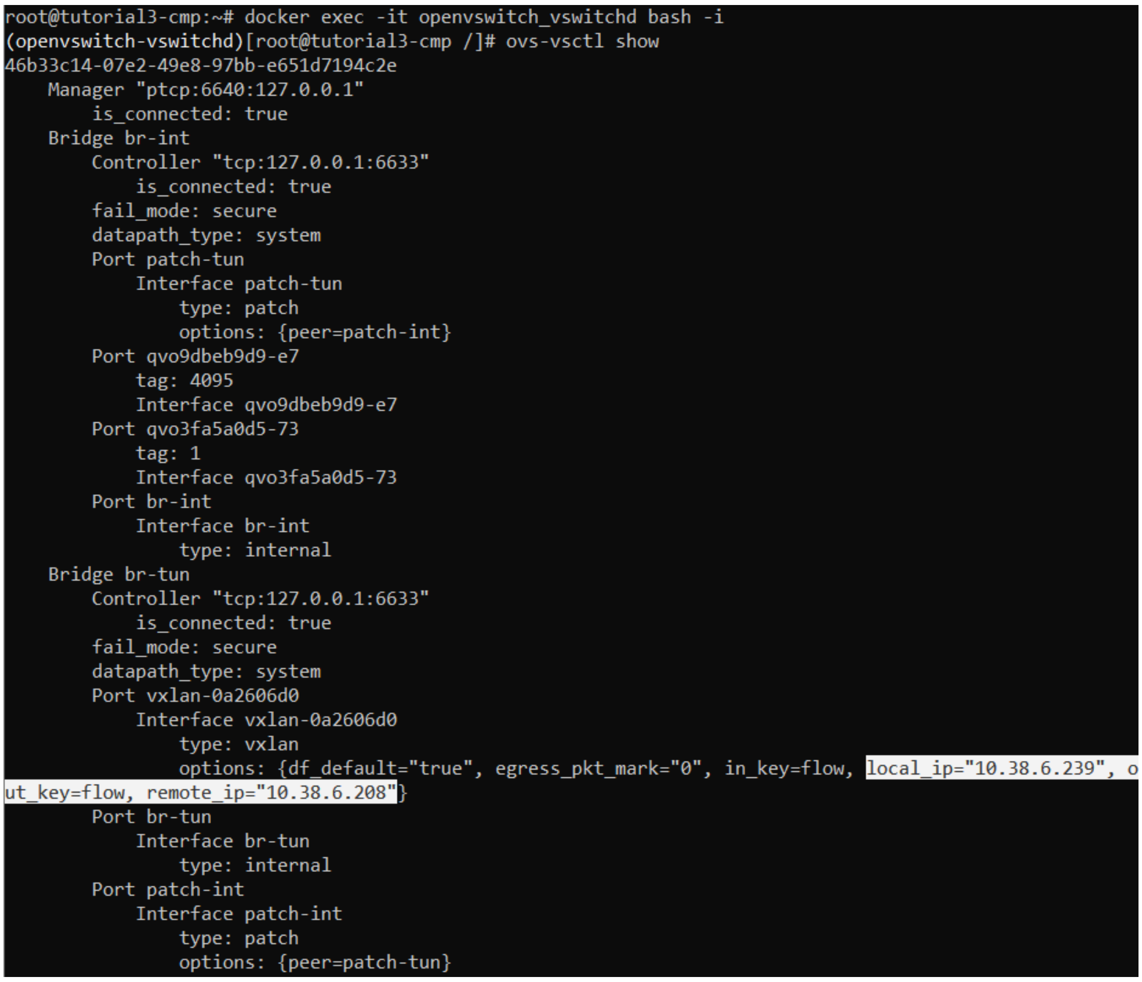
Tutorial 4

First part,

We will start with VTEP = Virtual Tunnel Endpoints, for that we will need to look inside the openv switch which is in a docker container.

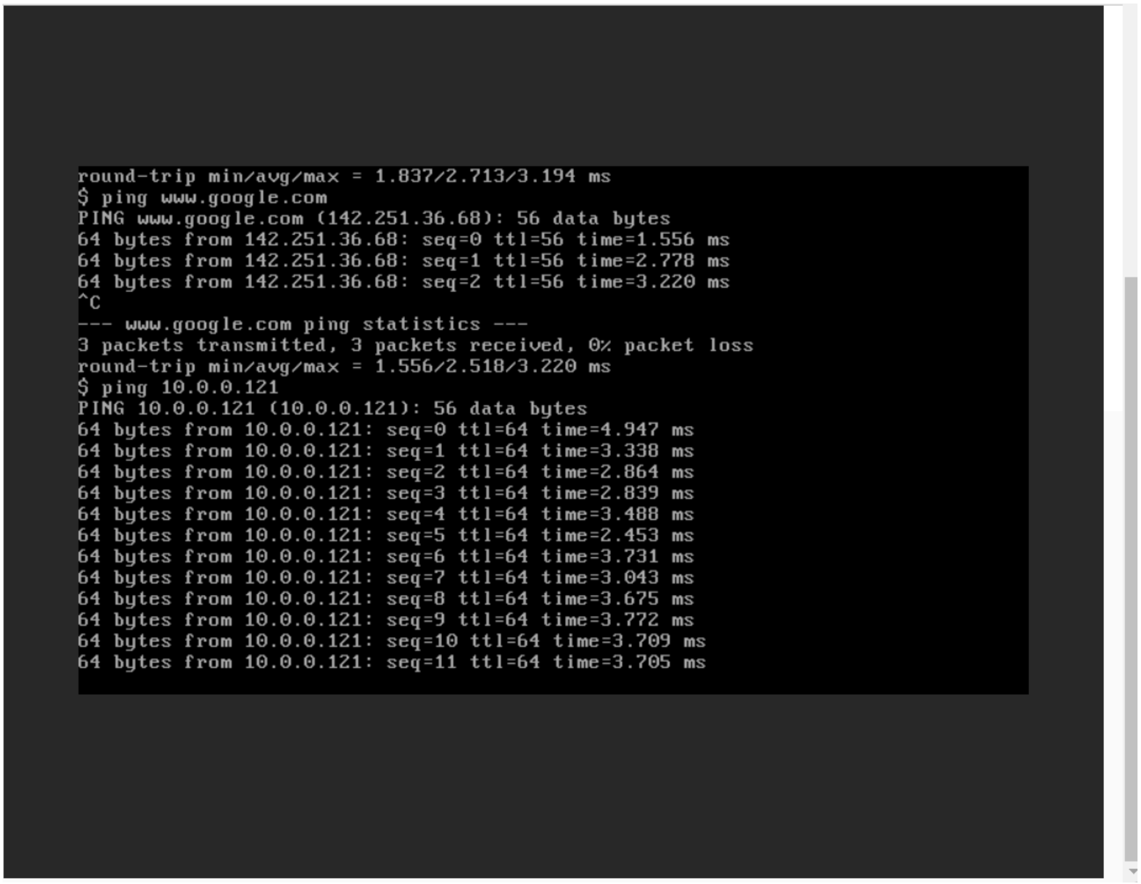


```
root@tutorial3-cmp:~# docker exec -it openvswitch_vswitchd bash -i
(openvswitch-vswitchd)[root@tutorial3-cmp /]# ovs-vsctl show
46b33c14-07e2-49e8-97bb-e651d7194c2e
    Manager "ptcp:6640:127.0.0.1"
        is_connected: true
    Bridge br-int
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        datapath_type: system
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
        Port qvo9dbeb9d9-e7
            tag: 4095
            Interface qvo9dbeb9d9-e7
        Port qvo3fa5a0d5-73
            tag: 1
            Interface qvo3fa5a0d5-73
        Port br-int
            Interface br-int
                type: internal
    Bridge br-tun
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        datapath_type: system
        Port vxlan-0a2606d0
            Interface vxlan-0a2606d0
                type: vxlan
                options: {df_default="true", egress_pkt_mark="0", in_key=flow, local_ip="10.38.6.239", o
ut_key=flow, remote_ip="10.38.6.208"}
        Port br-tun
            Interface br-tun
                type: internal
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
```

This command shows us all the ports of the virtual switch, besides we can see the vxlan interface and when traffic goes into that port is sent from the "local_ip" to the "remote_ip".

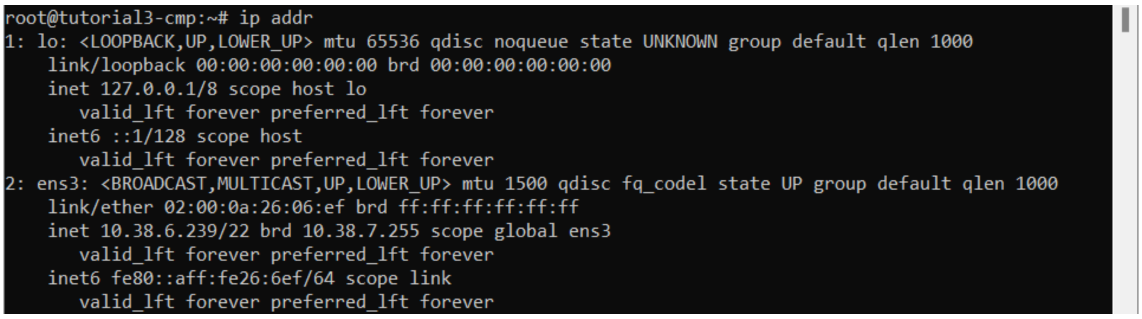
Now let's go to the openstack graphic interface and start generating some traffic, so let's ping the between the instances which are on different compute nodes.

Sergio Teodoro Castellano Betancor



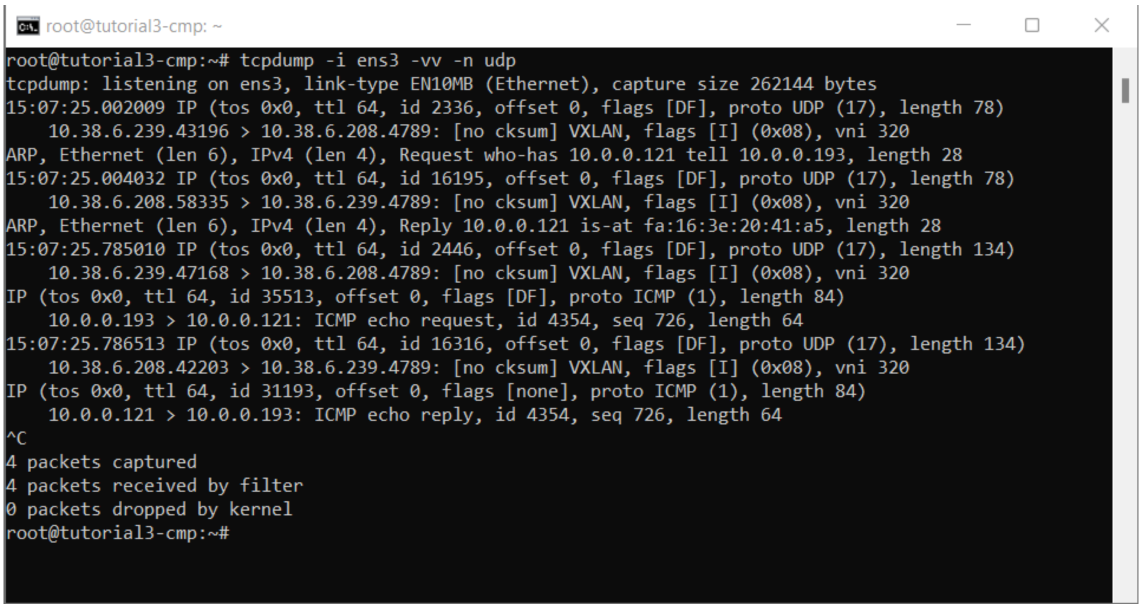
```
round-trip min/avg/max = 1.837/2.713/3.194 ms
$ ping www.google.com
PING www.google.com (142.251.36.68): 56 data bytes
64 bytes from 142.251.36.68: seq=0 ttl=56 time=1.556 ms
64 bytes from 142.251.36.68: seq=1 ttl=56 time=2.778 ms
64 bytes from 142.251.36.68: seq=2 ttl=56 time=3.220 ms
^C
--- www.google.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1.556/2.518/3.220 ms
$ ping 10.0.0.121
PING 10.0.0.121 (10.0.0.121): 56 data bytes
64 bytes from 10.0.0.121: seq=0 ttl=64 time=4.947 ms
64 bytes from 10.0.0.121: seq=1 ttl=64 time=3.338 ms
64 bytes from 10.0.0.121: seq=2 ttl=64 time=2.864 ms
64 bytes from 10.0.0.121: seq=3 ttl=64 time=2.839 ms
64 bytes from 10.0.0.121: seq=4 ttl=64 time=3.488 ms
64 bytes from 10.0.0.121: seq=5 ttl=64 time=2.453 ms
64 bytes from 10.0.0.121: seq=6 ttl=64 time=3.731 ms
64 bytes from 10.0.0.121: seq=7 ttl=64 time=3.043 ms
64 bytes from 10.0.0.121: seq=8 ttl=64 time=3.675 ms
64 bytes from 10.0.0.121: seq=9 ttl=64 time=3.772 ms
64 bytes from 10.0.0.121: seq=10 ttl=64 time=3.709 ms
64 bytes from 10.0.0.121: seq=11 ttl=64 time=3.705 ms
```

