Microservice Networking Leveraging VRF on the Host

David Ahern — Cumulus Networks

Netdev 1.2, October 2016

VRF on the Host

It’s gonna be HUUUUGE!

VRF on the Host

VRF recently added to Linux Networking Stack, now appearing in OS distributions

- Host can leverage VRF for traffic segmentation

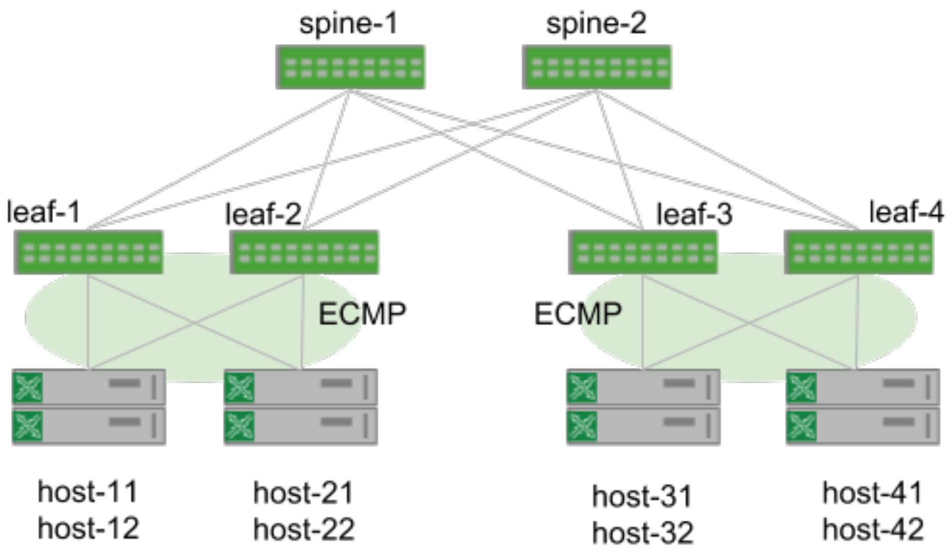
Intent of this tutorial is to get people thinking about how VRF can be used on the host

Network Diagram

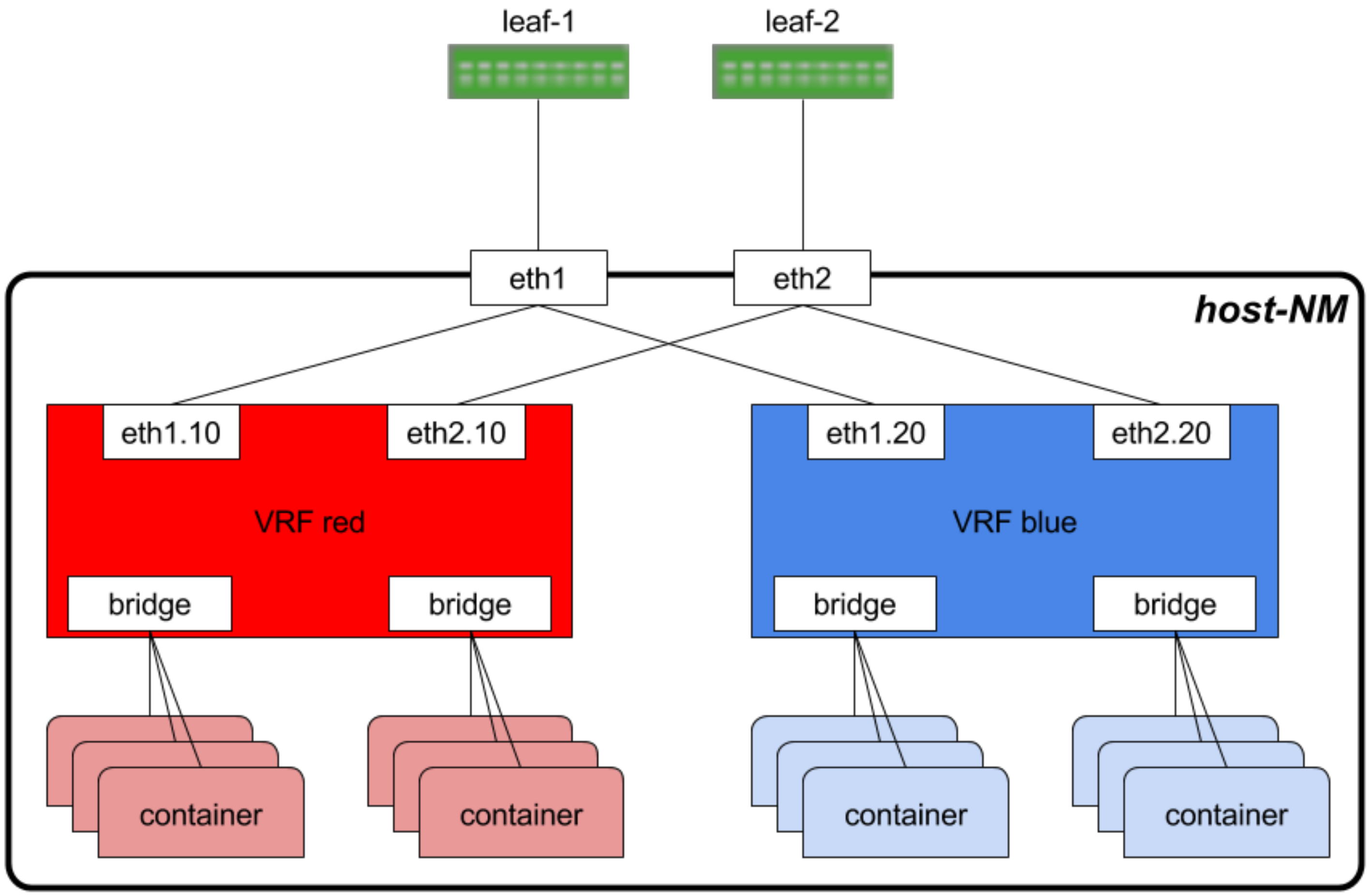
2 spines, 4 leafs, 8 hosts

- All leafs connected to all spines
- Each host connected to 2 leafs (ECMP default route)
- VRF provides traffic isolation at Layer 3
- VLANs for trunking

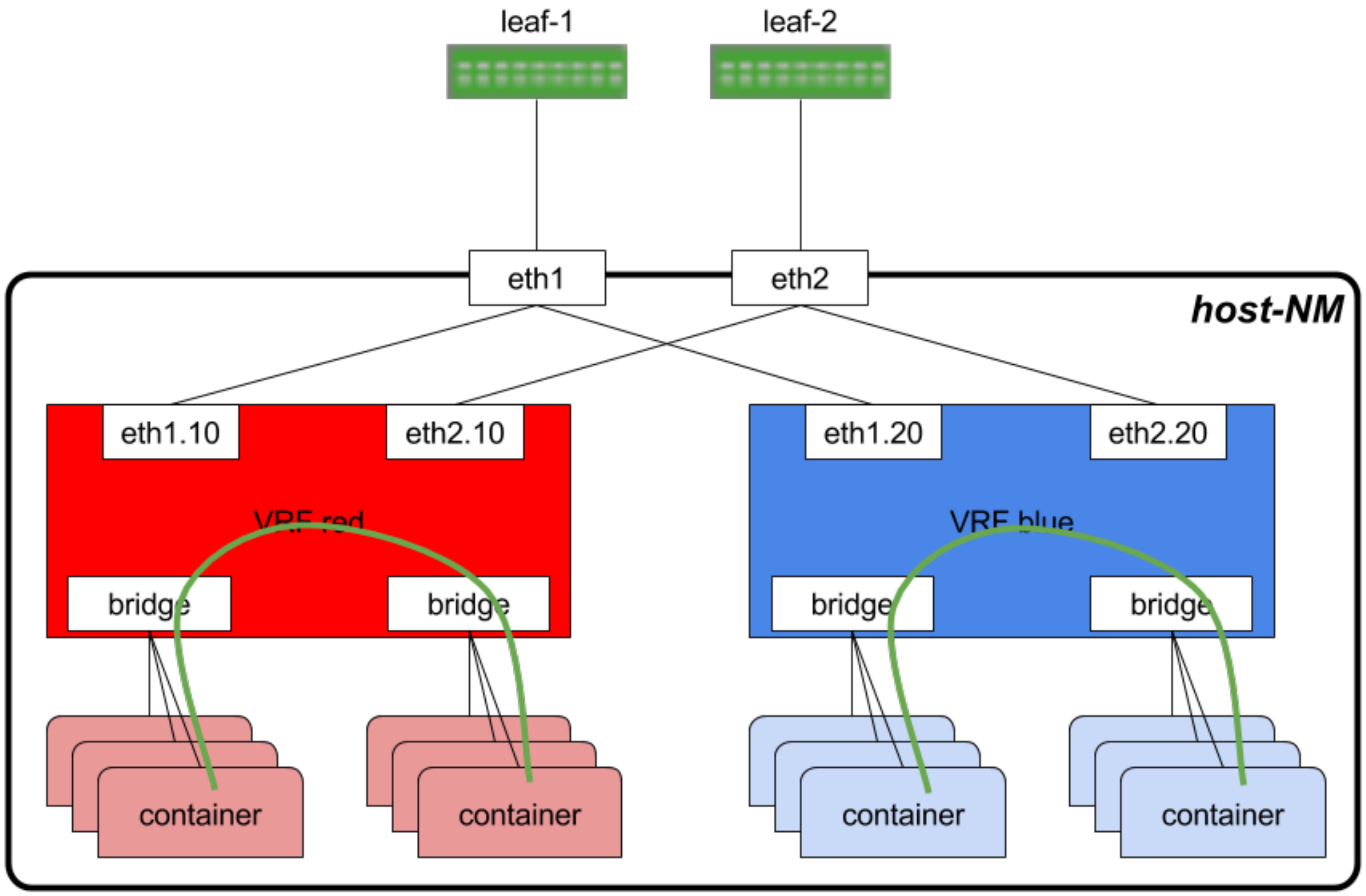
Readily scales out to more leafs, spines and VRFs



