Howto setup a `veth` virtual network

I'd like to setup three virtual network interfaces (`veth`) which can communicate with each other. To simulate a three node cluster, each program then binds to one `veth` interface. I'd like to do it without LXC if possible.

I tried using:

- Created three `veth` pairs: `sudo ip link add type veth`
- Created a bridge `sudo brctl addbr br0`
- Added one of each pair to the bridge:
  - `sudo brctl addif br0 veth1`
  - `sudo brctl addif br0 veth3`
  - `sudo brctl addif br0 veth5`
- Configured the interfaces:
  - `sudo ifconfig veth0 10.0.0.201 netmask 255.255.255.0 up`
  - `sudo ifconfig veth2 10.0.0.202 netmask 255.255.255.0 up`
  - `sudo ifconfig veth4 10.0.0.203 netmask 255.255.255.0 up`

Then I verified if is works using: `ping -I veth0 10.0.0.202` but it doesn't :(

The I added IP addresses to the `veth1` , `veth3` , `veth5` and `br0` interfaces in the 10.0.1.x/24 range. But that doesn't help.

Any ideas? or a guide, all I find in how to use it with LXC. Or am I trying something that isn't possible?

linux networking

edited Jun 20 '14 at 15:44
Oliver Salzburg
64.1k ● 47 ● 223 ● 274

asked Jun 7 '14 at 20:18
Reinder
100 ● 1 ● 1 ● 7

Is `br0` itself up? – grawity Jun 7 '14 at 20:23

Yes, it is up. Configured it like the veth's – Reinder Jun 7 '14 at 20:52

2 Answers

For veth to work, one end of the tunnel must be bridged with another interface. Since you want to keep this all virtual, you may bridge the vm1 end of the tunnel (vm2 is the other end of the tunnel) with a tap-type virtual interface, in a bridge called brm. Now you give IP addresses to brm and to vm2 (10.0.0.1 and 10.0.0.2, respectively), enable IPv4 forwarding by means of

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

bring all interfaces up, and add a route instructing the kernel how to reach IP addresses 10.0.0.0/24. That's all.

If you want to create more pairs, repeat the steps below with different subnets, for instance 10.0.1.0/24, 10.0.2.0/24, and so on. Since you enabled IPv4 forwarding and added appropriate routes to the kernel routing table, they will be able to talk to each other right away.

Also, remember that most of the commands you are using (brctl, ifconfig,...) are obsolete: the **iproute2** suite has commands to do all of this, see below my use of the *ip* command.

This is a correct sequence of commands for the use of interfaces of type *veth*:

first create all required interfaces,

```
ip link add dev vm1 type veth peer name vm2
ip link set dev vm1 up
ip tuntap add tapm mode tap
ip link set dev tapm up
ip link add brm type bridge
```

Notice we did not bring up brm and vm2 because we have to assign them IP addresses, but we did bring up tapm and vm1, which is necessary to include them into the bridge brm. Now enslave the interfaces tapm and vm1 to the bridge brm,

```
ip link set tapm master brm
ip link set vm1 master brm
```

now give addresses to the bridge and to the remaining veth interface vm2,

```
ip addr add 10.0.0.1/24 dev brm
ip addr add 10.0.0.2/24 dev vm2
```

now bring vm2 and brm up,

```
ip link set brm up
ip link set vm2 up
```

There is no need to add the route to the subnet 10.0.0.0/24 explicitly, it is automatically generated,, you may check with *ip route show*. This results in

```
ping -c1 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.035 m

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.035/0.035/0.035/0.000 ms
```

You can also do it backwards, *i.e.* from vm2 back to brm:

```
ping -I 10.0.0.2 -c1 10.0.0.1
PING 10.0.0.1 (10.0.0.1) from 10.0.0.2 : 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.045 ms

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.045/0.045/0.045/0.000 ms
```