To find our vxlan we should find it in our virtual tunel endpoint which is the address of our physical interface so we need to see in which interface is the local_ip (10.38.6.239 compute node) displayed on the openvswitch, if we execute the command "ip addr" in our compute node we will see that the interface with that ip associated is ens3.



```
root@tutorial3-cmp:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:00:0a:26:06:ef brd ff:ff:ff:ff:ff:ff
    inet 10.38.6.239/22 brd 10.38.7.255 scope global ens3
       valid_lft forever preferred_lft forever
    inet6 fe80::aff:fe26:6ef/64 scope link
       valid_lft forever preferred_lft forever
```

If we check the ens3 interface in the compute node with the command "tcpdump -i ens3 -vv udp" we filter udp traffic to see this ping communication.

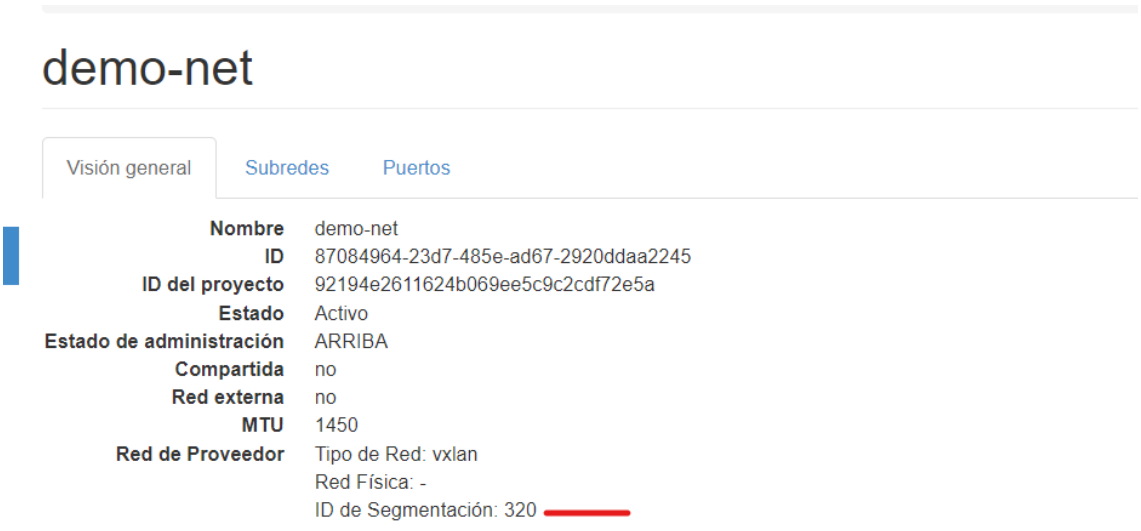


root@tutorial3-cmp: ~

```
root@tutorial3-cmp:~# tcpdump -i ens3 -vv -n udp
tcpdump: listening on ens3, link-type EN10MB (Ethernet), capture size 262144 bytes
15:07:25.002009 IP (tos 0x0, ttl 64, id 2336, offset 0, flags [DF], proto UDP (17), length 78)
    10.38.6.239.43196 > 10.38.6.208.4789: [no cksum] VXLAN, flags [I] (0x08), vni 320
ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 10.0.0.121 tell 10.0.0.193, length 28
15:07:25.004032 IP (tos 0x0, ttl 64, id 16195, offset 0, flags [DF], proto UDP (17), length 78)
    10.38.6.208.58335 > 10.38.6.239.4789: [no cksum] VXLAN, flags [I] (0x08), vni 320
ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.0.0.121 is-at fa:16:3e:20:41:a5, length 28
15:07:25.785010 IP (tos 0x0, ttl 64, id 2446, offset 0, flags [DF], proto UDP (17), length 134)
    10.38.6.239.47168 > 10.38.6.208.4789: [no cksum] VXLAN, flags [I] (0x08), vni 320
IP (tos 0x0, ttl 64, id 35513, offset 0, flags [DF], proto ICMP (1), length 84)
    10.0.0.193 > 10.0.0.121: ICMP echo request, id 4354, seq 726, length 64
15:07:25.786513 IP (tos 0x0, ttl 64, id 16316, offset 0, flags [DF], proto UDP (17), length 134)
    10.38.6.208.42203 > 10.38.6.239.4789: [no cksum] VXLAN, flags [I] (0x08), vni 320
IP (tos 0x0, ttl 64, id 31193, offset 0, flags [none], proto ICMP (1), length 84)
    10.0.0.121 > 10.0.0.193: ICMP echo reply, id 4354, seq 726, length 64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
root@tutorial3-cmp:~#
```

We can see now the pakages, with it's origin and where is it going.

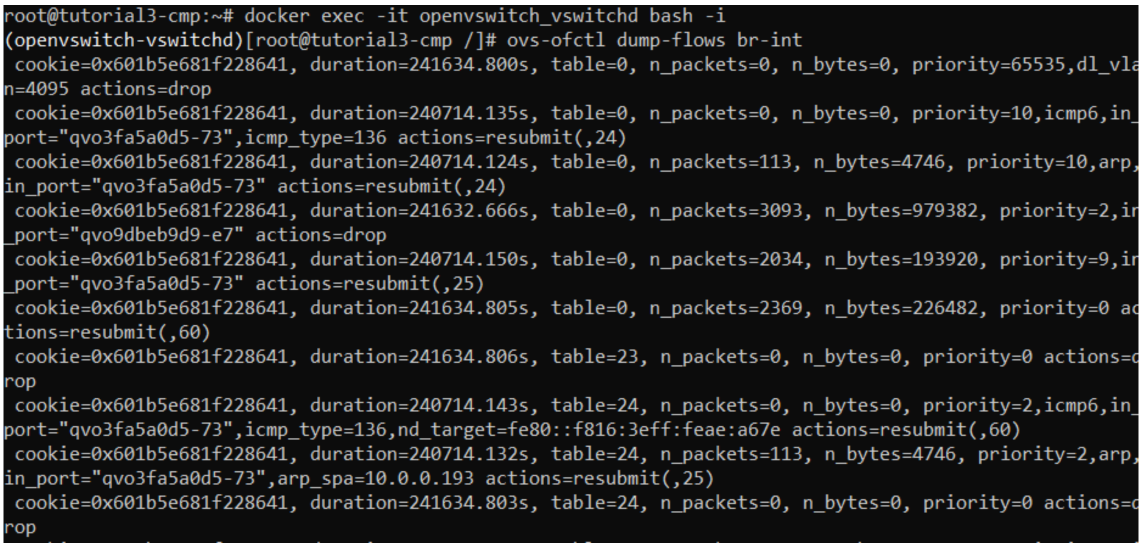
As we see we have VNI 320 which is the same of the id of segmentation of the network.



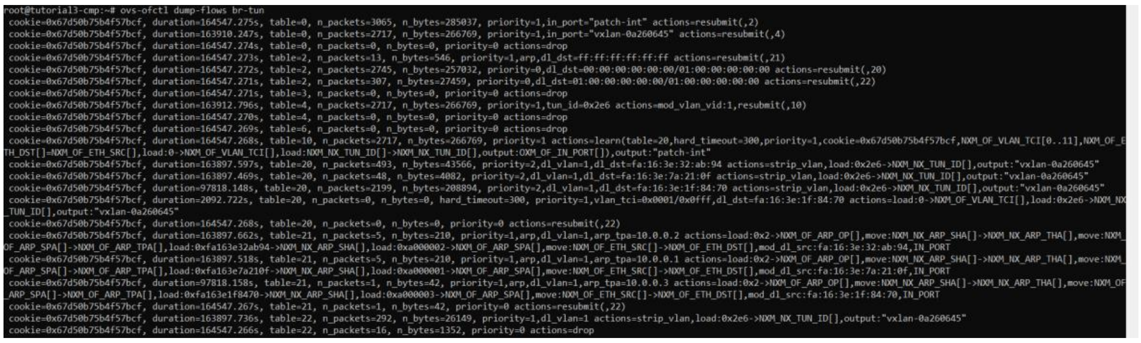
demo-net

Nombre	demo-net
ID	87084964-23d7-485e-ad67-2920ddaa2245
ID del proyecto	92194e2611624b069ee5c9c2cdf72e5a
Estado	Activo
Estado de administración	ARRIBA
Compartida	no
Red externa	no
MTU	1450
Red de Proveedor	Tipo de Red: vxlan Red Física: - ID de Segmentación: 320

Here we see the flow table on the integration bridge



```
root@tutorial3-cmp:~# docker exec -it openvswitch_vswitchd bash -i
(openvswitch-vswitchd)[root@tutorial3-cmp /]# ovs-ofctl dump-flows br-int
 cookie=0x601b5e681f228641, duration=241634.800s, table=0, n_packets=0, n_bytes=0, priority=65535,dl_vla
n=4095 actions=drop
 cookie=0x601b5e681f228641, duration=240714.135s, table=0, n_packets=0, n_bytes=0, priority=10,icmp6,in_
port="qvo3fa5a0d5-73",icmp_type=136 actions=resubmit(,24)
 cookie=0x601b5e681f228641, duration=240714.124s, table=0, n_packets=113, n_bytes=4746, priority=10,arp,
in_port="qvo3fa5a0d5-73" actions=resubmit(,24)
 cookie=0x601b5e681f228641, duration=241632.666s, table=0, n_packets=3093, n_bytes=979382, priority=2,in
_port="qvo9dbeb9d9-e7" actions=drop
 cookie=0x601b5e681f228641, duration=240714.150s, table=0, n_packets=2034, n_bytes=193920, priority=9,in
_port="qvo3fa5a0d5-73" actions=resubmit(,25)
 cookie=0x601b5e681f228641, duration=241634.805s, table=0, n_packets=2369, n_bytes=226482, priority=0 ac
tions=resubmit(,60)
 cookie=0x601b5e681f228641, duration=241634.806s, table=23, n_packets=0, n_bytes=0, priority=0 actions=d
rop
 cookie=0x601b5e681f228641, duration=240714.143s, table=24, n_packets=0, n_bytes=0, priority=2,icmp6,in_
port="qvo3fa5a0d5-73",icmp_type=136,nd_target=fe80::f816:3eff:feae:a67e actions=resubmit(,60)
 cookie=0x601b5e681f228641, duration=240714.132s, table=24, n_packets=113, n_bytes=4746, priority=2,arp,
in_port="qvo3fa5a0d5-73",arp_spa=10.0.0.193 actions=resubmit(,25)
 cookie=0x601b5e681f228641, duration=241634.803s, table=24, n_packets=0, n_bytes=0, priority=0 actions=d
rop
```

