan构建Docker网络

两台主机上均使用enp0s5网卡创建一个192.168.2.0网段的macvlan网络。macvlan驱动实际是利用的Linux macvlan内核驱动，这意味着这样子运行的容器，网络通讯将会直接送到下层vlan。这是目前最高网络效率的驱动。这里没有NAT，没有端口映射，通讯直接通过Vlan送出。

- master主机

### 创建macvlan网络

```

1  $ docker network create -d macvlan --subnet 192.168.2.0/24 --gateway 192.168.2.1
2  0ab39da89dd66238a1e9c75edbb70afa9b09d13dc5f1a041f5dd46c369856225

```

macvlan是kernel的模块名。  
 192.168.2.0/24是宿主机所在网络的网段。  
 192.168.2.1是网关。  
 enp0s5是宿主机接入192.168.2.0/24的网络设备。

### 查看macvlan是否创建成功

```
1  $ docker network ls
```

2	NETWORK ID	NAME	DRIVER	SCOPE
3	e13c13e22f73	bridge	bridge	local
4	a56cfbac20ed	docker_gwbridge	bridge	local
5	0b703c3d9cb8	host	host	local
6	ae041e911c	none	null	local
7	0ab39da89dd6	macnet	macvlan	local
8	856b6dbf7e83	weave	weavemesh	local

使用Macvlan构建Docker网络

使用Macvlan+VLAN构建Docker网络

创建两个使用macvlan容器

参考文档

创建容器c1

```
1 $ docker run -id --net macnet --ip 192.168.2.220 --name c1 busybox sh
2 6120c05c90ae75658f6703a269da453a4c516686144cc53dad027af7410cc93c
3
4 # 查看容器IP
5 $ docker exec c1 ip a
6
7 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
8     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
9     inet 127.0.0.1/8 scope host lo
10         valid_lft forever preferred_lft forever
11     inet6 ::1/128 scope host
12         valid_lft forever preferred_lft forever
13 30: eth0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
14     link/ether 02:42:c0:a8:02:dc brd ff:ff:ff:ff:ff:ff
15     inet 192.168.2.220/24 scope global eth0
16         valid_lft forever preferred_lft forever
17     inet6 fe80::42:c0ff:fea8:2dc/64 scope link
18         valid_lft forever preferred_lft forever
19
20 # 查看容器route
21 $ docker exec c1 route -n
22
23 Kernel IP routing table
24 Destination      Gateway           Genmask          Flags Metric Ref    Use Iface
25 0.0.0.0           192.168.2.1      0.0.0.0          UG    0      0      0 eth0
26 192.168.2.0       0.0.0.0          255.255.255.0    U     0      0      0 eth0
```

创建容器c2

```
1 $ docker run -id --net macnet --ip 192.168.2.221 --name c2 busybox sh
```

```

2  f8bcf7a103ac483fd46ee27ac675b23255bf64ad49c7e70be8785737add1ce06
3
4  # 查看容器IP
5  $ docker exec c2 ip a
6  Contents
7  1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
8  环境准备 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
9  使用Macvlan构建Docker网络 inet 127.0.0.1/8 scope host lo
10 valid_lft forever preferred_lft forever
11 使用Macvlan+VLAN构建Docker网络 inet6 ::1/128 scope host
12 一些问题 valid_lft forever preferred_lft forever
13 31: eth0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
14 参考文档 link/ether 02:42:c0:a8:02:dd brd ff:ff:ff:ff:ff:ff
15 inet 192.168.2.221/24 scope global eth0
16 valid_lft forever preferred_lft forever
17 inet6 fe80::42:c0ff:fea8:2dd/64 scope link
18 valid_lft forever preferred_lft forever
19
20
21 # 查看容器route
22 $ docker exec c2 route -n
23
24 Kernel IP routing table
25 Destination      Gateway            Genmask           Flags Metric Ref    Use Iface
26 0.0.0.0           192.168.2.1       0.0.0.0           UG    0      0      0 eth0
27 192.168.2.0       0.0.0.0           255.255.255.0     U      0      0      0 eth0

```

- node主机

## 创建macvlan网络

```

1  $ docker network create -d macvlan --subnet 192.168.2.0/24 --gateway 192.168.2.1
2  ad869482ddc115acf6ee004a500ce7c28386e976c4bb5eccdffaaa1afbcd07e1

