Management VRF and DNS

March 12, 2020

Someone recently asked me why apt-get was not working when he enabled management VRF on Ubuntu 18.04. After a few back and forths and a little digging I was reminded of why. The TL;DR is systemd-resolved. This blog post documents how I came to that conclusion and what you need to do to use management VRF with Ubuntu (or any OS using a DNS caching service such as systemd-resolved).

The following example is based on a newly created Ubuntu 18.04 VM. The VM comes up with the 4.15.0-66-generic kernel which is missing the VRF module:

```
$ modprobe vrf
modprobe: FATAL: Module vrf not found in directory /lib/modules/4.15.0-66-
generic
```

despite VRF being enabled and built:

```
$ grep VRF /boot/config-4.15.0-66-generic
CONFIG_NET_VRF=m
```

which is really weird.[4] So for this blog post I shifted to the v5.3 HWE kernel:

```
$ sudo apt-get install --install-recommends linux-generic-hwe-18.04
```

although nothing about the DNS problem is kernel specific. A 4.14 or better kernel with VRF enabled and usable will work.

First, let's enable Management VRF. All of the following commands need to be run as root. For simplicity getting started, you will want to enable this sysctl to allow sshd to work across VRFs:

```
    echo "net.ipv4.tcp_l3mdev_accept=1" >> /etc/sysctl.d/99-sysctl.conf
    sysctl -p /etc/sysctl.d/99-sysctl.conf
```

Advanced users can leave that disabled and use something like the systemd instances to run sshd in Management VRF only.[1]

Ubuntu has moved to netplan for network configuration, and

apparently netplan is still missing VRF support despite requests from multiple users since May 2018:
https://bugs.launchpad.net/netplan/+bug/1773522

One option to workaround the problem is to put the following in /etc/networkd-dispatcher/routable.d/50-ifup-hooks:

```
#!/bin/bash

ip link show dev mgmt 2>/dev/null
if [ $? -ne 0 ]
then
        # capture default route
        DEF=$(ip ro ls default)

        # only need to do this once
        ip link add mgmt type vrf table 1000
        ip link set mgmt up
        ip link set eth0 vrf mgmt
        sleep 1

        # move the default route
        ip route add vrf mgmt ${DEF}
        ip route del default

        # fix up rules to look in VRF table first
        ip ru add pref 32765 from all lookup local
        ip ru del pref 0
        ip -6 ru add pref 32765 from all lookup local
        ip -6 ru del pref 0
fi
ip route del default
```

The above assumes eth0 is the nic to put into Management VRF, and it has a static IP address. If using DHCP instead of a static route, create or update the dhclient-exit-hook to put the default route in the Management VRF table.[3] Another option is to use ifupdown2 for network management; it has good support for VRF.[1]

Reboot node to make the changes take effect.

WARNING: If you run these commands from an active ssh session, you will lose connectivity since you are shifting the L3 domain of eth0 and that impacts existing logins. You can avoid the reboot by running the above commands from console.

After logging back in to the node with Management VRF enabled, the first thing to remember is that when VRF is enabled network addresses become relative to the VRF – and that includes loopback addresses (they are not that special).

Any command that needs to contact a service over the Management VRF needs to be run in that context. If the command does not have native VRF support, then you can use 'ip vrf exec' as a helper to do the VRF binding. 'ip vrf exec' uses a small eBPF program to bind all IPv4 and IPv6 sockets opened by the command to the given device ('mgmt' in this case) which causes all routing lookups to go to the table associated with the VRF (table 1000 per the setting above).

Let's see what happens:

```
$ ip vrf exec mgmt apt-get update
Err:1 http://mirrors.digitalocean.com/ubuntu bionic InRelease
  Temporary failure resolving 'mirrors.digitalocean.com'
Err:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
  Temporary failure resolving 'security.ubuntu.com'
Err:3 http://mirrors.digitalocean.com/ubuntu bionic-updates InRelease
  Temporary failure resolving 'mirrors.digitalocean.com'
Err:4 http://mirrors.digitalocean.com/ubuntu bionic-backports InRelease
  Temporary failure resolving 'mirrors.digitalocean.com'
```

Theoretically, this should Just Work, but it clearly does not. Why?