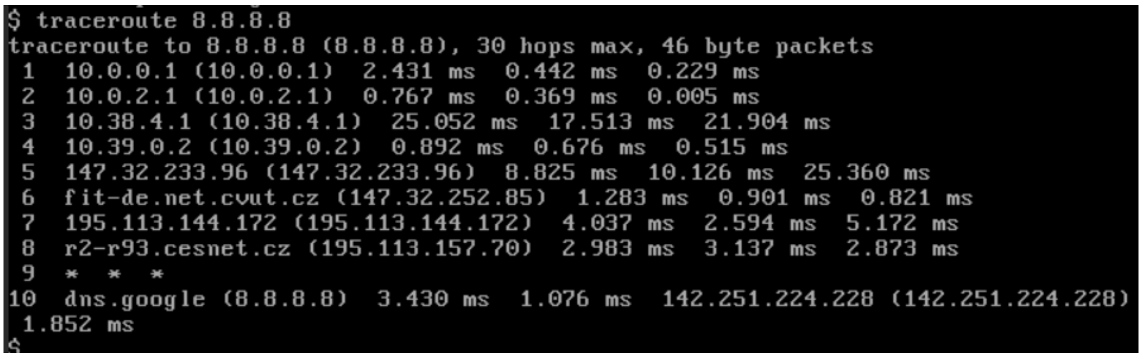


```
root@tutorial3-cmp:~# ovs-ofctl dump-flows br-tun
 cookie=0x67d50b75b4f57bcf, duration=164547.275s, table=0, n_packets=3065, n_bytes=285037, priority=1,in_port="patch-int" actions=resubmit(,2)
 cookie=0x67d50b75b4f57bcf, duration=163910.247s, table=0, n_packets=2717, n_bytes=266769, priority=1,in_port="vxlan-0a260645" actions=resubmit(,4)
 cookie=0x67d50b75b4f57bcf, duration=164547.274s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
 cookie=0x67d50b75b4f57bcf, duration=164547.273s, table=2, n_packets=13, n_bytes=546, priority=1,arp,dl_dst=ff:ff:ff:ff:ff:ff actions=resubmit(,21)
 cookie=0x67d50b75b4f57bcf, duration=164547.272s, table=2, n_packets=2745, n_bytes=257032, priority=0,dl_dst=00:00:00:00:00:00/01:00:00:00:00:00 actions=resubmit(,20)
 cookie=0x67d50b75b4f57bcf, duration=164547.271s, table=2, n_packets=307, n_bytes=27459, priority=0,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=resubmit(,22)
 cookie=0x67d50b75b4f57bcf, duration=164547.271s, table=3, n_packets=0, n_bytes=0, priority=0 actions=drop
 cookie=0x67d50b75b4f57bcf, duration=163912.796s, table=4, n_packets=2717, n_bytes=266769, priority=1,tun_id=0x2e6 actions=mod_vlan_vid:1,resubmit(,10)
 cookie=0x67d50b75b4f57bcf, duration=164547.270s, table=4, n_packets=0, n_bytes=0, priority=0 actions=drop
 cookie=0x67d50b75b4f57bcf, duration=164547.269s, table=6, n_packets=0, n_bytes=0, priority=0 actions=drop
 cookie=0x67d50b75b4f57bcf, duration=164547.268s, table=10, n_packets=2717, n_bytes=266769, priority=1 actions=learn(table=20,hard_timeout=300,priority=1,cookie=0x67d50b75b4f57bcf,NXM_OF_VLAN_TCI[0..11],NXM_OF_E
TH_DST[]=NXM_OF_ETH_SRC[],load:0->NXM_OF_VLAN_TCI[],load:NXM_NX_TUN_ID[]->NXM_NX_TUN_ID[],output:OXM_OF_IN_PORT[]),output:"patch-int"
 cookie=0x67d50b75b4f57bcf, duration=163897.597s, table=20, n_packets=493, n_bytes=43566, priority=2,dl_vlan=1,dl_dst=fa:16:3e:32:ab:94 actions=strip_vlan,load:0x2e6->NXM_NX_TUN_ID[],output:"vxlan-0a260645"
 cookie=0x67d50b75b4f57bcf, duration=163897.469s, table=20, n_packets=48, n_bytes=4082, priority=2,dl_vlan=1,dl_dst=fa:16:3e:7a:21:0f actions=strip_vlan,load:0x2e6->NXM_NX_TUN_ID[],output:"vxlan-0a260645"
 cookie=0x67d50b75b4f57bcf, duration=97818.148s, table=20, n_packets=2199, n_bytes=208894, priority=2,dl_vlan=1,dl_dst=fa:16:3e:1f:84:70 actions=strip_vlan,load:0x2e6->NXM_NX_TUN_ID[],output:"vxlan-0a260645"
 cookie=0x67d50b75b4f57bcf, duration=2092.722s, table=20, n_packets=0, n_bytes=0, hard_timeout=300, priority=1,vlan_tci=0x0001/0x0fff,dl_dst=fa:16:3e:1f:84:70 actions=load:0->NXM_OF_VLAN_TCI[],load:0x2e6->NXM_NX
_TUN_ID[],output:"vxlan-0a260645"
 cookie=0x67d50b75b4f57bcf, duration=164547.268s, table=20, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,22)
 cookie=0x67d50b75b4f57bcf, duration=163897.662s, table=21, n_packets=5, n_bytes=210, priority=1,arp,dl_vlan=1,arp_tpa=10.0.0.2 actions=load:0x2->NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_
OF_ARP_SPA[]->NXM_OF_ARP_TPA[],load:0xfa163e32ab94->NXM_NX_ARP_SHA[],load:0xa000002->NXM_OF_ARP_SPA[],move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],mod_dl_src:fa:16:3e:32:ab:94,IN_PORT
 cookie=0x67d50b75b4f57bcf, duration=163897.518s, table=21, n_packets=5, n_bytes=210, priority=1,arp,dl_vlan=1,arp_tpa=10.0.0.1 actions=load:0x2->NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_
OF_ARP_SPA[]->NXM_OF_ARP_TPA[],load:0xfa163e7a210f->NXM_NX_ARP_SHA[],load:0xa000001->NXM_OF_ARP_SPA[],move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],mod_dl_src:fa:16:3e:7a:21:0f,IN_PORT
 cookie=0x67d50b75b4f57bcf, duration=97818.158s, table=21, n_packets=1, n_bytes=42, priority=1,arp,dl_vlan=1,arp_tpa=10.0.0.3 actions=load:0x2->NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF
_ARP_SPA[]->NXM_OF_ARP_TPA[],load:0xfa163e1f8470->NXM_NX_ARP_SHA[],load:0xa000003->NXM_OF_ARP_SPA[],move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],mod_dl_src:fa:16:3e:1f:84:70,IN_PORT
 cookie=0x67d50b75b4f57bcf, duration=164547.267s, table=21, n_packets=1, n_bytes=42, priority=0 actions=resubmit(,22)
 cookie=0x67d50b75b4f57bcf, duration=163897.736s, table=22, n_packets=292, n_bytes=26149, priority=1,dl_vlan=1 actions=strip_vlan,load:0x2e6->NXM_NX_TUN_ID[],output:"vxlan-0a260645"
 cookie=0x67d50b75b4f57bcf, duration=164547.266s, table=22, n_packets=16, n_bytes=1352, priority=0 actions=drop
```

Now let's start with the assignment.

1. Find the xml descriptor of the virtual machine and the interface names in it. Traffic tracing between 2 VMs or between a VM and a router or DHCP server on the interface belonging to the virtual machine on the hypervisor.

In my case I'm going to analyse the traffic from an instance to the internet most specifically to google (8.8.8.8).

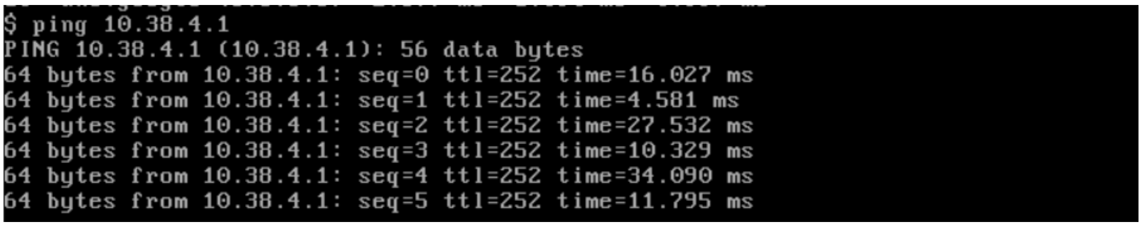


```
$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 46 byte packets
 1  10.0.0.1 (10.0.0.1)  2.431 ms  0.442 ms  0.229 ms
 2  10.0.2.1 (10.0.2.1)  0.767 ms  0.369 ms  0.005 ms
 3  10.38.4.1 (10.38.4.1)  25.052 ms  17.513 ms  21.904 ms
 4  10.39.0.2 (10.39.0.2)  0.892 ms  0.676 ms  0.515 ms
 5  147.32.233.96 (147.32.233.96)  8.825 ms  10.126 ms  25.360 ms
 6  fit-de.net.cvut.cz (147.32.252.85)  1.283 ms  0.901 ms  0.821 ms
 7  195.113.144.172 (195.113.144.172)  4.037 ms  2.594 ms  5.172 ms
 8  r2-r93.cesnet.cz (195.113.157.70)  2.983 ms  3.137 ms  2.873 ms
 9  *  *  *
10  dns.google (8.8.8.8)  3.430 ms  1.076 ms  142.251.224.228 (142.251.224.228)
 1.852 ms
$
```

Here we see the route of the packages from our instance until those packages arrive to 8.8.8.8

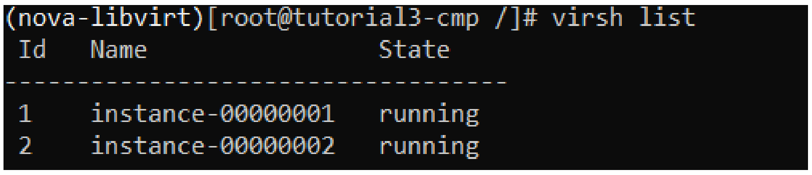
Now let's ping to outside to the router of the open nebula pinging to the

10.38.4.1 ip address



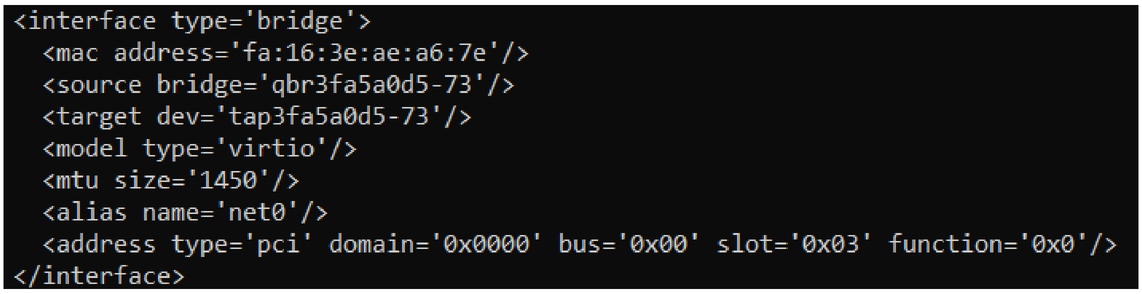
```
$ ping 10.38.4.1
PING 10.38.4.1 (10.38.4.1): 56 data bytes
64 bytes from 10.38.4.1: seq=0 ttl=252 time=16.027 ms
64 bytes from 10.38.4.1: seq=1 ttl=252 time=4.581 ms
64 bytes from 10.38.4.1: seq=2 ttl=252 time=27.532 ms
64 bytes from 10.38.4.1: seq=3 ttl=252 time=10.329 ms
64 bytes from 10.38.4.1: seq=4 ttl=252 time=34.090 ms
64 bytes from 10.38.4.1: seq=5 ttl=252 time=11.795 ms
```

Now to identify which instance is doing this communication, to know which instance is generating the traffic we have to take a look to the libvirt xml descriptor of the instance, for that we need to run the docker container which host libvirt with the command "docker exec -it nova_libvirt bash -i" and then with the command "virsh list" to get the instance name.



```
(nova-libvirt)[root@tutorial3-cmp /]# virsh list
 Id    Name                   State
----------------------------------
 1     instance-00000001      running
 2     instance-00000002      running
```

