Using the Linux VRF Solution

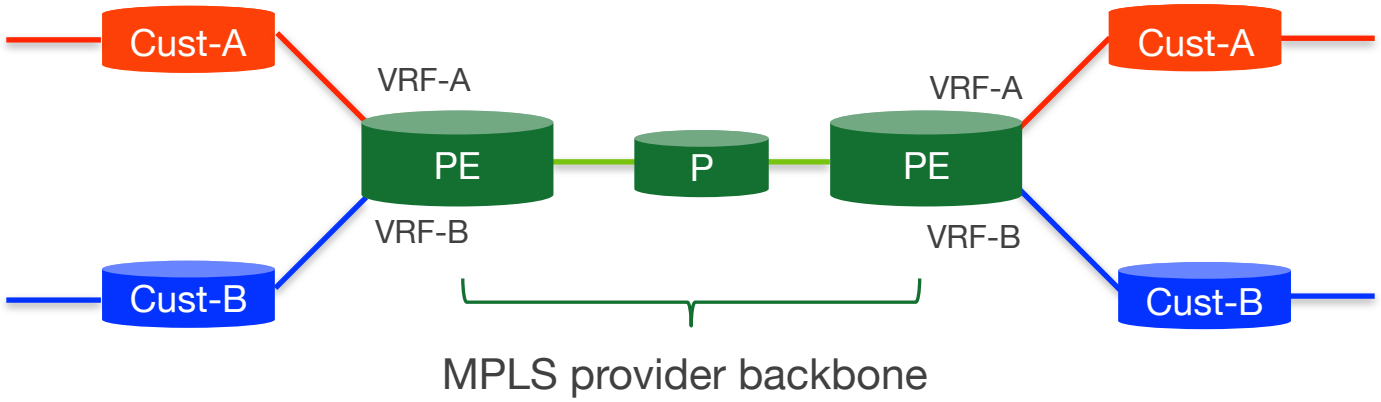
David Ahern — Cumulus Networks

Open Source Summit N.A., September 2017

What is a VRF?

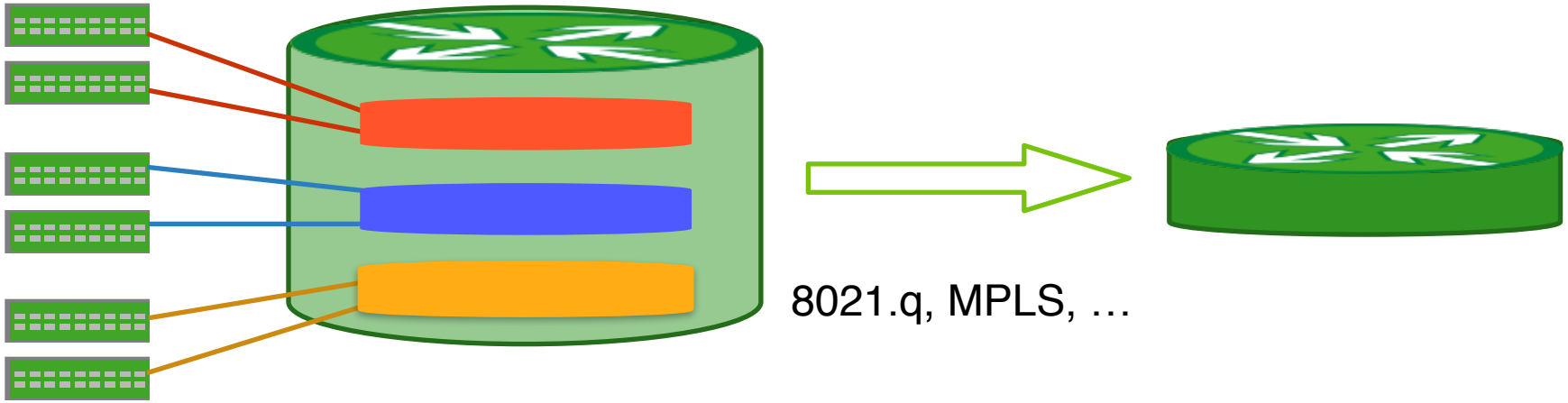
BGP/MPLS VPNs

RFC 2547 and 4364



Essentially a Virtual Router

Logically separate forwarding stacks within a node



Virtual Routing and Forwarding

cumulus

VRF = separate routing table

- per-node configuration providing Layer 3 traffic separation

Network interfaces can be associated with 1 and only 1 VRF

- Physical ports (ethN, ensN) or logical devices (e.g., VLAN, dummy, veth)
- Association makes network interfaces part of Layer 3 domain

Lookups for traffic ingressing / egressing interfaces restricted to VRF table

Widely used in NOS'es and networking deployments

Agenda

cumulus®

Linux VRF in a Nutshell

Managing VRFs

Applications and the User API for VRFs

Troubleshooting

Use Cases

Linux VRF in a Nutshell

VRF Design for Linux

VRF represented as a virtual network device

- VRF device correlates to FIB table via attribute on create

Network interfaces are enslaved to VRF device

- Device-VRF association makes device part of the L3 domain
- Host and connected routes moved to VRF table

Additional routes added to VRF table as needed

Policy routing via FIB rule directs lookups to VRF table

Impacts only IPv4 and IPv6 lookups

Network Namespace is not a VRF

cumulus

Network Namespace is the wrong model for VRF

- VRF is a **Layer 3** separation
- Network Namespace is a **complete stack** separation - network devices to sockets

In-depth discussion in a blog post

- https://cumulusnetworks.com/blog/vrf-for-linux/

VRF should only impact IPv4 and IPv6 lookups

- Device enslavement to VRF has no impact on LLDP or listing network interfaces

VRF as a Network Device

cumulus®

Network interface-to-VRF association similar to bridges

- Network interfaces enslaved to a VRF device makes those devices part of the L3 domain
- Familiar paradigm for networking

VRF as a Network Device

cumulus

Network interface-to-VRF association similar to bridges

netdevice is a fundamental construct in Linux networking stack

- Network addresses on VRF device - VRF-local loopback
- netfilter rules, tc rules, tcpdump on VRF device - apply to L3 domain as a whole

VRF as a Network Device

Network interface-to-VRF association similar to bridges

netdevice is a fundamental construct in Linux networking stack

Nesting of VRFs (L3) in a network namespace

- Follows existing paradigms for network interfaces and namespaces

VRF as a Network Device

Network interface-to-VRF association similar to bridges

netdevice is a fundamental construct in Linux networking stack

Nesting of VRFs (L3) in a network namespace

Applications use existing APIs

- SO_BINDTODEVICE, IP_PKTINFO, IP_UNICAST_IF

VRF as a Network Device

Network interface-to-VRF association similar to bridges

netdevice is a fundamental construct in Linux networking stack

Nesting of VRFs (L3) in a network namespace

Applications use existing APIs

Existing frameworks for configuration, monitoring, serviceability

- iproute2 commands (ip, ss), netlink, tracepoints

Linux Workflow

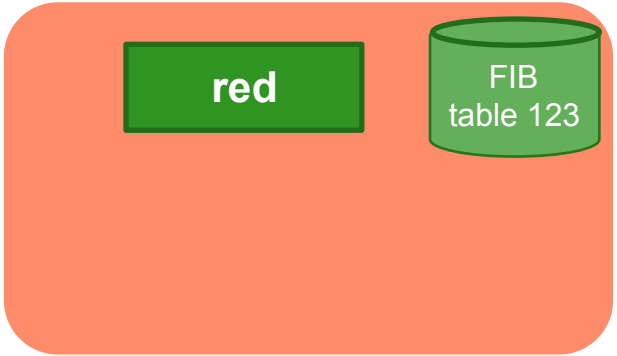
Create VRF device with table id

- ip link add red type vrf table 123

  VRF driver adds FIB rule on first device create

- ip link set red up

  Similar to bridging, VRF device must be admin up for packets to flow

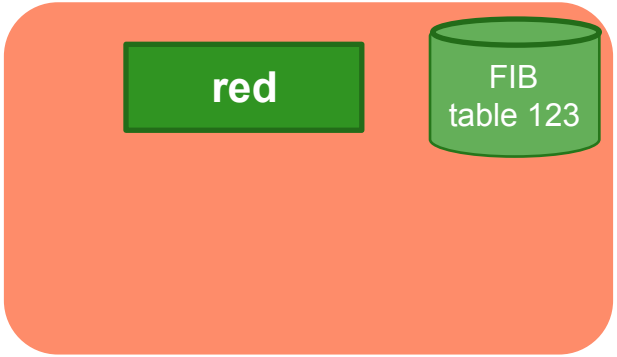


Linux Workflow

Create VRF device with table id

Add default route for VRF

- ip route add vrf red unreachable default metric 8192
- ip -6 route add vrf red unreachable default metric 8192