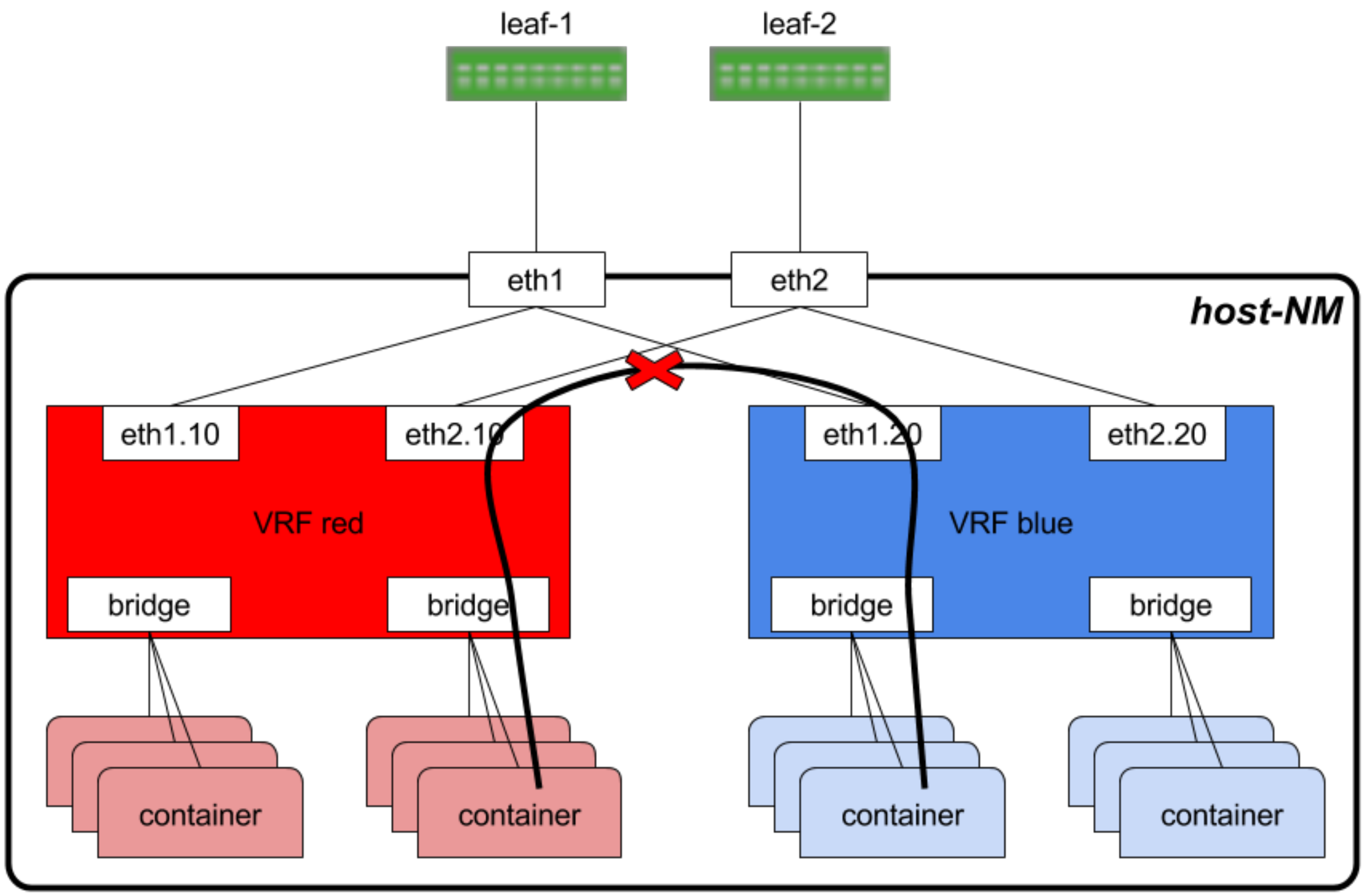
Host Networking Architecture



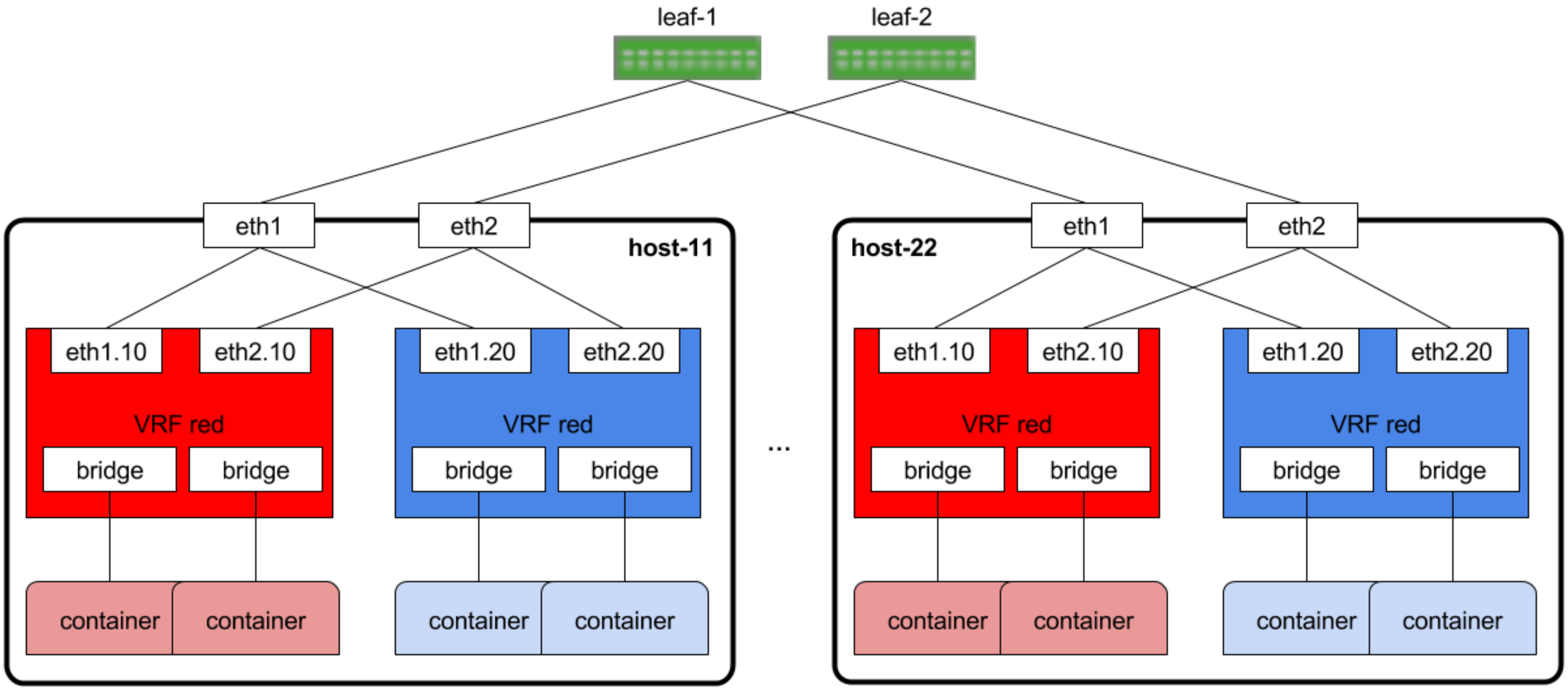
Host Networking - Intra-VRF allowed



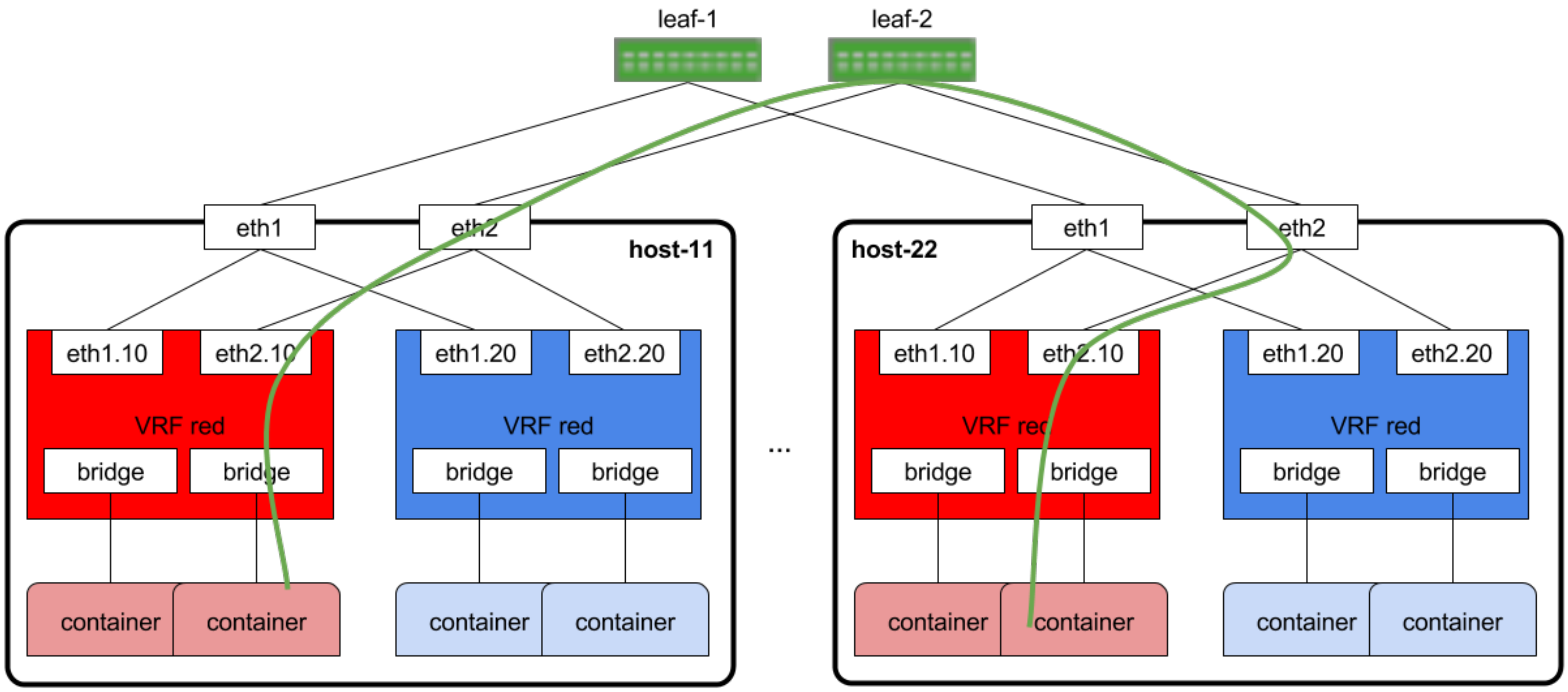
Host Networking - Cross VRF not allowed



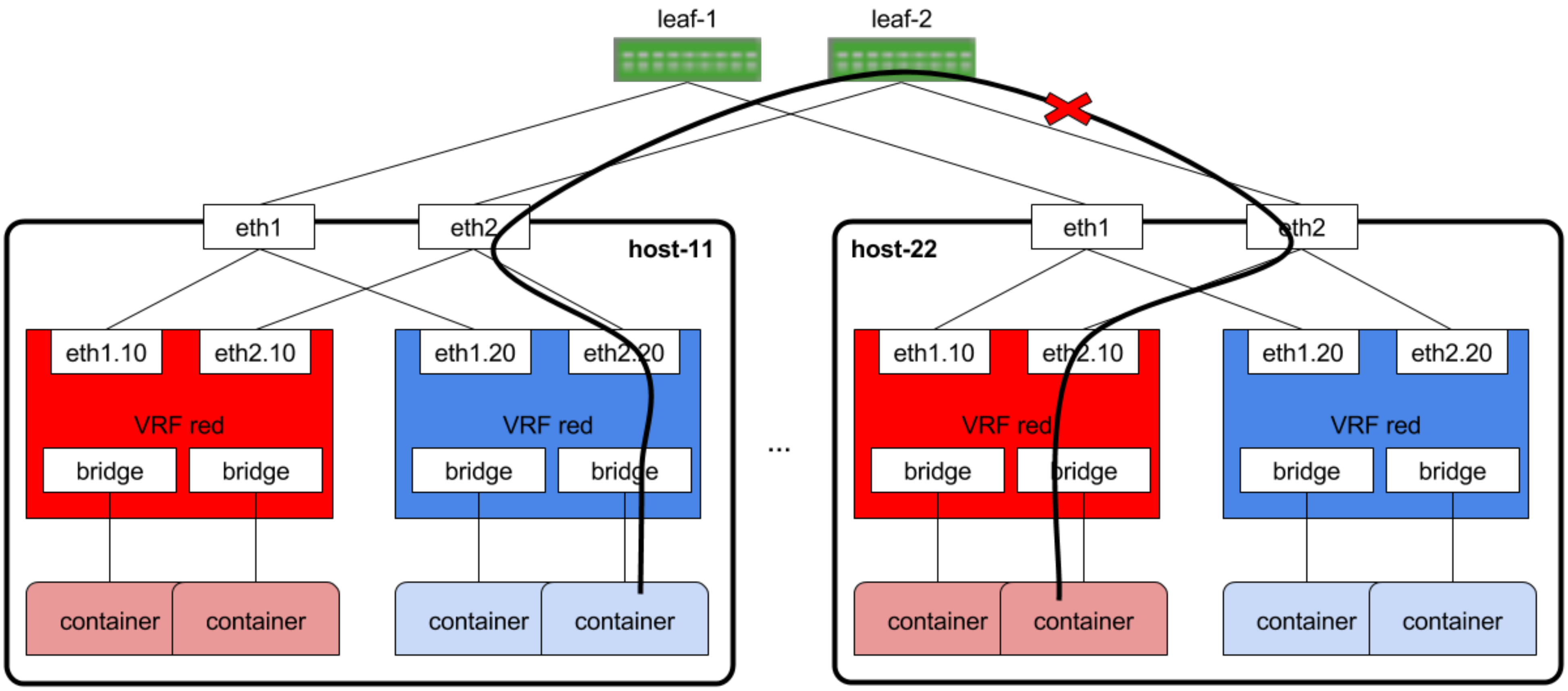
Multiple Host Networking



Multiple Host Networking - Intra-VRF allowed



Multiple Host Networking - Cross VRF not allowed



Spines and Leafs

Cumulus Linux 3.1

- No modifications

Vagrant box image

Spines have no routes to distribute; reflectors only

Spine-Leaf uses BGP unnumbered

Hosts

Ubuntu 16.04 - first release with VRF support

- 4.4 kernel
- No changes made to kernel; leveraging what exists
- Debian Stretch (4.6 kernel), Ubuntu 16.10 (4.8 ??)

Hosts

Ubuntu 16.04 - first release with VRF support

- 4.4 kernel
- No changes made to kernel; leveraging what exists
- Debian Stretch (4.6 kernel), Ubuntu 16.10 (4.8 ??)

Software add-ons to stock image

- ifupdown2 interface manager
- docker, experimental image

ifupdown2 Interface Manager

VRF Support

- Define/use VRF in /etc/network/interfaces

```
auto red
iface red
   vrf-table 1001
   up ip ro add table 1001 unreachable default metric 8192
```

- Add ‘vrf <name>’ to any iface stanza to add it to the VRF

https://support.cumulusnetworks.com/hc/en-us/articles/216130037-Using-ifupdown2-on-Ubuntu

ifupdown2 Interface Manager

VRF Support

- Define/use VRF in /etc/network/interfaces

```
auto red
iface red
    vrf-table 1001
    up ip ro add table 1001 unreachable default metric 8192
```

- Add ‘vrf <name>’ to any iface stanza to add it to the VRF

Using package built from github tree

- https://github.com/CumulusNetworks/ifupdown2
- Available via apt repositories as well

https://support.cumulusnetworks.com/hc/en-us/articles/216130037-Using-ifupdown2-on-Ubuntu

ifupdown2 Interface Manager

VRF Support

- Define/use VRF in /etc/network/interfaces

```
auto red
iface red
    vrf-table 1001
    up ip ro add table 1001 unreachable default metric 8192
```