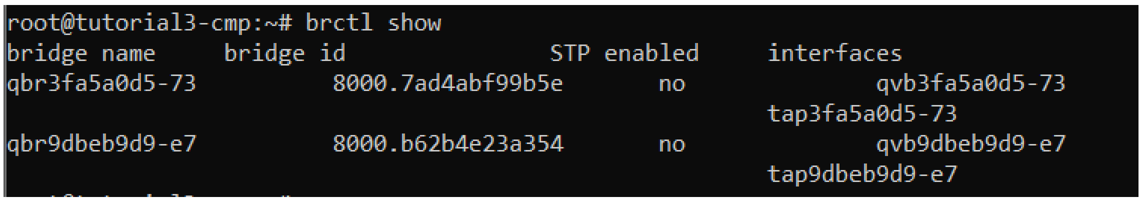
We can also see the the xml of the instance, now let's execute "virsh dumpxml instance-00000002"



```
<interface type='bridge'>
  <mac address='fa:16:3e:ae:a6:7e'/>
  <source bridge='qbr3fa5a0d5-73'/>
  <target dev='tap3fa5a0d5-73'/>
  <model type='virtio'/>
  <mtu size='1450'/>
  <alias name='net0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

Here we can see the definition of the network interface, here we can see data as the mac address and the tap interface name and, the name of the bridge.

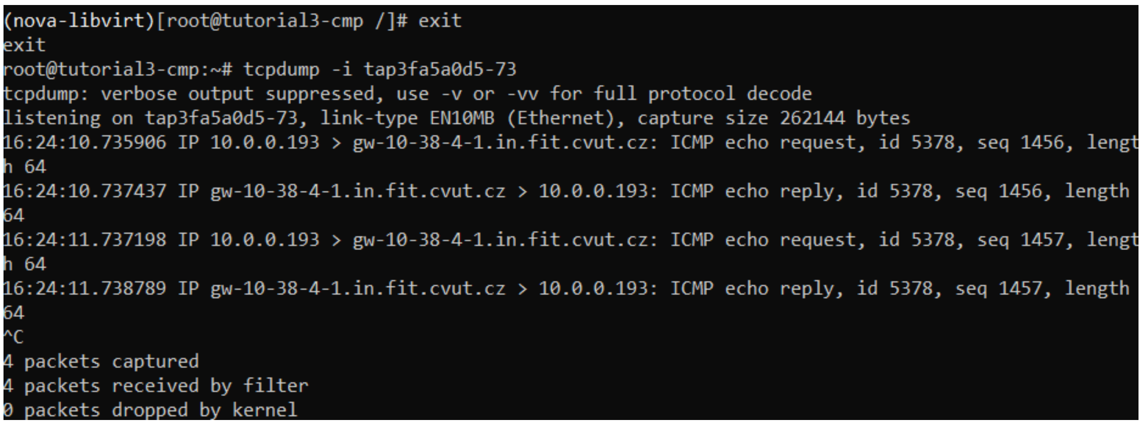
Now let's execute the command "brctl show" out of the container



```
root@tutorial3-cmp:~# brctl show
bridge name     bridge id               STP enabled     interfaces
qbr3fa5a0d5-73          8000.7ad4abf99b5e       no              qvb3fa5a0d5-73
                                                        tap3fa5a0d5-73
qbr9dbeb9d9-e7          8000.b62b4e23a354       no              qvb9dbeb9d9-e7
                                                        tap9dbeb9d9-e7
```

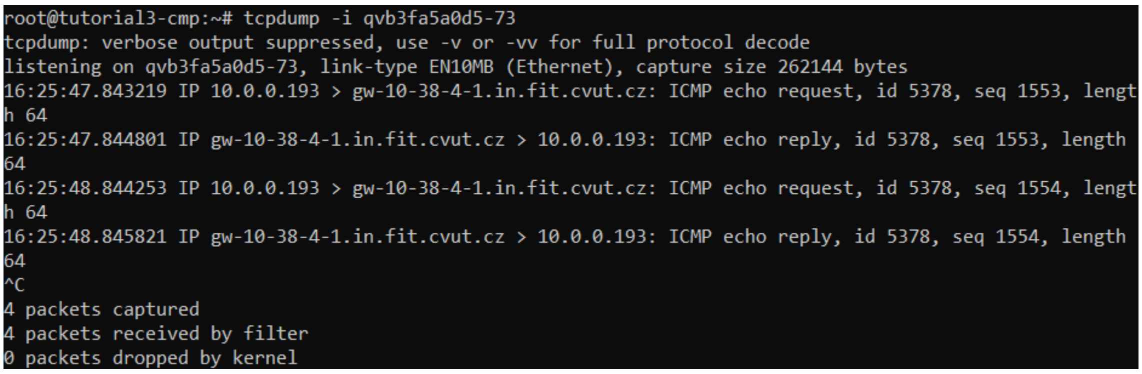
So in the first (qbr3fa5a…) bridge we can see port tab and port qvb which are the ports before and after the firewall.

If we check the tap interface of the first bridge, with the command "tcpdump -i tap3fa5a0d5-73", we will see our ping comunication.



```
(nova-libvirt)[root@tutorial3-cmp /]# exit
exit
root@tutorial3-cmp:~# tcpdump -i tap3fa5a0d5-73
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tap3fa5a0d5-73, link-type EN10MB (Ethernet), capture size 262144 bytes
16:24:10.735906 IP 10.0.0.193 > gw-10-38-4-1.in.fit.cvut.cz: ICMP echo request, id 5378, seq 1456, lengt
h 64
16:24:10.737437 IP gw-10-38-4-1.in.fit.cvut.cz > 10.0.0.193: ICMP echo reply, id 5378, seq 1456, length
64
16:24:11.737198 IP 10.0.0.193 > gw-10-38-4-1.in.fit.cvut.cz: ICMP echo request, id 5378, seq 1457, lengt
h 64
16:24:11.738789 IP gw-10-38-4-1.in.fit.cvut.cz > 10.0.0.193: ICMP echo reply, id 5378, seq 1457, length
64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
```

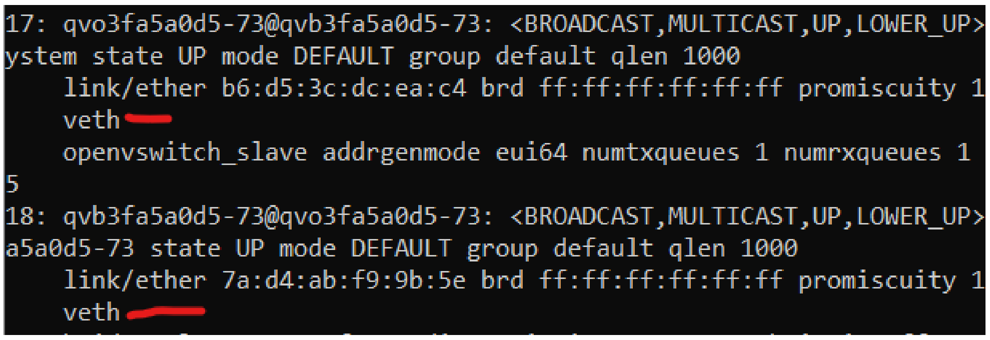
And if we check the qvb interface we will see our ping after going through the firewall



