Container Networking Use Cases

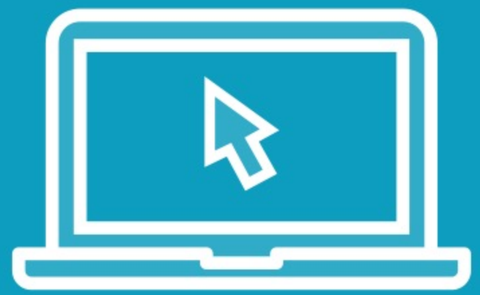
Nigel Poulton
Author & Trainer

@nigelpoulton nigelpoulton.com

Overview

- Single-host Bridge Networks
- Multi-host Overlay Networks
- Connecting to Existing Networks
  - MACVLAN
  - IPVLAN
- Recap

Demos



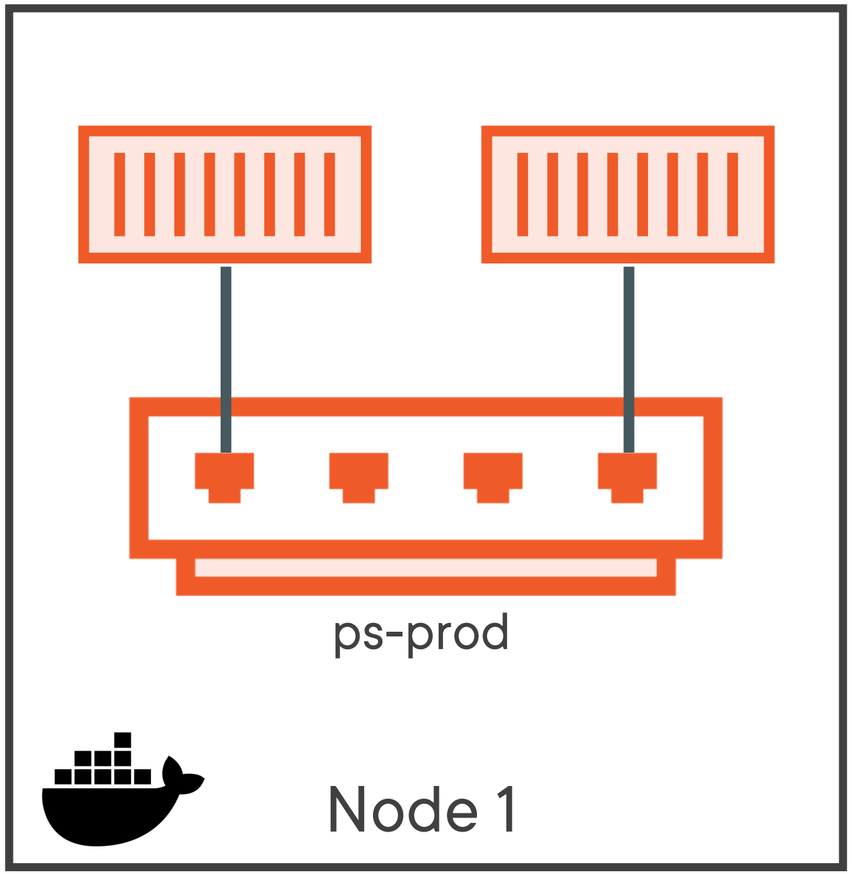
Lab environments

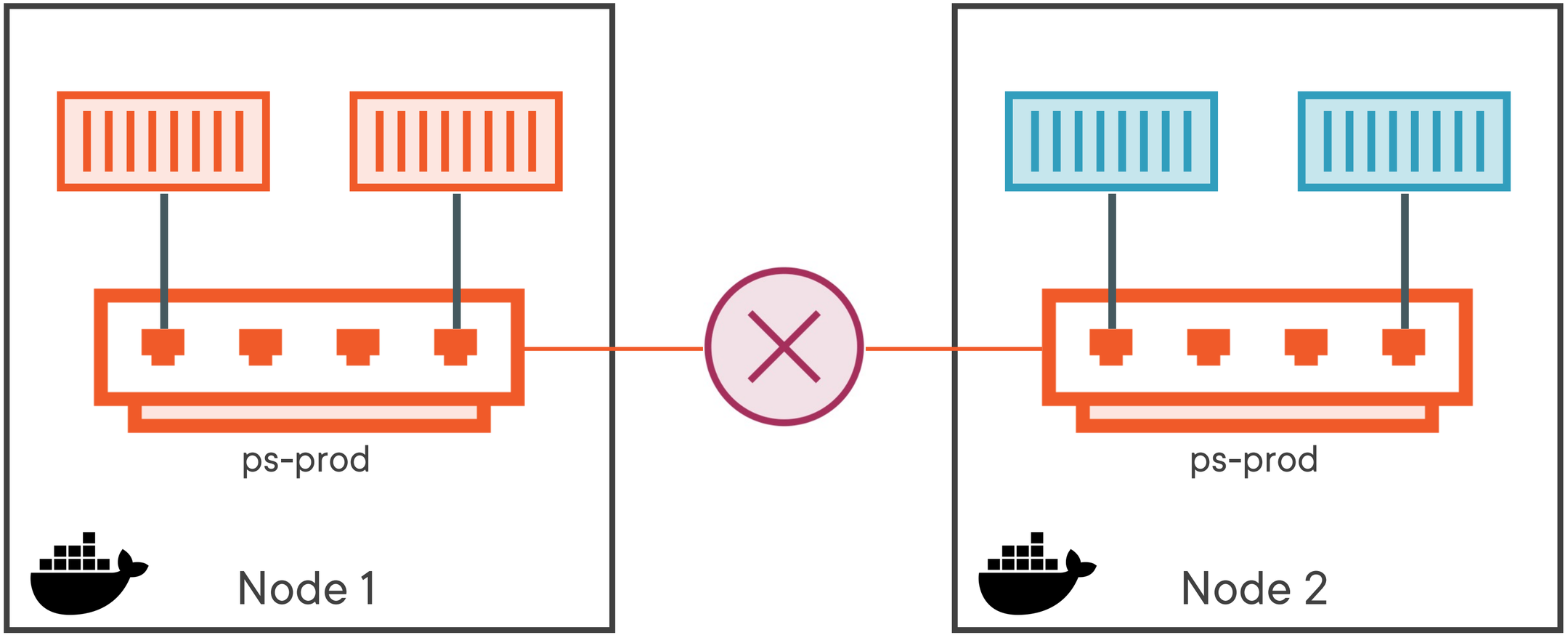
- Docker Desktop
- Cloud

Examples will be Linux

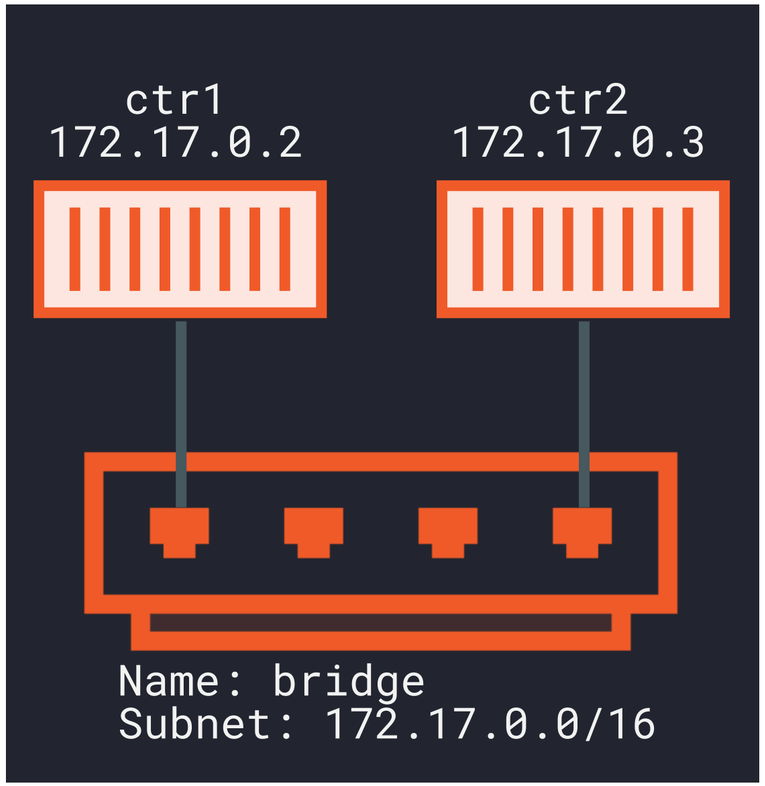
- Switch Docker Desktop on Windows into *Linux Containers mode*

