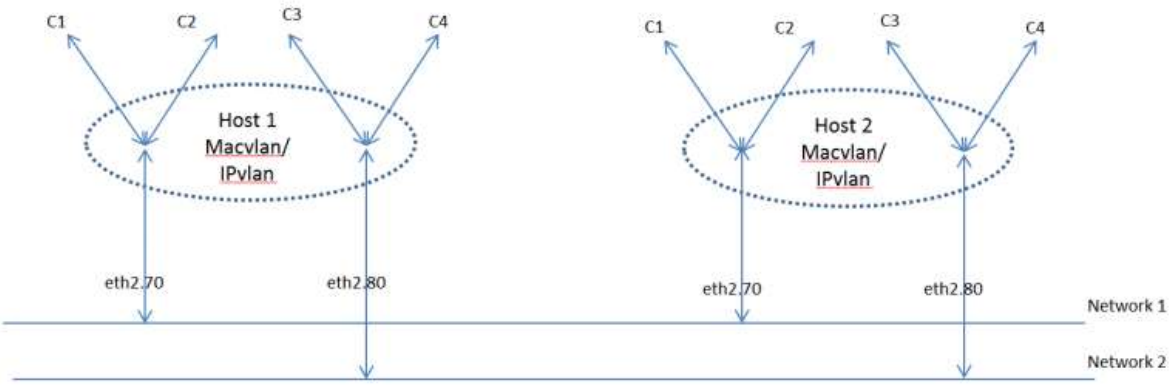
Docker macvlan and ipvlan network plugins

May 29, 2016 Containers, Docker, ipvlan, macvlan

This is a continuation of my previous blog on macvlan and ipvlan Linux network drivers. Docker has added support for macvlan and ipvlan drivers and its currently in experimental mode as of Docker release 1.11.

Example used in this blog

In this example, we will use Docker macvlan and ipvlan network plugins for Container communication across hosts. To illustrate macvlan and ipvlan concepts and usage, I have created the following example.



Following are details of the setup:

- First, we need to create two Docker hosts with experimental Docker installed. The experimental Docker has support for macvlan and ipvlan. To create experimental boot2docker image, please use the procedure here.
- I have used Virtualbox based environment. Macvlan network is created on top of host-only network adapter in Virtualbox. It is needed to enable promiscuous mode on the Virtualbox adapter. This allows for Container communication across hosts.
- There are four Containers in each host. Two Containers are in vlan70 network and two other Containers are in vlan80 network.
- We will use both macvlan and ipvlan drivers and illustrate Container network connectivity in same host and across hosts.

Following output shows the Docker experimental version running:

```
$ docker --version
Docker version 1.11.0-dev, build 6c2f438, experimental
```

Macvlan

In this section, we will illustrate macvlan based connectivity with macvlan bridge mode.

On host 1, create macvlan subinterface and Containers:

```
docker network  create  -d macvlan \
   --subnet=192.168.0.0/16 \
    --ip-range=192.168.2.0/24 \
        -o macvlan_mode=bridge \
    -o parent=eth2.70 macvlan70
docker run --net=macvlan70 -it --name macvlan70_1 --rm alpine /bin/sh
docker run --net=macvlan70 -it --name macvlan70_2 --rm alpine /bin/sh

docker network  create  -d macvlan \
   --subnet=192.169.0.0/16 \
    --ip-range=192.169.2.0/24 \
        -o macvlan_mode=bridge \
    -o parent=eth2.80 macvlan80
docker run --net=macvlan80 -it --name macvlan80_1 --rm alpine /bin/sh
docker run --net=macvlan80 -it --name macvlan80_2 --rm alpine /bin/sh
```

Containers in host 1 will get ip address in 192.168.2.0/24 network and 192.169.2.0/24 network based on the options mentioned above.

On host 2, create macvlan subinterface and Containers:

```
docker network  create  -d macvlan \
   --subnet=192.168.0.0/16 \
    --ip-range=192.168.3.0/24 \
        -o macvlan_mode=bridge \
    -o parent=eth2.70 macvlan70
docker run --net=macvlan70 -it --name macvlan70_3 --rm alpine /bin/sh
docker run --net=macvlan70 -it --name macvlan70_4 --rm alpine /bin/sh

docker network  create  -d macvlan \
   --subnet=192.169.0.0/16 \
    --ip-range=192.169.3.0/24 \
        -o macvlan_mode=bridge \
    -o parent=eth2.80 macvlan80
```

```
docker run --net=macvlan80 -it --name macvlan80_3 --rm alpine /bin/sh
docker run --net=macvlan80 -it --name macvlan80_4 --rm alpine /bin/sh
```

Containers in host 2 will get ip address in 192.168.3.0/24 network and 192.169.3.0/24 network based on the options mentioned above.

