VXLAN lab based on OpenVSwitch and LXD containers

Contents

1 Preamble

The very first idea when I started writing this lab was to illustrate the **Virtual Extensible LAN (VXLAN)** technology. Now that **OpenVSwitch** configuration is smoothely integrated in the Debian networking configuration files, this should have resulted in somewhat easy-to-read gist.

It was while advancing in the writing of the gist that things started to go wrong.

Although it's nice to have a VLAN (broadcast domain) sitting on top of an interconnection of different IP networks, what if the containers within this VLAN need failover between the two tenants where they stand ? That's when trouble comes. We need automatic IPv4 and IPv6 addressing for the containers with resiliency !

This is the way a short written gist that was supposed to stay short became a way too long document.

- Traffic flows are marked coming back and forth of each tenant and are sent to a dedicated routing table
- The gateway IPv4 and IPv6 addresses in the shared VLAN are handled by **keepalived**
- IPv4 dynamic addressing is handled by the ISC DHCP server with its failover feature
- IPv6 SLAAC (StateLess Automatic Address Configuration) is provided by **radvd**

Sorry for the inconvenience, but the result is worth it even if the reading is long and painful.

2 Lab architecture

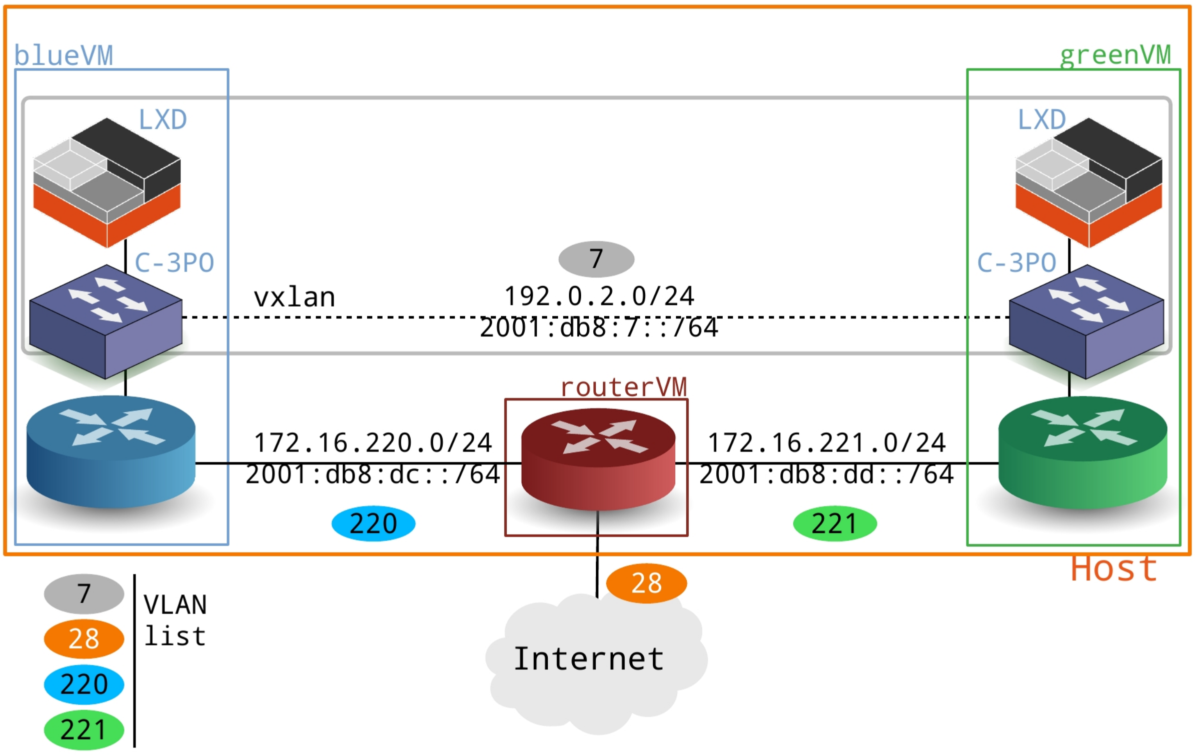

Figure 1: gist-vxlan

We use three virtual machines :

- redRouter is there to route the traffic between the two tenants. It also has to route the traffic coming from the containers to the Internet.
- blueTenant is the left side tenant which hosts some LXD containers on the VXLAN shared VLAN
- greenTenant is the right side tenant which hosts some other LXD containers on the same VXLAN shared VLAN

IMPORTANT ! All commands are run as normal user as this user account must belong to the mandatory system groups.

- **kvm** group for *virtualisation*
- **lxd** group for *containerisation*

Otherwise, atlmost all the command examples below should be preceded by sudo.

3 Prepare Host system

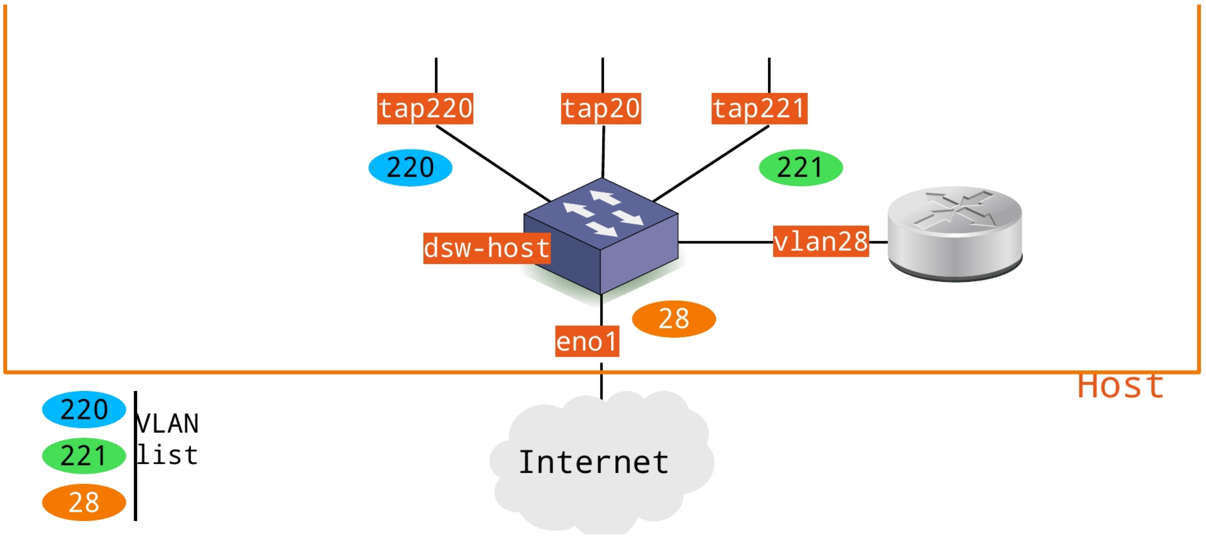

Figure 2: gist-host-vxlan

In order to run the three virtual machines we have a few preparation steps.

3.1 Create virtual machines files from a pre-existing master image

```
ionice -c3 cp $HOME/masters/debian-testing-amd64.qcow2 blueTenant.qcow2
ionice -c3 cp $HOME/masters/debian-testing-amd64.qcow2 greenTenant.qcow2
ionice -c3 cp $HOME/masters/debian-testing-amd64.qcow2 redRouter.qcow2
```

3.2 Download the ovs-startup.sh shell script file to run the KVM virtual machines

```
wget https://raw.githubusercontent.com/platu/inetdoc/master/guides/vm/files/ovs-startup.sh
chmod +x ovs-startup.sh
```

This shell script starts a KVM hypervisor based virtual machine. It runs with three parameters :

- virtual machine image file
- initial amount of guest RAM memory
- network tap interface number

Notice that the MAC address of the virtual machine is built from the **ba:ad:ca:fe** prefix followed by the tap interface number converted to hexadecimal on the two bytes on the right. If the MAC address prefix is already used in your infrastructure, the script has to be edited to change it.

3.3 Check that the mandatory packages for networking and virtualization are there and installed

- openvswitch-switch
- qemu-system-x86

```
aptitude versions openvswitch-switch
i   3.1.0~git20230108.006e1c6-1+b1    testing    500

aptitude versions qemu-system-x86
i   1:7.2+dfsg-2                      testing    500
```

If your host system has a GUI, you may want to access to virtual machines screen through the SPICE protocol. Therefore, you have to install the **spice-client-gtk** package and then use the **spicy** tool.