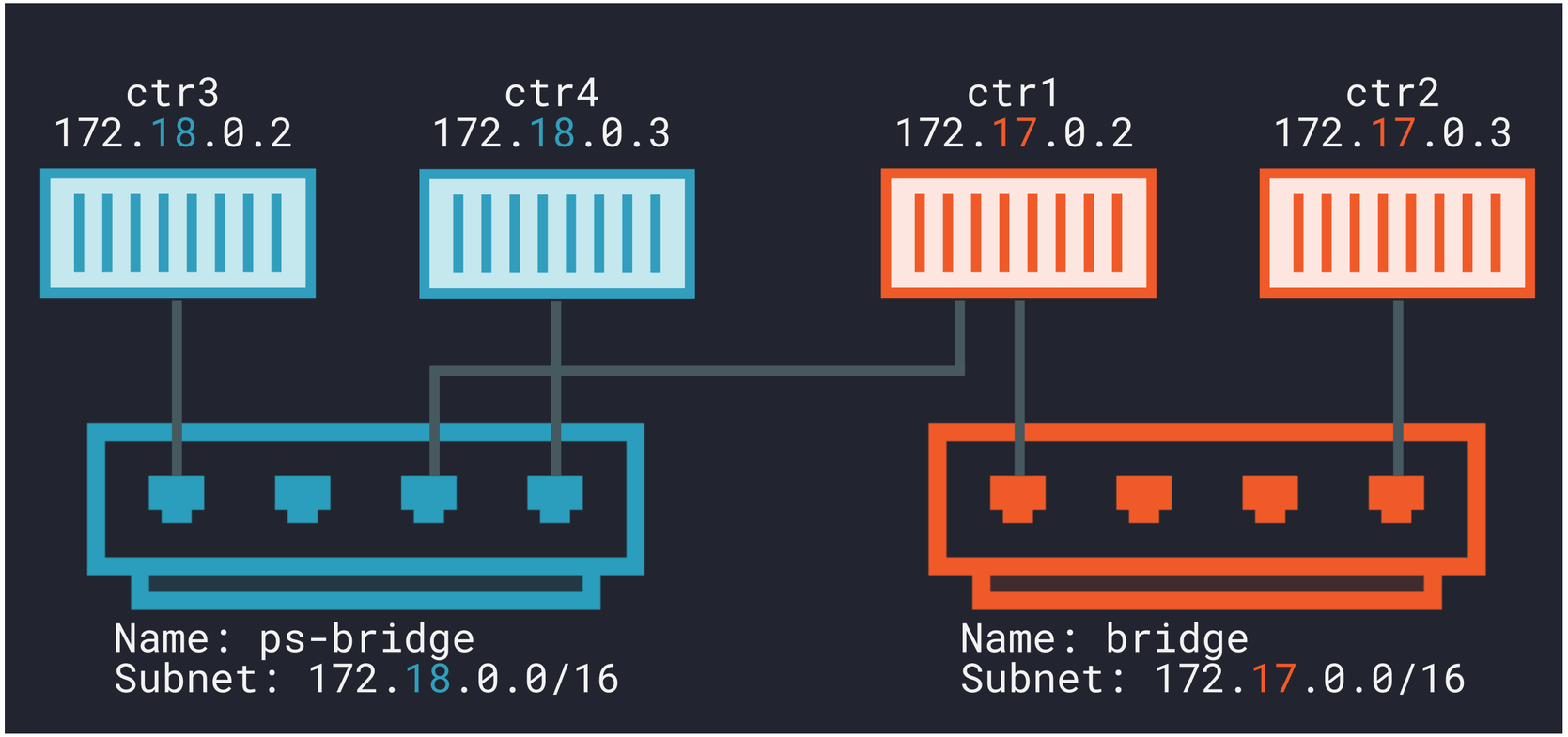
Single-host Bridge Networks

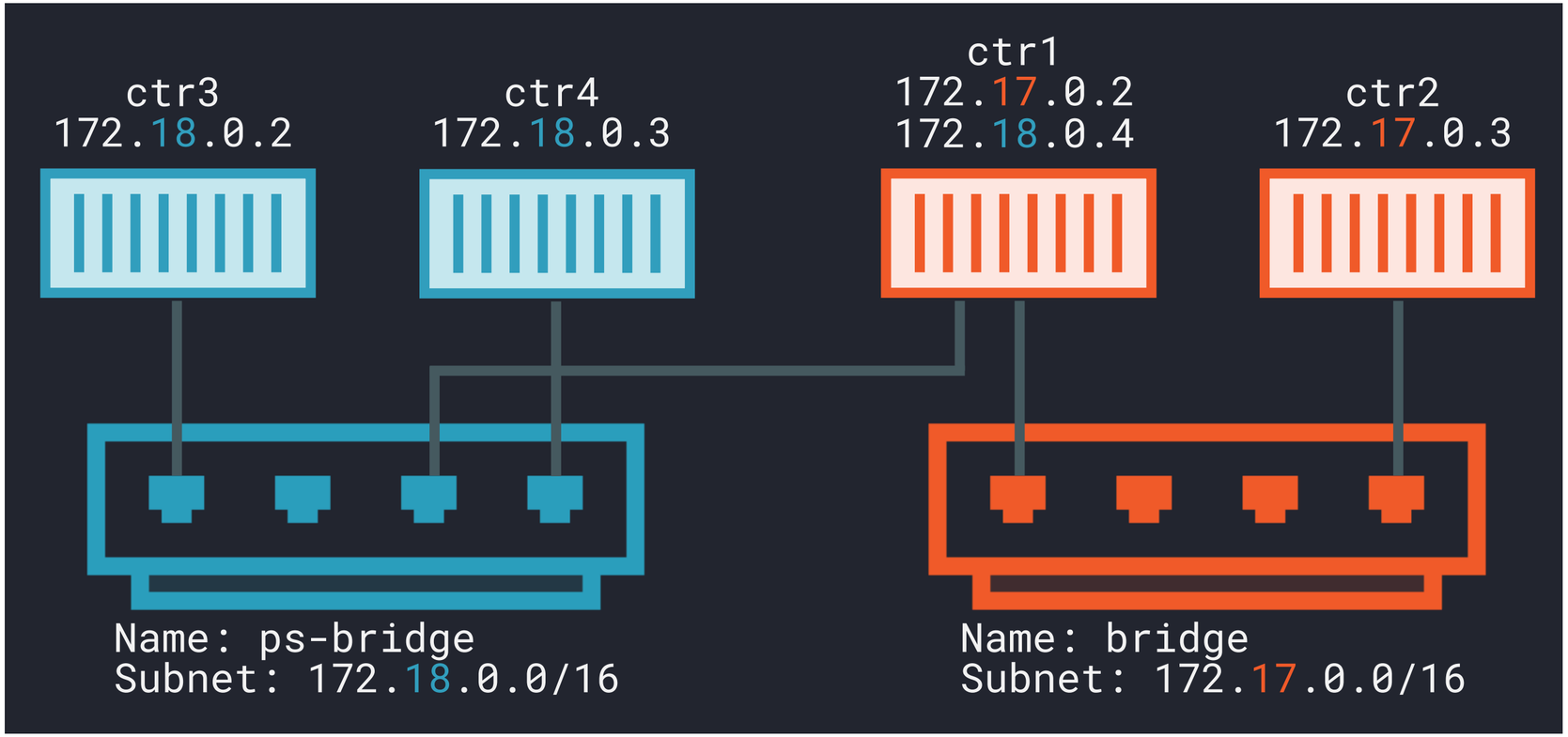

ps-prod
Node 1

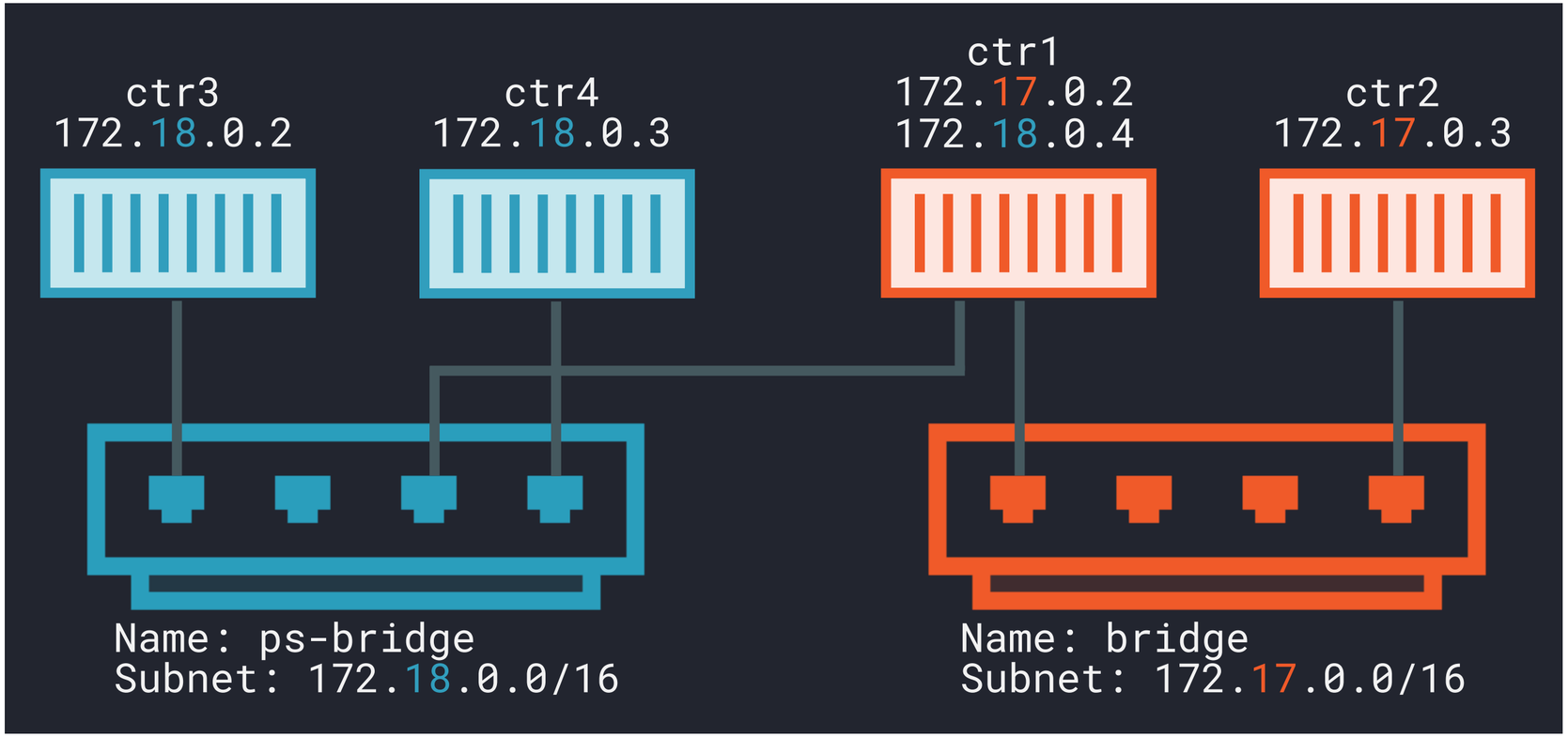


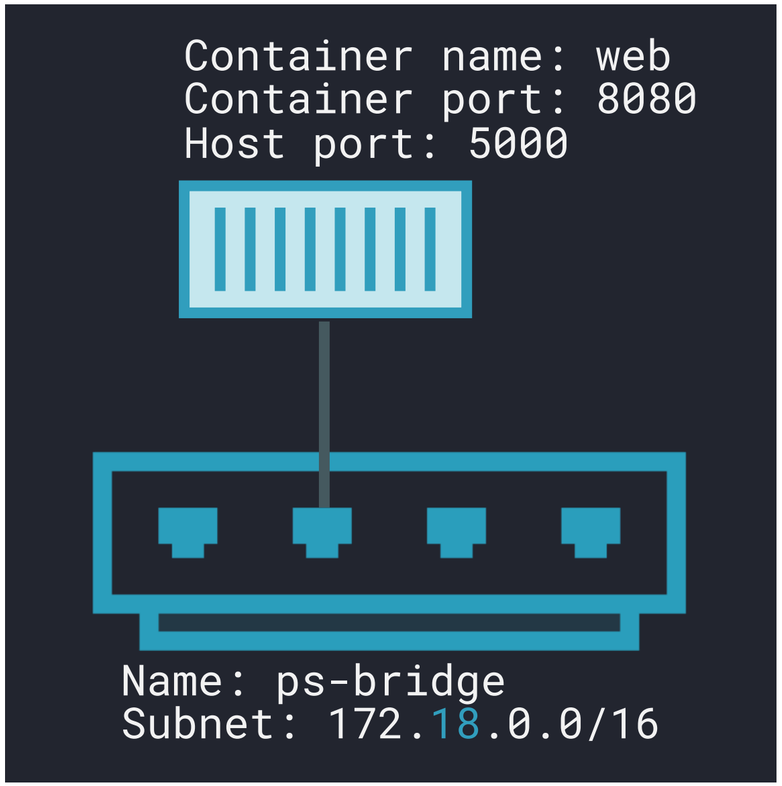

```
NETWORK ID     NAME      DRIVER    SCOPE
f9e9b3c13e37   bridge    bridge    local
```


ctr1
172.17.0.2