Lets look at Docker networks created in host 1, we can see the macvlan networks “macvlan70” and “macvlan80” as shown below.

```
$ docker network ls
NETWORK ID          NAME                DRIVER
e5f5f6add03d        bridge              bridge
a1b89ce4bd84        host                host
90b7d5ba61b9        macvlan70           macvlan
bedeca9839e1        macvlan80           macvlan
```

Lets check connectivity on ip subnet 192.168.x.x/16(vlan70) between Containers in same host and across hosts:

```
Here, we are inside macvlan70_1 Container in host1:
# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:C0:A8:02:01
          inet addr:192.168.2.1  Bcast:0.0.0.0  Mask:255.255.0.0
# ping -c1 192.168.2.2
PING 192.168.2.2 (192.168.2.2): 56 data bytes
64 bytes from 192.168.2.2: seq=0 ttl=64 time=0.137 ms

--- 192.168.2.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.137/0.137/0.137 ms
/ # ping -c1 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: seq=0 ttl=64 time=2.596 ms

--- 192.168.3.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 2.596/2.596/2.596 ms
/ # ping -c1 192.168.3.2
PING 192.168.3.2 (192.168.3.2): 56 data bytes
64 bytes from 192.168.3.2: seq=0 ttl=64 time=1.400 ms
```

```
--- 192.168.3.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.400/1.400/1.400 ms
```

Connectivity is also successful in ip subnet 192.169.x.x/16(vlan80) between Containers in same host and across hosts.

Connecting macvlan container to host

By default containers in macvlan network cannot directly talk to host and this is intentional. It is needed to create a macvlan interface in the host to allow the communication between host and container. Also, Containers can expose tcp/udp ports using macvlan network and it can directly be accessed from underlay network.

Lets use an example to illustrate host to container connectivity in macvlan network.

Create a macvlan interface on host sub-interface:

```
docker network create -d macvlan \
–subnet=192.168.0.0/16 \
–ip-range=192.168.2.0/24 \
-o macvlan_mode=bridge \
-o parent=eth2.70 macvlan70
```

Create container on that macvlan interface:

```
docker run -d –net=macvlan70 –name nginx nginx
```

Find ip address of Container:

```
docker inspect nginx | grep IPAddress
“SecondaryIPAddresses”: null,
“IPAddress”: “”,
“IPAddress”: “192.168.2.1”,
```

At this point, we cannot ping container IP “192.168.2.1” from host machine.

Now, let's create macvlan interface in host with address "192.168.2.10" in same network.

```
sudo ip link add mymacvlan70 link eth2.70 type macvlan mode bridge
sudo ip addr add 192.168.2.10/24 dev mymacvlan70
sudo ifconfig mymacvlan70 up
```

Now, we should be able to ping the Container IP as well as access "nginx" container from host machine.

```
$ ping -c1 192.168.2.1
PING 192.168.2.1 (192.168.2.1): 56 data bytes
64 bytes from 192.168.2.1: seq=0 ttl=64 time=0.112 ms

— 192.168.2.1 ping statistics —
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.112/0.112/0.112 ms
```

ipvlan

In this section, we will illustrate ipvlan based connectivity with ipvlan l2 mode.

On host 1, create ipvlan sub-interface and Containers.

```
docker network  create  -d ipvlan \
   --subnet=192.168.0.0/16 \
    --ip-range=192.168.2.0/24 \
        -o ipvlan_mode=l2 \
    -o parent=eth2.70 ipvlan70
docker run --net=ipvlan70 -it --name ipvlan70_1 --rm alpine /bin/sh
eth0      Link encap:Ethernet  HWaddr 08:00:27:FA:9D:0C
          inet addr:192.168.2.1  Bcast:0.0.0.0  Mask:255.255.0.0
docker run --net=ipvlan70 -it --name ipvlan70_2 --rm alpine /bin/sh

docker network  create  -d ipvlan \
   --subnet=192.169.0.0/16 \
    --ip-range=192.169.2.0/24 \
        -o ipvlan_mode=l2 \
    -o parent=eth2.80 ipvlan80
docker run --net=ipvlan80 -it --name ipvlan80_1 --rm alpine /bin/sh
docker run --net=ipvlan80 -it --name ipvlan80_2 --rm alpine /bin/sh
```

On host 2, create ipvlan sub-interface and Containers.

```
docker network  create  -d ipvlan \
   --subnet=192.168.0.0/16 \
    --ip-range=192.168.3.0/24 \
        -o ipvlan_mode=l2 \
    -o parent=eth2.70 ipvlan70
docker run --net=ipvlan70 -it --name ipvlan70_3 --rm alpine /bin/sh
docker run --net=ipvlan70 -it --name ipvlan70_4 --rm alpine /bin/sh

docker network  create  -d ipvlan \
   --subnet=192.169.0.0/16 \
    --ip-range=192.169.3.0/24 \
        -o ipvlan_mode=l2 \
    -o parent=eth2.80 ipvlan80
docker run --net=ipvlan80 -it --name ipvlan80_3 --rm alpine /bin/sh
docker run --net=ipvlan80 -it --name ipvlan80_4 --rm alpine /bin/sh
```