- Add ‘vrf <name>’ to any iface stanza to add it to the VRF

Using package built from github tree

- https://github.com/CumulusNetworks/ifupdown2
- Available via apt repositories as well

Works with Debian and Ubuntu

https://support.cumulusnetworks.com/hc/en-us/articles/216130037-Using-ifupdown2-on-Ubuntu

Example ifupdown2 Configuration - Leafs

```
auto red
iface red
    vrf-table 1001

auto blue
iface blue
    vrf-table 1002


# leaf number
<% n = 1 %>
```

```
%for i in range(1,3):
auto swp${i}.10
iface swp${i}.10
    vrf red
%endfor

%for i in range(1,3):
auto swp${i}.20
iface swp${i}.20
    vrf blue
%endfor
```

```
%for i in range(3,7):
auto swp${i}.10
iface swp${i}.10
    address 10.1.${n}.${(i-3)*16}/31
    vrf red
%endfor

%for i in range(3,7):
auto swp${i}.20
iface swp${i}.20
    address 10.1.${n}.${(i-3)*16 + 2}/31
    vrf blue
%endfor
```

Host Networking

Cumulus Quagga in Docker container

- Container runs in privileged mode with host network
- Ease/consistency across OS’es; deb packages exist as well

Host Networking

Cumulus Quagga in Docker container

- Container runs in privileged mode with host network
- Ease/consistency across OS'es; deb packages exist as well

ECMP default route to each leaf in each VRF

- Installed by quagga, learned from leafs

Host Networking

Cumulus Quagga in Docker container

- Container runs in privileged mode with host network
- Ease/consistency across OS’es; deb packages exist as well

ECMP default route to each leaf in each VRF

- Installed by quagga, learned from leafs

Container networks distributed to leafs

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

Leaf-Host Configuration

VRF support in 4.4 kernel does not handle IPv6 linklocal addresses

- Can not use BGP unnumbered
- An option for 4.8 kernel and up

/31 addresses on host-leaf ports

Separate addresses for each VLAN interface

Container Technology

Using Docker as an example

Works for any container or VM

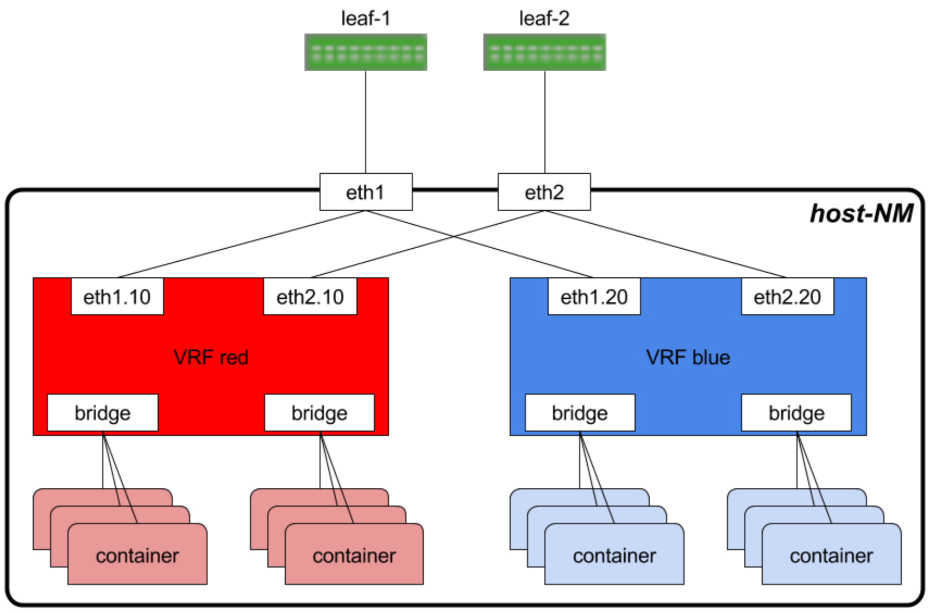
Scenario 1: Container networking via bridges