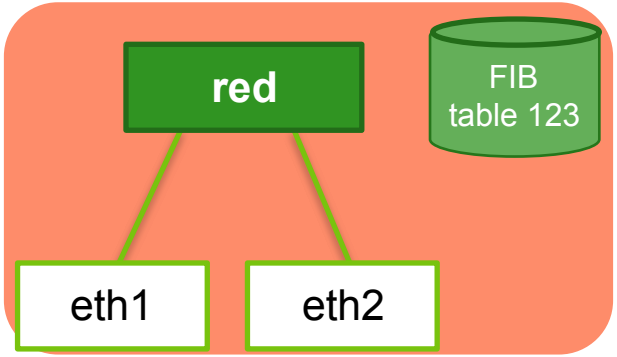
Linux Workflow

Create VRF device with table id

Add default route for VRF

Enslave interfaces

- ip link set eth1 vrf red
- ip link set eth2 vrf red



Linux Workflow

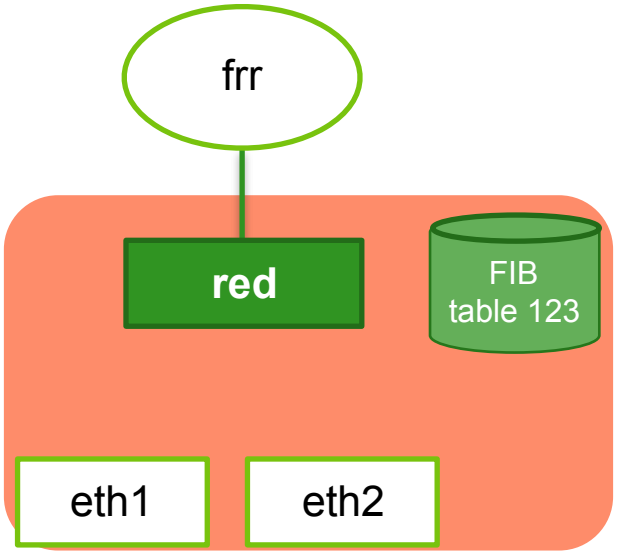
Create VRF device with table id

Add default route for VRF

Enslave interfaces

Add routes to VRF table as needed

- ip route add vrf red …



Linux Workflow

Create VRF device with table id

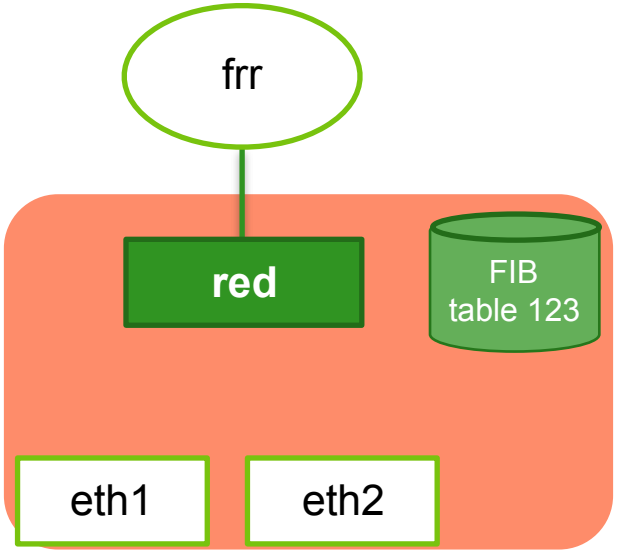
Add default route for VRF

Enslave interfaces

Add routes to VRF table as needed

Bind socket to VRF device

- Connects socket to L3 domain



Linux Workflow

Create VRF device with table id

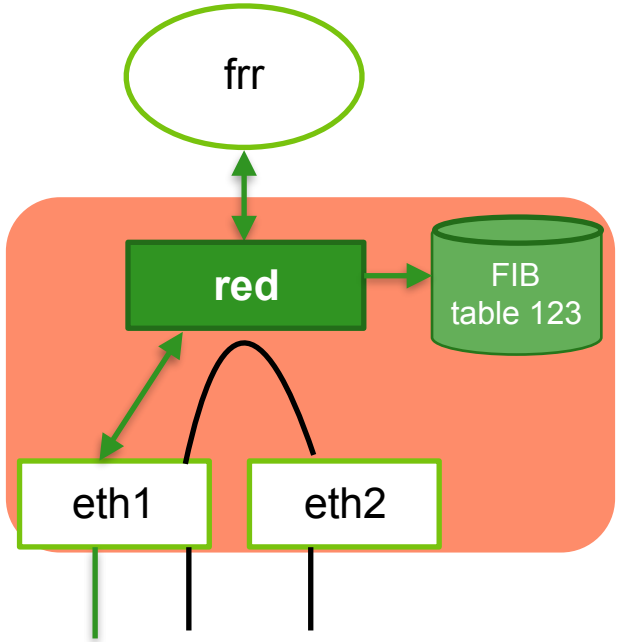
Add default route for VRF

Enslave interfaces

Bind socket to VRF device

Add routes to VRF table as needed

Send / receive packets



Managing VRFs

Netlink API

Create a VRF

- RTM_NEWLINK with IFLA_INFO_KIND = “vrf”
- IFLA_INFO_DATA attribute with IFLA_VRF_TABLE

  *IFLA_VRF_TABLE is a **required** attribute*

Add network interfaces to L3 domain

- RTM_NEWLINK or RTM_SETLINK for network interface with IFLA_MASTER set to VRF device index

Add routes

- RTM_NEWROUTE with RTA_TABLE set to VRF table

iproute2

Need iproute2 version that correlates to kernel version

- Need v4.10 for ‘ip vrf’ subcommand
- Or use top of tree

Subcommands support vrf keyword

- link, address, route, neighbor

iproute2

Need iproute2 version that correlates to kernel version or better

Most subcommands support vrf keyword

Create a VRF

- ip link add NAME type vrf table TABLE
- ip link set NAME up

Add network interfaces to L3 domain

- ip link set DEV vrf NAME

Add routes

- ip route add vrf NAME …

ifupdown2 Interface Manager

Alternative interface manager for Ubuntu and Debian

- Available from Debian and Ubuntu repositories

VRF add-on module

- Simple configuration options
- Takes care of the details of managing VRFs

Written in python

- should work on Red Hat based OS’es as well

ifupdown2 and Ubuntu

Version in Ubuntu 16 does not have VRF support

- dpkg -L ifupdown2 | grep vrf
- Bug to have it updated

  https://bugs.launchpad.net/zesty-backports/+bug/1712665
- ifupdown2 in Debian 9 has support and can be used with Ubuntu

  wget http://ftp.us.debian.org/debian/pool/main/i/ifupdown2/ifupdown2_1.0~git20170314-1_all.deb

  dpkg -i ifupdown2_1.0~git20170314-1_all.deb

Build from source repository

https://github.com/CumulusNetworks/ifupdown2

VRF with ifupdown2

Define VRF in /etc/network/interfaces

```
auto red
iface red
    address 10.10.10.10/32
    vrf-table 1001
    up ip route add vrf red unreachable default metric 8192
```

VRF with ifupdown2

Define VRF in /etc/network/interfaces

Add ‘vrf <name>’ to any iface stanza to add interface to VRF

```
auto eth1
iface eth1
    address 10.1.1.1/24
    vrf red
```

VRF with systemd-networkd

cumulus®

VRF support added in v230 by Andreas Rammhold

Configuraton files in /etc/systemd/network

VRF with systemd-networkd

Files in /etc/systemd/network

Define VRF - e.g., 25-mgmt.netdev

```
[NetDev]
Name=mgmt
Kind=vrf

[VRF]
TableId=252
```