```

macvlan是kernel的模块名。  
 192.168.2.0/24是宿主机所在网络的网段。  
 192.168.2.1是网关。  
 enp0s5是宿主机接入192.168.2.0/24的网络设备。

## 查看macvlan是否创建成功

```
1  $ docker network ls
```

2	NETWORK ID	NAME	DRIVER	SCOPE
3	e13c13e22f73	bridge	bridge	local
4	a56cfbac20ed	docker_gwbridge	bridge	local
5	0b703c3d9cb8	host	host	local
6	2b206e165d4	macnet	macvlan	local
7	abdb4b9f751c	none	null	local
8	856b6dbf7e83	weave	weavemesh	local

使用Macvlan构建Docker网络

使用Macvlan+VLAN构建Docker网络

## 创建两个使用macvlan容器

参考文档

## 创建容器c3

```

1  $ docker run -id --net macnet --ip 192.168.2.222 --name c3 busybox sh
2  8b6a83f73af9dd54ef24be80bf946f0e10959e48ead12a09514d3c59a287927c
3
4
5  # 查看容器IP
6  $ docker exec c3 ip a
7
8  1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
9      link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
10     inet 127.0.0.1/8 scope host lo
11         valid_lft forever preferred_lft forever
12     inet6 ::1/128 scope host
13         valid_lft forever preferred_lft forever
14 18: eth0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
15     link/ether 02:42:c0:a8:02:ca brd ff:ff:ff:ff:ff:ff
16     inet 192.168.2.222/24 scope global eth0
17         valid_lft forever preferred_lft forever
18     inet6 fe80::42:c0ff:fea8:2ca/64 scope link
19         valid_lft forever preferred_lft forever
20
21 # 查看容器route
22 $ docker exec c3 route -n
23
24 Kernel IP routing table
25 Destination      Gateway            Genmask           Flags Metric Ref    Use Iface
26 0.0.0.0           192.168.2.1       0.0.0.0           UG      0      0      0 eth0
27 192.168.2.0       0.0.0.0           255.255.255.0     U       0      0      0 eth0

```

## 创建容器c4

```
1  $ docker run -id --net macnet --ip 192.168.2.223 --name c4 busybox sh
```

```

2 c7be679190467564023aec1fd908bb7418a55f41ab548d59d77cabf187dc665c
3
4
5 # 查看容器IP
6 $ docker exec c4 ip a
7
8 环境准备
9 1. 准备 <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
10 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
11 使用Macvlan构建Docker网络
12 inet 127.0.0.1/8 scope host lo
13 valid_lft forever preferred_lft forever
14 使用Macvlan+VLAN构建Docker网络
15 inet6 ::1/128 scope host
16 valid_lft forever preferred_lft forever
17 一些问题
18 2. eth0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
19 link/ether 02:42:c0:a8:02:df brd ff:ff:ff:ff:ff:ff
20 inet 192.168.2.223/24 scope global eth0
21 valid_lft forever preferred_lft forever
22 inet6 fe80::42:c0ff:fea8:2df/64 scope link
23 valid_lft forever preferred_lft forever
24
25 # 查看容器route
26 $ docker exec c4 route -n
27
28 Kernel IP routing table
29
30 Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
31 0.0.0.0           192.168.2.1    0.0.0.0         UG    0      0          0 eth0
32 192.168.2.0       0.0.0.0        255.255.255.0   U     0      0          0 eth0

```

## 测试网络连通情况

所有测试在master主机上进行。

- ping网关

```

1 $ docker exec c1 ping -c 3 192.168.2.1
2 PING 192.168.2.1 (192.168.2.1): 56 data bytes
3 64 bytes from 192.168.2.1: seq=0 ttl=255 time=1.594 ms
4 64 bytes from 192.168.2.1: seq=1 ttl=255 time=0.659 ms
5 64 bytes from 192.168.2.1: seq=2 ttl=255 time=0.753 ms
6
7 --- 192.168.2.1 ping statistics ---
8 3 packets transmitted, 3 packets received, 0% packet loss
9 round-trip min/avg/max = 0.659/1.002/1.594 ms