3.4 Setup the three tap interfaces in order to plug the virtual machines on Host distribution switch dsw-host

As we use the Debian GNU/Linux network interfaces configuration is under `ifupdown` control.

Here is an excerpt from Host `/etc/network/interfaces` file:

```
# Host main distribution switch
# The Host physical port is named eno1. Change it to fit your context.
auto dsw-host
iface dsw-host inet manual
    ovs_type OVSBridge
    ovs_ports eno1 vlan28 tap20 tap220 tap221

# Host physical port
# Configuration mode may be changed from manual to static or dhcp depending on context
allow-dsw-host eno1
iface eno1 inet manual
    ovs_bridge dsw-host
    ovs_type OVSPort
    up ip link set dev $IFACE up
    down ip link set dev $IFACE down

# Host SVI interface
# This network layer Switch Virtual Interface is in access mode and belongs to
# VLAN 28.
allow-dsw-host vlan28
iface vlan28 inet static
    ovs_bridge dsw-host
    ovs_type OVSIntPort
    ovs_options tag=28 vlan_mode=access
    address 198.18.28.20/24

iface vlan28 inet6 static
    ovs_bridge dsw-host
    ovs_type OVSIntPort
    ovs_options tag=28 vlan_mode=access
    address 2001:678:3fc:1c::14/64

# redRouter -> trunk mode
allow-dsw-host tap20
iface tap20 inet manual
    ovs_bridge dsw-host
    ovs_type OVSPort
    ovs_options vlan_mode=trunk
    pre-up ip tuntap add mode tap dev $IFACE group kvm multi_queue
    up ip link set dev $IFACE up
    down ip link set dev $IFACE down
    post-down ip tuntap del mode tap dev $IFACE multi_queue

# blueTenant left side tenant -> access mode on VLAN 220
allow-dsw-host tap220
iface tap220 inet manual
    ovs_bridge dsw-host
    ovs_type OVSPort
    ovs_options tag=220 vlan_mode=access
    pre-up ip tuntap add mode tap dev $IFACE group kvm multi_queue
    up ip link set dev $IFACE up
    down ip link set dev $IFACE down
    post-down ip tuntap del mode tap dev $IFACE multi_queue

# greenTenant right side tenant -> access mode on VLAN 221
allow-dsw-host tap221
iface tap221 inet manual
    ovs_bridge dsw-host
    ovs_type OVSPort
    ovs_options tag=221 vlan_mode=access
    pre-up ip tuntap add mode tap dev $IFACE group kvm multi_queue
    up ip link set dev $IFACE up
    down ip link set dev $IFACE down
    post-down ip tuntap del mode tap dev $IFACE multi_queue
```

Don’t forget to run **ifdown** and/or **ifup** on the above interfaces before to go further !

For instance:

```
sudo ifdown dsw-host && sudo ifup dsw-host
```

Here is a way to check interfaces status :

```
ip a ls | egrep 'state (UP|UNKNOWN)'
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq master ovs-system state UP group
     default qlen 1000
7: dsw-host: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UNKNOWN group
    default qlen 1000
10: vlan28: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UNKNOWN group
    default qlen 1000
36: tap20: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq master ovs-system state UP
    group default qlen 1000
37: tap220: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq master ovs-system state UP
    group default qlen 1000
38: tap221: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq master ovs-system state UP
    group default qlen 1000
```

3.5 Turn on IPv4 and IPv6 routing on Host system which is also a router

Edit /etc/sysctl.conf in order to get the following result

```
egrep -v '(^#|^$)' /etc/sysctl.conf
net.ipv4.conf.default.rp_filter=2
net.ipv4.conf.all.rp_filter=2
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians = 1
```

Don’t forget to turn routing on after editing the /etc/sysctl.conf file

```
sudo sysctl --system
```

3.6 Finally, launch the three virtual machines

Remember that the normal user account has to be part of the kvm system group.

```
id | grep -o kvm
 kvm
```

Now we are ready to launch the virtual machines.

```
#!/bin/bash

$HOME/masters/scripts/ovs-startup.sh redRouter.qcow2 1024 20
$HOME/masters/scripts/ovs-startup.sh blueTenant.qcow2 1024 220
$HOME/masters/scripts/ovs-startup.sh greenTenant.qcow2 1024 221

exit 0
```

Take note of the MAC addresses below. They will be useful in the following steps of the lab.

```
bash startup.sh
~> Virtual machine filename   : redRouter.qcow2
~> RAM size                   : 1024MB
~> SPICE VDI port number      : 5920
~> telnet console port number : 2320
~> MAC address                : b8:ad:ca:fe:00:14
~> Switch port interface      : tap20, access mode
~> IPv6 LL address            : fe80::baad:caff:fefe:14%vlan28
~> Virtual machine filename   : blueTenant.qcow2
~> RAM size                   : 1024MB
~> SPICE VDI port number      : 6120
~> telnet console port number : 2520
~> MAC address                : b8:ad:ca:fe:00:dc
~> Switch port interface      : tap220, trunk mode
~> IPv6 LL address            : fe80::baad:caff:fefe:dc%dsw-host
~> Virtual machine filename   : greenTenant.qcow2
~> RAM size                   : 1024MB
~> SPICE VDI port number      : 6121
~> telnet console port number : 2521
~> MAC address                : b8:ad:ca:fe:00:dd
~> Switch port interface      : tap221, trunk mode
~> IPv6 LL address            : fe80::baad:caff:fefe:dd%dsw-host
```

4 Configure router virtual machine : redRouter

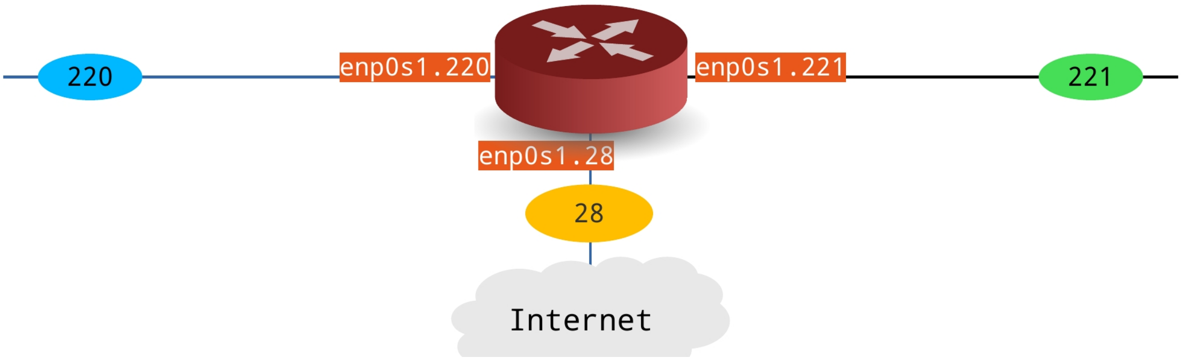

Figure 3: gist-routerVm

4.1 Turn on IPv4 and IPv6 routing

As it is the main purpose of this virtual machine, it would be stupid to forget it !

Edit /etc/sysctl.conf in order to get the following result

```
egrep -v '(^#|^$)' /etc/sysctl.conf
net.ipv4.conf.default.rp_filter=2
net.ipv4.conf.all.rp_filter=2
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians = 1
```

Turn routing on after editing the /etc/sysctl.conf file

```
sudo sysctl --system
```

4.2 Add two new IPv4 routing tables

Traffic coming back and forth from the containers to the Internet may follow two different paths. One through the link between the **Blue** tenant and another one through the **Green** tenant. With IPv4, this could lead to an asymetric routing which is not compatible with connection tracking introduced by firewall rules.

This is why we choose to setup two separate routing tables : one for each path. We have to add two new lines at the end of the /etc/iproute2/rt_tables file:

```
#
# reserved values
#
255     local
254     main
253     default
0       unspec
#
# local
#
```

```
#1      inr.ruhep
220     blue
221     green
```

- The new routing tables entries will be set in the network interfaces configuration file
- The `rule` to send traffic to a specific routing table is also set in the network interfaces configuration file
- The traffic will be **marked** by rules in the `mangle` table of netfilter/iptables configuration file

4.3 Configure three network interfaces : one per VLAN

The router's main interface named `enp0s1` is a trunk port connected to `tap20` on Host system. We have to set one subinterface per VLAN.