Container Networking with Bridges

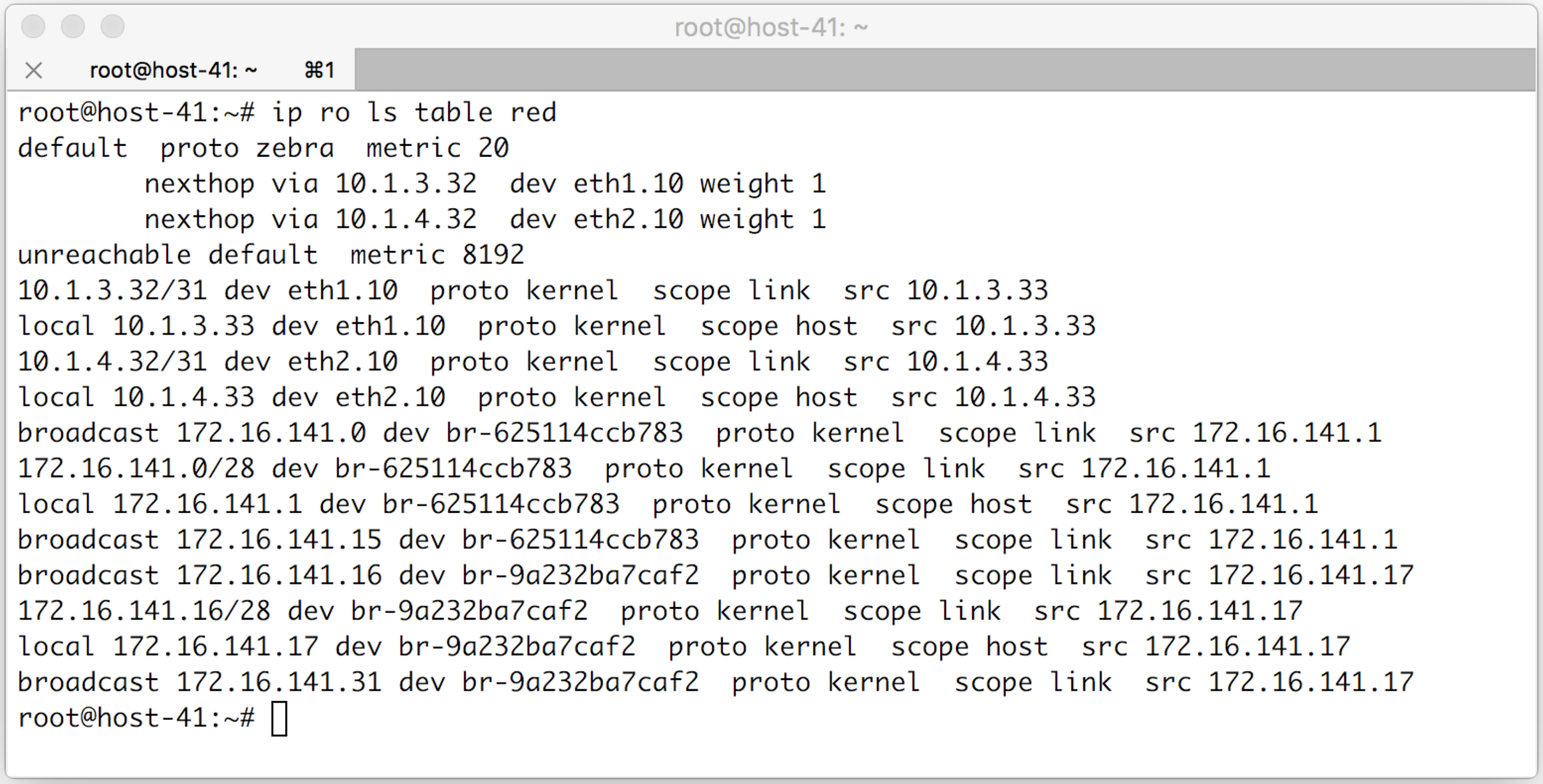
Typical use of Docker's bridge driver

- Create bridge with subnet allocation

Add Bridge to VRF



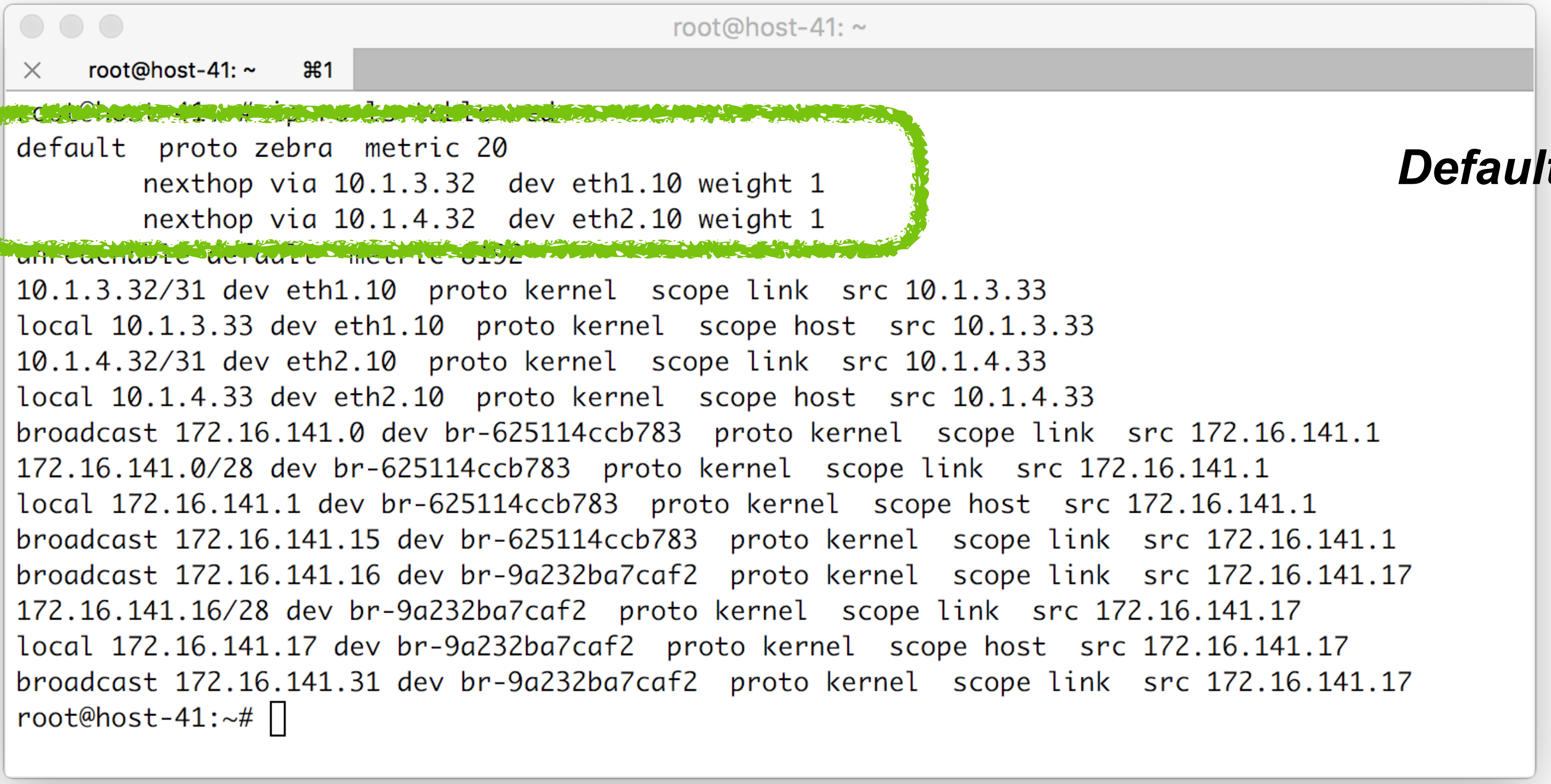
Example VRF Table on Host


root@host-41: ~

root@host-41: ~ ⌘1

```
root@host-41:~# ip ro ls table red
default  proto zebra  metric 20
        nexthop via 10.1.3.32  dev eth1.10 weight 1
        nexthop via 10.1.4.32  dev eth2.10 weight 1
unreachable default  metric 8192
10.1.3.32/31 dev eth1.10  proto kernel  scope link  src 10.1.3.33
local 10.1.3.33 dev eth1.10  proto kernel  scope host  src 10.1.3.33
10.1.4.32/31 dev eth2.10  proto kernel  scope link  src 10.1.4.33
local 10.1.4.33 dev eth2.10  proto kernel  scope host  src 10.1.4.33
broadcast 172.16.141.0 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
172.16.141.0/28 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
local 172.16.141.1 dev br-625114ccb783  proto kernel  scope host  src 172.16.141.1
broadcast 172.16.141.15 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
broadcast 172.16.141.16 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
172.16.141.16/28 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
local 172.16.141.17 dev br-9a232ba7caf2  proto kernel  scope host  src 172.16.141.17
broadcast 172.16.141.31 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
root@host-41:~# 
```

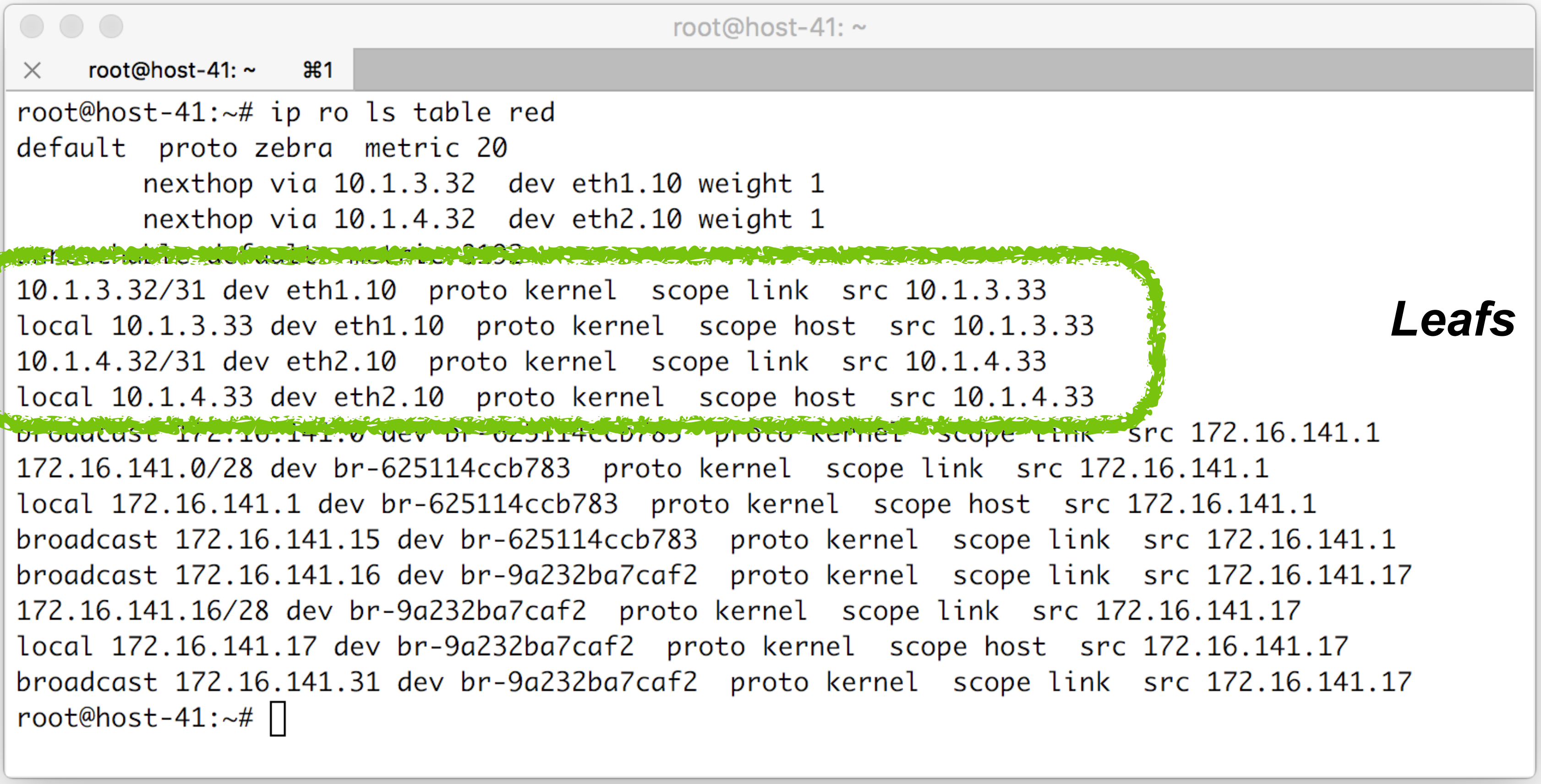
Example Host VRF Table



```
root@host-41:~# ip ro ls table red
default  proto zebra  metric 20
        nexthop via 10.1.3.32  dev eth1.10 weight 1
        nexthop via 10.1.4.32  dev eth2.10 weight 1
unreachable default  metric 8192
10.1.3.32/31 dev eth1.10  proto kernel  scope link  src 10.1.3.33
local 10.1.3.33 dev eth1.10  proto kernel  scope host  src 10.1.3.33
10.1.4.32/31 dev eth2.10  proto kernel  scope link  src 10.1.4.33
local 10.1.4.33 dev eth2.10  proto kernel  scope host  src 10.1.4.33
broadcast 172.16.141.0 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
172.16.141.0/28 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
local 172.16.141.1 dev br-625114ccb783  proto kernel  scope host  src 172.16.141.1
broadcast 172.16.141.15 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
broadcast 172.16.141.16 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
172.16.141.16/28 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
local 172.16.141.17 dev br-9a232ba7caf2  proto kernel  scope host  src 172.16.141.17
broadcast 172.16.141.31 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
root@host-41:~# 
```