(Note: TableId is changed to Table in v234+)

VRF with systemd-networkd

Files in /etc/systemd/network

Define VRF

Add interface to L3 domain - e.g., 30-ens4.network

```
[Match]
Name=ens4

[Network]
VRF=mgmt
DHCP=yes

[DHCP]
RouteTable=252
```

VRF with systemd-networkd

Files in /etc/systemd/network

Define VRF

Add interface to L3 domain

Bring up VRF devices - 90-vrf.network

```
[Match]
Driver=vrf
```

cumulus

VRF support added as of 3.2.27

API

```
struct rtnl_link *rtnl_link_vrf_alloc(void);
int rtnl_link_is_vrf(struct rtnl_link *link);
int rtnl_link_vrf_get_tableid(struct rtnl_link *link, uint32_t
*id);
int rtnl_link_vrf_set_tableid(struct rtnl_link *link, uint32_t
id);

void     rtnl_link_set_master(struct rtnl_link *, int);
void     rtnl_route_set_table(struct rtnl_route *, uint32_t);
```

Applications and VRF

VRF and Applications

cumulus

Network addresses and routes are relative to VRF

- Insert flashing lights, sounds, etc
- Addresses can be duplicated across VRFs

Need to specify VRF to use

- Clients (outgoing connections) require it
- Simplification options for servers / daemons - will get to those

Default context is main table

- a.k.a., Default VRF
- If a VRF is not specified, connections use the main table

  modulo policy routing and FIB rules

User API

Linux device APIs

- setsockopt
- cmsg + IP{6}_PKTINFO
- IP{6}_UNICAST_IF

MUST be specified before socket is used

- connect or bind to address
- cmsg in sendmsg for UDP and raw sockets

User API: setsockopt

cumulus

Open a socket

```
sd = socket(PF_INET{6}, ... );
```

Bind socket to device

```
if (setsockopt(sd, SOL_SOCKET, SO_BINDTODEVICE, name, strlen(name)))
        // handle error
```

Non-root option for UDP and raw sockets

```
int ifindex = ...;

if (setsockopt(sd, SOL_IP, IP_UNICAST_IF, ifindex, sizeof(ifindex)))
        // handle error

if (setsockopt(sd, SOL_IPV6, IPV6_UNICAST_IF, ifindex, sizeof(ifindex)))
        // handle error
```

User API: cmsg and IP_PKTINFO

```
unsigned char cmsgbuf[64] = {};
struct cmsghdr *cm = (struct cmsghdr *)cmsgbuf;
struct msghdr m = { .msg_control = (caddr_t)cm };

if (version == AF_INET) {
    struct in_pktinfo *pi;

    cm->cmsg_level = SOL_IP;
    cm->cmsg_type = IP_PKTINFO;
    cm->cmsg_len = CMSG_LEN(sizeof(struct in_pktinfo));
    pi = (struct in_pktinfo *)CMSG_DATA(cm);
    pi->ipi_ifindex = ifindex;

    m.msg_controllen = cm->cmsg_len;
```

Terminology

Default VRF

- If you don't specify a VRF, it defaults to the main table — aka, Default VRF

VRF Global

- Daemons / services that can work across all VRFs

VRF Local

- Daemons / services that work only in a VRF

VRF Aware

- Understands the bind-to-device (L3 domain) semantics

VRF Unaware

- Opposite of VRF aware: huh? what is SO_BINDTODEVICE?

VRF-Global Services

One daemon that works across all VRFs

- one listen socket not bound to any device (VRF or other)
- Remember: daemon owns port across ALL VRFs

Accepts connections or messages across all VRFs

- Child sockets (from accept call) bound to VRF of ingress device

Requires sysctl settings

- net.ipv4.tcp_l3mdev_accept = 1
- net.ipv4.udp_l3mdev_accept = 1

VRF-Local Services

Service or socket per VRF

- Listen socket bound to a single VRF
- Accepts any connection / message for any network interface enslaved to VRF

Device scope (kernel 4.14 and up)

- Socket bound to enslaved device
- Only accepts connections through network interface

VRF-aware Services

Ideally all services are “VRF-aware”

- They have a bind-to-device configuration option

Device APIs have been around for ages

- Few code bases support them
- Easy to add, but takes time - Lot of Open Source Software!
- Recommended convention for configuration option: <addr>%<vrf>

  use '%' as the delimiter between the address and vrf

VRF aware apps

- ping (-I), traceroute (-i), rsyslog (8.24), frr

VRF helper

VRF helper can set “context” for command

- ‘ip vrf exec’ (4.10 kernel and newer)
- intentionally follows ‘ip netns exec’ semantics

VRF helper

cumulus

VRF helper can set “context” that is inherited parent-to-child

Shortcut - set context on shell that is passed to commands

- ip vrf exec <name> bash
- Similar to ‘ip netns exec <name>’ bash

Used as a passthrough for commands run in a shell

- e.g., apt-get, dnf

VRF helper caveats

Shortcuts have limitations

VRF Context *ONLY* affects IPv4 and IPv6 sockets

- ‘ip vrf exec NAME ls’ has no impact on ‘ls’ command

  Similar to how ‘ip netns exec NAME ls’ has no impact on ‘ls’ command[1]

[1] With the exception of namespace based network files that 'ip netns’ sets up

VRF helper caveats

VRF Context *ONLY* affects IPv4 and IPv6 sockets

VRF Context can get "lost"

- Daemon is running in the background
- Command run by user in the foreground in a VRF context
- Command communicates with daemon over unix socket

VRF context of shell executing foreground command is lost!

systemd and docker are examples

- systemd runs in Default VRF, processes started by it default to Default VRF

VRF and systemd

cumulus

Want to start services bound to a VRF

- VRF-Local services
- Each VRF instance can have a different configuration

Need to work on systemd directive

- Vincent Bernat has created a patch
- Works for the l3mdev cgroup (implementation used in Cumulus Linux 3.x)

systemd: VRF as an Instance

cumulus®

Leverage systemd's instance capability

- VRF is the instance name

systemd: VRF as an Instance

Leverages systemd's instance capability

Instance files auto-generated based on service config

- VRF systemd generator creates <name>@.service config file
- Prepends Exec lines with 'ip vrf exec %i'
- https://github.com/CumulusNetworks/vrf

systemd: VRF as an Instance

Leverages systemd's instance capability

Instance files auto-generated based on service config

Bash CLI

- systemctl <action> <service>@<vrf>

Ansible

```
 name: Stop NTP service in default VRF
service: name=ntp state=stopped

name: Start NTP service in Mgmt VRF
service: name=ntp@mgmt state=started
```

systemd: VRF as an Instance

Leverages systemd's instance capability

Instance files auto-generated based on service config

Bash CLI / Ansible

Programmatic way to start / stop services bound to VRF

- VRF destroy and re-create

systemd: Caveats using vrf helper

Package provided instance file needs to be compatible with ‘ip vrf exec’

- ssh has an instance file that is not (uses –i option with sshd which assumes launched by inetd)
- Cumulus Linux and vrf package has a workaround for ssh

systemd: Caveats using vrf helper

Collision with instance file provided by package

Some service arguments are not compatible with VRF as an instance

- Socket activation may not work

  docker wants to use '–H fd://'; does not work with Management VRF

- Root user only (hope to fix this soon)

  Can not specify non-root user with User= configuration

systemd: Caveats using vrf helper

Collision with instance file provided by package

Some service arguments are not compatible with VRF as an instance

Service in Default VRF may need to be stopped before VRF instances can run

- If a process in Default VRF does a global bind on a port, that process owns the port across all VRFs
- Must be stopped before VRF instance will run

systemd: Caveats using vrf helper

Collision with instance file provided by package

Some service arguments are not compatible with VRF design

Service in Default VRF may need to be stopped before VRF instances can run

Not all services need to be run in VRF context

- Only services that use IPv4 or IPv6 sockets

Some apps have hidden gotchas

snmpd in net-snmp package

- Always sends a message with IP_PKTINFO and index set to 0

  booooo…..