```
root@tutorial3-cmp:~# tcpdump -i qvb3fa5a0d5-73
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on qvb3fa5a0d5-73, link-type EN10MB (Ethernet), capture size 262144 bytes
16:25:47.843219 IP 10.0.0.193 > gw-10-38-4-1.in.fit.cvut.cz: ICMP echo request, id 5378, seq 1553, lengt
h 64
16:25:47.844801 IP gw-10-38-4-1.in.fit.cvut.cz > 10.0.0.193: ICMP echo reply, id 5378, seq 1553, length
64
16:25:48.844253 IP 10.0.0.193 > gw-10-38-4-1.in.fit.cvut.cz: ICMP echo request, id 5378, seq 1554, lengt
h 64
16:25:48.845821 IP gw-10-38-4-1.in.fit.cvut.cz > 10.0.0.193: ICMP echo reply, id 5378, seq 1554, length
64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
```

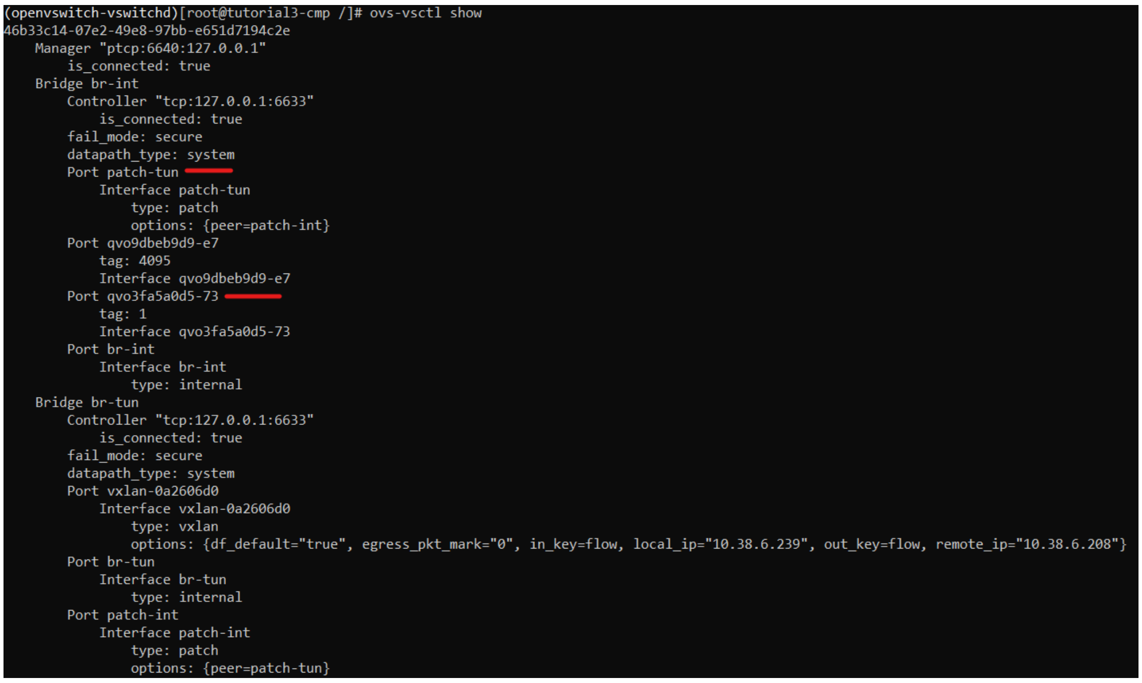
This is possible because our firewall is configured in a way that it let's ICMP traffic goes through.

So, we are on the last component of the linux bridge and to reach the OVS Integration Bridge we need to go trought veth which will connect us to qvo interface, this is visible with the result of the command "ip -d link" the qvb interfacte uses veth and also qvo too



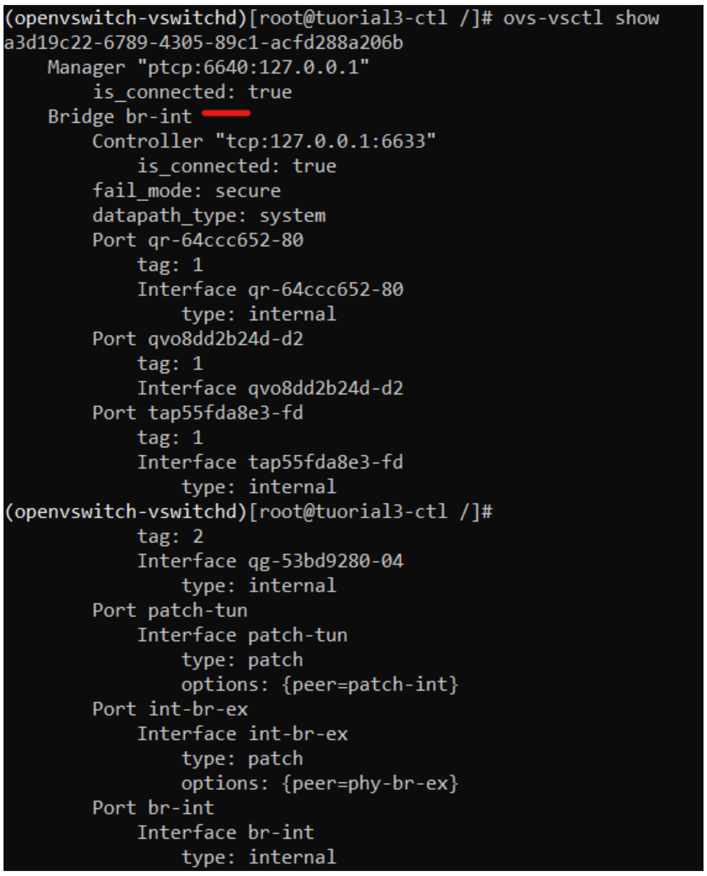
```
17: qvo3fa5a0d5-73@qvb3fa5a0d5-73: <BROADCAST,MULTICAST,UP,LOWER_UP>
ystem state UP mode DEFAULT group default qlen 1000
    link/ether b6:d5:3c:dc:ea:c4 brd ff:ff:ff:ff:ff:ff promiscuity 1
    veth
    openvswitch_slave addrgenmode eui64 numtxqueues 1 numrxqueues 1
5
18: qvb3fa5a0d5-73@qvo3fa5a0d5-73: <BROADCAST,MULTICAST,UP,LOWER_UP>
a5a0d5-73 state UP mode DEFAULT group default qlen 1000
    link/ether 7a:d4:ab:f9:9b:5e brd ff:ff:ff:ff:ff:ff promiscuity 1
    veth
```

Now inside the docker openvswitch container we will execute "ovs-vsctl show"



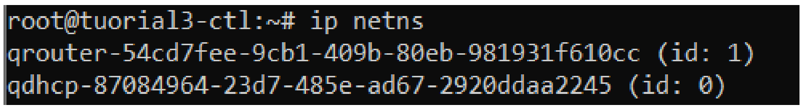
```
(openvswitch-vswitchd)[root@tutorial3-cmp /]# ovs-vsctl show
46b33c14-07e2-49e8-97bb-e651d7194c2e
    Manager "ptcp:6640:127.0.0.1"
        is_connected: true
    Bridge br-int
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        datapath_type: system
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
        Port qvo9dbeb9d9-e7
            tag: 4095
            Interface qvo9dbeb9d9-e7
        Port qvo3fa5a0d5-73
            tag: 1
            Interface qvo3fa5a0d5-73
        Port br-int
            Interface br-int
                type: internal
    Bridge br-tun
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        datapath_type: system
        Port vxlan-0a2606d0
            Interface vxlan-0a2606d0
                type: vxlan
                options: {df_default="true", egress_pkt_mark="0", in_key=flow, local_ip="10.38.6.239", out_key=flow, remote_ip="10.38.6.208"}
        Port br-tun
            Interface br-tun
                type: internal
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
```

Here we can see the Patch turn which connects the OVS Integration Bridge with the OVS Tunnel Bridge, we see that there is only one way packages go out and it is through the vxlan interface, after that, our data will go to the bridge tunel on the network node, and from tunel bridge of the network node we will go to the integration bridge of the network node.



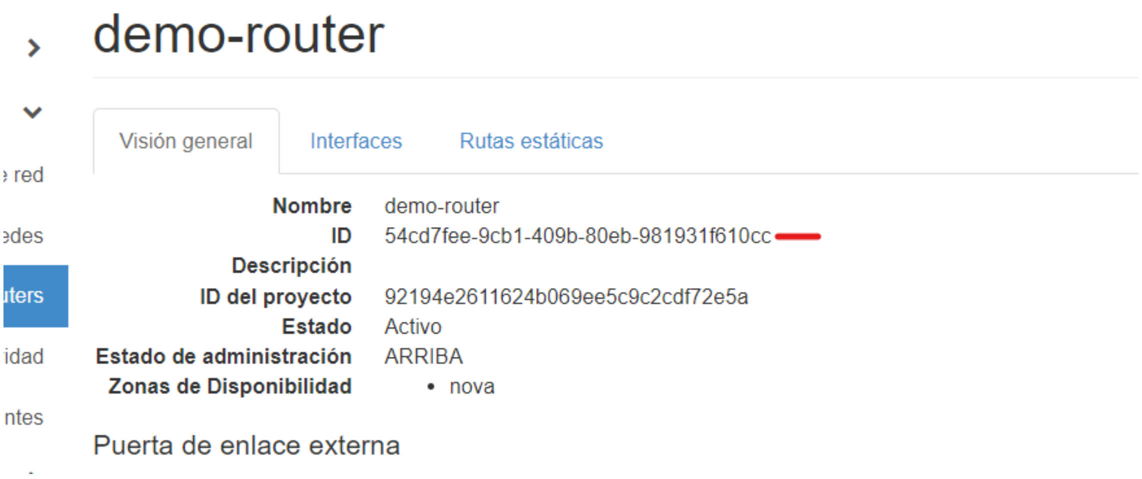
```
(openvswitch-vswitchd)[root@tuorial3-ctl /]# ovs-vsctl show
a3d19c22-6789-4305-89c1-acfd288a206b
    Manager "ptcp:6640:127.0.0.1"
        is_connected: true
    Bridge br-int
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        datapath_type: system
        Port qr-64ccc652-80
            tag: 1
            Interface qr-64ccc652-80
                type: internal
        Port qvo8dd2b24d-d2
            tag: 1
            Interface qvo8dd2b24d-d2
        Port tap55fda8e3-fd
            tag: 1
            Interface tap55fda8e3-fd
                type: internal
(openvswitch-vswitchd)[root@tuorial3-ctl /]#
            tag: 2
            Interface qg-53bd9280-04
                type: internal
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
        Port int-br-ex
            Interface int-br-ex
                type: patch
                options: {peer=phy-br-ex}
        Port br-int
            Interface br-int
                type: internal
```

Now we want to go to the router which is implemented as namespace according to the diagram, this router separates: interfaces, routing tables, firewalls, etc.



```
root@tuorial3-ctl:~# ip netns
qrouter-54cd7fee-9cb1-409b-80eb-981931f610cc (id: 1)
qdhcp-87084964-23d7-485e-ad67-2920ddaa2245 (id: 0)
```

Here we see the router namespace name which should have the same name as the router id of our router in our cloud in openstack



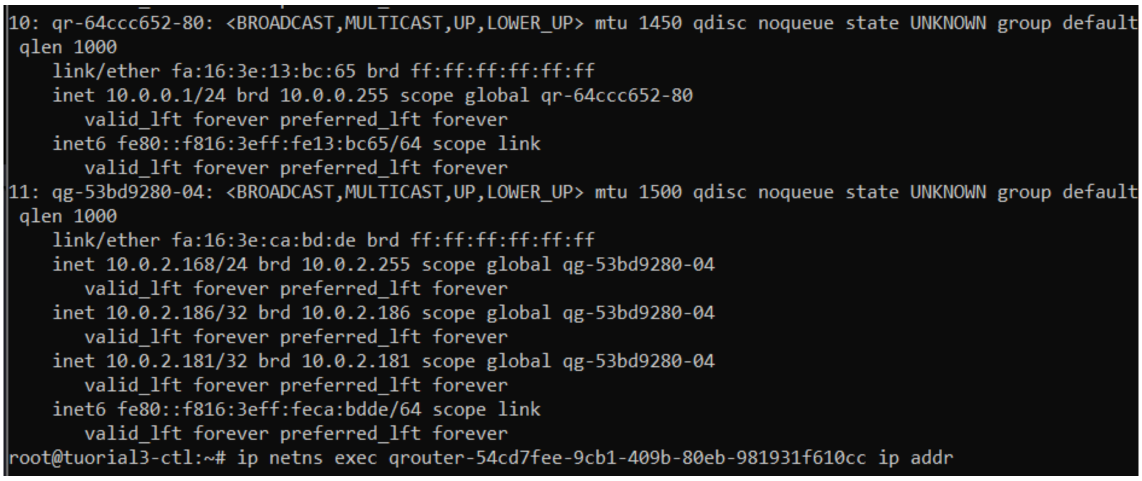
demo-router

Visión general | Interfaces | Rutas estáticas

Nombre	demo-router
ID	54cd7fee-9cb1-409b-80eb-981931f610cc
Descripción	
ID del proyecto	92194e2611624b069ee5c9c2cdf72e5a
Estado	Activo
Estado de administración	ARRIBA
Zonas de Disponibilidad	• nova

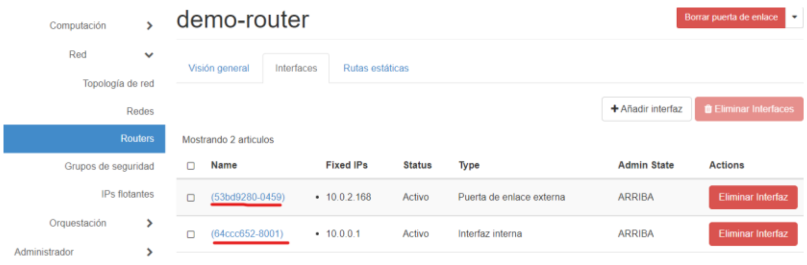
Puerta de enlace externa

Then to access to the name space we will run "ip netns exec qrouter-54cd7fee-9cb1-409b-80eb-981931f610cc ip addr"



```
10: qr-64ccc652-80: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default
 qlen 1000
    link/ether fa:16:3e:13:bc:65 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global qr-64ccc652-80
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe13:bc65/64 scope link
       valid_lft forever preferred_lft forever
11: qg-53bd9280-04: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default
 qlen 1000
    link/ether fa:16:3e:ca:bd:de brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.168/24 brd 10.0.2.255 scope global qg-53bd9280-04
       valid_lft forever preferred_lft forever
    inet 10.0.2.186/32 brd 10.0.2.186 scope global qg-53bd9280-04
       valid_lft forever preferred_lft forever
    inet 10.0.2.181/32 brd 10.0.2.181 scope global qg-53bd9280-04
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:feca:bdde/64 scope link
       valid_lft forever preferred_lft forever
root@tuorial3-ctl:~# ip netns exec qrouter-54cd7fee-9cb1-409b-80eb-981931f610cc ip addr
```

And we will see the two interface qr and qg



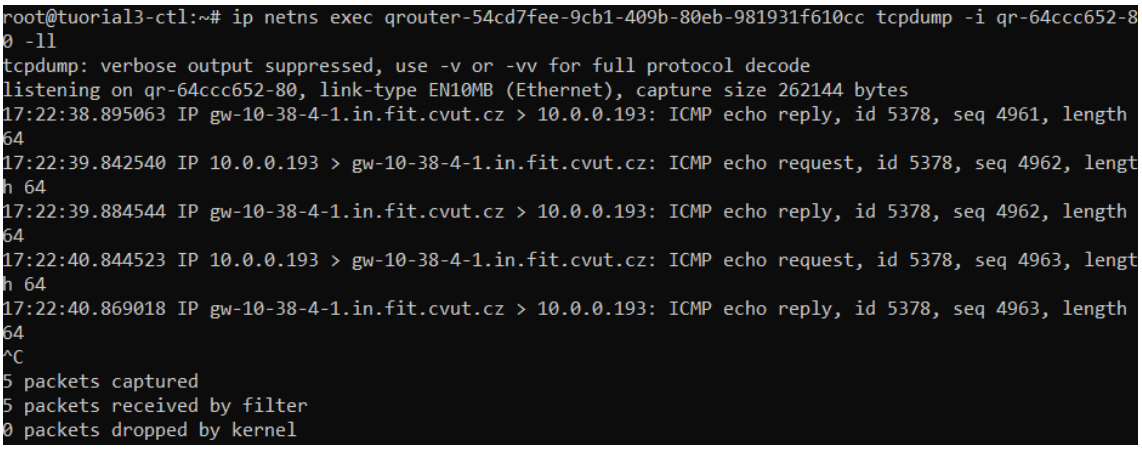
demo-router

Mostrando 2 articulos

Name	Fixed IPs	Status	Type	Admin State	Actions
(53bd9280-0459)	• 10.0.2.168	Activo	Puerta de enlace externa	ARRIBA	Eliminar Interfaz
(64ccc652-8001)	• 10.0.0.1	Activo	Interfaz interna	ARRIBA	Eliminar Interfaz

Which are the internal and external interfaces of the router.

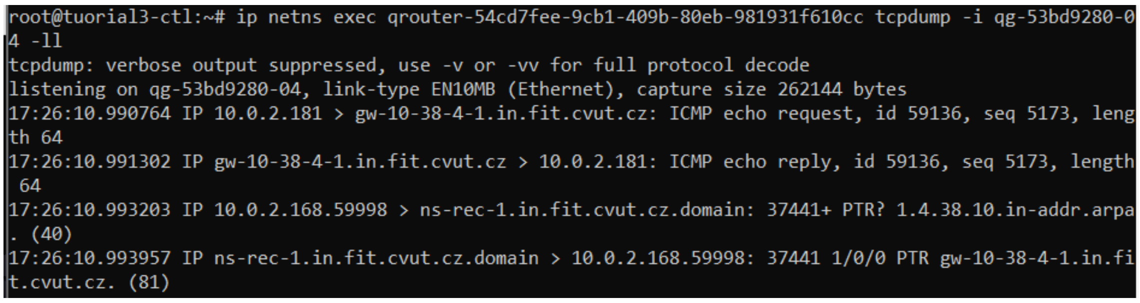
Now if we execute “ip netns exec qrouter-54cd7fee-9cb1-409b-80eb-981931f610cc tcpdump -i qr-64ccc652-80 -ll” to inspect before the router