Default route

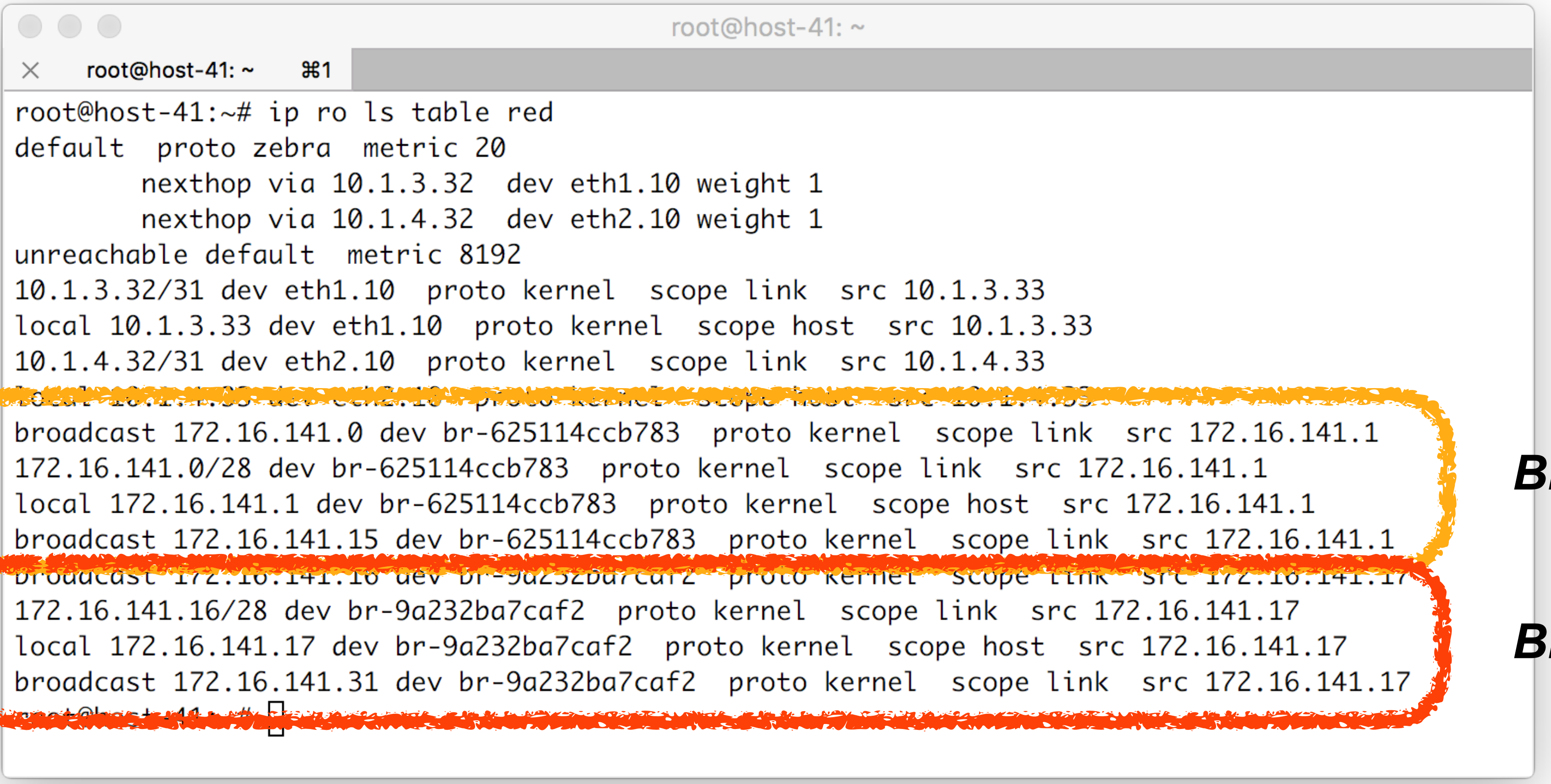
Example Host VRF Table



```
root@host-41:~# ip ro ls table red
default  proto zebra  metric 20
        nexthop via 10.1.3.32  dev eth1.10 weight 1
        nexthop via 10.1.4.32  dev eth2.10 weight 1
unreachable default  metric 8192
10.1.3.32/31 dev eth1.10  proto kernel  scope link  src 10.1.3.33
local 10.1.3.33 dev eth1.10  proto kernel  scope host  src 10.1.3.33
10.1.4.32/31 dev eth2.10  proto kernel  scope link  src 10.1.4.33
local 10.1.4.33 dev eth2.10  proto kernel  scope host  src 10.1.4.33
broadcast 172.16.141.0 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
172.16.141.0/28 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
local 172.16.141.1 dev br-625114ccb783  proto kernel  scope host  src 172.16.141.1
broadcast 172.16.141.15 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
broadcast 172.16.141.16 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
172.16.141.16/28 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
local 172.16.141.17 dev br-9a232ba7caf2  proto kernel  scope host  src 172.16.141.17
broadcast 172.16.141.31 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
root@host-41:~#
```

Leafs

Example Host VRF Table



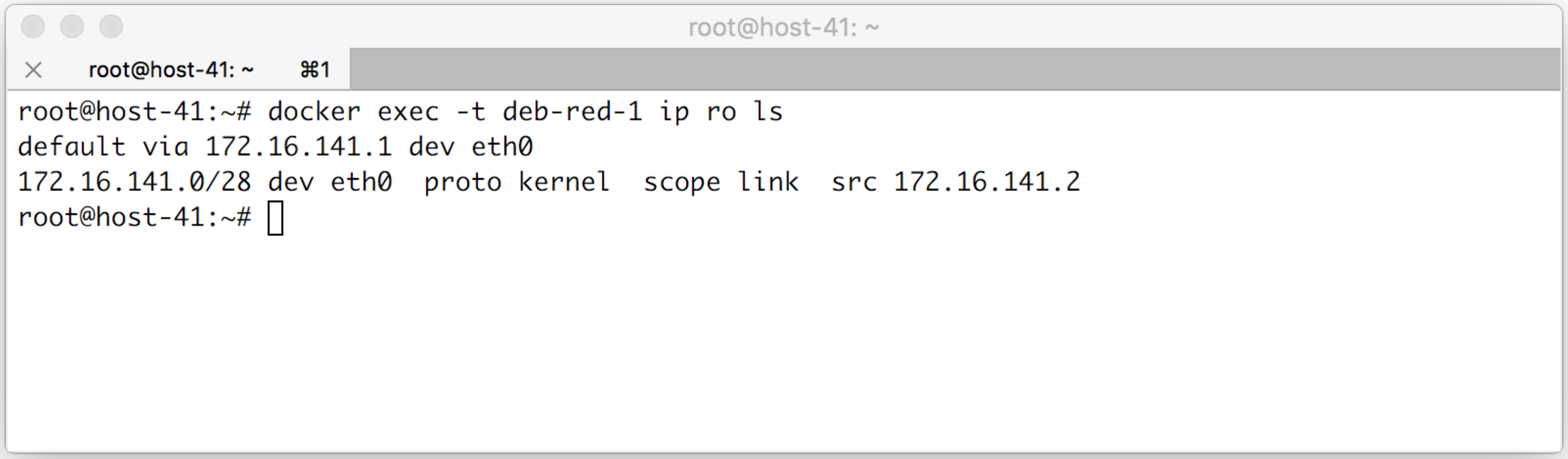
```
root@host-41:~# ip ro ls table red
default  proto zebra  metric 20
        nexthop via 10.1.3.32  dev eth1.10 weight 1
        nexthop via 10.1.4.32  dev eth2.10 weight 1
unreachable default  metric 8192
10.1.3.32/31 dev eth1.10  proto kernel  scope link  src 10.1.3.33
local 10.1.3.33 dev eth1.10  proto kernel  scope host  src 10.1.3.33
10.1.4.32/31 dev eth2.10  proto kernel  scope link  src 10.1.4.33
local 10.1.4.33 dev eth2.10  proto kernel  scope host  src 10.1.4.33
broadcast 172.16.141.0 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
172.16.141.0/28 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
local 172.16.141.1 dev br-625114ccb783  proto kernel  scope host  src 172.16.141.1
broadcast 172.16.141.15 dev br-625114ccb783  proto kernel  scope link  src 172.16.141.1
broadcast 172.16.141.16 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
172.16.141.16/28 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
local 172.16.141.17 dev br-9a232ba7caf2  proto kernel  scope host  src 172.16.141.17
broadcast 172.16.141.31 dev br-9a232ba7caf2  proto kernel  scope link  src 172.16.141.17
root@host-41:~# 
```