- overrides the SO_BINDTODEVICE and blows up the VRF context

yes, Cumulus Linux has a patch for that

Some apps have hidden gotchas

snmpd in net-snmp package

sshd ListenAddress directive

- Addresses are relative to a VRF
- If ListenAddress is set, it needs to agree with VRF context of sshd

API Summary

Applications need to specify VRF

Options

- Configuration option specific to program (config file or command line option)

  for programs that understand bind-to-device APIs

- VRF helper - ip vrf exec
- Systemd and VRF instances: ip vrf exec prepended to Exec lines
- Future: Systemd directive once it gets implemented

Pros and cons to each

Troubleshooting

iproute2: List VRFs

List VRFs that exist in the kernel

```
# ip vrf show
Name              Table
-----------------------
red               1001
blue              1002
green             1003
```

iproute2: List VRF-specific data

Commands with vrf keyword

- ip {link|addr|neigh|route} show vrf NAME

List network interfaces in VRF red

- ip link show vrf red

List interfaces and addresses for VRF red

- ip address show vrf red

Show neighbor entries for VRF red

- ip neighbor show vrf red

iproute2: Remember, ‘ip’ is a netlink command!

cumulus

‘ip’ is a netlink based command

- netlink commands are **NOT** affected by VRF context
- ip vrf exec red ip route show == ip route show

Use ‘vrf’ keyword

- ip route show **vrf** red

Important to remember if using VRF context for login shells (Management VRF)

- Setting context on a login shell is only a short cut for adding VRF context to commands that open an IPv4/v6 socket (e.g., apt-get)

iproute2: About listing routes in a VRF …

‘ip route show vrf NAME’ vs ‘ip route show table N’

- means to filter out local, IPv4 broadcast and IPv6 multicast routes (tend to clutter the screen)
- Similar to local table versus main table

Show only unicast routes for VRF table

- ip route show vrf red

Show every route entry in table

- ip route show table N

iproute2: ip route show vrf NAME

```
# ip ro ls vrf red
unreachable default metric 8192
10.100.1.0/24 dev eth1 proto kernel scope link src 10.100.1.3
10.100.2.0/24 dev eth2 proto kernel scope link src 10.100.2.3

# ip -6 ro ls vrf red
2001:db8:1::/120 dev eth1 proto kernel metric 256 pref medium
anycast 2001:db8:2:: dev red proto kernel metric 0 pref medium
2001:db8:2::/120 dev eth2 proto kernel metric 256 pref medium
anycast fe80:: dev lo proto kernel metric 0 pref medium
anycast fe80:: dev lo proto kernel metric 0 pref medium
fe80::/64 dev eth1 proto kernel metric 256 pref medium
fe80::/64 dev eth2 proto kernel metric 256 pref medium
ff00::/8 dev eth1 metric 256 pref medium
ff00::/8 dev eth2 metric 256 pref medium
unreachable default dev lo metric 8192 error -113 pref medium
```

iproute2: ip route show table N

```
# ip ro ls table 1001
unreachable default metric 8192
broadcast 10.100.1.0 dev eth1 proto kernel scope link src 10.100.1.3
10.100.1.0/24 dev eth1 proto kernel scope link src 10.100.1.3
local 10.100.1.3 dev eth1 proto kernel scope host src 10.100.1.3
broadcast 10.100.1.255 dev eth1 proto kernel scope link src 10.100.1.3
broadcast 10.100.2.0 dev eth2 proto kernel scope link src 10.100.2.3
10.100.2.0/24 dev eth2 proto kernel scope link src 10.100.2.3
local 10.100.2.3 dev eth2 proto kernel scope host src 10.100.2.3
broadcast 10.100.2.255 dev eth2 proto kernel scope link src 10.100.2.3

# ip -6 ro ls table 1001
anycast 2001:db8:1:: dev red proto kernel metric 0 pref medium
local 2001:db8:1::3 dev red proto kernel metric 0 pref medium
2001:db8:1::/120 dev eth1 proto kernel metric 256 pref medium
...
(too much output for a slide)
```

iproute2: List pids bound to VRF context

ip vrf pids

```
$ ip vrf pids mgmt
 1319  ntpd
 1392  systemd
 1425  (sd-pam)
12868  sshd
12902  sshd
12903  sftp-server
18115  sshd
18149  sshd
18150  bash
 4784  bash
 4859  screen
13246  bash
13491  ip
```

iproute2: Show VRF context for process

VRF binding for current shell

- ip vrf id

VRF binding for any process

- ip vrf id <pid>

Again, both follow ‘ip netns’ syntax

If ‘ip vrf exec’ fails

‘ip vrf exec’ installs a BPF program in a cgroup

- Requires locked memory (1 page)
- Some OS’es set limit (see ‘ulimit -l’)

If it fails:

```
# ip vrf exec red ls
Failed to load BPF prog: 'Operation not permitted’
```

Check for other users — e.g., running perf session

Bump the limit for locked memory

- ulimit -l <higher-number>

iproute2: Show sockets bound to a VRF

‘ss’ command with device filter

- ss -a ‘dev == NAME’
- Add -K to close the sockets (e.g., when VRF is deleted)

```
$ ss -a 'dev == mgmt'
Netid  State      Recv-Q Send-Q  Local Address:Port         Peer Address:Port
udp    UNCONN     0      0       192.168.1.23%mgmt:ntp                   *:*
udp    UNCONN     0      0          127.0.0.1%mgmt:ntp                   *:*
udp    UNCONN     0      0                  *%mgmt:ntp                   *:*
udp    UNCONN     0      0                ::1%mgmt:ntp                  :::*
udp    UNCONN     0      0                 ::%mgmt:ntp                  :::*
tcp    ESTAB      0      0       192.168.1.23%mgmt:ssh    192.168.2.105:56749
tcp    ESTAB      0      0       192.168.1.23%mgmt:ssh     192.168.0.50:64561
tcp    ESTAB      0      0       192.168.1.23%mgmt:ssh     192.168.0.50:64505
tcp    ESTAB      0      592     192.168.1.23%mgmt:ssh     192.168.0.50:65415
```

FIB Tracepoints

cumulus

Pre-defined tracepoints

- fib:fib_table_lookup
- fib:fib_table_lookup_nh
- fib:fib_validate_source
- fib6:fib6_table_lookup

FIB Tracepoints: IPv4 Example

```
# perf record -e fib:* -- ip route get fibmatch 192.168.1.1
default via 10.1.1.253 dev eth0
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.002 MB perf.data (2 samples) ]

# perf script -F trace:event,trace
   fib:fib_table_lookup: table 255 oif 0 iif 0 src 0.0.0.0 dst 192.168.1.1 tos 0 scope 0 flags 0
   fib:fib_table_lookup: table 254 oif 0 iif 0 src 0.0.0.0 dst 192.168.1.1 tos 0 scope 0 flags 0
fib:fib_table_lookup_nh: nexthop dev eth0 oif 2 src 10.1.1.3
```

FIB Tracepoints: IPv4 Example

Add a return probe for clarity

```
# perf probe fib_table_lookup%return ret=%ax       # for x86

# perf record -e fib:fib_table_lookup -e probe:fib_table_lookup ...

# perf script -F trace:event,trace
   fib:fib_table_lookup: table 255 oif 0 iif 0 src 0.0.0.0 dst 192.168.1.1 tos 0 scope 0 flags 0
 probe:fib_table_lookup: (ffffffff814668a7 <- ffffffff8146c662) ret=0xffffffff5
   fib:fib_table_lookup: table 254 oif 0 iif 0 src 0.0.0.0 dst 192.168.1.1 tos 0 scope 0 flags 0
fib:fib_table_lookup_nh: nexthop dev eth0 oif 2 src 10.1.1.3
 probe:fib_table_lookup: (ffffffff814668a7 <- ffffffff8146c662) ret=0x0
```

tcpdump

packet capture / tcpdump on enslaved device or VRF device

tcpdump on VRF device

- All packets into and out of any device enslaved to the VRF
  few exceptions: IPv4 multicast, IPv6 link local, multicast
- VRF global view of packets

systemd Controlled Services

systemctl status <service>@<vrf>.service

- e.g., systemctl status ntpd@mgmt.service

systemctl list-units *.service | grep <vrf>

Services need to be restarted if VRF device is destroyed and re-created

Connection failures

Global services?