We can check the `tap20` switch port configuration from the **Host** system. The main parameter Here is `vlan_mode` at the end of the screen copy below.

```
sudo ovs-vsctl list port tap20
_uuid               : d1cbfd72-d44f-429f-aee1-fe6e54c5430b
bond_active_slave   : []
bond_downdelay      : 0
bond_fake_iface     : false
bond_mode           : []
bond_updelay        : 0
cvlans              : []
external_ids        : {}
fake_bridge         : false
interfaces          : [44e1f871-4dae-4110-9d69-4a83e0e5ac8d]
lacp                : []
mac                 : []
name                : tap20
other_config        : {}
protected           : false
qos                 : []
rstp_statistics     : {}
rstp_status         : {}
statistics          : {}
status              : {}
tag                 : 28
trunks              : []
vlan_mode           : trunk
```

Here is an excerpt of redRouter networking configuration file : `/etc/network/interfaces`

```
# The main network interface
# Interface name enp0s1 may have to be changed !
auto enp0s1
iface enp0s1 inet manual
    up ip link set dev $IFACE up
    down ip link set dev $IFACE down

# The Host system link
auto enp0s1.28
iface enp0s1.28 inet static
    address 198.18.28.200/24
    gateway 198.18.28.1
    ## Use quad9.net as DNS resolver
    dns-nameservers 9.9.9.9

iface enp0s1.28 inet6 static
    address 2001:678:3fc:1c::c8/64
    gateway fe80:1c::1
    pre-up echo 0 > /proc/sys/net/ipv6/conf/$IFACE/accept_dad

# The Blue tenant link
```

```
auto enp0s1.220
iface enp0s1.220 inet static
    address 172.16.220.1/24
    ## IPv4 Blue routing table
    post-up ip route add 172.16.220.0/24 dev $IFACE table blue
    post-up ip route add 192.0.2.0/24 via 172.16.220.2 dev $IFACE table blue
    post-up ip route add 198.18.28.0/24 dev enp0s1.28 table blue
    post-up ip route add default via 198.18.28.1 dev enp0s1.28 table blue
    ## Traffic marked with 220 goes to the IPv4 Blue routing table
    post-up ip rule add fwmark 220 table blue

iface enp0s1.220 inet6 static
    address 2001:db8:dc::1/64
    up ip -6 addr add fe80:dc::1/64 dev $IFACE
    ## IPv6 Blue routing table
    post-up ip -6 route add 2001:db8:dc::/64 dev $IFACE table blue
    post-up ip -6 route add 2001:db8:7::/64 via 2001:db8:dc::2 dev $IFACE table blue
    post-up ip -6 route add 2001:678:3fc:1c::/64 dev enp0s1.28 table blue
    post-up ip -6 route add default via fe80:1c::1 dev enp0s1.28 table blue
    ## Traffic marked with 220 goes to the IPv6 Blue routing table
    post-up ip -6 rule add fwmark 220 table blue
    pre-up echo 0 > /proc/sys/net/ipv6/conf/$IFACE/accept_dad

# The Green tenant link
auto enp0s1.221
iface enp0s1.221 inet static
    address 172.16.221.1/24
    ## IPv4 Green routing table
    post-up ip route add 172.16.221.0/24 dev $IFACE table green
    post-up ip route add 192.0.2.0/24 via 172.16.221.2 dev $IFACE table green
    post-up ip route add 198.18.28.0/24 dev enp0s1.28 table green
    post-up ip route add default via 198.18.28.1 dev enp0s1.28 table green
    ## Traffic marked with 221 goes to the IPv4 Green routing table
    post-up ip rule add fwmark 221 table green

iface enp0s1.221 inet6 static
    address 2001:db8:dd::1/64
    up ip -6 addr add fe80:dd::1/64 dev $IFACE
    ## IPv6 Green routing table
    post-up ip -6 route add 2001:db8:dd::/64 dev $IFACE table green
    post-up ip -6 route add 2001:db8:7::/64 via 2001:db8:dd::2 dev $IFACE table green
    post-up ip -6 route add 2001:678:3fc:1c::/64 dev enp0s1.28 table green
    post-up ip -6 route add default via fe80:1c::1 dev enp0s1.28 table green
    ## Traffic marked with 221 goes to the IPv6 Green routing table
    post-up ip -6 rule add fwmark 221 table green
    pre-up echo 0 > /proc/sys/net/ipv6/conf/$IFACE/accept_dad
```

In order to check the results of the network interfaces configurations use commands like these :

- List rules for marked traffic :

```
ip rule ls
0:    from all lookup local
32764:    from all fwmark 0xdd lookup green <-- firewall mark 221 leads to green IPv4 and
     IPv6 routing tables
32765:    from all fwmark 0xdc lookup blue <-- firewall mark 220 leads to blue IPv4 and
    IPv6 routing tables
32766:    from all lookup main
32767:    from all lookup default
```

- List IPv4 Green tenant routing table entries

```
ip route ls table green
default via 198.18.28.1 dev enp0s1.28
172.16.221.0/24 dev enp0s1.221 scope link
192.0.2.0/24 via 172.16.221.2 dev enp0s1.221
198.18.28.0/23 dev enp0s1.28 scope link
```

- List IPv6 Blue tenant routing table entries

```
ip -6 route ls table blue
2001:678:3fc:1c::/64 dev enp0s1.28 metric 1024 pref medium
2001:db8:7::/64 via 2001:db8:dc::2 dev enp0s1.220 metric 1024 pref medium
2001:db8:dc::/64 dev enp0s1.220 metric 1024 pref medium
default via fe80:1c::1 dev enp0s1.28 metric 1024 pref medium
```

4.4 Set the "ugly" NAT rules on Host link and the mangle rules on links to the tenants

The link between the **redRouter** and the **Host system** is the lab boarder to the real world. This is where connection tracking stands.

Check that the `iptables-persistent` package is installed on redRouter, so we have to save the rules for IPv4 and IPv6 in the `/etc/iptbales` directory.

- First, the `nat` table holds the source address translations for both IPv4 and IPv6.
- Second, the `mangle` table is used to set marks to the IPv4 and IPv6 traffic coming back and forth from the Blue or Green tenants

In the `/etc/iptables/rules.v4` file :

```
## ~~~~~~~~~~~~~~~~ NAT table ~~~~~~
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -o enp0s1.28 -p tcp --syn -j TCPMSS --clamp-mss-to-pmtu
-A POSTROUTING -o enp0s1.28 -j MASQUERADE
COMMIT
## ~~~~~~~~~~~~~~~~ MANGLE table ~~~~~~
*mangle
:PREROUTING ACCEPT [0:0]
-A PREROUTING -j CONNMARK --restore-mark
-A PREROUTING -i enp0s1.220 -s 192.0.2.0/24 -j MARK --set-mark 220
-A PREROUTING -i enp0s1.221 -s 192.0.2.0/24 -j MARK --set-mark 221
-A PREROUTING -j CONNMARK --save-mark
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
COMMIT
```

In the `/etc/iptables/rules.v6` file :

```
## ~~~~~~~~~~~~~~~~ NAT table ~~~~~~
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -o enp0s1.28 -j MASQUERADE
COMMIT
## ~~~~~~~~~~~~~~~~ MANGLE table ~~~~~~
*mangle
:PREROUTING ACCEPT [0:0]
-A PREROUTING -j CONNMARK --restore-mark
-A PREROUTING -i enp0s1.220 -s 2001:db8:7::/64 -j MARK --set-mark 220
-A PREROUTING -i enp0s1.221 -s 2001:db8:7::/64 -j MARK --set-mark 221
-A PREROUTING -j CONNMARK --save-mark
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
```

```
COMMIT
```

One way to apply the rules written in the two files is :

```
sudo sh -c "iptables-restore </etc/iptables/rules.v4"
sudo sh -c "ip6tables-restore </etc/iptables/rules.v6"
```

In order to check the results of these firewalling rules, use the following commands:

- List IPv4 NAT rules

```
sudo iptables -vnL -t nat
Chain PREROUTING (policy ACCEPT 27 packets, 5876 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 15 packets, 2087 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 43 packets, 5589 bytes)
 pkts bytes target      prot opt in     out          source               destination
    0     0 TCPMSS      tcp  --  *      enp0s1.28  0.0.0.0/0            0.0.0.0/0  tcp flags
        :0x17/0x02 TCPMSS clamp to PMTU
   16   947 MASQUERADE  all  --  *      enp0s1.28  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 47 packets, 2747 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

- List IPv6 NAT rules