The most useful application of NICs of the *veth* kind is a *network namespace*, which is what is used in Linux containers (LXC). You start one called nnsm as follows

```
ip netns add nnsm
```

then we transfer vm2 to it,

```
ip link set vm2 netns nnsm
```

we endow the new network namespace with a lo interface (absolutely necessary),

```
ip netns exec nnsm  ip link set dev lo up
```

we allow NATting in the main machine,

```
iptables -t nat -A POSTROUTING -o brm -j MASQUERADE
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

(if you are connected to the Internet via *eth0*, otherwise change accordingly), start a shell in the new network namespace,

```
ip netns exec nnsm xterm &
```

and now, if you start typing in the new xterm, you will find you are in a separate virtual machine with IP address 10.0.0.2, but you can reach the Internet. The advantage of this is that the new network namespace has its own stack, which means, for instance, you can start a VPN in it while the rest of your pc is **not** on the VPN. This is the contraption LXCs are based on.

EDIT:

I made a mistake, bringing the vm2 interface brings it down and clears its address. Thus you need to add these commands, from within the xterm:

```
ip addr add 10.0.0.2/24 dev vm2
ip link set dev  vm2 up
ip route add default via 10.0.0.1
echo "nameserver 8.8.8.8" >> /etc/resolv.conf
echo "nameserver 8.8.4.4" >> /etc/resolv.conf
```

and now you can navigate from within xterm.

The `ip` commands can also be done before the xterm with

```
ip -netns nnsm addr add 10.0.0.2/24 dev vm2
ip -netns nnsm link set dev vm2 up
ip -netns nnsm route add default via 10.0.0.1
```

edited Jul 14 '17 at 3:27
Larry
3 ● 2

answered Jun 8 '14 at 8:02
MariusMatutiae
35.8k ● 9 ● 45 ● 89

Thank you for the explanation. I only see lo in xterm, the vm2 interface is missing. – Reinder Jun 9 '14 at 20:52

Thanks again. I've made a script to setup three xterm's, and the can ping to each other :) – Reinder Jun 10 '14 at 19:11

Only one issue....When I send a UPD broadcast in one xterm the others receive the packet from 10.0.0.254 (brm). For my script see: here (can't post it in the comment) – Reinder Jun 10 '14 at 19:19

I have problem bringing up vm1 up :(# ip link add dev vm1 type veth peer name vm2 ip: RTNETLINK answers: File exists # ip link set dev vm1 up ip: SIOCGIFFLAGS: No such device – resultsway Jul 17 '14 at 23:13

@ MariusMatutiae I had to try couple of times so i agree with the first command but somehow my copy paste is not proper for the second (Iam using minicom to the device) so in short I followed exactly as suggested . I think I dont have iproute2 package . – resultsway Jul 18 '14 at 17:25

Here is a 5 node bridge setup that I use that works. You should be able to use ifconfig to assign addresses onto the NodeX interfaces

```
ip link add dev Node1s type veth peer name Node1
ip link add dev Node2s type veth peer name Node2
ip link add dev Node3s type veth peer name Node3
ip link add dev Node4s type veth peer name Node4
ip link add dev Node5s type veth peer name Node5

ip link set Node1 up
ip link set Node2 up
ip link set Node3 up
ip link set Node4 up
ip link set Node5 up

ip link set Node1s up
ip link set Node2s up
ip link set Node3s up
ip link set Node4s up
ip link set Node5s up

brctl addbr Br
ifconfig Br up

brctl addif Br Node1s
brctl addif Br Node2s
brctl addif Br Node3s
brctl addif Br Node4s
brctl addif Br Node5s
```

and to clean up

```
brctl delif Br Node1s
brctl delif Br Node2s
brctl delif Br Node3s
brctl delif Br Node4s
brctl delif Br Node5s
brctl delif Br Node1
brctl delif Br Node2
brctl delif Br Node3
brctl delif Br Node4
brctl delif Br Node5

ifconfig Br down
brctl delbr Br

ip link del dev Node1
ip link del dev Node2
ip link del dev Node3
ip link del dev Node4
ip link del dev Node5
```

answered Mar 17 at 3:45

Neil McGill
171 • 7