- check that l3mdev sysctl settings are enabled!

Miscellaneous Notes

Limits

How many VRFs does Linux support?

- How much memory does your server / switch have?
- Linux kernel does not have a limit

Subsystem limits might require sysctl settings to be increased

- e.g., net.ipv6.route.max_size defaults to 4096

System limit on the total number of devices

- device index is a 4-byte signed integer
- 2,147,483,647

Default Route for each VRF

Add a high metric default route to each VRF table

- Ensures lookup terminates in that table
- ip route add vrf red default unreachable metric 8192
- Can add a second default route with lower metric

Default routes from DHCP Server need to be installed in VRF table

- Use a dhclient exit hook
- Table / TableId parameter for systemd-networkd

Move FIB Rule for Local Table

cumulus

False hits in local table due to FIB rule

Lower the priority of the rule for local table

- FIB rule for local table defaults to preference of 0
- main table is at 32766
- ip rule add from all lookup local pref 32765 && ip rule delete from all lookup local

  Do on console or in a script

  Will break networking if rule for local table does not exist

About the l3mdev FIB Rule

l3mdev is a dynamic rule - works for all VRF instances

```
$ ip ru ls
 1000:    from all lookup [l3mdev-table]
32765:    from all lookup local
32766:    from all lookup main
32767:    from all lookup default
```

l3mdev rule - table id is fetched from device

- Lookups directed to that table

Can use per-VRF FIB rules - performance impacts

- 1 iif rule, 1 oif rule per VRF - required prior to kernel v4.8

Tips & Tricks: Inter-VRF Routing

cumulus

Routing Between VRFs

Explicit route in a table

- ip route add vrf red 1.1.1.0/24 dev eth2
- eth2 is in alternate VRF

Full lookup in VRF table

- ip route add vrf red 1.1.1.0/24 dev green

Tips & Tricks: MPLS

cumulus®

Label pop and lookup

- ip -f mpls route add LABEL dev VRF
- e.g., ip -f mpls route add 101 dev red

Lookup in VRF table to decide fate of packet

Key Features by Linux Kernel Version

cumulus

Version	Feature
4.3	Basic IPv4 support, FIB tracepoint
4.4	Basic IPv6 support (global addresses only)
4.5	“VRF-all” TCP sockets, FIB6 tracepoint
4.6	Keep IPv6 global addresses on admin down
4.7	Fix ingress device for IP_PKTINFO, IP6_PKTINFO
4.8	IPv6 linklocal and multicast, local VRF traffic, l3mdev FIB rule, performance improvements, inet_diag filter on device

Key Features by Linux Kernel Version

cumulus®

Version	Feature
4.9	IPv6 and Router Advertisements
4.10	Initial IPv4 multicast, BPF for cgroups - basis for ‘ip vrf exec’
4.11	“VRF-all” UDP sockets
4.12	More performance improvements
4.13	IPv4 multicast, IGMP bug fixes
4.14	Bind sockets to enslaved devices LTS kernel - first one with a complete VRF implementation

VRF and Distributions

cumulus

Ubuntu 16.04

- 4.4 kernel, option to upgrade to hwe and 4.10 kernel
- iproute2 version is 4.3; need a newer version!

  https://github.com/dsahern/iproute2 debian-builds
- install ifupdown2 from Debian 9 (see earlier slide)

Debian 9 / Stretch

- 4.9 kernel, 4.9 iproute2

  need newer version of both for 'ip vrf exec' capability
- good version of ifupdown2

VRF and Distributions

Fedora 26

- 4.12 kernel, 4.11 iproute2 — yea!
- systemd-networkd has VRF support

Remember to enable l3mdev sysctl settings if enabling Management VRF!

bash prompt

Add VRF to bash prompt as queue to VRF context

vrf package adds /etc/profile.d/vrf.sh

- adds VRF to PS1
- PS1='\u@\h${VRF}:\w\$ ‘
- dsa@ubuntu16:mgmt:~$

source file to use it

- add ‘. /etc/profile.d/vrf.sh’ to .bash_profile
- or add ‘. /etc/profile’ which includes all files under /etc/profile.d

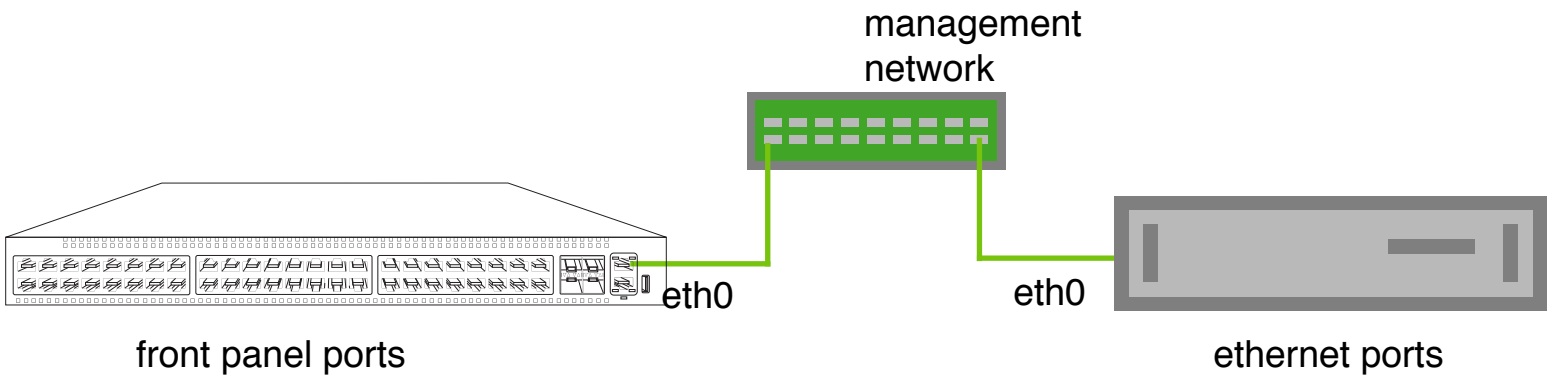
Use Case: Management VRF

Use Case: Management VRF

Separate management plane from data plane

Common deployment for switches and routers

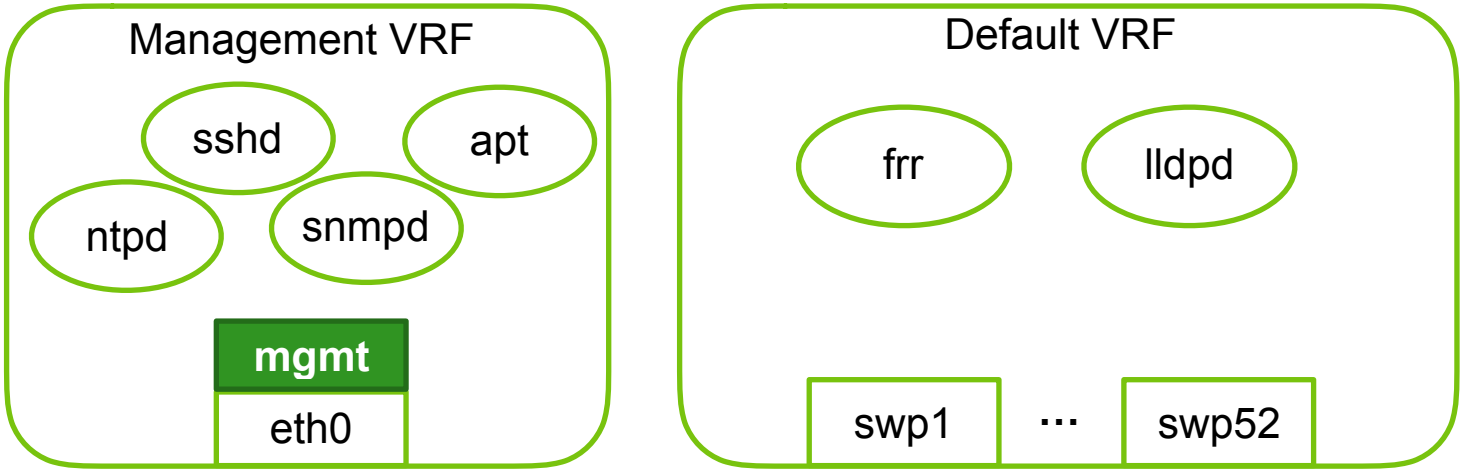
- Applicable to server and host deployments as well



Use Case: Management VRF

Management traffic only over mgmt VRF

- sshd, ntp, snmp, ansible, apt-get
- systemctl with VRF instances!