Bridge 1

Bridge 2

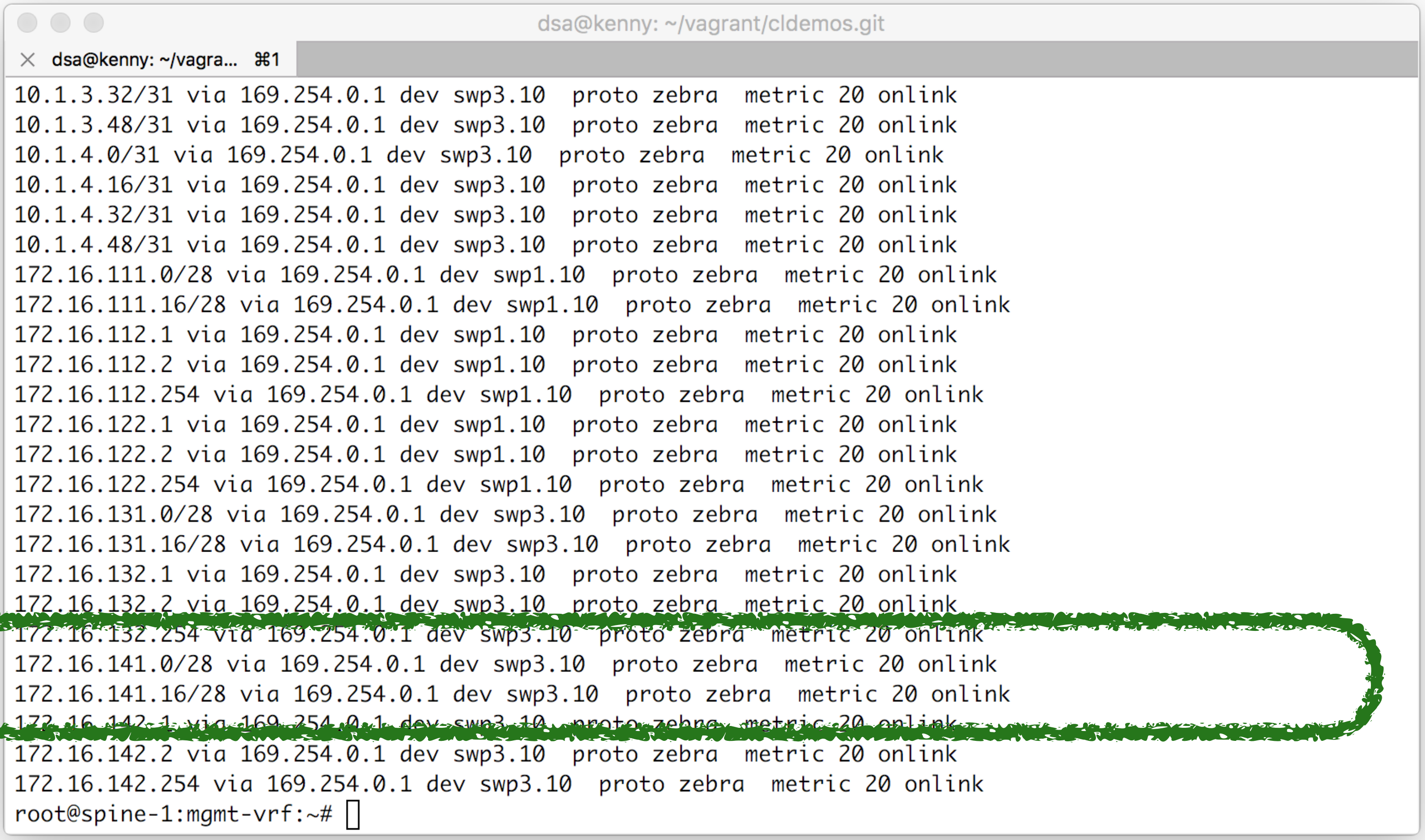
Example Container Routes

Containers have only connected route + default route



```
root@host-41:~# docker exec -t deb-red-1 ip ro ls
default via 172.16.141.1 dev eth0
172.16.141.0/28 dev eth0  proto kernel  scope link  src 172.16.141.2
root@host-41:~# 
```

Spine-Leaf Fabric Learns Container Routes



dsa@kenny: ~/vagrant/cldemos.git

dsa@kenny: ~/vagra... ⌘1

```
10.1.3.32/31 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
10.1.3.48/31 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
10.1.4.0/31 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
10.1.4.16/31 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
10.1.4.32/31 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
10.1.4.48/31 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.111.0/28 via 169.254.0.1 dev swp1.10  proto zebra  metric 20 onlink
172.16.111.16/28 via 169.254.0.1 dev swp1.10  proto zebra  metric 20 onlink
172.16.112.1 via 169.254.0.1 dev swp1.10  proto zebra  metric 20 onlink
172.16.112.2 via 169.254.0.1 dev swp1.10  proto zebra  metric 20 onlink
172.16.112.254 via 169.254.0.1 dev swp1.10  proto zebra  metric 20 onlink
172.16.122.1 via 169.254.0.1 dev swp1.10  proto zebra  metric 20 onlink
172.16.122.2 via 169.254.0.1 dev swp1.10  proto zebra  metric 20 onlink
172.16.122.254 via 169.254.0.1 dev swp1.10  proto zebra  metric 20 onlink
172.16.131.0/28 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.131.16/28 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.132.1 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.132.2 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.132.254 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.141.0/28 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.141.16/28 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.142.1 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.142.2 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
172.16.142.254 via 169.254.0.1 dev swp3.10  proto zebra  metric 20 onlink
root@spine-1:mgmt-vrf:~# 
```

Host-41 bridges

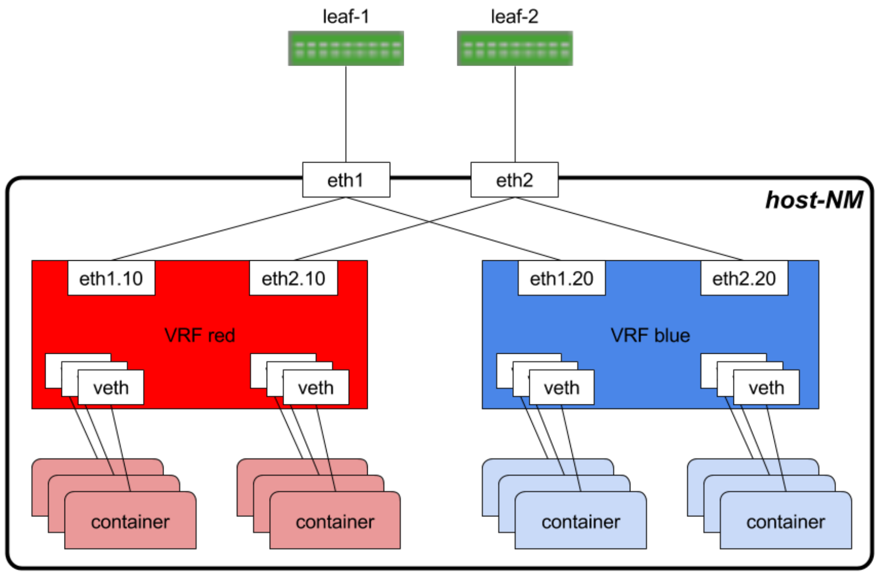
Scenario 2: Container networking with /32 routes

Container Networking with veth and /32 routes

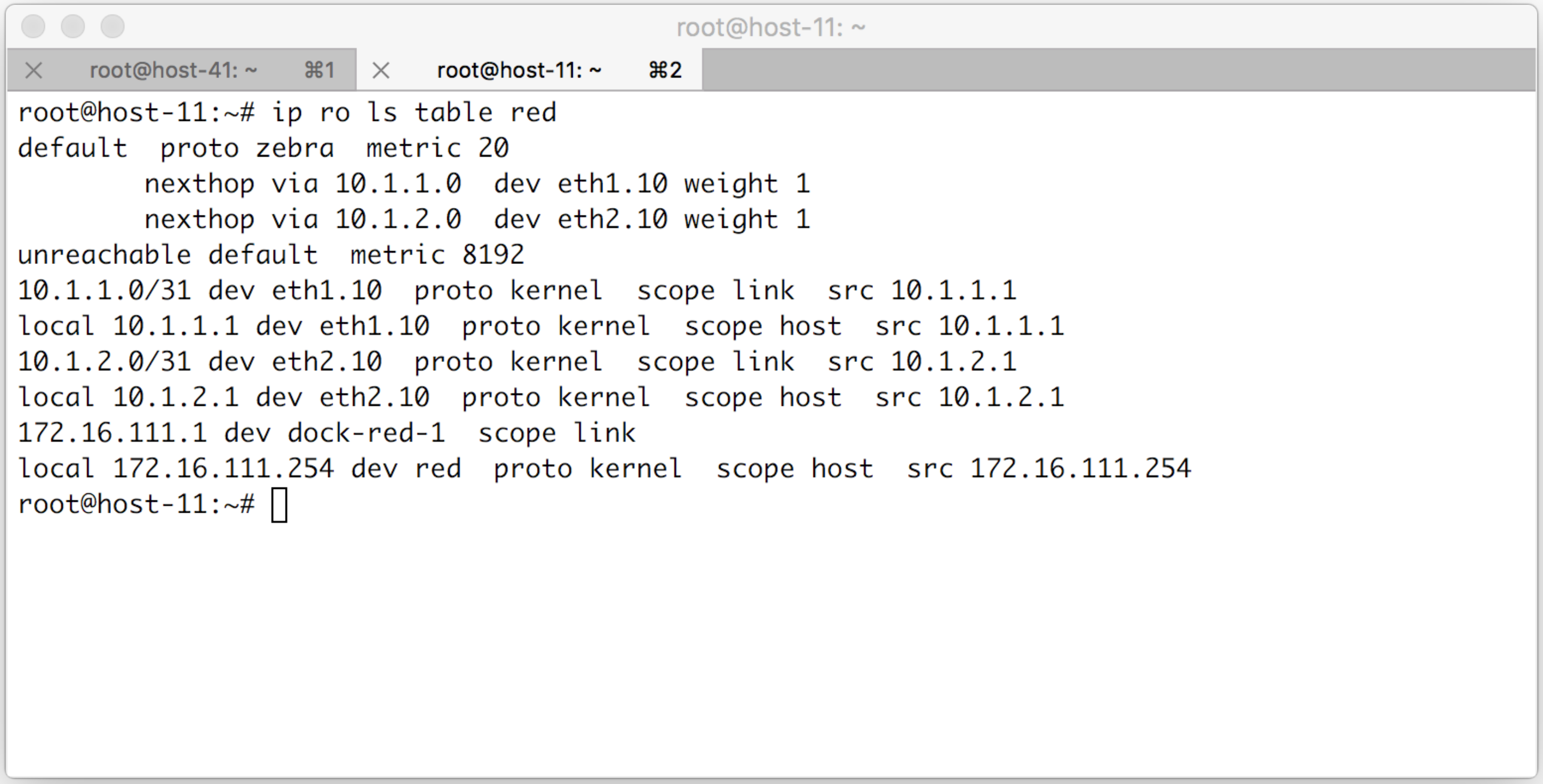
Docker network = none

Networking for container "manually" created after start

- /32 route added to VRF in host
- /32 addresss in container
- Default route passed to container
  - Limitation of VRF in v4.4