Let's look at networks created in host 1, we can see ipvlan networks "ipvlan70" and "ipvlan80" as shown below.

```
$ docker network ls
NETWORK ID          NAME                DRIVER
e5f5f6add03d        bridge              bridge
a1b89ce4bd84        host                host
1a1262e008d3        ipvlan70            ipvlan
080b230b892e        ipvlan80            ipvlan
```

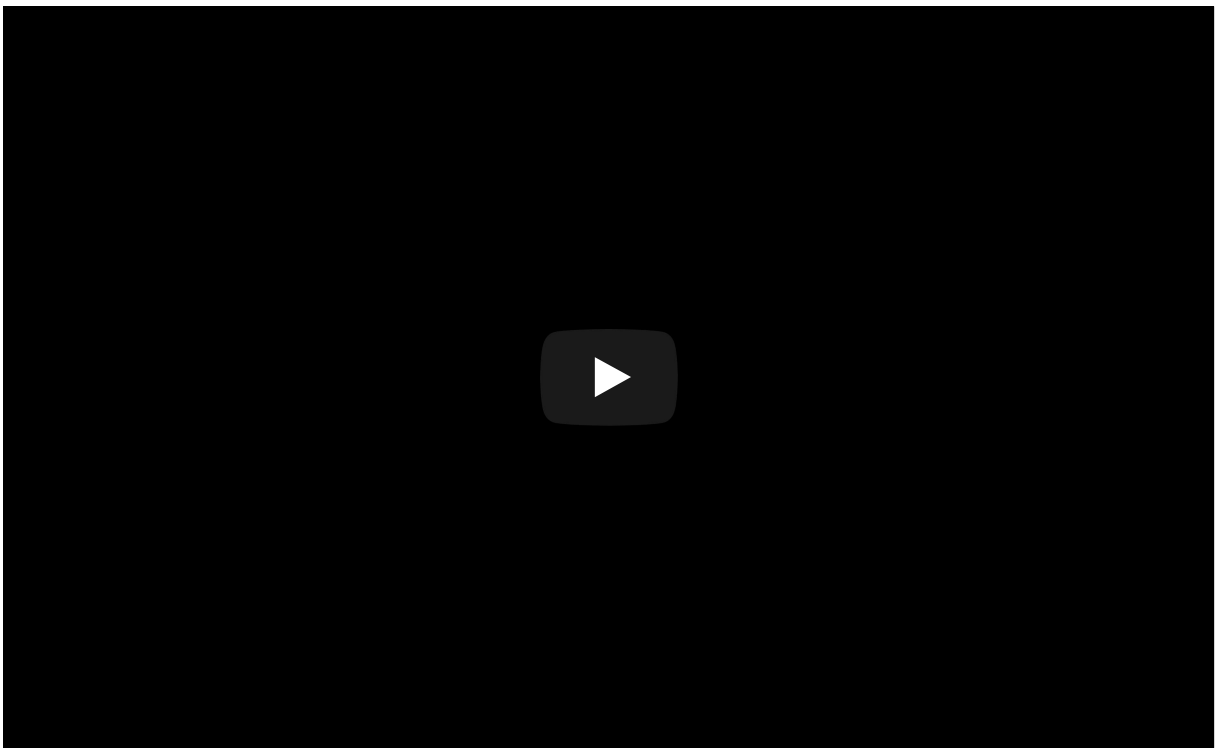
Connectivity is successful in ip subnet 192.168.x.x/16(vlan70) and ip subnet 192.169.x.x/16(vlan80) between Containers in same host and across hosts.

ipvlan l3 mode

There are some issues in getting ipvlan l3 mode to work across hosts. Following example shows setting up ipvlan l3 mode across Containers in a single host:

```
docker network  create  -d ipvlan \
   --subnet=192.168.2.0/24 \
   --subnet=192.169.2.0/24 \
        -o ipvlan_mode=l3 \
    -o parent=eth2.80 ipvlan
docker run --net=ipvlan --ip=192.168.2.10 -it --name ipvlan_1 --rm alpine
/bin/sh
docker run --net=ipvlan --ip=192.169.2.10 -it --name ipvlan_2 --rm alpine
/bin/sh
```

For a quick reference on Docker macvlan driver, you can refer my video here:



References

- Docker macvlan and ipvlan network drivers
- Feedback on Docker macvlan, ipvlan experimental drivers