Use Case: Management VRF

Convenient to set login shell to mgmt VRF

- Avoids having to set mgmt context on each command
- e.g., libpamscript can set login shell to VRF cgroup

Use Case: Management VRF

cumulus

Convenient to set login shell to mgmt VRF

Daemon Configuration uses Management address

- sshd: ListenAddress
- snmpd: agentAddress
- MUST be bound to mgmt VRF

```
for s in ssh ntp snmpd; do
   systemctl stop ${s}; systemctl disable ${s}
   systemctl start ${s}@mgmt; systemctl enable ${s}@mgmt
done
```

Requires vrf and mgmt-vrf packages from Cumulus Networks to generate systemd files

Use Case: Management VRF

Convenient to set login shell to mgmt VRF

Daemon Configuration uses Management address

rsyslog forwarding

- Forwarding messages to a remote host over mgmt VRF
- Use new syntax to set Device option for omfwd module

  action(type="omfwd" Target="<ip>" Device="mgmt" Port="514" Protocol="udp")
- Version 8.24 and up

Use Case: Multitenancy

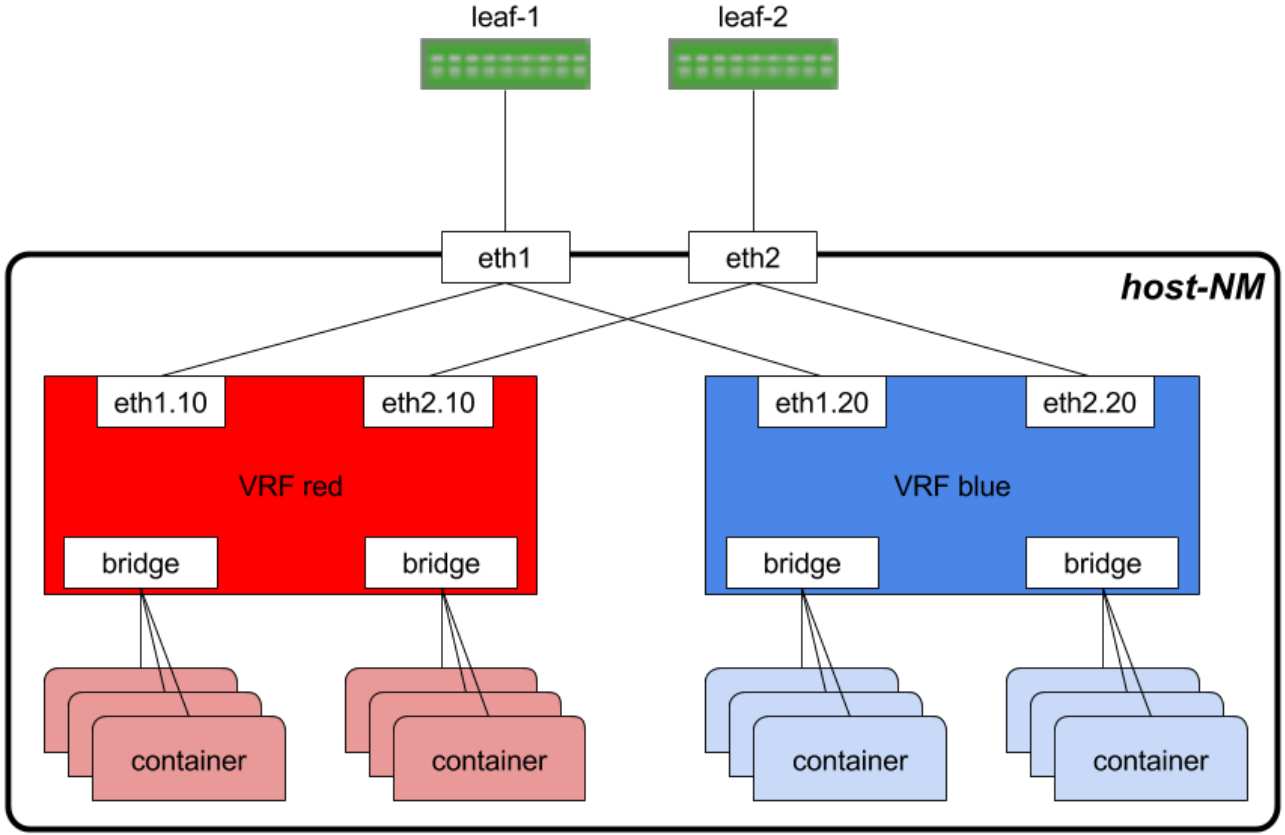
Use Case: Multitenancy

cumulus®

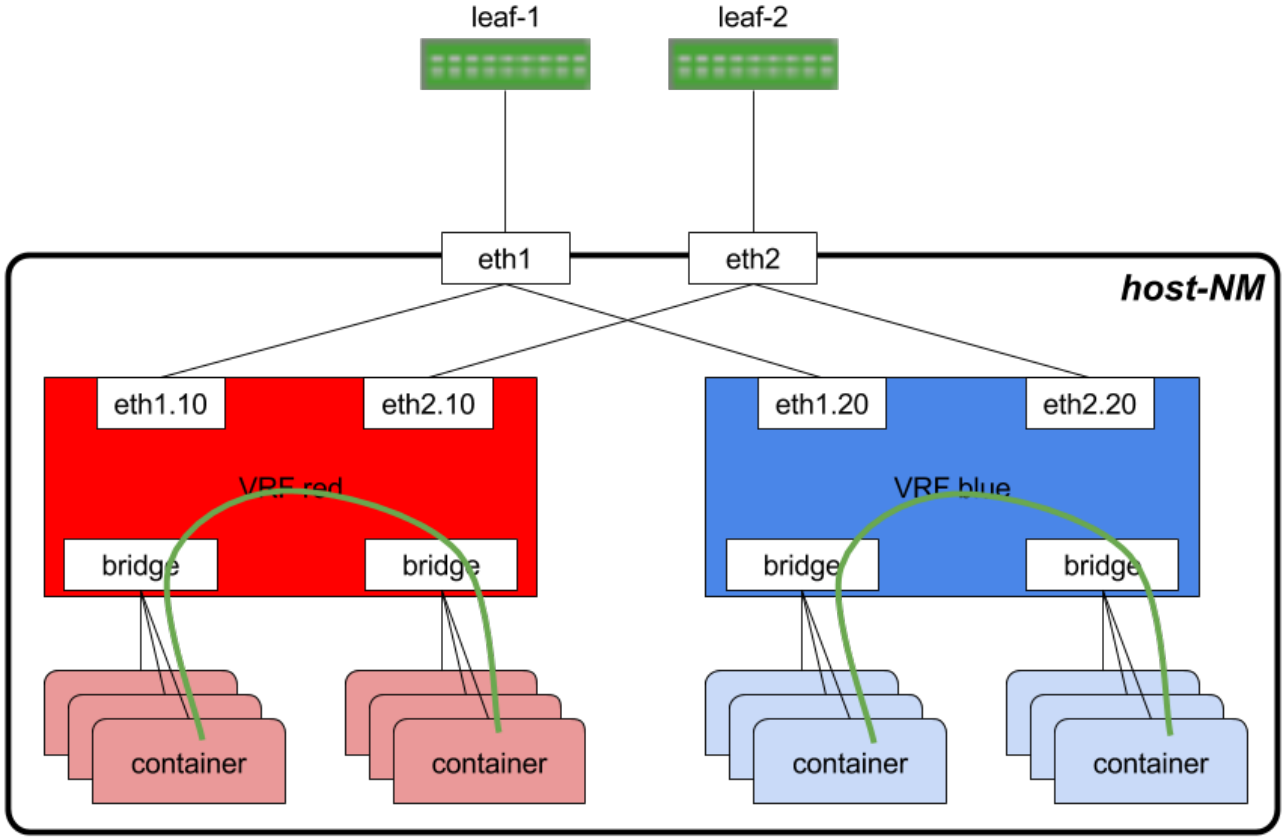
Separate container and VM traffic on host