ctr2
172.17.0.3
Name: bridge
Subnet: 172.17.0.0/16

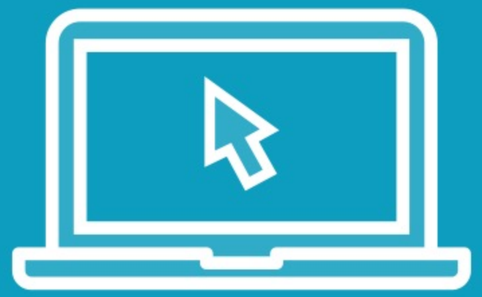

ctr3
172.18.0.2
ctr4
172.18.0.3
ctr1
172.17.0.2
ctr2
172.17.0.3
Name: ps-bridge
Subnet: 172.18.0.0/16
Name: bridge
Subnet: 172.17.0.0/16


ctr3
172.18.0.2
ctr4
172.18.0.3
ctr1
172.17.0.2
172.18.0.4
ctr2
172.17.0.3
Name: ps-bridge
Subnet: 172.18.0.0/16
Name: bridge
Subnet: 172.17.0.0/16


ctr3
172.18.0.2
ctr4
172.18.0.3
ctr1
172.17.0.2
172.18.0.4
ctr2
172.17.0.3
Name: ps-bridge
Subnet: 172.18.0.0/16
Name: bridge
Subnet: 172.17.0.0/16


Container name: web
Container port: 8080
Host port: 5000
Name: ps-bridge
Subnet: 172.18.0.0/16