```
root@tuorial3-ctl:~# ip netns exec qrouter-54cd7fee-9cb1-409b-80eb-981931f610cc tcpdump -i qr-64ccc652-8
0 -ll
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on qr-64ccc652-80, link-type EN10MB (Ethernet), capture size 262144 bytes
17:22:38.895063 IP gw-10-38-4-1.in.fit.cvut.cz > 10.0.0.193: ICMP echo reply, id 5378, seq 4961, length
64
17:22:39.842540 IP 10.0.0.193 > gw-10-38-4-1.in.fit.cvut.cz: ICMP echo request, id 5378, seq 4962, lengt
h 64
17:22:39.884544 IP gw-10-38-4-1.in.fit.cvut.cz > 10.0.0.193: ICMP echo reply, id 5378, seq 4962, length
64
17:22:40.844523 IP 10.0.0.193 > gw-10-38-4-1.in.fit.cvut.cz: ICMP echo request, id 5378, seq 4963, lengt
h 64
17:22:40.869018 IP gw-10-38-4-1.in.fit.cvut.cz > 10.0.0.193: ICMP echo reply, id 5378, seq 4963, length
64
^C
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

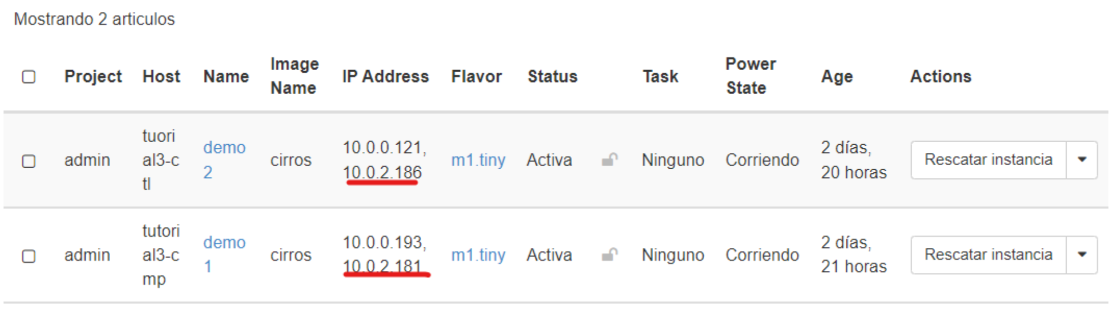
It shows us those pings which are no longer on the vxlan and we can see it goes from the address of our instance to the gateway on the internet.

Now if we inspect after the router with the command “ip netns exec qrouter-54cd7fee-9cb1-409b-80eb-981931f610cc tcpdump -i qg-53bd9280-04 -ll”



```
root@tuorial3-ctl:~# ip netns exec qrouter-54cd7fee-9cb1-409b-80eb-981931f610cc tcpdump -i qg-53bd9280-0
4 -ll
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on qg-53bd9280-04, link-type EN10MB (Ethernet), capture size 262144 bytes
17:26:10.990764 IP 10.0.2.181 > gw-10-38-4-1.in.fit.cvut.cz: ICMP echo request, id 59136, seq 5173, leng
th 64
17:26:10.991302 IP gw-10-38-4-1.in.fit.cvut.cz > 10.0.2.181: ICMP echo reply, id 59136, seq 5173, length
 64
17:26:10.993203 IP 10.0.2.168.59998 > ns-rec-1.in.fit.cvut.cz.domain: 37441+ PTR? 1.4.38.10.in-addr.arpa
. (40)
17:26:10.993957 IP ns-rec-1.in.fit.cvut.cz.domain > 10.0.2.168.59998: 37441 1/0/0 PTR gw-10-38-4-1.in.fi
t.cvut.cz. (81)
```

We are going from the external gateway which means that we are doing nat normally. Now to access the instance the other way, we need to use floating IP to expose instances to the internet by using public IP (floating IPs has been added by me in the previous tutorials that’s why I’m not going to document the process here).

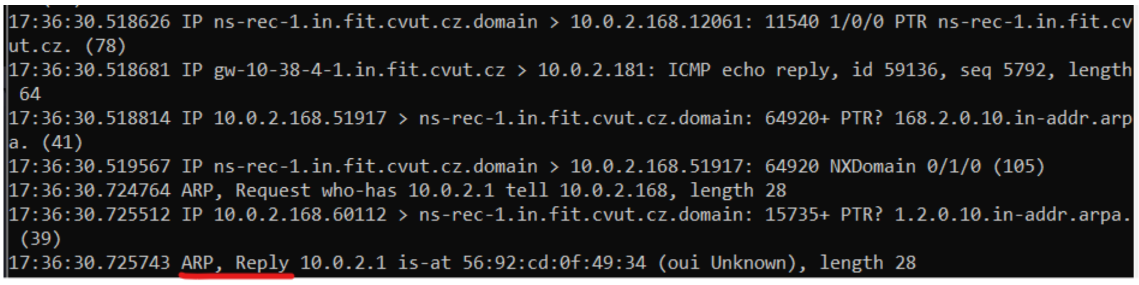


Mostrando 2 articulos

☐	Project	Host	Name	Image Name	IP Address	Flavor	Status		Task	Power State	Age	Actions
☐	admin	tuorial3-ctl	demo2	cirros	10.0.0.121, 10.0.2.186	m1.tiny	Activa		Ninguno	Corriendo	2 días, 20 horas	Rescatar instancia
☐	admin	tutorial3-cmp	demo1	cirros	10.0.0.193, 10.0.2.181	m1.tiny	Activa		Ninguno	Corriendo	2 días, 21 horas	Rescatar instancia

This why in the execution of the previous command we see some ARP replies

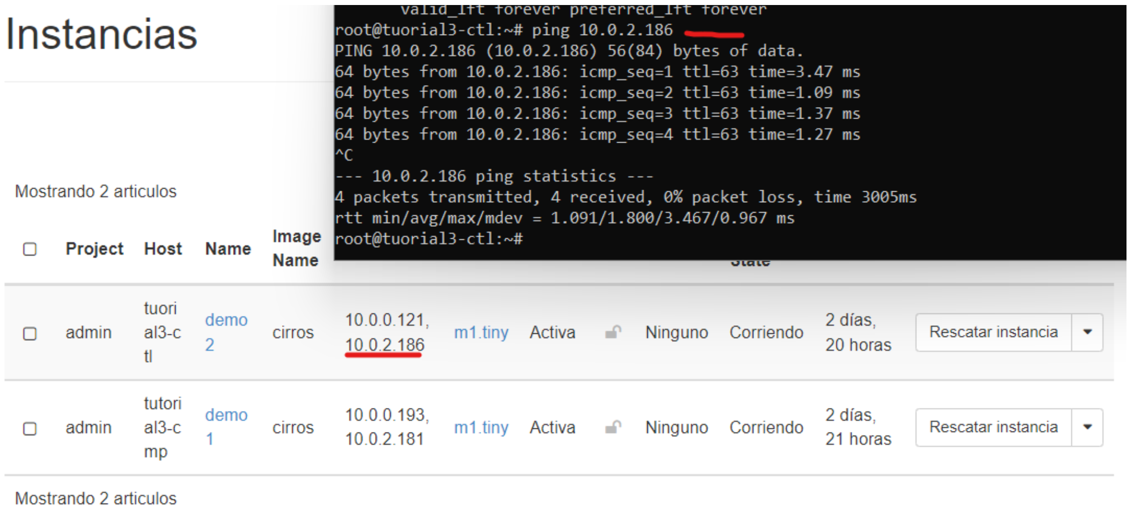
Sergio Teodoro Castellano Betancor



```
17:36:30.518626 IP ns-rec-1.in.fit.cvut.cz.domain > 10.0.2.168.12061: 11540 1/0/0 PTR ns-rec-1.in.fit.cv
ut.cz. (78)
17:36:30.518681 IP gw-10-38-4-1.in.fit.cvut.cz > 10.0.2.181: ICMP echo reply, id 59136, seq 5792, length
 64
17:36:30.518814 IP 10.0.2.168.51917 > ns-rec-1.in.fit.cvut.cz.domain: 64920+ PTR? 168.2.0.10.in-addr.arp
a. (41)
17:36:30.519567 IP ns-rec-1.in.fit.cvut.cz.domain > 10.0.2.168.51917: 64920 NXDomain 0/1/0 (105)
17:36:30.724764 ARP, Request who-has 10.0.2.1 tell 10.0.2.168, length 28
17:36:30.725512 IP 10.0.2.168.60112 > ns-rec-1.in.fit.cvut.cz.domain: 15735+ PTR? 1.2.0.10.in-addr.arpa.
 (39)
17:36:30.725743 ARP, Reply 10.0.2.1 is-at 56:92:cd:0f:49:34 (oui Unknown), length 28
```

This explains why before when we execute the command "ip netns exec qrouter-54cd7fee-9cb1-409b-80eb-981931f610cc ip addr" the router takes the floating ip of the instance and answers the ARP requests to that address, so this is how it is done in layer 2.

So now if we want, we should be able to get to the instance from the outside so if we ping one of the instances to their floating IP they should replay.



```
    valid_lft forever preferred_lft forever
root@tuorial3-ctl:~# ping 10.0.2.186
PING 10.0.2.186 (10.0.2.186) 56(84) bytes of data.
64 bytes from 10.0.2.186: icmp_seq=1 ttl=63 time=3.47 ms
64 bytes from 10.0.2.186: icmp_seq=2 ttl=63 time=1.09 ms
64 bytes from 10.0.2.186: icmp_seq=3 ttl=63 time=1.37 ms
64 bytes from 10.0.2.186: icmp_seq=4 ttl=63 time=1.27 ms
^C
--- 10.0.2.186 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.091/1.800/3.467/0.967 ms
root@tuorial3-ctl:~#
```

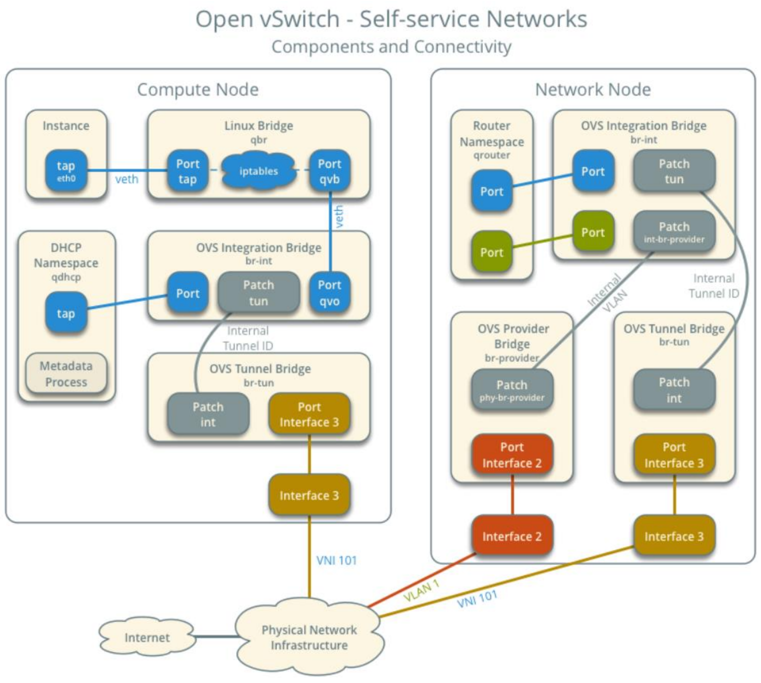
Instancias

Mostrando 2 articulos

☐	Project	Host	Name	Image Name								
☐	admin	tuorial3-ctl	demo2	cirros	10.0.0.121, 10.0.2.186	m1.tiny	Activa	🔓	Ninguno	Corriendo	2 días, 20 horas	Rescatar instancia ▾
☐	admin	tutorial3-cmp	demo1	cirros	10.0.0.193, 10.0.2.181	m1.tiny	Activa	🔓	Ninguno	Corriendo	2 días, 21 horas	Rescatar instancia ▾