```

测试结果:通

- 使用容器名ping本主机容器



```

1 $ docker exec c1 ping -c3 c1
2 PING c1 (192.168.2.220): 56 data bytes
3 64 bytes from 192.168.2.220: seq=0 ttl=64 time=0.053 ms
4 64 bytes from 192.168.2.220: seq=1 ttl=64 time=0.073 ms
5 64 bytes from 192.168.2.220: seq=2 ttl=64 time=0.048 ms
6
7 环境准备ping statistics ---
8 3 packets transmitted, 3 packets received, 0% packet loss
9 round-trip min/avg/max = 0.048/0.058/0.073 ms

```

使用Macvlan构建Docker网络

使用Macvlan+VLAN构建Docker网络

一些问题

测试结果:通

- ping本主机容器

```

1 $ docker exec c1 ping -c3 192.168.2.220
2 PING 192.168.2.220 (192.168.2.220): 56 data bytes
3 64 bytes from 192.168.2.220: seq=0 ttl=64 time=0.069 ms
4 64 bytes from 192.168.2.220: seq=1 ttl=64 time=0.172 ms
5 64 bytes from 192.168.2.220: seq=2 ttl=64 time=0.079 ms
6
7 --- 192.168.2.220 ping statistics ---
8 3 packets transmitted, 3 packets received, 0% packet loss
9 round-trip min/avg/max = 0.069/0.106/0.172 ms

```

测试结果:通

- ping另一主机容器

```

1 $ docker exec c1 ping -c2 192.168.2.222
2 PING 192.168.2.222 (192.168.2.222): 56 data bytes
3 64 bytes from 192.168.2.222: seq=0 ttl=255 time=0.788 ms
4 64 bytes from 192.168.2.222: seq=1 ttl=255 time=0.383 ms
5
6 --- 192.168.2.222 ping statistics ---
7 2 packets transmitted, 2 packets received, 0% packet loss
8 round-trip min/avg/max = 0.383/0.585/0.788 ms

```

测试结果:通

- 使用容器名ping另一主机容器:

```
1 $ docker exec c1 ping -c2 c3
2 ping: bad address 'c3'
```

## Contents

测试结果:不通

环境准备

- 使用Macvlan构建Docker网络
- 本主机ping本主机容器: 不通
- 使用Macvlan+VLAN构建Docker网络

一些问题

参考文档

```
1 $ ping -c 2 192.168.2.220
2 PING 192.168.2.220 (192.168.2.220) 56(84) bytes of data.
3 From 192.168.2.210 icmp_seq=1 Destination Host Unreachable
4 From 192.168.2.210 icmp_seq=2 Destination Host Unreachable
5
6 --- 192.168.2.220 ping statistics ---
7 2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1000ms
```

测试结果:不通

- 本主机ping另一主机容器

```
1 $ ping -c 2 192.168.2.222
2 PING 192.168.2.222 (192.168.2.222) 56(84) bytes of data.
3 64 bytes from 192.168.2.222: icmp_seq=1 ttl=255 time=0.308 ms
4 64 bytes from 192.168.2.222: icmp_seq=2 ttl=255 time=0.343 ms
5
6 --- 192.168.2.222 ping statistics ---
7 2 packets transmitted, 2 received, 0% packet loss, time 1000ms
8 rtt min/avg/max/mdev = 0.308/0.325/0.343/0.025 ms
```

测试结果:通

- 本主机上容器到另一台宿主IP

```
1 $ docker exec c2 ping -c 2 192.168.2.211
2 PING 192.168.2.211 (192.168.2.211): 56 data bytes
3 64 bytes from 192.168.2.211: seq=0 ttl=64 time=0.494 ms
4 64 bytes from 192.168.2.211: seq=1 ttl=64 time=0.495 ms
5
6 --- 192.168.2.211 ping statistics ---
7 2 packets transmitted, 2 packets received, 0% packet loss
8 round-trip min/avg/max = 0.494/0.494/0.495 ms
```

# 使用Macvlan+VLAN构建Docker网络

## VLAN简介

**环境准备**  
**使用Macvlan构建Docker网络**  
**使用Macvlan+VLAN构建Docker网络**  
 VLAN(Virtual Local Area Network)又称虚拟局域网，是指在局域网的基础上，采用网络管理软件构建的可跨越不同网段、不同网络的端到端的逻辑网络。