Allow some containers on host to communicate with each other

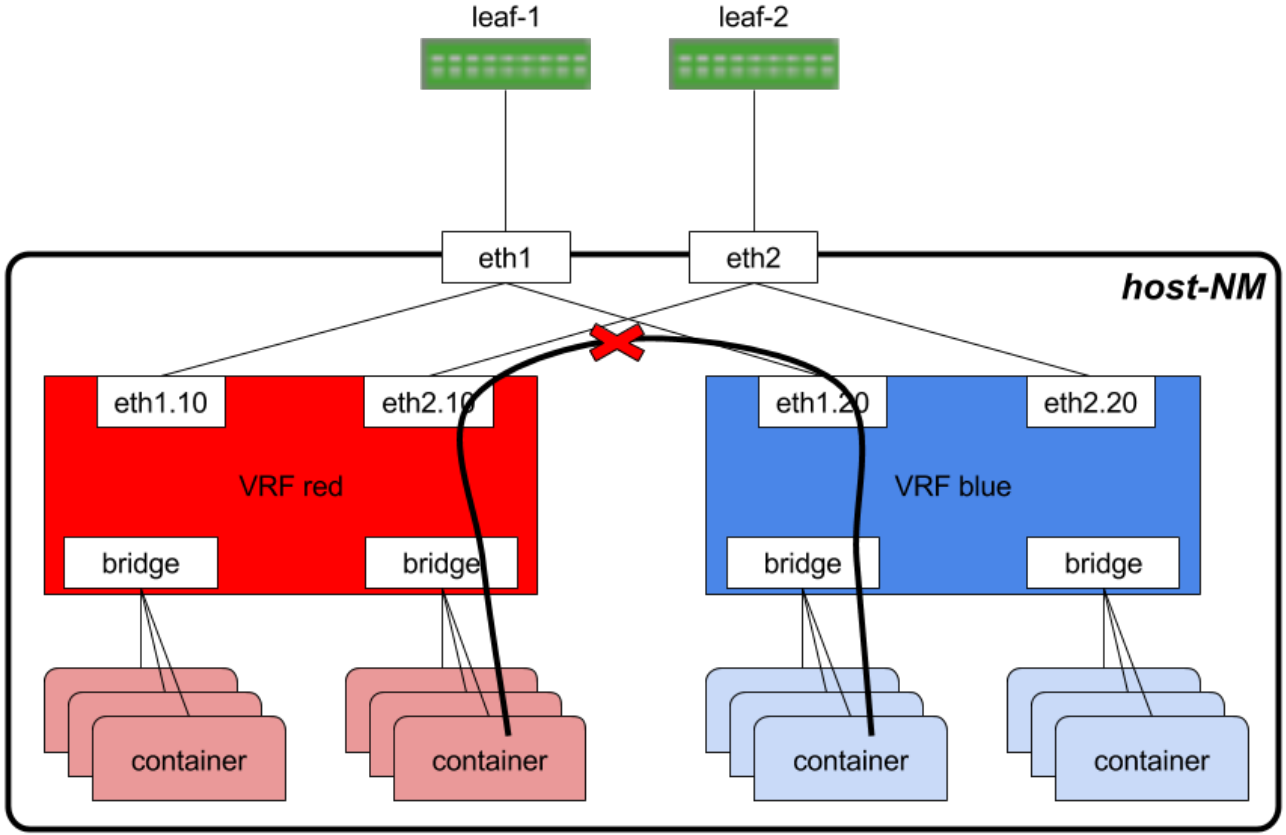
Multitenancy: Host Networking Architecture