Mostrando 2 articulos

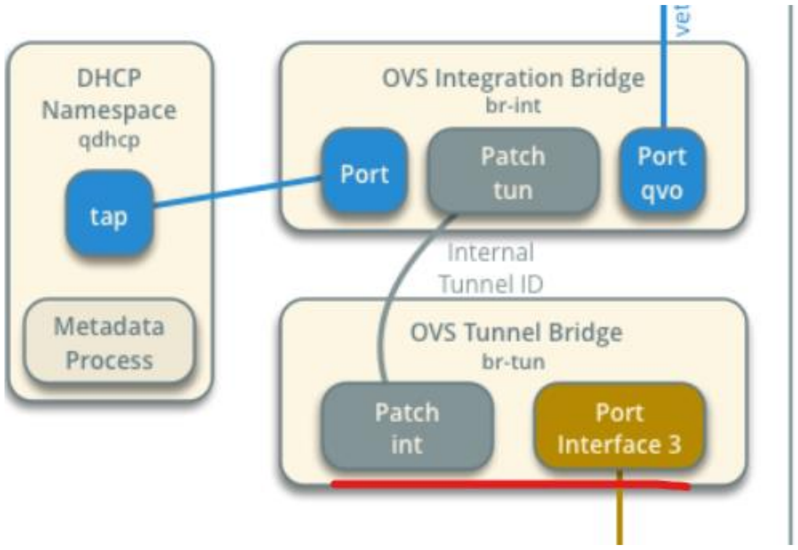
So this was north-south and back again.



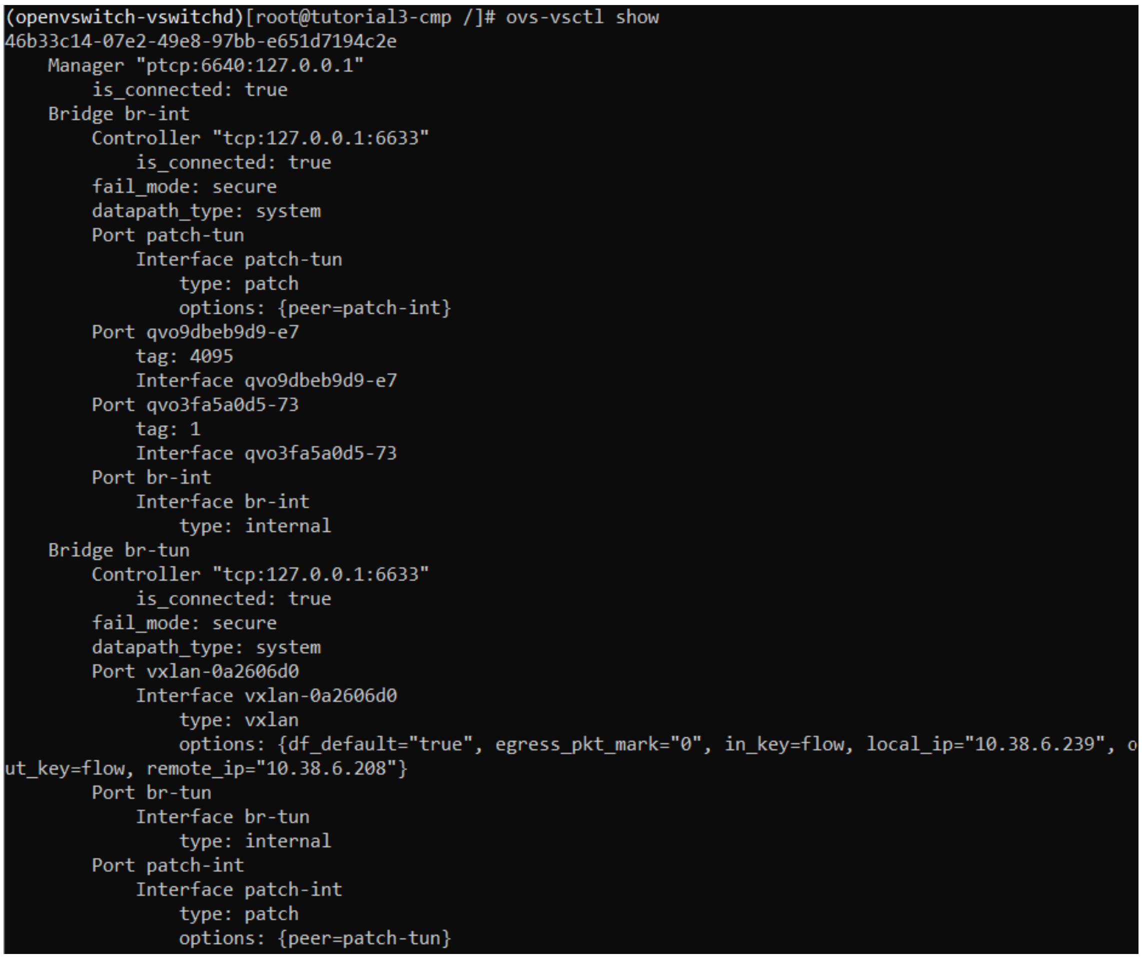
Sergio Teodoro Castellano Betancor

2. Trace of VXLAN tunnelled traffic between two nodes, displayed in the Wireshark graphical tool.

For this, considering that I have two instances each running in a different compute node so I will start pinging from one to the other, now

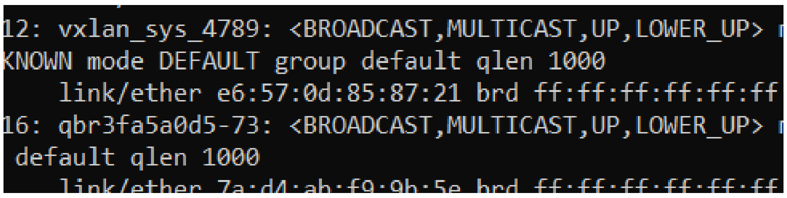


We are interested in the br-tun bridge and if we check the specifications of it’s with ovs-vsctl show



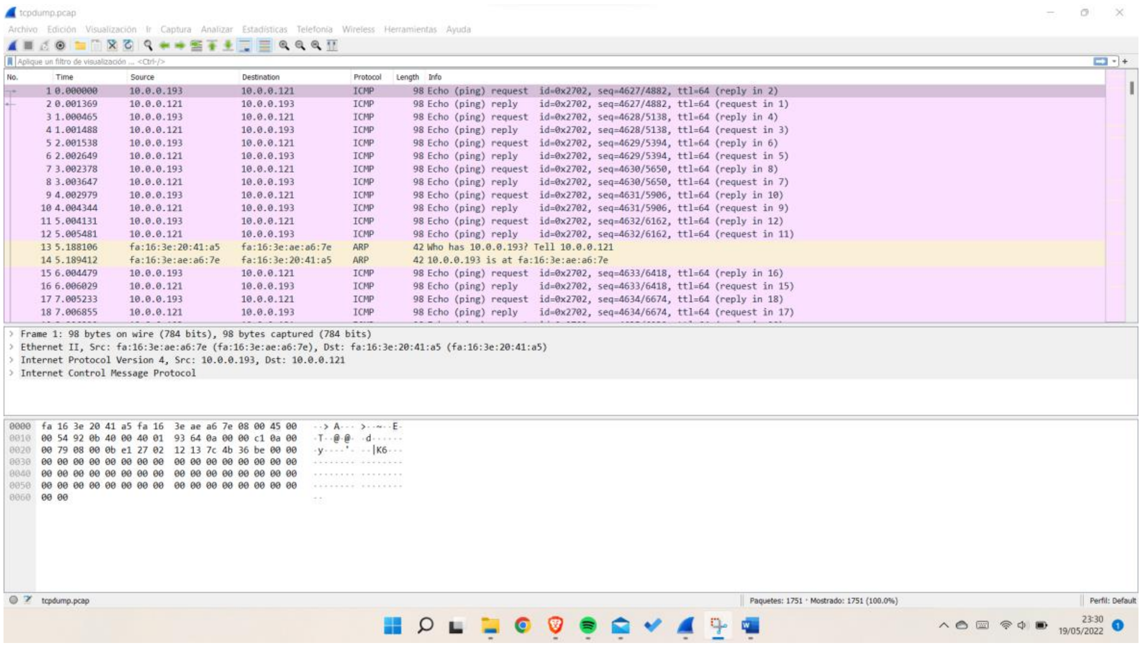
```
(openvswitch-vswitchd)[root@tutorial3-cmp /]# ovs-vsctl show
46b33c14-07e2-49e8-97bb-e651d7194c2e
    Manager "ptcp:6640:127.0.0.1"
        is_connected: true
    Bridge br-int
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        datapath_type: system
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
        Port qvo9dbeb9d9-e7
            tag: 4095
            Interface qvo9dbeb9d9-e7
        Port qvo3fa5a0d5-73
            tag: 1
            Interface qvo3fa5a0d5-73
        Port br-int
            Interface br-int
                type: internal
    Bridge br-tun
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        datapath_type: system
        Port vxlan-0a2606d0
            Interface vxlan-0a2606d0
                type: vxlan
                options: {df_default="true", egress_pkt_mark="0", in_key=flow, local_ip="10.38.6.239", o
ut_key=flow, remote_ip="10.38.6.208"}
        Port br-tun
            Interface br-tun
                type: internal
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
```

Our port is the vxlan… and if we inspect our interfaces with "ip link"



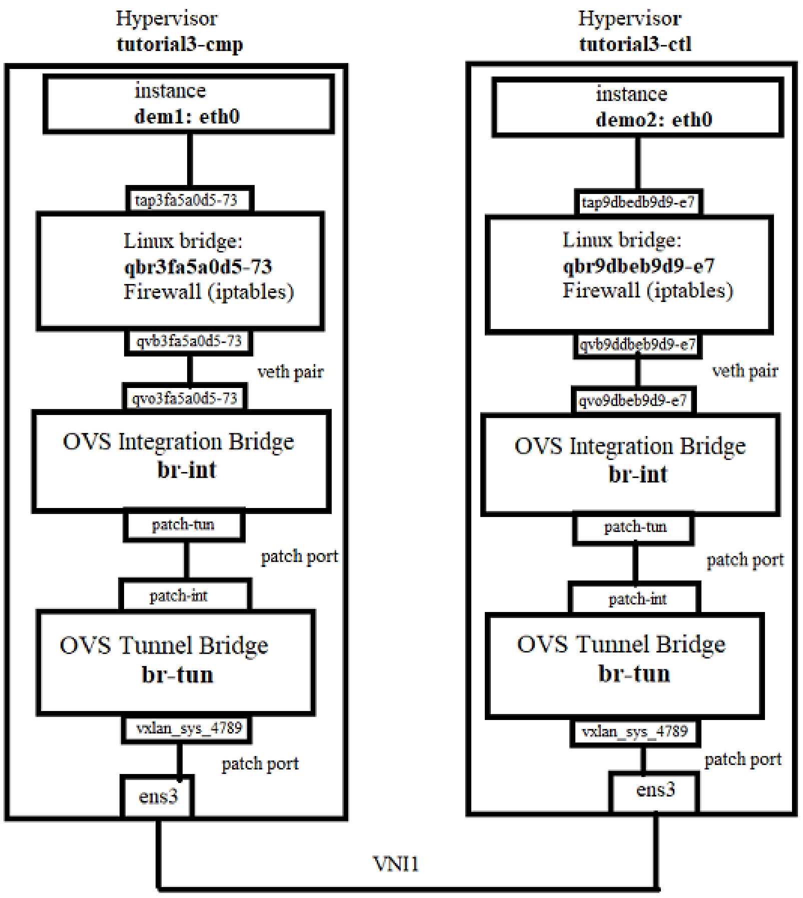
```
12: vxlan_sys_4789: <BROADCAST,MULTICAST,UP,LOWER_UP>
KNOWN mode DEFAULT group default qlen 1000
    link/ether e6:57:0d:85:87:21 brd ff:ff:ff:ff:ff:ff
16: qbr3fa5a0d5-73: <BROADCAST,MULTICAST,UP,LOWER_UP>
 default qlen 1000
    link/ether 7a:d4:ab:f9:9b:5e brd ff:ff:ff:ff:ff:ff
```