**一些问题**  
**参考文档**  
 一个VLAN组成一个逻辑子网，即一个逻辑广播域，它可以覆盖多个网络设备，允许处于不同地理位置的网络用户加入到一个逻辑子网中。使用VLAN功能后，能够将网络分割成多个广播域。

Linux支持在物理网卡上创建vlan子接口。每个vlan子接口属于不同的二层域，所有的vlan子接口拥有相同的MAC地址。这点是和Macvlan子接口不同的地方。

- master主机

## 创建VLAN

```

1  # 为物理网卡enp0s5创建Macvlan子接口
2  $ ip link add link enp0s5 name enp0s5.200 type vlan id 200
3
4  # 将MACVLAN设备加入到容器的network space
5  $ ip link list enp0s5.200
6  33: ip enp0s5.200@enp0s5: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN n
7      link/ether 00:1c:42:97:53:2a brd ff:ff:ff:ff:ff:ff
8
9  # 启用enp0s5.200
10 $ ip link set enp0s5.200 up
11
12 $ ip link list enp0s5.200
13 33: enp0s5.200@enp0s5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
14     link/ether 00:1c:42:97:53:2a brd ff:ff:ff:ff:ff:ff
15
16 # 设置enp0s5.200为混杂模式
17 $ ip link set enp0s5.200 promisc on
18 $ ifconfig enp0s5.200
19 enp0s5.200 Link encap:Ethernet  HWaddr 00:1c:42:97:53:2a
20             inet6 addr: fe80::21c:42ff:fe97:532a/64 Scope:Link
21             UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
22             RX packets:0 errors:0 dropped:0 overruns:0 frame:0
23             TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
24             collisions:0 txqueuelen:1000
25             RX bytes:0 (0.0 B)  TX bytes:648 (648.0 B)
  
```

## 设置macvlan的ip和网关

```

1 $ ip addr add 192.168.200.10/24 dev enp0s5.200
2 # 删除原默认路由，否则下面加默认路由时会报错。
3 $ ip route del default
4 $ ip route add default via 192.168.200.1 dev enp0s5.200

```

环境准备

使用Macvlan构建Docker网络

使用Macvlan+VLAN构建Docker网络

一些问题

```

1 $ docker network create -d macvlan --subnet=192.168.200.0/24 --gateway=192.168.2
2 b2fb555bbcdc1f0b724cc9e76b154630e632184566b9e8bb314cde825ad9ff9d
3
4
5 # 查看macvlan是否创建成功
6
7 $ docker network ls
8 NETWORK ID          NAME                DRIVER              SCOPE
9 6dc2160d4026        bridge             bridge              local
10 a56cfbac20ed        docker_gwbridge    bridge              local
11 0b703c3d9cb8        host               host                local
12 9282605b5107        macnet             macvlan             local
13 d129db56fcc1        macvlan200         macvlan             local
14 abdb4b9f751c        none               null                local
15 856b6dbf7e83        weave              weavemesh           local

```

## 创建两个使用macvlan容器

### 创建容器c5

```

1 $ docker run --net=macvlan200 --ip=192.168.200.100 -id --name c5 busybox sh
2 28ee719d4784696eff10805f1f9d992b245b7b886b031c1781dbaf4037bbb231

```

### 创建容器c6

```

1 $ docker run --net=macvlan200 --ip=192.168.200.101 -id --name c6 busybox sh
2 7bd2a36e24846b383ab3aca6b9b48227600aed7352956c2c974db1ceb97efb54

```

- node主机

## 创建Vlan

```

1  # 为物理网卡enp0s5创建Macvlan子接口
2  $ ip link add link enp0s5 name enp0s5.200 type vlan id 200
3
4  # 将MACVLAN设备加入到容器的network space
5  Contents list enp0s5.200
6  24: enp0s5.200@enp0s5: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode
7  环境准备nk/ether 00:1c:42:67:23:07 brd ff:ff:ff:ff:ff:ff
8
9  使用Macvlan构建Docker网络
10 # 启用enp0s5.200
11 使用Macvlan+VLAN构建Docker网络
12 $ ip link set enp0s5.200 up
13 一些问题
14 $ ip link list enp0s5.200
15 24: enp0s5.200@enp0s5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
16 link/ether 00:1c:42:67:23:07 brd ff:ff:ff:ff:ff:ff
17
18 # 设置enp0s5.200为混杂模式
19 $ ip link set enp0s5.200 promisc on
20 $ ifconfig enp0s5.200
21 enp0s5.200 Link encap:Ethernet HWaddr 00:1c:42:67:23:07
22 inet6 addr: fe80::21c:42ff:fe67:2307/64 Scope:Link
23 UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
24 RX packets:0 errors:0 dropped:0 overruns:0 frame:0
25 TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
   collisions:0 txqueuelen:1000
   RX bytes:0 (0.0 B) TX bytes:648 (648.0 B)