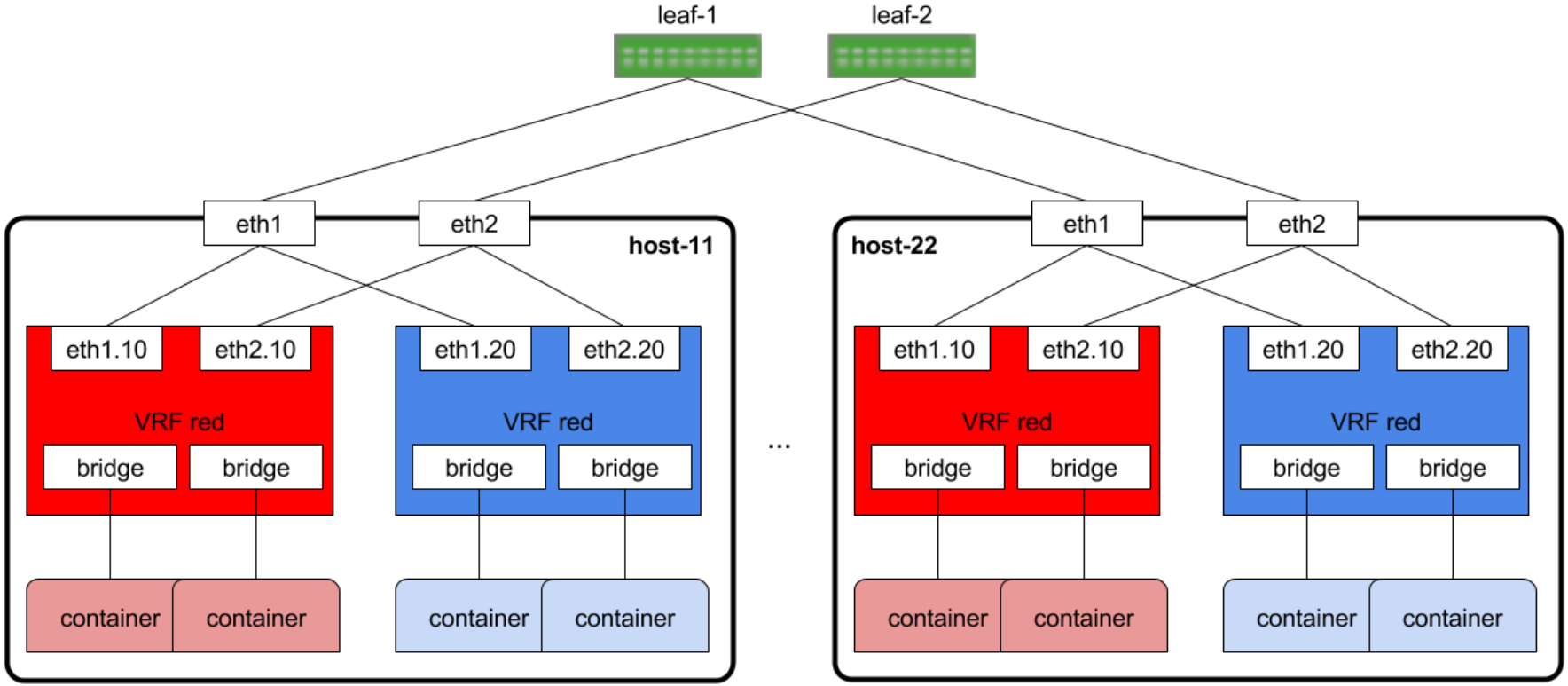
Host Networking - Intra-VRF allowed



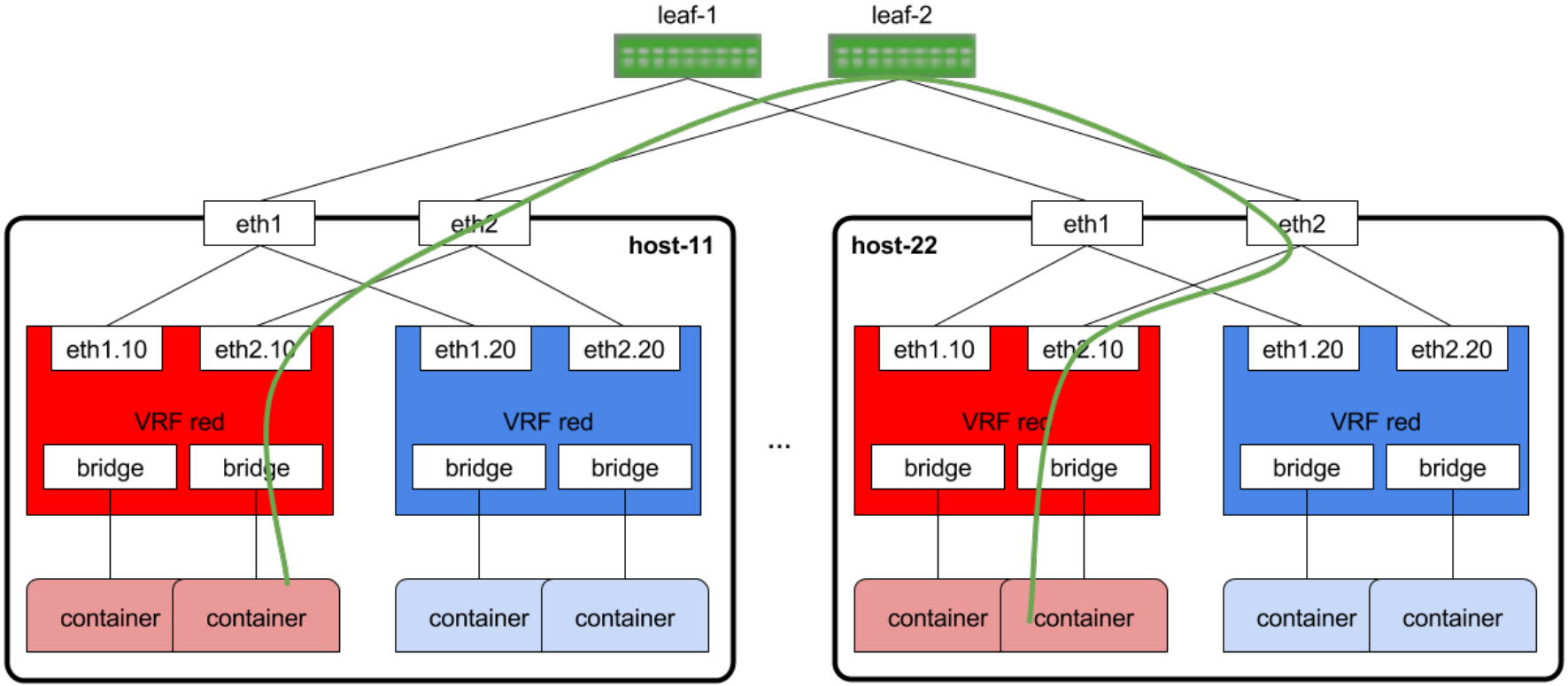
Host Networking - Cross VRF not allowed



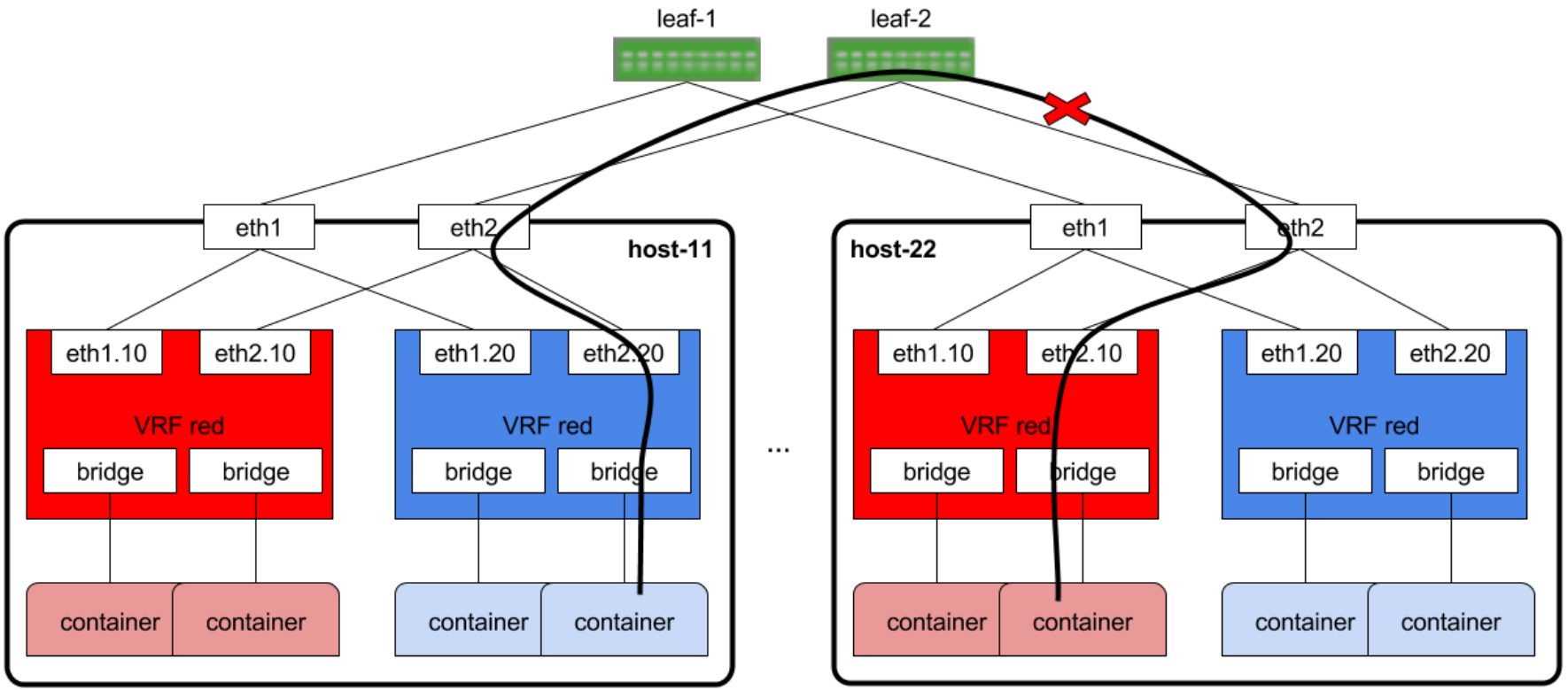
Multiple Host Networking



Multiple Host Networking - Intra-VRF allowed



Multiple Host Networking - Cross VRF not allowed



Multitenancy: Routing on the Host

cumulus®

Extend L3 fabric to host

- Run frr in the host
- BGP unnumbered simplifies the configuration

Multitenancy: Routing on the Host

Extend L3 fabric to host

ECMP default route to each leaf in each VRF

- Learned from leafs by frr and installed in host

Multitenancy: Routing on the Host

Extend L3 fabric to host

ECMP default route to each leaf in each VRF

Container networking configuration

- Address + default route

Multitenancy: Routing on the Host

Extend L3 fabric to host

ECMP default route to each leaf in each VRF

Container networking configuration

Container networks distributed to leafs

- Network fabric learns about container networks as they come on line
- Forwarding isolation provided by VRF

Demonstration using Vagrant

cumulus

Vagrant used for topology orchestration

Ansible for configuring the nodes

Files available from github:

- https://github.com/dsahern/cldemos

Other Use Cases

cumulus®

VRF and VLANs, VRF with MPLS

http://www.netdevconf.org/1.1/proceedings/slides/ahern-vrf-tutorial.pdf

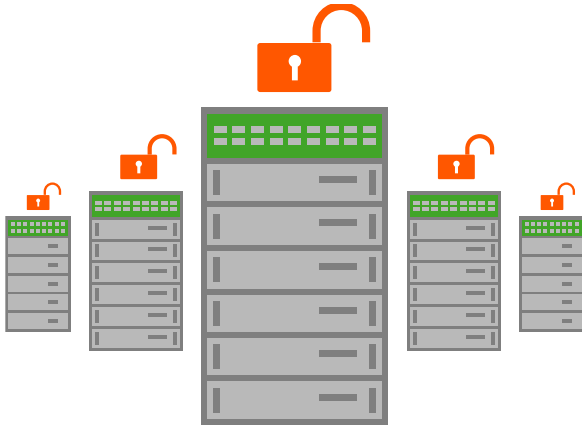
Q & A

cumulus®

Simple concept - multiple routing tables

Lot of Details

Unleashing the Power of Open Networking



Thank You!