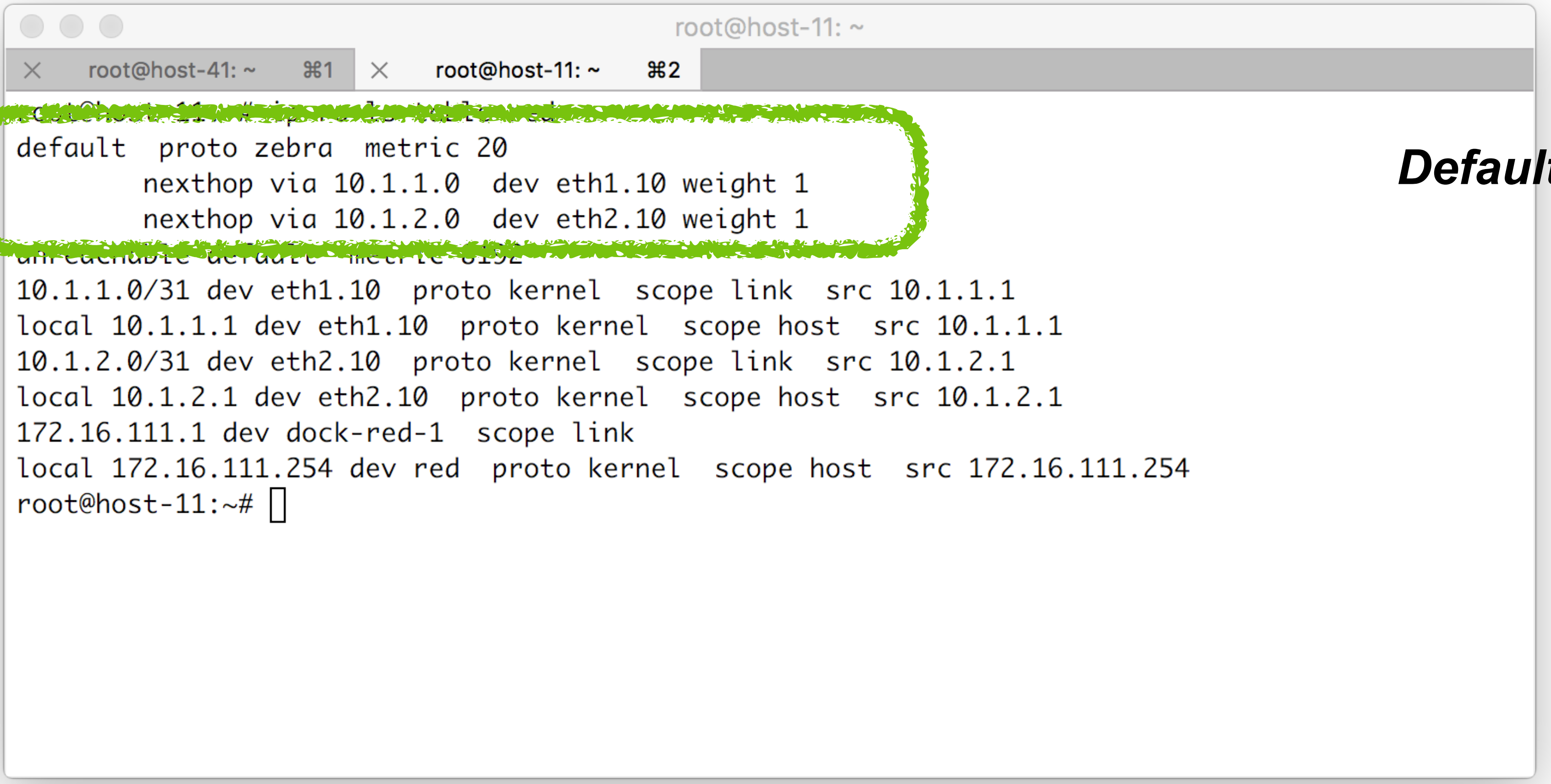


Example Host VRF Table



```
root@host-11:~# ip ro ls table red
default  proto zebra  metric 20
        nexthop via 10.1.1.0  dev eth1.10 weight 1
        nexthop via 10.1.2.0  dev eth2.10 weight 1
unreachable default  metric 8192
10.1.1.0/31 dev eth1.10  proto kernel  scope link  src 10.1.1.1
local 10.1.1.1 dev eth1.10  proto kernel  scope host  src 10.1.1.1
10.1.2.0/31 dev eth2.10  proto kernel  scope link  src 10.1.2.1
local 10.1.2.1 dev eth2.10  proto kernel  scope host  src 10.1.2.1
172.16.111.1 dev dock-red-1  scope link
local 172.16.111.254 dev red  proto kernel  scope host  src 172.16.111.254
root@host-11:~# 
```

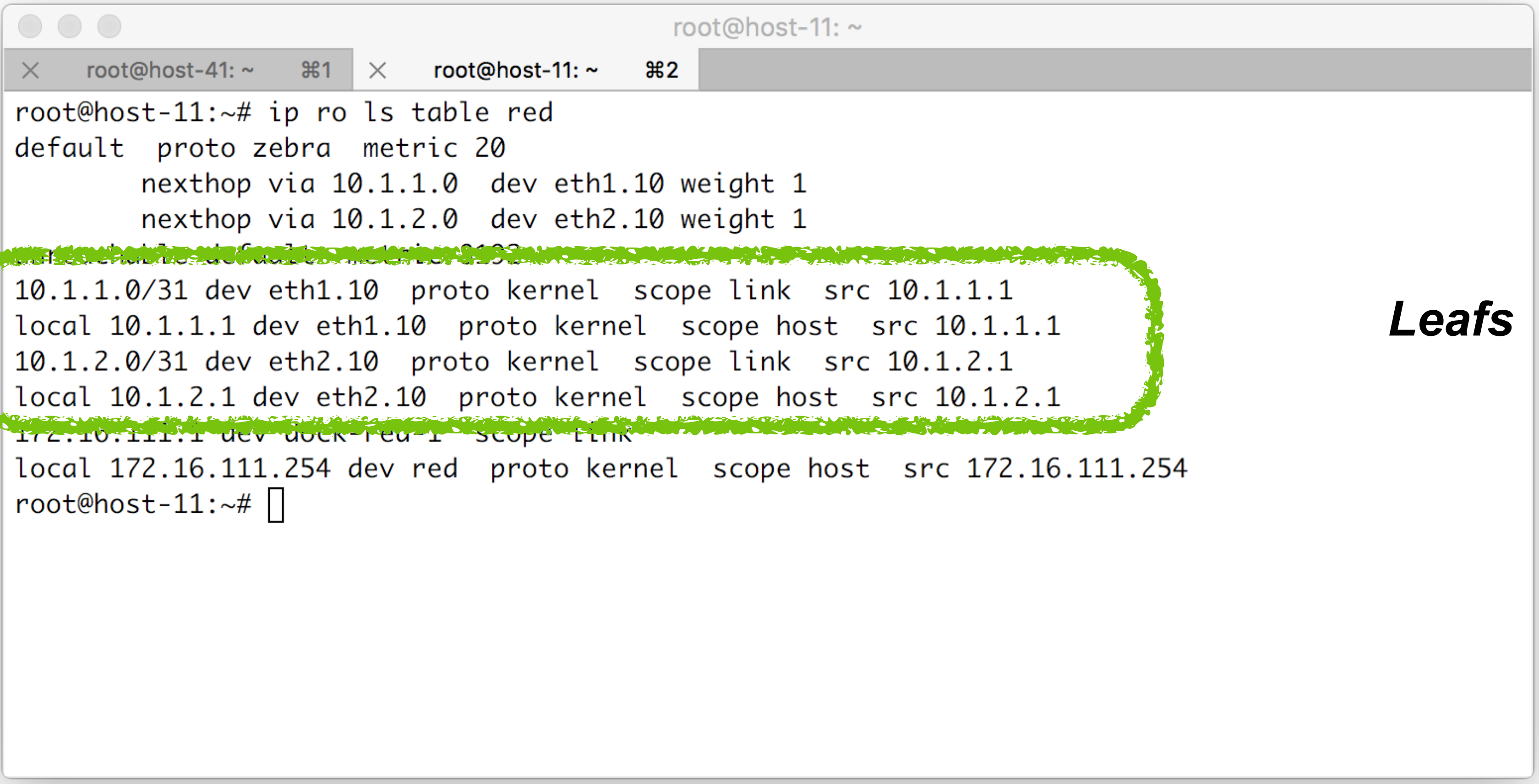
Example Host VRF Table



```
root@host-11:~# ip ro ls table red
default  proto zebra  metric 20
        nexthop via 10.1.1.0  dev eth1.10 weight 1
        nexthop via 10.1.2.0  dev eth2.10 weight 1
unreachable default  metric 8192
10.1.1.0/31 dev eth1.10  proto kernel  scope link  src 10.1.1.1
local 10.1.1.1 dev eth1.10  proto kernel  scope host  src 10.1.1.1
10.1.2.0/31 dev eth2.10  proto kernel  scope link  src 10.1.2.1
local 10.1.2.1 dev eth2.10  proto kernel  scope host  src 10.1.2.1
172.16.111.1 dev dock-red-1  scope link
local 172.16.111.254 dev red  proto kernel  scope host  src 172.16.111.254
root@host-11:~# 
```

Default route

Example Host VRF Table

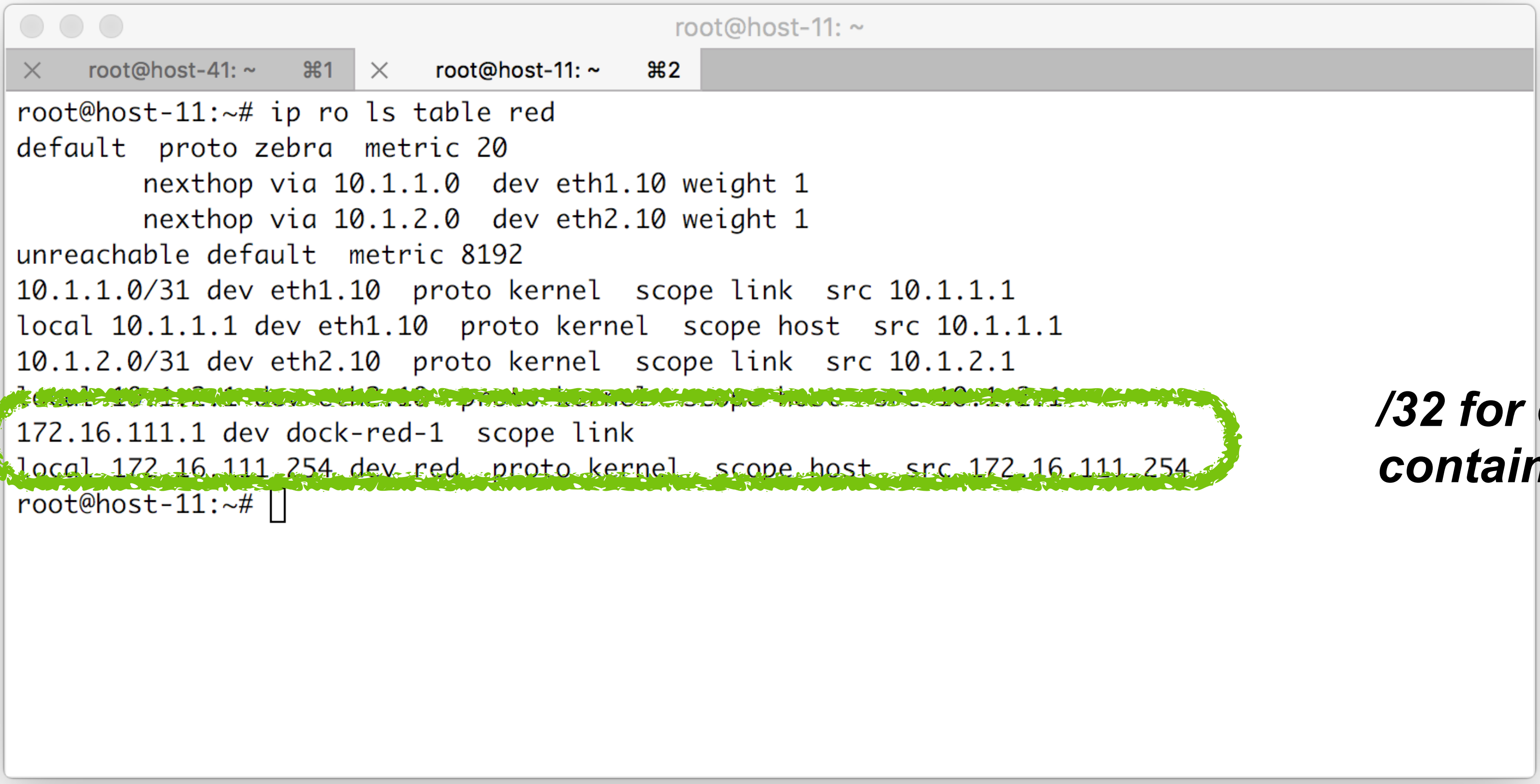


root@host-41: ~ ⌘1 | root@host-11: ~ ⌘2

```
root@host-11:~# ip ro ls table red
default  proto zebra  metric 20
        nexthop via 10.1.1.0  dev eth1.10 weight 1
        nexthop via 10.1.2.0  dev eth2.10 weight 1
unreachable default  metric 8192
10.1.1.0/31 dev eth1.10  proto kernel  scope link  src 10.1.1.1
local 10.1.1.1 dev eth1.10  proto kernel  scope host  src 10.1.1.1
10.1.2.0/31 dev eth2.10  proto kernel  scope link  src 10.1.2.1
local 10.1.2.1 dev eth2.10  proto kernel  scope host  src 10.1.2.1
172.16.111.1 dev dock-red-1  scope link
local 172.16.111.254 dev red  proto kernel  scope host  src 172.16.111.254
root@host-11:~# 
```