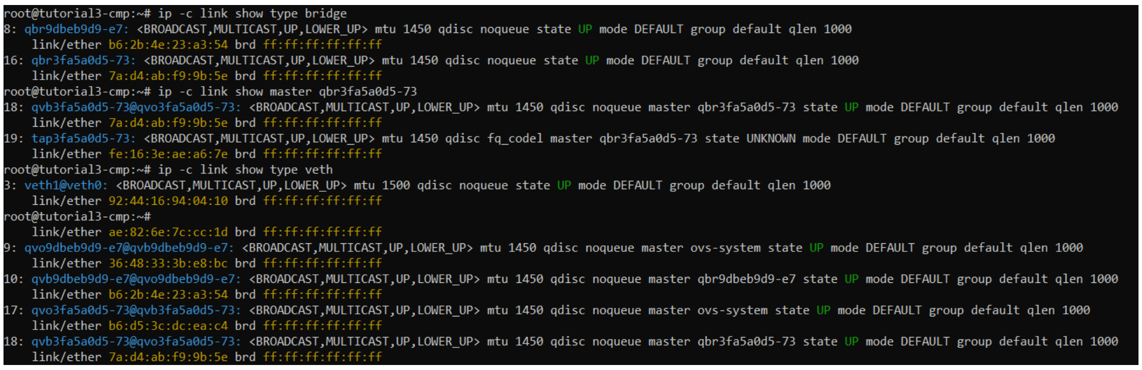
The name of the interface we want is vxlan_sys_4789, then lets execute the command "tcpdump -i vxlan_sys_4789 -w tcpdump.pcap -s 65535" to create a file which later we will send to our computer via sftp and read with wireshark.


tcpdump.pcap
Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda
Aplique un filtro de visualización ... <Ctrl-/>

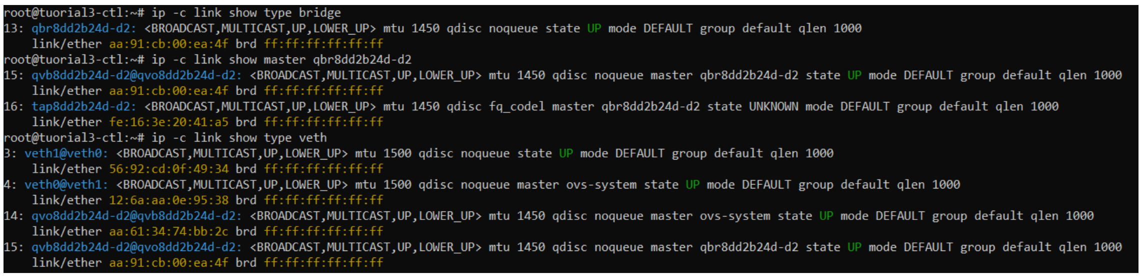
```
No.   Time      Source             Destination        Protocol Length Info
    1 0.000000  10.0.0.193         10.0.0.121         ICMP     98 Echo (ping) request  id=0x2702, seq=4627/4882, ttl=64 (reply in 2)
    2 0.001369  10.0.0.121         10.0.0.193         ICMP     98 Echo (ping) reply    id=0x2702, seq=4627/4882, ttl=64 (request in 1)
    3 1.000465  10.0.0.193         10.0.0.121         ICMP     98 Echo (ping) request  id=0x2702, seq=4628/5138, ttl=64 (reply in 4)
    4 1.001488  10.0.0.121         10.0.0.193         ICMP     98 Echo (ping) reply    id=0x2702, seq=4628/5138, ttl=64 (request in 3)
    5 2.001538  10.0.0.193         10.0.0.121         ICMP     98 Echo (ping) request  id=0x2702, seq=4629/5394, ttl=64 (reply in 6)
    6 2.002649  10.0.0.121         10.0.0.193         ICMP     98 Echo (ping) reply    id=0x2702, seq=4629/5394, ttl=64 (request in 5)
    7 3.002378  10.0.0.193         10.0.0.121         ICMP     98 Echo (ping) request  id=0x2702, seq=4630/5650, ttl=64 (reply in 8)
    8 3.003647  10.0.0.121         10.0.0.193         ICMP     98 Echo (ping) reply    id=0x2702, seq=4630/5650, ttl=64 (request in 7)
    9 4.002979  10.0.0.193         10.0.0.121         ICMP     98 Echo (ping) request  id=0x2702, seq=4631/5906, ttl=64 (reply in 10)
   10 4.004344  10.0.0.121         10.0.0.193         ICMP     98 Echo (ping) reply    id=0x2702, seq=4631/5906, ttl=64 (request in 9)
   11 5.004131  10.0.0.193         10.0.0.121         ICMP     98 Echo (ping) request  id=0x2702, seq=4632/6162, ttl=64 (reply in 12)
   12 5.005481  10.0.0.121         10.0.0.193         ICMP     98 Echo (ping) reply    id=0x2702, seq=4632/6162, ttl=64 (request in 11)
   13 5.188106  fa:16:3e:20:41:a5  fa:16:3e:ae:a6:7e  ARP      42 Who has 10.0.0.193? Tell 10.0.0.121
   14 5.189412  fa:16:3e:ae:a6:7e  fa:16:3e:20:41:a5  ARP      42 10.0.0.193 is at fa:16:3e:ae:a6:7e
   15 6.004479  10.0.0.193         10.0.0.121         ICMP     98 Echo (ping) request  id=0x2702, seq=4633/6418, ttl=64 (reply in 16)
   16 6.006029  10.0.0.121         10.0.0.193         ICMP     98 Echo (ping) reply    id=0x2702, seq=4633/6418, ttl=64 (request in 15)
   17 7.005233  10.0.0.193         10.0.0.121         ICMP     98 Echo (ping) request  id=0x2702, seq=4634/6674, ttl=64 (reply in 18)
   18 7.006855  10.0.0.121         10.0.0.193         ICMP     98 Echo (ping) reply    id=0x2702, seq=4634/6674, ttl=64 (request in 17)

> Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
> Ethernet II, Src: fa:16:3e:ae:a6:7e (fa:16:3e:ae:a6:7e), Dst: fa:16:3e:20:41:a5 (fa:16:3e:20:41:a5)
> Internet Protocol Version 4, Src: 10.0.0.193, Dst: 10.0.0.121
> Internet Control Message Protocol

0000  fa 16 3e 20 41 a5 fa 16  3e ae a6 7e 08 00 45 00   ··> A··· >··~··E·
0010  00 54 92 0b 40 00 40 01  93 64 0a 00 00 c1 0a 00   ·T··@·@· ·d······
0020  00 79 08 00 0b e1 27 02  12 13 7c 4b 36 be 00 00   ·y····'· ··|K6···
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00                                              ··
```

Draw a diagram of the network elements through which the east-west traffic passes (between two VMs on different hypervisors), with specific interface names in your OpenStack installation (supported by random screenshots).

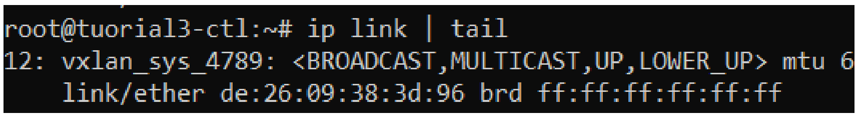




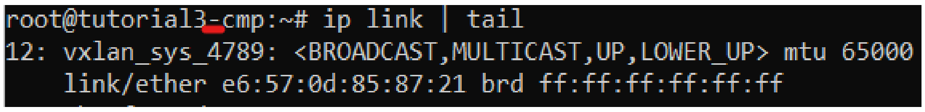
```
root@tutorial3-cmp:~# ip -c link show type bridge
8: qbr9dbeb9d9-e7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP mode DEFAULT group default qlen 1000
    link/ether b6:2b:4e:23:a3:54 brd ff:ff:ff:ff:ff:ff
16: qbr3fa5a0d5-73: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP mode DEFAULT group default qlen 1000
    link/ether 7a:d4:ab:f9:9b:5e brd ff:ff:ff:ff:ff:ff
root@tutorial3-cmp:~# ip -c link show master qbr3fa5a0d5-73
18: qvb3fa5a0d5-73@qvo3fa5a0d5-73: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master qbr3fa5a0d5-73 state UP mode DEFAULT group default qlen 1000
    link/ether 7a:d4:ab:f9:9b:5e brd ff:ff:ff:ff:ff:ff
19: tap3fa5a0d5-73: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc fq_codel master qbr3fa5a0d5-73 state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether fe:16:3e:ae:a6:7e brd ff:ff:ff:ff:ff:ff
root@tutorial3-cmp:~# ip -c link show type veth
3: veth1@veth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen 1000
    link/ether 92:44:16:94:04:10 brd ff:ff:ff:ff:ff:ff
root@tutorial3-cmp:~#
    link/ether ae:82:6e:7c:cc:1d brd ff:ff:ff:ff:ff:ff
9: qvo9dbeb9d9-e7@qvb9dbeb9d9-e7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 36:48:33:3b:e8:bc brd ff:ff:ff:ff:ff:ff
10: qvb9dbeb9d9-e7@qvo9dbeb9d9-e7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master qbr9dbeb9d9-e7 state UP mode DEFAULT group default qlen 1000
    link/ether b6:2b:4e:23:a3:54 brd ff:ff:ff:ff:ff:ff
17: qvo3fa5a0d5-73@qvb3fa5a0d5-73: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether b6:d5:3c:dc:ea:c4 brd ff:ff:ff:ff:ff:ff
18: qvb3fa5a0d5-73@qvo3fa5a0d5-73: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master qbr3fa5a0d5-73 state UP mode DEFAULT group default qlen 1000
    link/ether 7a:d4:ab:f9:9b:5e brd ff:ff:ff:ff:ff:ff
```



```
root@tuorial3-ctl:~# ip -c link show type bridge
13: qbr8dd2b24d-d2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP mode DEFAULT group default qlen 1000
    link/ether aa:91:cb:00:ea:4f brd ff:ff:ff:ff:ff:ff
root@tuorial3-ctl:~# ip -c link show master qbr8dd2b24d-d2
15: qvb8dd2b24d-d2@qvo8dd2b24d-d2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master qbr8dd2b24d-d2 state UP mode DEFAULT group default qlen 1000
    link/ether aa:91:cb:00:ea:4f brd ff:ff:ff:ff:ff:ff
16: tap8dd2b24d-d2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc fq_codel master qbr8dd2b24d-d2 state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether fe:16:3e:20:41:a5 brd ff:ff:ff:ff:ff:ff
root@tuorial3-ctl:~# ip -c link show type veth
3: veth1@veth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen 1000
    link/ether 56:92:cd:0f:49:34 brd ff:ff:ff:ff:ff:ff
4: veth0@veth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 12:6a:aa:0e:95:38 brd ff:ff:ff:ff:ff:ff
14: qvo8dd2b24d-d2@qvb8dd2b24d-d2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether aa:61:34:74:bb:2c brd ff:ff:ff:ff:ff:ff
15: qvb8dd2b24d-d2@qvo8dd2b24d-d2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master qbr8dd2b24d-d2 state UP mode DEFAULT group default qlen 1000
    link/ether aa:91:cb:00:ea:4f brd ff:ff:ff:ff:ff:ff
```



```
root@tuorial3-ctl:~# ip link | tail
12: vxlan_sys_4789: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 6
    link/ether de:26:09:38:3d:96 brd ff:ff:ff:ff:ff:ff
```



```
root@tutorial3-cmp:~# ip link | tail
12: vxlan_sys_4789: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65000
    link/ether e6:57:0d:85:87:21 brd ff:ff:ff:ff:ff:ff
```