```
sudo ip6tables -vnL -t nat
Chain PREROUTING (policy ACCEPT 4 packets, 379 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 1 packets, 138 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 11 packets, 968 bytes)
 pkts bytes target      prot opt in     out          source               destination
   10  1005 MASQUERADE  all      *      enp0s1.28  ::/0                 ::/0

Chain OUTPUT (policy ACCEPT 18 packets, 1732 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

- List IPv4 mangle rules

```
sudo iptables -vnL -t mangle
Chain PREROUTING (policy ACCEPT 240K packets, 31M bytes)
 pkts bytes target     prot opt in          out     source               destination
 240K   31M CONNMARK   all  --  *          *       0.0.0.0/0            0.0.0.0/0
     CONNMARK restore
   18  1290 MARK       all  --  enp0s1.220 *       192.0.2.0/24         0.0.0.0/0
       MARK set 0xdc
   36  2580 MARK       all  --  enp0s1.221 *       192.0.2.0/24         0.0.0.0/0
       MARK set 0xdd
 240K   31M CONNMARK   all  --  *          *       0.0.0.0/0            0.0.0.0/0
     CONNMARK save

Chain INPUT (policy ACCEPT 25319 packets, 6506K bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 214K packets, 24M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 5495 packets, 1014K bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 220K packets, 25M bytes)
```

```
 pkts bytes target     prot opt in     out     source               destination
```

- List IPv6 mangle rules

```
sudo ip6tables -vnL -t mangle
Chain PREROUTING (policy ACCEPT 28476 packets, 45M bytes)
 pkts bytes target     prot opt in         out     source               destination
28476   45M CONNMARK   all      *          *       ::/0                 ::/0
    CONNMARK restore
  272 23796 MARK       all      enp0s1.220 *       2001:db8:7::/64      ::/0
      MARK set 0xdc
 1586  119K MARK       all      enp0s1.221 *       2001:db8:7::/64      ::/0
     MARK set 0xdd
28476   45M CONNMARK   all      *          *       ::/0                 ::/0
    CONNMARK save

Chain INPUT (policy ACCEPT 2576 packets, 287K bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 24514 packets, 45M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 1031 packets, 74320 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 25597 packets, 45M bytes)
 pkts bytes target     prot opt in     out     source               destination
```

5 Configure Blue tenant virtual machine : blueTenant

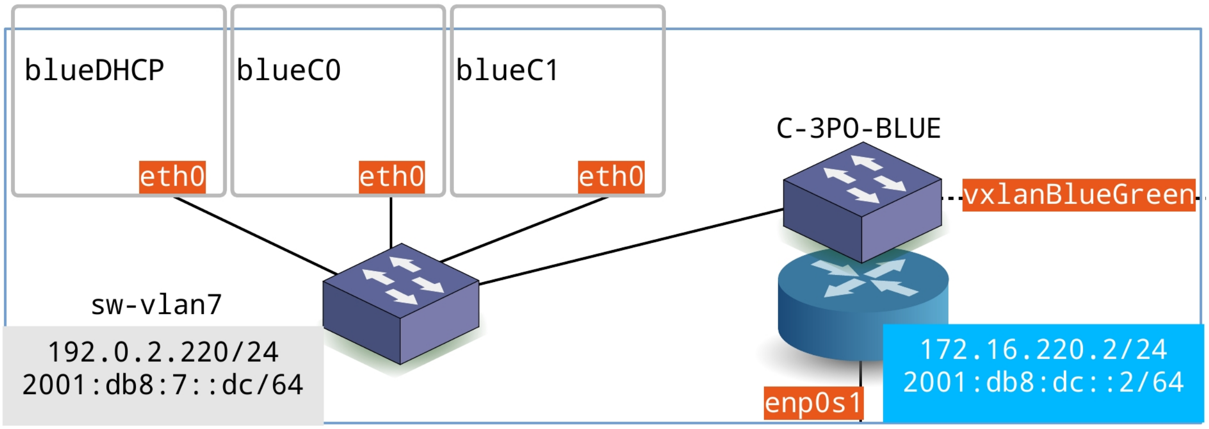

Figure 4: gist-blueVm

5.1 Configure network interfaces and switches for LXD

Here is an excerpt of blueTenant networking configuration file : `/etc/network/interfaces`

```
# The primary network interface
auto enp0s1
iface enp0s1 inet static
    address 172.16.220.2/24
    gateway 172.16.220.1
    dns-nameservers 9.9.9.9

iface enp0s1 inet6 static
    address 2001:db8:dc::2/64
    gateway fe80:dc::1
    dns-nameservers 2620:fe::fe

auto C-3PO-BLUE
iface C-3PO-BLUE inet manual
    ovs_type OVSBridge
    ovs_ports sw-vlan7 vxlanBlueGreen
    ovs_mtu 9000
    up ip link set dev $IFACE up
    down ip link set dev $IFACE down

allow-C-3PO-BLUE sw-vlan7
iface sw-vlan7 inet static
    ovs_type OVSBridge
    ovs_bridge C-3PO-BLUE
    ovs_options C-3PO-BLUE 7
    ovs_mtu 9000
    address 192.0.2.220/24

iface sw-vlan7 inet6 static
    ovs_type OVSBridge
    ovs_bridge C-3PO-BLUE
    ovs_options C-3PO-BLUE 7
    ovs_mtu 9000
    address 2001:db8:7::dc/64
    up ip -6 addr add fe80:7::dc/64 dev $IFACE

allow-C-3PO-BLUE vxlanBlueGreen
iface vxlanBlueGreen inet manual
    ovs_bridge C-3PO-BLUE
    ovs_type OVSTunnel
    ovs_tunnel_type vxlan
    ovs_tunnel_options options:remote_ip=172.16.221.2 options:key=WhereIsMyTCAM
```

5.2 Turn on IPv4 and IPv6 routing

This blue virtual machine is, just as the others, a router. We want the traffic coming back and forth from the containers to be routed.

Edit `/etc/sysctl.conf` in order to get the following result.

```
$ egrep -v '(^#|^$)' /etc/sysctl.conf
net.ipv4.conf.default.rp_filter=2
net.ipv4.conf.all.rp_filter=2
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians = 1
```

Turn routing on after editing the above `/etc/sysctl.conf` file.

```
$ sudo sysctl --system
```

5.3 Install lxd

```
$ sudo apt install snapd
$ sudo snap install lxd
$ sudo adduser etu lxd
```

Normal user account is the UNprivileged user and must belong to `lxd` group. Log out and log back in to make it effective.

```
$ id | grep -o lxd
lxd
```

List the installed snaps

```
$ snap list
Name     Version     Rev    Tracking       Publisher   Notes
core     16-2.54.3   12725  latest/stable ✓ canonical  core
core18   20211215    2284   latest/stable ✓ canonical  base
core20   20220304    1376   latest/stable ✓ canonical  base
lxd      4.23        22525  latest/stable ✓ canonical  -
```

5.4 Initial configuration and/or profile

```
$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]: no
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (btrfs, ceph, dir, lvm) [default=btrfs]:
Create a new BTRFS pool? (yes/no) [default=yes]:
Would you like to use an existing block device? (yes/no) [default=no]:
Size in GB of the new loop device (1GB minimum) [default=15GB]:
Would you like to connect to a MAAS server? (yes/no) [default=no]:
```

```
Would you like to create a new local network bridge? (yes/no) [default=yes]: no  <-- CHANGE
    TO NO
Would you like to configure LXD to use an existing bridge or host interface? (yes/no) [
    default=no]: yes  <-- CHANGE TO YES
Name of the existing bridge or host interface: sw-vlan7    <-- HERE WE USE OUR OWN SWITCH
Would you like LXD to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:no
```

We have to change the **nictype:** from **macvlan** to **bridged** and we are done with the default profile.

```
$ lxc profile device set default eth0 nictype bridged
$ lxc profile device get default eth0 nictype
bridged
```

5.5 Launch the three lxd new containers

```
$ lxc launch images:debian/12 blueDHCP
Creating blueDHCP
Starting blueDHCP

$ lxc launch images:debian/12 blueC0
Creating blueC0
Starting blueC0

$ lxc launch images:debian/12 blueC1
Creating blueC1
Starting blueC1
```

5.6 Configure IPv6 SLAAC with radvd for containers

We choose to use **radvd** on **blueTenant** as we want resilient and failover addressing on both **Blue** and **Green** tenants.

