docker network through a specific physical interface [closed]

Asked 6 years, 8 months ago Modified 10 months ago Viewed 23k times

4

Closed. This question is not about programming or software development. It is not currently accepting answers.

This question does not appear to be about a specific programming problem, a software algorithm, or software tools primarily used by programmers. If you believe the question would be on-topic on another Stack Exchange site, you can leave a comment to explain where the question may be able to be answered.

Closed 10 months ago.

Improve this question

So I'm trying to create a network (`docker network create`) so that its traffic will pass through an specific physical network interface (NIC); I have two: `<iface1>` (internal), and `<iface2>` (external).

I need the traffics of both NICs to be physically separated.

METHOD 1:

I think `macvlan` is the driver should use to create such network. For most of what I found on the internet, the solutions refer to Pipework (deprecated now) and temporary docker-plugins (deprecated too). For what most closely has helped me is this1

```
docker network create -d macvlan \
    --subnet 192.168.0.0/16 \
    --ip-range 192.168.2.0/24 \
    -o parent=wlp8s0.1 \
    -o macvlan_mode=bridge \
    macvlan0
```

Then, in order for the container to be visible from the host, I need to do this in the host:

```
sudo ip link add macvlan0 link wlp8s0.1 type macvlan mode bridge
sudo ip addr add 192.168.2.10/16 dev macvlan0
sudo ifconfig macvlan0 up
```

Now the container and the host see each other :) BUT the container can't access the local network. The idea, is that the container can access internet.

METHOD 2:

As I will use `<iface2>` manually, I'm ok if by default the traffic goes through `<iface1>`. But no matter in which order I get the NICs up (I also tried removing the LKM for `<iface2>` temporarely); the whole traffic is always overtaken by the external NIC `<iface2>`. And I found that it happens because the route table updates automatically at some "random" time. In order to force the traffic to go through `<iface1>`, I have to (in the host):

```
sudo route del -net <net> gw 0.0.0.0 netmask 255.0.0.0 dev <iface2>
sudo route del default <iface2>
```

Now, I can verify (in several ways) that the traffic just goes through `<iface1>`. But the moment that the route table updates (automatically), all traffic moves to `<iface2>`. Damn! I'm sure there's a way to make the route table "static" or "persistent".

EDIT (18/Jul/2018): The main idea is to be able to access internet through a docker container using **only one** of two available physical network interfaces.

docker network-programming docker-network nic macvlan

Share Improve this question Follow edited Jul 18, 2018 at 20:20

asked Jul 17, 2018 at 3:03

issrc

49 ● 1 ● 1 ● 4

1 Answer

Sorted by: Highest score (default)

0

My environment:

On the host created for vm virbr0 bridge with ip address 192.168.122.1 and up vm instance with interface ens3 and ip address 192.168.122.152.

192.168.122.1 - is gateway for 192.168.122.0/24 network.

Into vm:

Create network:

```
 # docker network create --subnet 192.168.122.0/24 --gateway 192.168.122.1 --driver
macvlan -o parent=ens3 vmnet
```

Create docker container:

```
# docker run -ti --network vmnet alpine ash
```

Check:

```
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
12: eth0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:c0:a8:7a:02 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.2/24 brd 192.168.122.255 scope global eth0
       valid_lft forever preferred_lft forever
/ # ping 192.168.122.152
PING 192.168.122.152 (192.168.122.152): 56 data bytes
^C
--- 192.168.122.152 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
/ # ping 192.168.122.1
PING 192.168.122.1 (192.168.122.1): 56 data bytes
64 bytes from 192.168.122.1: seq=0 ttl=64 time=0.471 ms
^C
--- 192.168.122.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.471/0.471/0.471 ms
```

Ok, I up another vm with ip address 192.168.122.73 and check from docker:

```
/ # ping 192.168.122.73 -c2
PING 192.168.122.73 (192.168.122.73): 56 data bytes
64 bytes from 192.168.122.73: seq=0 ttl=64 time=1.630 ms
64 bytes from 192.168.122.73: seq=1 ttl=64 time=0.984 ms

--- 192.168.122.73 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.984/1.307/1.630 ms
```

From docker instance I can't ping interface on vm, but I can access to local network.

```
/ # ip n|grep 192.168.122.152
192.168.122.152 dev eth0  used 0/0/0 probes 6 FAILED
```

On vm I add macvlan0 nic:

```
# ip link add macvlan0 link ens3 type macvlan mode bridge
# ip addr add 192.168.122.100/24 dev macvlan0
```

```
# ip l set macvlan0 up
```

From the docker I can ping 192.168.122.100:

```
/ # ping 192.168.122.100 -c2
PING 192.168.122.100 (192.168.122.100): 56 data bytes
64 bytes from 192.168.122.100: seq=0 ttl=64 time=0.087 ms
64 bytes from 192.168.122.100: seq=1 ttl=64 time=0.132 ms

--- 192.168.122.100 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.087/0.109/0.132 ms
```

Share Improve this answer Follow

answered Jul 18, 2018 at 17:33

user4776812