Ubuntu uses systemd-resolved service by default with /etc/resolv.conf configured to send DNS lookups to it:

```
$ ls -l /etc/resolv.conf
lrwxrwxrwx 1 root root 39 Oct 21 15:48 /etc/resolv.conf -> ../run/systemd
/resolve/stub-resolv.conf

$ cat /etc/resolv.conf
...
nameserver 127.0.0.53
options edns0
```

So when a process (e.g., apt) does a name lookup, the message is sent to 127.0.0.53/53. In theory, systemd-resolved gets the request and attempts to contact the actual nameserver.

Where does the theory breakdown? In 3 places.

First, 127.0.0.53 is not configured for Management VRF, so attempts to reach it fail:

```
$ ip ro get vrf mgmt 127.0.0.53
```

```
127.0.0.53 via 157.245.160.1 dev eth0 table 1000 src 157.245.160.132 uid 0
    cache
```

That one is easy enough to fix. The VRF device is meant to be the loopback for a VRF, so let's add the loopback addresses to it:

```
    $ ip addr add dev mgmt 127.0.0.1/8
    $ ip addr add dev mgmt ::1/128

    $ ip ro get vrf mgmt 127.0.0.53
    127.0.0.53 dev mgmt table 1000 src 127.0.0.1 uid 0
        cache
```

The second problem is systemd-resolved binds its socket to the loopback device:

```
    $ ss -apn | grep systemd-resolve
    udp  UNCONN   0      0       127.0.0.53%lo:53     0.0.0.0:*     users:
(("systemd-resolve",pid=803,fd=12))
    tcp  LISTEN   0      128     127.0.0.53%lo:53     0.0.0.0:*     users:
(("systemd-resolve",pid=803,fd=13))
```

The loopback device is in the default VRF and can not be moved to Management VRF. A process bound to the Management VRF can not communicate with a socket bound to the loopback device.

The third issue is that systemd-resolved runs in the default VRF, so its attempts to reach the real DNS server happen over the default VRF. Those attempts fail since the servers are only reachable from the Management VRF and systemd-resolved has no knowledge of it.

Since systemd-resolved is hardcoded (from a quick look at the source) to bind to the loopback device, there is no option but to disable it. It is not compatible with Management VRF – or VRF at all.

```
$ rm /etc/resolv.conf
$ grep nameserver /run/systemd/resolve/resolv.conf > /etc/resolv.conf
$ systemctl stop systemd-resolved.service
$ systemctl disable systemd-resolved.service

With that it works as expected:
$ ip vrf exec mgmt apt-get update
Get:1 http://mirrors.digitalocean.com/ubuntu bionic InRelease [242 kB]
Get:2 http://mirrors.digitalocean.com/ubuntu bionic-updates InRelease [88.7
kB]
Get:3 http://mirrors.digitalocean.com/ubuntu bionic-backports InRelease [74.6
kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://mirrors.digitalocean.com/ubuntu bionic-updates/universe amd64
Packages [1054 kB]
```

When using Management VRF, it is convenient (ie., less typing) to bind the shell to the VRF and let all commands run by it inherit the VRF binding:
$ ip vrf exec mgmt su – dsahern

Now all commands run will automatically use Management VRF. This can be done at login using libpamscript[2].

Personally, I like a reminder about the network bindings in my bash prompt. I do that by adding the following to my .bashrc:

```
NS=$(ip netns identify)
[ -n "$NS" ] && NS=":${NS}"

VRF=$(ip vrf identify)
[ -n "$VRF" ] && VRF=":${VRF}"
```

And then adding '${NS}${VRF}' after the host in PS1:

```
PS1='${debian_chroot:+($debian_chroot)}\u@\h${NS}${VRF}:\w\$ '
```

For example, now the prompt becomes:
dsahern@myhost:mgmt:~$

References:

[1] VRF tutorial, Open Source Summit, North America, Sept 2017
http://schd.ws/hosted_files/ossna2017/fe/vrf-tutorial-oss.pdf

[2] VRF helpers, e.g., systemd instances for VRF
https://github.com/CumulusNetworks/vrf

[3] Example using VRF in dhclient-exit-hook
https://github.com/CumulusNetworks/vrf/blob/master/etc/dhcp/dhclient-exit-hooks.d/vrf

[4] Vincent Bernat informed me that some modules were moved to non-standard package; installing linux-modules-extra-$(uname -r)-

generic provides the vrf module. Thanks Vincent.