This daemon is installed on the virtual machine as the **sw-vlan7** Switched Virtual Interface (SVI) stands on this machine.

```
$ sudo apt install radvd
$ sudo systemctl enable radvd
```

Here is a copy of the `/etc/radvd.conf` file for VLAN 7. Do not forget to restart service after editing this configuration file.

```
interface sw-vlan7
{
    AdvSendAdvert on;

    AdvRASrcAddress {
        fe80:7::de;
    };

    prefix 2001:db8:7::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
    };

    RDNSS 2620:fe::fe
    {
    };
};
```

The most important part of the above file is the router or gateway address advertised : `fe80:7::de`. This address is under control of the **keepalived** daemon.

5.7 Configure IPv4 ISC DHCP server into the blueDHCP container

We choose to use the ISC DHCP server as failover for dynamic IPv4 addressing is available.

We also choose to run this server into a container as it is an application layer service.

. Here is a copy of the `/etc/network/interfaces` of the blueDHCP container. This container uses a static IPv4 address.

```
$ lxc exec blueDHCP -- cat /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.0.2.250/24
    # Gateway address is 192.0.2.220 before keepalived setup
    gateway 192.0.2.222
    dns-nameservers 9.9.9.9
```

. Install ISC DHCP server package.

```
$ lxc exec blueDHCP -- apt install isc-dhcp-server
```

. Edit the `/etc/default/isc-dhcp-server` file to designate the network interface `eth0`.

```
$ lxc exec blueDHCP -- sed -i 's/INTERFACESv4=""/INTERFACESv4="eth0"/g' /etc/default/isc-dhcp
    -server
$ lxc exec blueDHCP -- egrep -v '(^#|^$)' /etc/default/isc-dhcp-server
INTERFACESv4="eth0"
INTERFACESv6=""
```

. Edit the `/etc/dhcp/dhcpd.conf` to setup failover and address range for dynamic IPv4 addressing.

```
$ lxc exec blueDHCP -- grep -v ^# /etc/dhcp/dhcpd.conf | cat -s

default-lease-time 600;
max-lease-time 7200;

ddns-update-style none;

failover peer "failover-partner" {
    primary;
    address 192.0.2.250;
    port 519;
    peer address 192.0.2.251;
    peer port 520;
    max-response-delay 60;
    max-unacked-updates 10;
    mclt 3600;
    split 128;
    load balance max seconds 3;
}

subnet 192.0.2.0 netmask 255.255.255.0 {
    option domain-name-servers 9.9.9.9;
    option routers 192.0.2.222;
    pool {
        failover peer "failover-partner";
        range 192.0.2.10 192.0.2.90;
    }
}
```

As in the case of IPv6 addressing, the most important part of the above file is the router or gateway address advertised : `192.0.2.222`. This address is under control of the **keepalived** daemon.

5.8 Configure keepalived for IPv4 and IPv6 gateway resiliency between Blue and Green tenants

Let's place three new scripts to be used with keepalived configuration in the /usr/local/bin directory.

- Check IPv4 connectivity to the Internet from the keepalived daemon with the /usr/local/bin/keepalived_check_ipv4.sh script.

```
#!/bin/bash

/usr/bin/ping -c 1 -W 1 9.9.9.9 > /dev/null 2>&1
```

- Check IPv6 connectivity to the Internet from the keepalived daemon with the /usr/local/bin/keepalived_check_ipv6.sh script.

```
#!/bin/bash

/usr/bin/ping -c 1 -W 1 2620:fe::fe > /dev/null 2>&1
```

- Keepalived daemon status notifications with the /usr/local/bin/keepalived_notify.sh script.

```
#!/bin/bash

echo "$1 $2 has transitioned to the $3 state with a priority of $4" > /var/run/
    keepalived_status
```

All these scripts should be executable.

```
$ sudo chmod +x /usr/local/bin/keepalived_*
```

After package installation, we edit the main configuration file.

```
$ sudo apt install keepalived
```

Here is a copy of the /etc/keepalived/keepalived.conf configuration file. As usual, do not forget to restart service after editing.

```
! Configuration File for keepalived

global_defs {
        notification_email {
                root@localhost
        }
        notification_email_from etu@localhost
        smtp_server localhost
        smtp_connect_timeout 30

        enable_script_security
        script_user etu

        vrrp_version 3
}

vrrp_script keepalived_check_ipv4 {
        script "/usr/local/bin/keepalived_check_ipv4.sh"
        interval 1
        timeout 5
        rise 3
        fall 3
}

vrrp_script keepalived_check_ipv6 {
        script "/usr/local/bin/keepalived_check_ipv6.sh"
        interval 1
        timeout 5
        rise 3
        fall 3
}

vrrp_sync_group vrrp_group {
        group {
                VXLAN_7_IPv4
                VXLAN_7_IPv6
        }
        track_script {
                keepalived_check_ipv4
        }
        notify "/usr/local/bin/keepalived_notify.sh"
}

vrrp_instance VXLAN_7_IPv4 {
        state MASTER
        interface sw-vlan7
        virtual_router_id 74
        priority 220
        advert_int 1
        virtual_ipaddress {
                192.0.2.222/24
        }
}

vrrp_instance VXLAN_7_IPv6 {
        state MASTER
        interface sw-vlan7
        virtual_router_id 76
        priority 220
        advert_int 1
        virtual_ipaddress {
                fe80:7::de/64
        }
}
```

We must not forget to restart the daemon

```
$ sudo systemctl restart keepalived.service
```

6 Configure Green tenant virtual machine : greenTenant

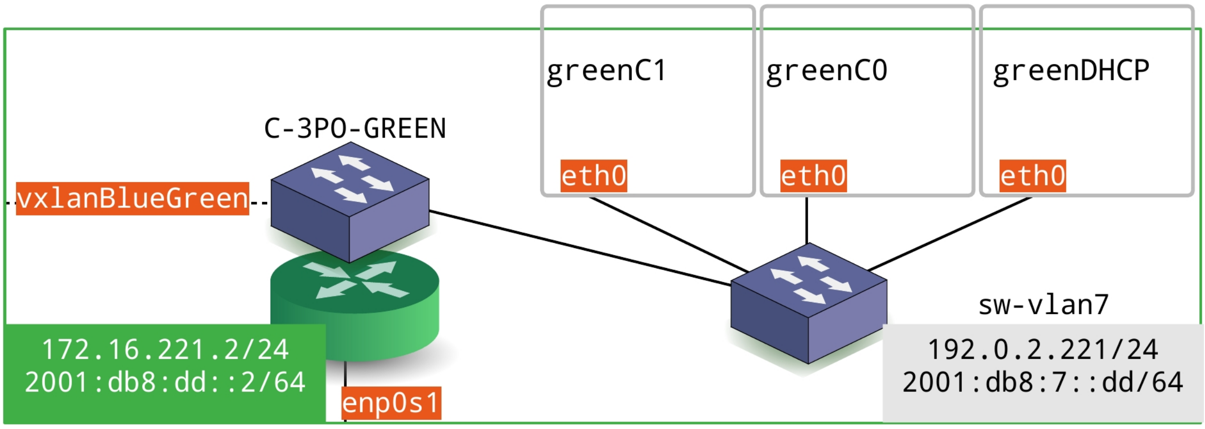

Figure 5: gist-greenVm

The configuration files of the Blue and Green tenant are very similar. IP addresses and names are switched from blue to green.