```

## 设置macvlan的ip和网关

```

1  $ ip addr add 192.168.200.11/24 dev enp0s5.200
2  # 删除原默认路由，否则下面加默认路由时会报错。
3  $ ip route del default
4  $ ip route add default via 192.168.200.1 dev enp0s5.200

```

## 创建Docker macvlan网络

```

1  $ docker network create -d macvlan --subnet=192.168.200.0/24 --gateway=192.168.2
2  b2fb555bbcdc1f0b724cc9e76b154630e632184566b9e8bb314cde825ad9ff9d
3
4
5  # 查看macvlan是否创建成功
6  $ docker network ls
7
8  NETWORK ID          NAME           DRIVER         SCOPE
9  4886f72179a8         bridge        bridge         local
10 0b703c3d9cb8         host         host           local
11 149f5dcc20b3         macnet        macvlan        local
12 8ee3174430ab         macvlan200    macvlan        local
13 abdb4b9f751c         none         null           local
14 bce1daa0a925         weave        weavemesh      local

```

## 创建两个使用macvlan容器

### 创建容器c7

#### Contents

环境准备

- 1 使用Macvlan构建Docker网络
- 2 使用Macvlan+VLAN构建Docker网络

一些问题

参考文档

### 创建容器c8

- 1 \$ docker run --net=macvlan200 --ip=192.168.200.103 -id --name c8 busybox sh
- 2 7bd2a36e24846b383ab3aca6b9b48227600aed7352956c2c974db1ceb97efb54

## 测试网络连通情况

所有测试在master主机上进行。

- ping网关

- 1 \$ docker exec c5 ping -c2 192.168.200.1

测试结果:不通

- ping本地macvlannet地址

- 1 \$ docker exec c5 ping -c2 192.168.200.10
- 2 PING 192.168.200.10 (192.168.200.10): 56 data bytes
- 3
- 4 --- 192.168.200.10 ping statistics ---
- 5 2 packets transmitted, 0 packets received, 100% packet loss

测试结果:不通

- ping另一个主机的macvlannet地址

```

1  $ docker exec c5 ping -c2 192.168.200.11
2
3  PING 192.168.200.11 (192.168.200.11): 56 data bytes
4  64 bytes from 192.168.200.11: seq=0 ttl=64 time=0.535 ms
5  64 bytes from 192.168.200.11: seq=1 ttl=64 time=0.368 ms
6
7  --- 192.168.200.11 ping statistics ---
8  2 packets transmitted, 2 packets received, 0% packet loss
9  round-trip min/avg/max = 0.368/0.451/0.535 ms

```

使用Macvlan+VLAN构建Docker网络

一些问题

测试结果:通

- ping本主机容器

```

1  $ docker exec c5 ping -c2 192.168.200.100
2
3  PING 192.168.200.100 (192.168.200.100): 56 data bytes
4  64 bytes from 192.168.200.100: seq=0 ttl=64 time=0.091 ms
5  64 bytes from 192.168.200.100: seq=1 ttl=64 time=0.046 ms
6
7  --- 192.168.200.100 ping statistics ---
8  2 packets transmitted, 2 packets received, 0% packet loss
9  round-trip min/avg/max = 0.046/0.068/0.091 ms

```

测试结果:通

- ping另一主机容器

```

1  $ docker exec c5 ping -c2 192.168.200.102
2  PING 192.168.200.102 (192.168.200.102): 56 data bytes
3  64 bytes from 192.168.200.102: seq=0 ttl=64 time=0.815 ms
4  64 bytes from 192.168.200.102: seq=1 ttl=64 time=1.051 ms
5
6  --- 192.168.200.102 ping statistics ---
7  2 packets transmitted, 2 packets received, 0% packet loss
8  round-trip min/avg/max = 0.815/0.933/1.051 ms