Leafs

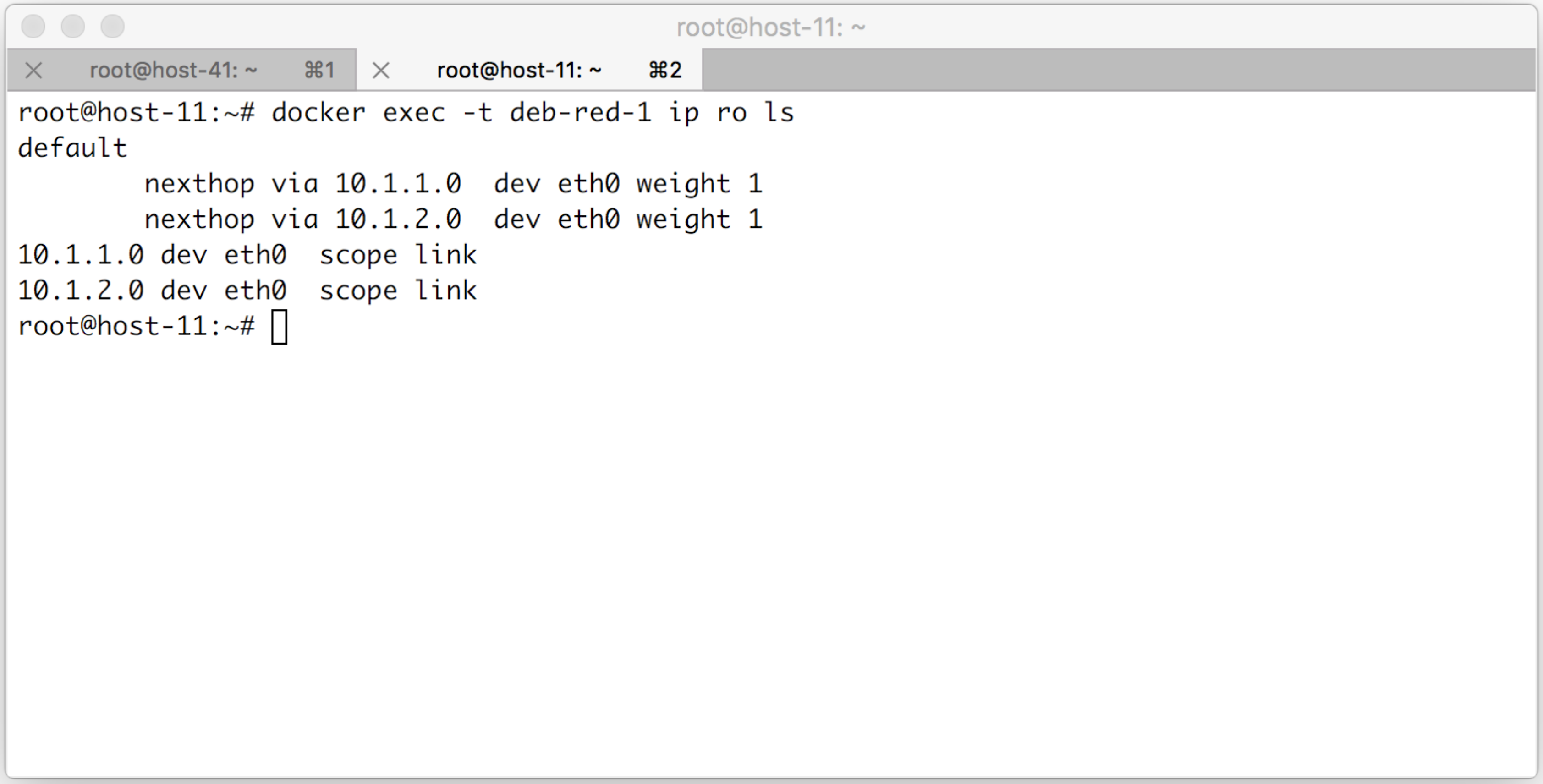
Example Host VRF Table



```
root@host-11:~# ip ro ls table red
default  proto zebra  metric 20
        nexthop via 10.1.1.0  dev eth1.10 weight 1
        nexthop via 10.1.2.0  dev eth2.10 weight 1
unreachable default  metric 8192
10.1.1.0/31 dev eth1.10  proto kernel  scope link  src 10.1.1.1
local 10.1.1.1 dev eth1.10  proto kernel  scope host  src 10.1.1.1
10.1.2.0/31 dev eth2.10  proto kernel  scope link  src 10.1.2.1
local 10.1.2.1 dev eth2.10  proto kernel  scope host  src 10.1.2.1
172.16.111.1 dev dock-red-1  scope link
local 172.16.111.254 dev red  proto kernel  scope host  src 172.16.111.254
root@host-11:~# 
```

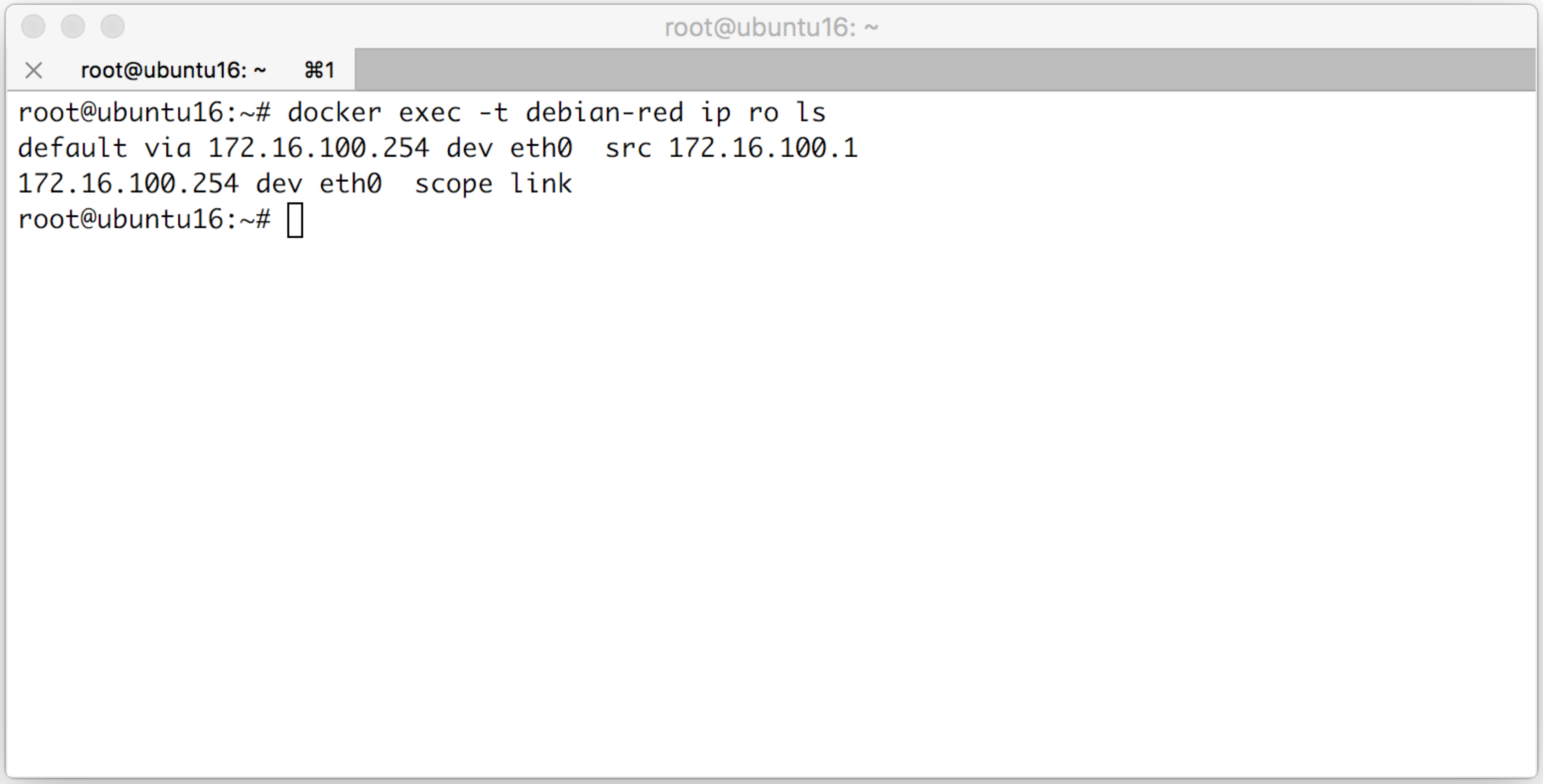
/32 for each container

Example Container Routes



```
root@host-11:~# docker exec -t deb-red-1 ip ro ls
default
        nexthop via 10.1.1.0  dev eth0 weight 1
        nexthop via 10.1.2.0  dev eth0 weight 1
10.1.1.0 dev eth0  scope link
10.1.2.0 dev eth0  scope link
root@host-11:~# 
```

Example Container Routes - v4.8 kernel



```
root@ubuntu16:~# docker exec -t debian-red ip ro ls
default via 172.16.100.254 dev eth0  src 172.16.100.1
172.16.100.254 dev eth0  scope link
root@ubuntu16:~#
```

Demonstration

Vagrant used for topology orchestration

Ansible for configuring the nodes

Files available from github:

https://github.com/dsahern/cldemos/tree/roh-vrf-netdev-1.2

Vagrant, ansible, ifupdown2, quagga and docker scripts

Q & A

Unleashing the Power of Open Networking

Thank You!