6.1 Configure network interfaces and switches for LXD

Here is an excerpt of blueTenant networking configuration file : /etc/network/interfaces

```
# The primary network interface
auto enp0s1
iface enp0s1 inet static
    address 172.16.221.2/24
    gateway 172.16.221.1
    dns-nameservers 9.9.9.9

iface enp0s1 inet6 static
    address 2001:db8:dd::2/64
    gateway fe80:dd::1
    dns-nameservers 2620:fe::fe

auto C-3PO-GREEN
iface C-3PO-GREEN inet manual
    ovs_type OVSBridge
    ovs_ports sw-vlan7 vxlanGreenBlue
    ovs_mtu 9000
    up ip link set dev $IFACE up
    down ip link set dev $IFACE down

allow-C-3PO-GREEN sw-vlan7
iface sw-vlan7 inet static
    ovs_type OVSBridge
    ovs_bridge C-3PO-GREEN
    ovs_options C-3PO-GREEN 7
    ovs_mtu 9000
    address 192.0.2.221/24
```

```
iface sw-vlan7 inet6 static
    ovs_type OVSBridge
    ovs_bridge C-3PO-GREEN
    ovs_options C-3PO-GREEN 7
    ovs_mtu 9000
    address 2001:db8:7::dd/64
    up ip -6 addr add fe80:7::dd/64 dev $IFACE

allow-C-3PO-GREEN vxlanGreenBlue
iface vxlanGreenBlue inet manual
    ovs_bridge C-3PO-GREEN
    ovs_type OVSTunnel
    ovs_tunnel_type vxlan
    ovs_tunnel_options options:remote_ip=172.16.220.2 options:key=WhereIsMyTCAM
```

6.2 Turn on IPv4 and IPv6 routing

This green virtual machine is, just as the others, a router. We want the traffic coming back and forth from the containers to be routed.

Edit `/etc/sysctl.conf` in order to get the following result

```
$ egrep -v '(^#|^$)' /etc/sysctl.conf
net.ipv4.conf.default.rp_filter=2
net.ipv4.conf.all.rp_filter=2
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians = 1
```

Turn routing on after editing the above `/etc/sysctl.conf` file

```
$ sudo sysctl --system
```

6.3 Install lxd

```
$ sudo apt install snapd
$ sudo snap install lxd
$ sudo adduser etu lxd
```

Normal user account is the UNprivileged user and must belong to **lxd** group. Log out and log back in to make it effective.

```
$ id | grep -o lxd
lxd
```

List the installed snaps

```
$ snap list
Name    Version   Rev    Tracking       Publisher   Notes
core    16-2.49   10859  latest/stable  canonical  core
core18  20210128  1988   latest/stable  canonical  base
lxd     4.12      19766  latest/stable  canonical  -
```

6.4 Initial configuration and/or profile

```
$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]: no
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (btrfs, ceph, dir, lvm) [default=btrfs]:
```

```
Create a new BTRFS pool? (yes/no) [default=yes]:
Would you like to use an existing block device? (yes/no) [default=no]:
Size in GB of the new loop device (1GB minimum) [default=15GB]:
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]: no <-- CHANGE TO
     NO
Would you like to configure LXD to use an existing bridge or host interface? (yes/no) [
    default=no]: yes  <-- CHANGE TO YES
Name of the existing bridge or host interface: sw-vlan7    <-- HERE WE USE OUR OWN SWITCH
Would you like LXD to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:no
```

We have to change the **nictype:** from **macvlan** to **bridged** and we are done with the default profile.

```
$ lxc profile device set default eth0 nictype bridged
$ lxc profile device get default eth0 nictype
bridged
```

6.5 Launch the three lxd new containers

```
$ lxc launch images:debian/12 greenDHCP
Creating greenDHCP
Starting greenDHCP

$ lxc launch images:debian/12 greenC0
Creating greenC0
Starting greenC0

$ lxc launch images:debian/12 greenC1
Creating greenC1
Starting greenC1
```

6.6 Configure IPv6 SLAAC with radvd for containers

We choose to use `radvd` on **greenTenant** as we want resilient and failover addressing on both **Blue** and **Green** tenants.

This daemon is installed on the virtual machine as the **sw-vlan7** Switched Virtual Interface (SVI) stands on this machine.

```
$ sudo apt install radvd
$ sudo systemctl enable radvd
```

Here is a copy of the `/etc/radvd.conf` file for VLAN 7. Do not forget to restart service after editing this configuration file.

```
interface sw-vlan7
{
    AdvSendAdvert on;

    AdvRASrcAddress {
        fe80:7::de;
    };

    prefix 2001:db8:7::de/64
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
    };

    RDNSS 2620:fe::fe
    {
    };
};
```

The most important part of the above file is the router or gateway address advertised : `fe80:7::de`. This address is under control of the **keepalived** daemon.

6.7 Configure IPv4 ISC DHCP server into the greenDHCP container

We choose to use the ISC DHCP server as failover for dynamic IPv4 addressing is available.

We also choose to run this server into a container as it is an application layer service.

. Here is a copy of the `/etc/network/interfaces` of the greenDHCP container. This container uses a static IPv4 address.

```
$ lxc exec greenDHCP -- cat /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.0.2.251/24
    # Gateway address is 192.0.2.221 before keepalived setup
    gateway 192.0.2.222
    dns-nameservers 9.9.9.9
```

. Install ISC DHCP server package

```
$ lxc exec greenDHCP -- apt install isc-dhcp-server
```

. Edit the `/etc/default/isc-dhcp-server` file to designate the network interface `eth0`

```
$ lxc exec greenDHCP -- sed -i 's/INTERFACESv4=""/INTERFACESv4="eth0"/g' /etc/default/isc-dhcp-server
$ lxc exec greenDHCP -- egrep -v '(^#|^$)' /etc/default/isc-dhcp-server
INTERFACESv4="eth0"
INTERFACESv6=""
```

. Edit the `/etc/dhcp/dhcpd.conf` to setup failover and address range for dynamic IPv4 addressing

```
$ lxc exec greenDHCP -- grep -v ^# /etc/dhcp/dhcpd.conf | cat -s

default-lease-time 600;
max-lease-time 7200;

ddns-update-style none;

failover peer "failover-partner" {
     secondary;
     address 192.0.2.251;
     port 520;
     peer address 192.0.2.250;
     peer port 519;
     max-response-delay 60;
     max-unacked-updates 10;
     load balance max seconds 3;
}

subnet 192.0.2.0 netmask 255.255.255.0 {
        option domain-name-servers 9.9.9.9;
        option routers 192.0.2.222;
        pool {
                failover peer "failover-partner";
                range 192.0.2.10 192.0.2.90;
        }
}
```

As in the case of IPv6 addressing, the most important part of the above file is the router or gateway address advertised : **192.0.2.222**. This address is under control of the **keepalived** daemon.

6.8 Configure keepalived for IPv4 and IPv6 gateway resiliency between Blue and Green tenants

Just as it is done on the Blue tenant, we first have to place three new scripts to be used by keepalived configuration.

- Check IPv4 connectivity to the Internet from the keepalived daemon with the `/usr/local/bin/keepalived_check_ipv4.sh` script.

```
#!/bin/bash

/usr/bin/ping -c 1 -W 1 9.9.9.9 > /dev/null 2>&1
```

- Check IPv6 connectivity to the Internet from the keepalived daemon with the `/usr/local/bin/keepalived_check_ipv6.sh` script.

```
#!/bin/bash

/usr/bin/ping -c 1 -W 1 2620:fe::fe > /dev/null 2>&1
```

- Keepalived daemon status notifications with the `/usr/local/bin/keepalived_notify.sh` script.

```
#!/bin/bash

echo "$1 $2 has transitioned to the $3 state with a priority of $4" > /var/run/
    keepalived_status
```

All these scripts should be executable.

```
$ sudo chmod +x /usr/local/bin/keepalived_*
```

When the above scripts are there, we carry on with package installation and configuration.

```
$ sudo apt install keepalived
```

Here is a copy of the `/etc/keepalived/keepalived.conf` configuration file. As usual, do not forget to restart service after editing.

```
! Configuration File for keepalived

global_defs {
        notification_email {
                root@localhost
        }
        notification_email_from greenTenant@localhost
        smtp_server localhost
        smtp_connect_timeout 30

        enable_script_security
        script_user etu

        vrrp_version 3
}

vrrp_script keepalived_check_ipv4 {
        script "/usr/local/bin/keepalived_check_ipv4.sh"
        interval 1
        timeout 5
        rise 3
        fall 3
}
```

```
vrrp_script keepalived_check_ipv6 {
        script "/usr/local/bin/keepalived_check_ipv6.sh"
        interval 1
        timeout 5
        rise 3
        fall 3
}

vrrp_sync_group vrrp_group {
        group {
                VXLAN_7_IPv4
                VXLAN_7_IPv6
        }
        track_script {
                keepalived_check_ipv4
        }
        notify "/usr/local/bin/keepalived_notify.sh"
}

vrrp_instance VXLAN_7_IPv4 {
        state MASTER
        interface sw-vlan7
        virtual_router_id 74
        priority 221
        advert_int 1
        virtual_ipaddress {
                192.0.2.222/24
        }
}

vrrp_instance VXLAN_7_IPv6 {
        state MASTER
        interface sw-vlan7
        virtual_router_id 76
        priority 221
        advert_int 1
        virtual_ipaddress {
                fe80:7::de/64
        }
}
```