```

测试结果:通

- 使用容器名ping本主机容器

```

1 $ docker exec c5 ping -c2 c6
2 PING c6 (192.168.200.101): 56 data bytes
3 64 bytes from 192.168.200.101: seq=0 ttl=64 time=0.152 ms
4 64 bytes from 192.168.200.101: seq=1 ttl=64 time=0.108 ms
5
6 --- c6 ping statistics ---
7 2 packets transmitted, 2 packets received, 0% packet loss
8 round-trip min/avg/max = 0.108/0.130/0.152 ms

```

## Contents

使用Macvlan+VLAN构建Docker网络

一些问题

测试结果:通

参考文档

- 使用容器名ping另一主机容器

```

1 $ docker exec c5 ping -c2 c7
2 ping: bad address 'c7'

```

测试结果:不通

- ping跨网络主机

```

1 $ docker exec c5 ping -c2 192.168.200.102
2 PING 192.168.200.102 (192.168.200.102): 56 data bytes
3 64 bytes from 192.168.200.102: seq=0 ttl=64 time=0.518 ms
4 64 bytes from 192.168.200.102: seq=1 ttl=64 time=0.458 ms
5
6 --- 192.168.200.102 ping statistics ---
7 2 packets transmitted, 2 packets received, 0% packet loss
8 round-trip min/avg/max = 0.458/0.488/0.518 ms

```

测试结果:通

- 本主机ping本主机容器

```

1 $ ping 192.168.200.100 -c2
2 PING 192.168.200.100 (192.168.200.100) 56(84) bytes of data.
3 From 192.168.200.10 icmp_seq=1 Destination Host Unreachable
4 From 192.168.200.10 icmp_seq=2 Destination Host Unreachable
5
6 --- 192.168.200.100 ping statistics ---
7 2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1006ms
8 pipe 2

```



测试结果:不通

本主机ping另一主机容器

```
Contents

环境准备

1 使用Macvlan构建Docker网络
2 使用MacvlanVLAN构建Docker网络
3 一些问题
4 参考文档
5 --- 192.168.200.102 ping statistics ---
6 2 packets transmitted, 2 received, 0% packet loss, time 1001ms
7 rtt min/avg/max/mdev = 0.319/0.472/0.625/0.153 ms
```

测试结果:通

一些问题

- macvlan网络在创建时要指定parent.其中parent仅能使用一次,即eth0在创建一个macvlan网络时使用了,则在创建另一个的时候就无法再使用了。
- 两种模式下都有测试网络不通的情况，不知道是不是虚拟机网络环境问题。理论上应该是通的。待在真实环境验证后才得知。

参考文档

<http://www.google.com> (<http://www.google.com>)  
<http://t.cn/RXYNXqK> (<http://t.cn/RXYNXqK>)  
<http://t.cn/RXQU43D> (<http://t.cn/RXQU43D>)  
<http://hustcat.github.io/docker-macvlan/> (<http://hustcat.github.io/docker-macvlan/>)

撰写评论

Contents

发布

- 环境准备
- 使用Macvlan构建Docker网络
- 使用Macvlan+VLAN构建Docker网络
- 一些问题
- 参考文档

还没有评论，快来抢沙发吧！


© LiveRe.

FEATURED TAGS (/TAGS/)

- Docker (/tags/#Docker)
- Kubernetes (/tags/#Kubernetes)

FRIENDS

简单.生活 (<http://www.mike.org.cn/>)  
技术交流群 (<http://shang.qq.com/wpa/qunwpa?idkey=ea4c43493c2269428ac6ef6141de4b6d78e5ab2d41380ca4099b833b62884ee9>)  
0 0

 (/atom.xml)

 (<https://www.zhihu.com/people/80imike>)

 (<http://weibo.com/2093524665>)

Copyright © Mike 2018 | Hosted by Coding Pages (<https://pages.coding.me>)  
Theme by BeanTech (<http://beantech.org>) ♥ re-Ported by 胡伟煌 (<http://www.huweihuang.com>) |  
Star 179