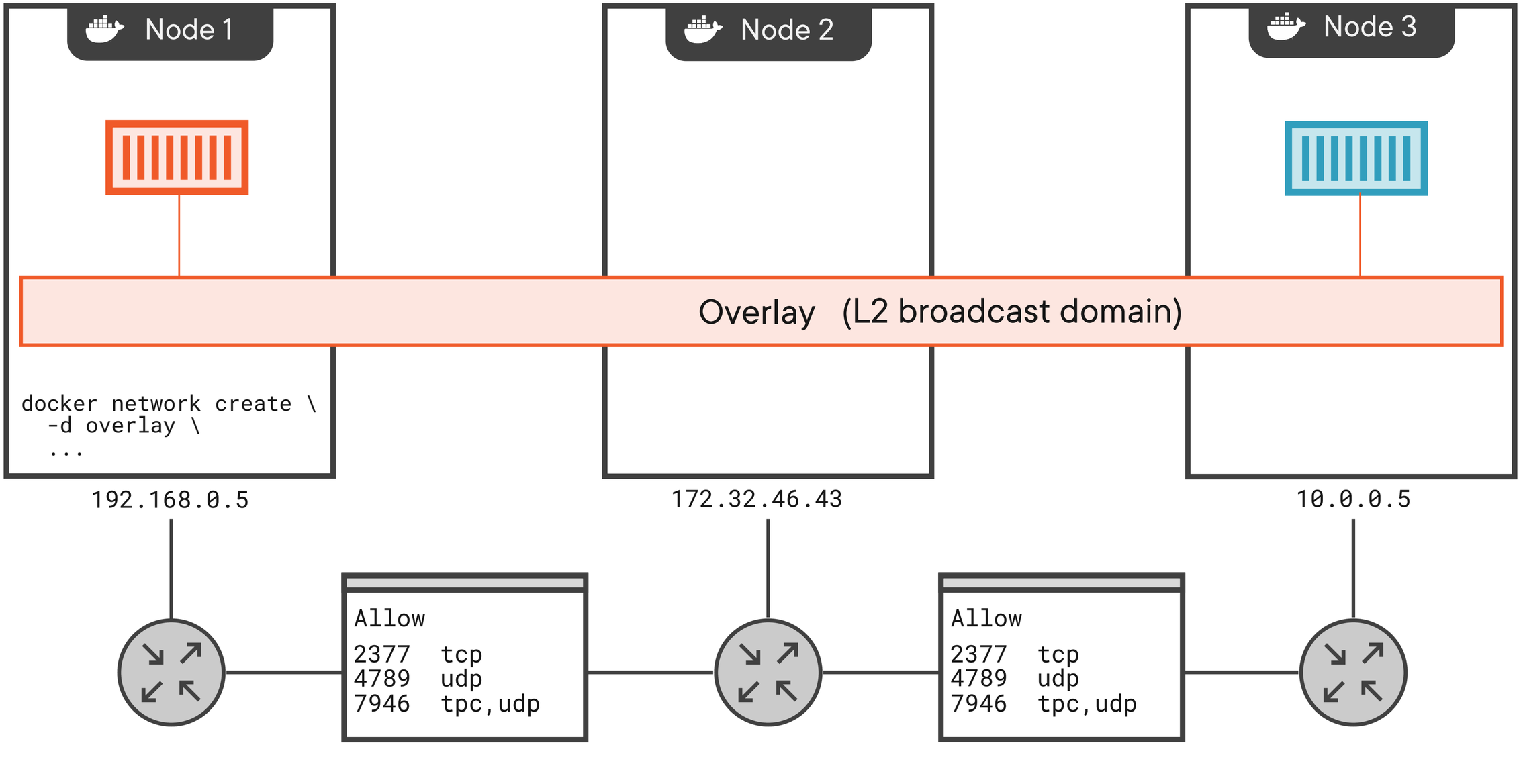
Demos

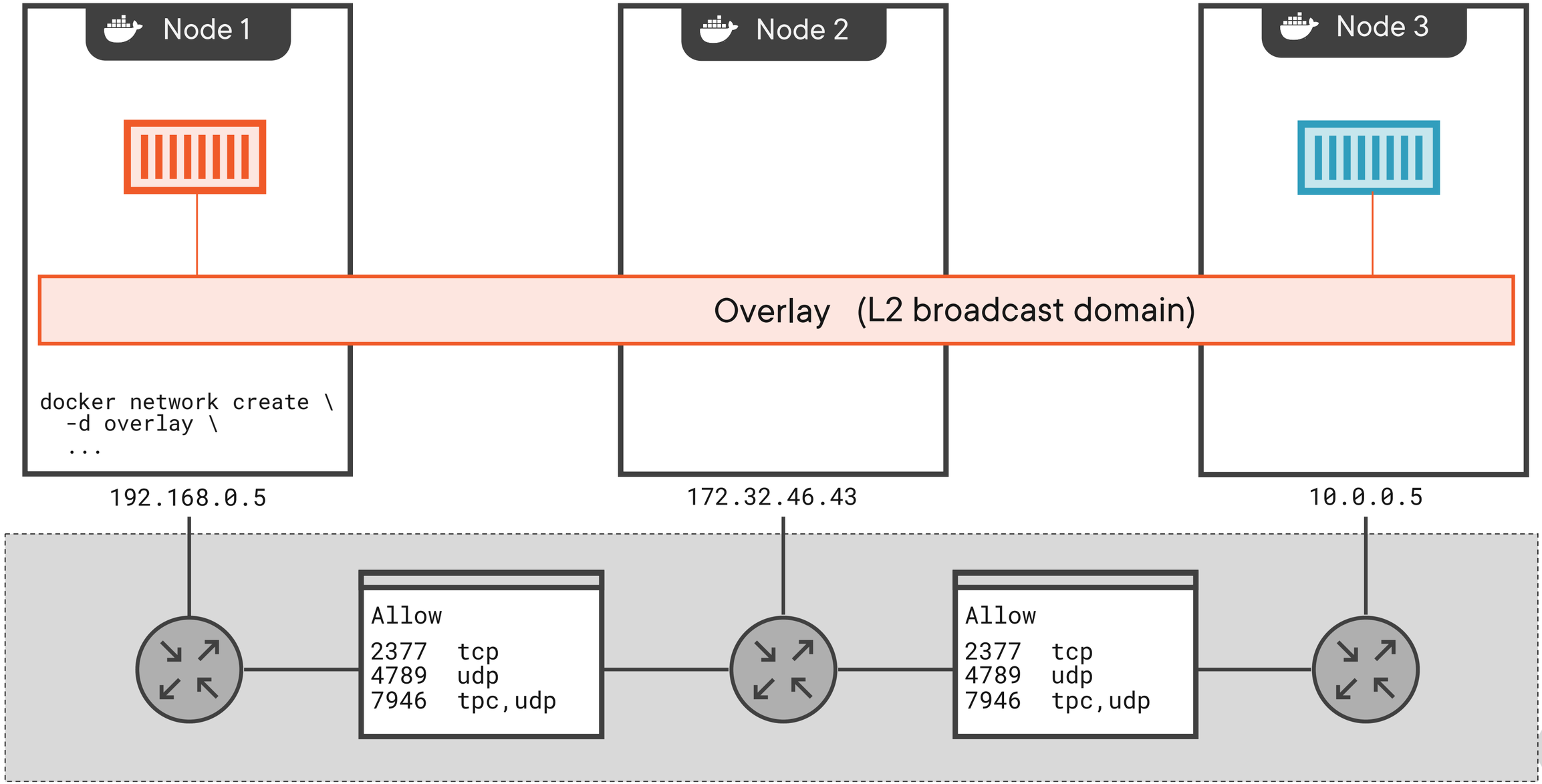


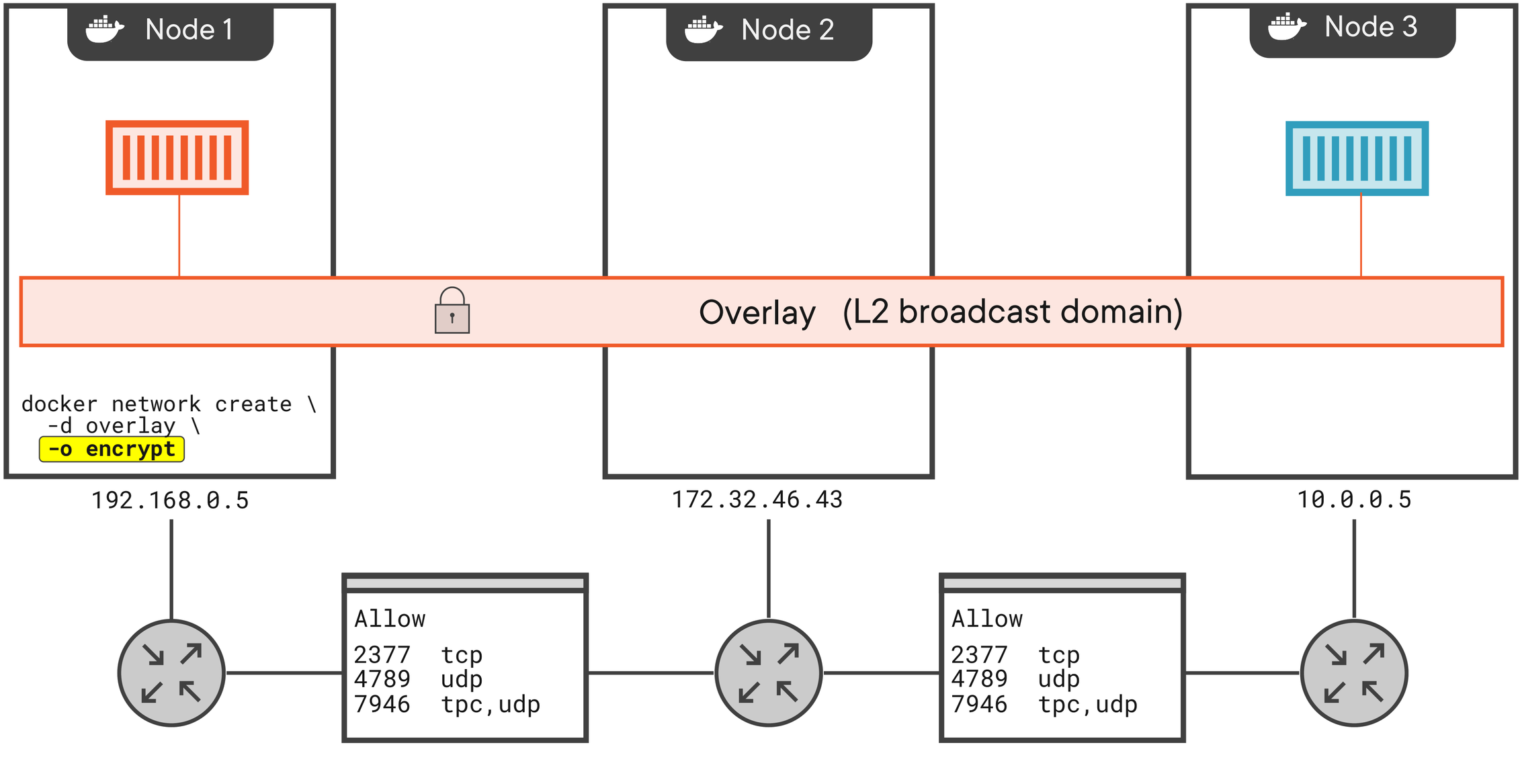
- Test the default bridge network
- Test a user-defined bridge network
- Test a dual-homed container
- Expose an app