We must not forget to restart the daemon

```
sudo systemctl restart keepalived.service
```

7 Check the results

7.1 Look at VXLAN broadcast domain

. First, VXLAN ends know each other. Here are IPv4 and IPv6 tests from **Green** to **Blue**.

```
$ ping -q -c2 192.0.2.220
PING 192.0.2.220 (192.0.2.220) 56(84) bytes of data.

--- 192.0.2.220 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 1.099/1.189/1.280/0.090 ms
```

```
$ ping -q -c2 fe80:7::dc%sw-vlan7
PING fe80:7::dc%sw-vlan7(fe80:7::dc%sw-vlan7) 56 data bytes

--- fe80:7::dc%sw-vlan7 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.142/1.745/2.349/0.603 ms
```

. Second, we take a look at the switch TCAM (Ternary Content Addressable Memory) table for VLAN 7.

```
sudo ovs-appctl fdb/show C-3PO-GREEN
 port  VLAN  MAC                Age
    1     7  3a:98:95:a9:6b:44    1
    5     7  00:16:3e:b2:71:28    1
    2     7  00:16:3e:1e:f5:70    1
    2     7  4e:e8:f5:75:ec:4f    1
    2     7  00:16:3e:2d:4f:16    1
    2     7  00:16:3e:0b:7b:bc    1
    3     7  00:16:3e:04:a9:e8    1
    4     7  00:16:3e:70:16:53    1
```

7.2 Look at container addressing

. On the Blue tenant side.

```
$ lxc ls -c ns46
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
+----------+---------+--------------------+----------------------------------------+
|   NAME   |  STATE  |        IPV4        |                  IPV6                  |
+----------+---------+--------------------+----------------------------------------+
| blueC0   | RUNNING | 192.0.2.60 (eth0)  | 2001:db8:7:0:216:3eff:fe2d:4f16 (eth0) |
+----------+---------+--------------------+----------------------------------------+
| blueC1   | RUNNING | 192.0.2.53 (eth0)  | 2001:db8:7:0:216:3eff:fe0b:7bbc (eth0) |
+----------+---------+--------------------+----------------------------------------+
| blueC2   | RUNNING | 192.0.2.59 (eth0)  | 2001:db8:7:0:216:3eff:fee2:4c76 (eth0) |
+----------+---------+--------------------+----------------------------------------+
| blueDHCP | RUNNING | 192.0.2.250 (eth0) | 2001:db8:7:0:216:3eff:fe1e:f570 (eth0) |
+----------+---------+--------------------+----------------------------------------+
```

. On the Green tenant side.

```
lxc ls -c ns46
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
+-----------+---------+--------------------+-----------------------------------------+
|   NAME    |  STATE  |        IPV4        |                  IPV6                   |
+-----------+---------+--------------------+-----------------------------------------+
| greenC0   | RUNNING | 192.0.2.19 (eth0)  | 2001:db8:7:0:70be:e8ff:fe7a:a9e0 (eth0) |
+-----------+---------+--------------------+------------------------------------------+
| greenC1   | RUNNING | 192.0.2.20 (eth0)  | 2001:db8:7:0:216:3eff:fe70:1653 (eth0)   |
+-----------+---------+--------------------+------------------------------------------+
| greenDHCP | RUNNING | 192.0.2.251 (eth0) | 2001:db8:7:0:216:3eff:feb2:7128 (eth0)   |
+-----------+---------+--------------------+------------------------------------------+
```

7.3 Look at gateway management with keepalived

. On the Blue tenant side.

```
$ lxc exec blueC1 -- ip route ls
default via 192.0.2.222 dev eth0
192.0.2.0/24 dev eth0 proto kernel scope link src 192.0.2.53
```

```
$ lxc exec blueC1 -- ip -6 route ls
2001:db8:7::/64 dev eth0 proto kernel metric 256 expires 86289sec pref medium
fe80::/64 dev eth0 proto kernel metric 256 pref medium
default via fe80:7::de dev eth0 proto ra metric 1024 expires 1689sec hoplimit 64 pref medium
```

. On the Green tenant side.

```
$ lxc exec greenC1 -- ip route ls
default via 192.0.2.222 dev eth0
192.0.2.0/24 dev eth0 proto kernel scope link src 192.0.2.55
```

```
$ lxc exec greenC1 -- ip -6 route ls
2001:db8:7::/64 dev eth0 proto kernel metric 256 expires 86175sec pref medium
fe80::/64 dev eth0 proto kernel metric 256 pref medium
default via fe80:7::de dev eth0 proto ra metric 1024 expires 1575sec hoplimit 64 pref medium
```

7.4 Look at the traffic coming back and forth from the containers

. ICMP tests on the Blue tenant side: **0% packet loss**.

```
$ for c in blueC0 blueC1 blueDHCP; do echo ------------// $c && lxc exec $c -- ping -q -c3
    9.9.9.9; done
------------// blueC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 14.588/16.602/20.451/2.722 ms
------------// blueC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 14.378/15.808/16.893/1.055 ms
------------// blueDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 14.776/15.699/16.276/0.659 ms
```

```
$ for c in blueC0 blueC1 blueDHCP; do echo ------------// $c && lxc exec $c -- ping -q -c3
    2620:fe::fe; done
------------// blueC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
```

```
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 40.765/41.883/44.098/1.566 ms
------------// blueC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 40.502/42.337/44.911/1.874 ms
------------// blueDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 41.009/43.974/45.594/2.099 ms
```

. ICMP tests on the Green tenant side: **0% packet loss**.

```
$ for c in greenC0 greenC1 greenDHCP; do echo ------------// $c && lxc exec $c -- ping -q -c3
     9.9.9.9; done
------------// greenC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 13.106/13.386/13.805/0.301 ms
------------// greenC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 12.997/15.535/19.150/2.624 ms
------------// greenDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 13.140/13.546/13.751/0.287 ms
```

```
$ for c in greenC0 greenC1 greenDHCP; do echo ------------// $c && lxc exec $c -- ping -q -c3
     2620:fe::fe; done
------------// greenC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 44.135/44.822/46.163/0.948 ms
------------// greenC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 39.431/42.982/45.335/2.554 ms
------------// greenDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 43.803/44.232/44.768/0.400 ms
```

. HTTP(s) on the Blue tenant side.

```
$ for c in blueC0 blueC1 blueDHCP; do echo ------------// $c && lxc exec $c -- apt update;
    done
------------// blueC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
------------// blueC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
------------// blueDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

. HTTP(s) on the Green tenant side.

```
$ for c in greenC0 greenC1 greenDHCP; do echo ------------// $c && lxc exec $c -- apt update;
     done
------------// greenC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
------------// greenC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
------------// greenDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

7.5 Look at connection tracking on the redRouter Internet link

The two following commands give the list of the stateful entries stored in *netfilter* tables.

```
$ sudo conntrack -f ipv4 -L | grep mark=221
udp      17 3 src=192.0.2.251 dst=9.9.9.9 sport=59798 dport=53 src=9.9.9.9 dst=198.18.28.200
    sport=53 dport=59798 [ASSURED] mark=221 use=1
udp      17 0 src=192.0.2.20 dst=9.9.9.9 sport=34668 dport=53 src=9.9.9.9 dst=198.18.28.200
    sport=53 dport=34668 [ASSURED] mark=221 use=1
udp      17 0 src=192.0.2.20 dst=9.9.9.9 sport=57195 dport=53 src=9.9.9.9 dst=198.18.28.200
    sport=53 dport=57195 mark=221 use=1
conntrack v1.4.6 (conntrack-tools): 86 flow entries have been shown.
udp      17 3 src=192.0.2.251 dst=9.9.9.9 sport=43648 dport=53 src=9.9.9.9 dst=198.18.28.200
    sport=53 dport=43648 mark=221 use=1
```

```
$ sudo conntrack -f ipv6 -L | grep mark=221
conntrack v1.4.6 (conntrack-tools): 6 flow entries have been shown.
tcp      6 105 TIME_WAIT src=2001:db8:7:0:216:3eff:feb2:7128 dst=2a04:4e42:2d::644 sport
    =52746 dport=80 src=2a04:4e42:2d::644 dst=2001:678:3fc:1c::c8 sport=80 dport=52746 [
    ASSURED] mark=221 use=1
tcp      6 99 TIME_WAIT src=2001:db8:7:0:70be:e8ff:fe7a:a9e0 dst=2a04:4e42:2d::644 sport
    =51062 dport=80 src=2a04:4e42:2d::644 dst=2001:678:3fc:1c::c8 sport=80 dport=51062 [
    ASSURED] mark=221 use=1
tcp      6 102 TIME_WAIT src=2001:db8:7:0:216:3eff:fe70:1653 dst=2a04:4e42:2d::644 sport
    =38244 dport=80 src=2a04:4e42:2d::644 dst=2001:678:3fc:1c::c8 sport=80 dport=38244 [
    ASSURED] mark=221 use=1
```

7.6 Failover test

. First, we have to locate the IPv4 and IPv6 gateway addresses. Stated that the **Green** tenant has the highest VRRP priority, the sw-vlan7 interface on the Green side hold the gateway adresses.

```
$ ip addr ls dev sw-vlan7
6: sw-vlan7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UNKNOWN group
    default qlen 1000
    link/ether c2:8e:8d:87:0c:4c brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.221/24 brd 192.0.2.255 scope global sw-vlan7
       valid_lft forever preferred_lft forever
    inet 192.0.2.222/24 scope global secondary sw-vlan7
       valid_lft forever preferred_lft forever
    inet6 2001:db8:7::dd/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80:7::de/64 scope link nodad
       valid_lft forever preferred_lft forever
    inet6 fe80:7::dd/64 scope link
       valid_lft forever preferred_lft forever
    inet6 fe80::c08e:8dff:fe87:c4c/64 scope link
       valid_lft forever preferred_lft forever
```

We also have the ability to check the logs for VRRP state.

```
$ grep 'vrrp.*state' /var/log/syslog | tail -1
Mar 14 09:48:04 greenTenant Keepalived_vrrp[1339]: VRRP_Group(vrrp_group) Syncing instances
    to MASTER state
```

. Second, we take a look at the default gateway IPv4 and IPv6 addresses on the **Blue** tenant containers.

```
$ for c in blueC0 blueC1 blueDHCP; do echo ------------// $c && lxc exec $c -- ip route ls
    default; done
------------// blueC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
default via 192.0.2.222 dev eth0
------------// blueC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
default via 192.0.2.222 dev eth0
------------// blueDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
default via 192.0.2.222 dev eth0 onlink
```

```
$ for c in blueC0 blueC1 blueDHCP; do echo ------------// $c && lxc exec $c -- ip -6 route ls
     default; done
------------// blueC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
default via fe80:7::de dev eth0 proto ra metric 1024 expires 1328sec hoplimit 64 pref medium
------------// blueC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
default via fe80:7::de dev eth0 proto ra metric 1024 expires 1328sec hoplimit 64 pref medium
------------// blueDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
```

```
default via fe80:7::de dev eth0 proto ra metric 1024 expires 1328sec hoplimit 64 pref medium
```

. Third, we simulate Internet connectivity loss on the **Green** side with an `iptables` rule which drops traffic.

```
$ sudo iptables -A OUTPUT -d 9.9.9.9/32 -j DROP
```

Then we check the logs.

```
$ tail -n 7 /var/log/syslog
Mar 14 19:03:06 greenTenant Keepalived_vrrp[1339]: VRRP_Script(keepalived_check_ipv4) failed
    (exited with status 1)
Mar 14 19:03:06 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv4) Entering FAULT STATE
Mar 14 19:03:06 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv4) sent 0 priority
Mar 14 19:03:06 greenTenant Keepalived_vrrp[1339]: VRRP_Group(vrrp_group) Syncing instances
    to FAULT state
Mar 14 19:03:06 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv6) Entering FAULT STATE
Mar 14 19:03:06 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv6) Entering FAULT STATE
Mar 14 19:03:06 greenTenant avahi-daemon[385]: Withdrawing address record for 192.0.2.222 on
    sw-vlan7.
```

. Fourth, The **Green** has lost the gateway addresses.

```
$ ip addr ls dev sw-vlan7
6: sw-vlan7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UNKNOWN group
    default qlen 1000
    link/ether c2:8e:8d:87:0c:4c brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.221/24 brd 192.0.2.255 scope global sw-vlan7
       valid_lft forever preferred_lft forever
    inet6 2001:db8:7::dd/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80:7::dd/64 scope link
       valid_lft forever preferred_lft forever
    inet6 fe80::c08e:8dff:fe87:c4c/64 scope link
       valid_lft forever preferred_lft forever
```

The **Blue** is now holding the gateway addresses.

```
$ ip addr ls dev sw-vlan7
6: sw-vlan7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UNKNOWN group
    default qlen 1000
    link/ether 6e:aa:78:68:4d:49 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.220/24 brd 192.0.2.255 scope global sw-vlan7
       valid_lft forever preferred_lft forever
    inet 192.0.2.222/24 scope global secondary sw-vlan7
       valid_lft forever preferred_lft forever
    inet6 2001:db8:7::dc/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80:7::de/64 scope link nodad
       valid_lft forever preferred_lft forever
    inet6 fe80:7::dc/64 scope link
       valid_lft forever preferred_lft forever
    inet6 fe80::6caa:78ff:fe68:4d49/64 scope link
       valid_lft forever preferred_lft forever
```

. Fifth, we run connectivity tests from the containers of both tenants.

```
$ for c in blueC0 blueC1 blueDHCP; do echo ------------// $c && lxc exec $c -- apt update;
    done
------------// blueC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
------------// blueC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
------------// blueDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

```
$ for c in greenC0 greenC1 greenDHCP; do echo ------------// $c && lxc exec $c -- apt update;
     done
------------// greenC0
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
------------// greenC1
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
------------// greenDHCP
WARNING: cgroup v2 is not fully supported yet, proceeding with partial confinement
Hit:1 http://deb.debian.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

. Sixth, we check the logs to see the **Blue** tenant is now the VRRP master.

```
$ grep vrrp.*state /var/log/syslog | tail -1
Mar 14 19:03:06 blueTenant Keepalived_vrrp[1323]: VRRP_Group(vrrp_group) Syncing instances to
     MASTER state
```

. Finally, things go back to initial state when the `iptables` rule is deleted on the **Green** side.

```
$ sudo iptables -D OUTPUT -d 9.9.9.9/32 -j DROP
```

```
$ tail -n 15 /var/log/syslog
Mar 14 19:12:42 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv6) Entering BACKUP STATE
Mar 14 19:12:42 greenTenant Keepalived_vrrp[1339]: VRRP_Group(vrrp_group) Syncing instances
     to BACKUP state
Mar 14 19:12:42 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv4) Entering BACKUP STATE
Mar 14 19:12:43 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv6) received lower priority
    (220) advert from fe80:7::dc - discarding
Mar 14 19:12:43 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv4) received lower priority
    (220) advert from 192.0.2.220 - discarding
Mar 14 19:12:44 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv4) received lower priority
    (220) advert from 192.0.2.220 - discarding
Mar 14 19:12:44 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv6) received lower priority
    (220) advert from fe80:7::dc - discarding
Mar 14 19:12:45 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv4) received lower priority
    (220) advert from 192.0.2.220 - discarding
Mar 14 19:12:45 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv6) received lower priority
    (220) advert from fe80:7::dc - discarding
Mar 14 19:12:46 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv4) Entering MASTER STATE
Mar 14 19:12:46 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv4) using locally configured
    advertisement interval (1000 milli-sec)
Mar 14 19:12:46 greenTenant Keepalived_vrrp[1339]: VRRP_Group(vrrp_group) Syncing instances
     to MASTER state
```

```
Mar 14 19:12:46 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv6) Entering MASTER STATE
Mar 14 19:12:46 greenTenant Keepalived_vrrp[1339]: (VXLAN_7_IPv6) using locally configured
    advertisement interval (1000 milli-sec)
Mar 14 19:12:46 greenTenant avahi-daemon[385]: Registering new address record for 192.0.2.222
     on sw-vlan7.IPv4.
```

8 The ending words

This lab illustrates network gateway resiliency for containers hosted among two different tenants.

The VXLAN technology allows to share a single LAN or broadcast domain over an IP interconnection between these two tenants.

Hope this will give you ideas to experiment and investigate further ;).