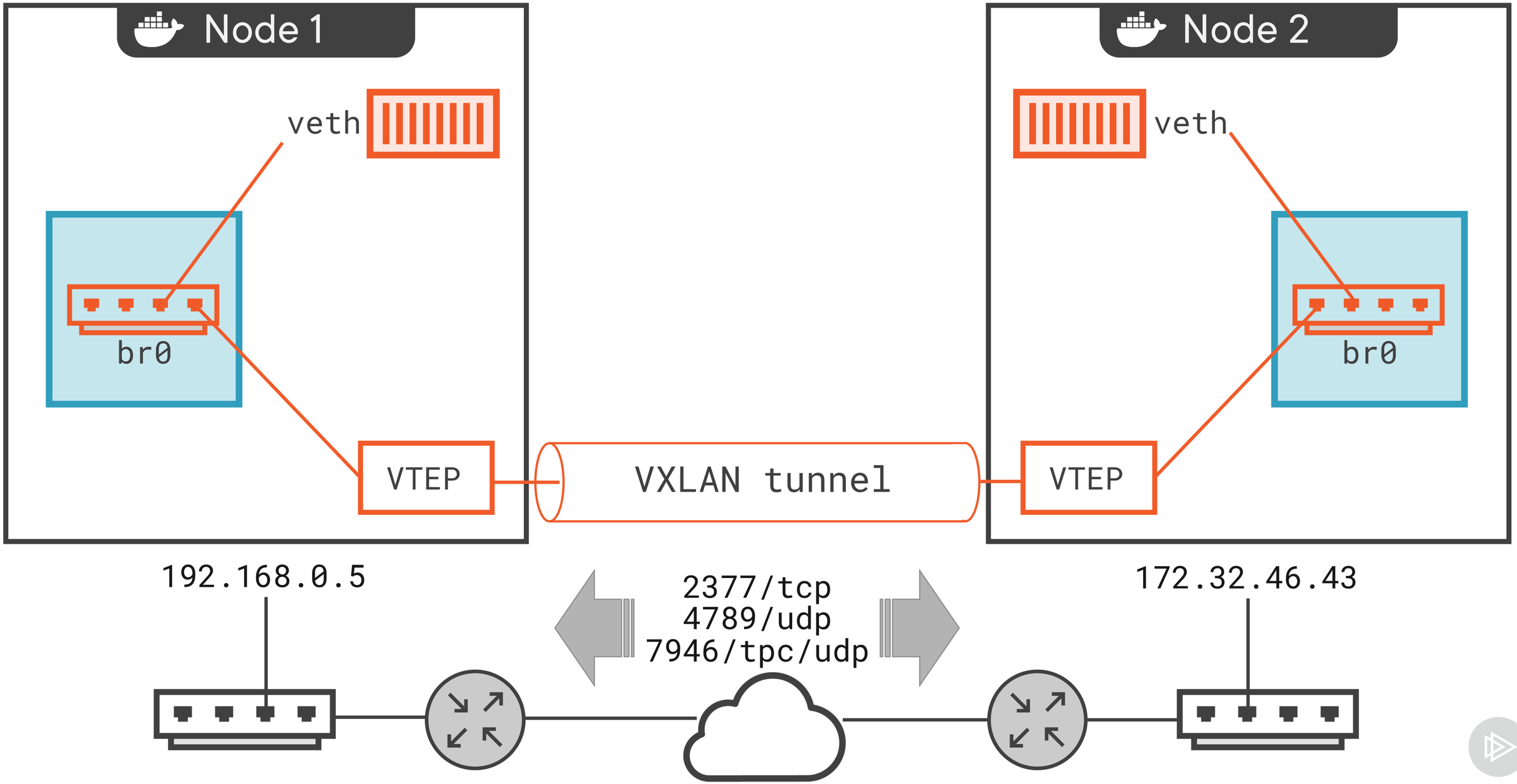
Up Next:
Multi-host Overlay Networks

Multi-host Overlay Networks


Node 1
Node 2
Node 3
Overlay (L2 broadcast domain)
docker network create \
-d overlay \
...
192.168.0.5
172.32.46.43
10.0.0.5
Allow
2377 tcp
4789 udp
7946 tpc,udp
Allow
2377 tcp
4789 udp
7946 tpc,udp


Node 1
Node 2
Node 3
Overlay (L2 broadcast domain)
docker network create \
-d overlay \
...
192.168.0.5
172.32.46.43
10.0.0.5
Allow
2377 tcp
4789 udp
7946 tpc,udp
Allow
2377 tcp
4789 udp
7946 tpc,udp

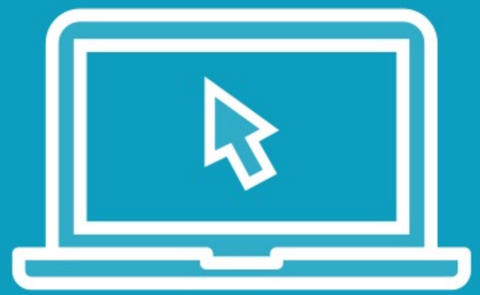

Node 1
Node 2
Node 3
Overlay (L2 broadcast domain)
docker network create \
-d overlay \
-o encrypt
192.168.0.5
172.32.46.43
10.0.0.5
Allow
2377 tcp
4789 udp
7946 tpc,udp
Allow
2377 tcp
4789 udp
7946 tpc,udp


Node 1
veth
br0
VTEP
VXLAN tunnel
Node 2
veth
br0
VTEP
192.168.0.5
2377/tcp
4789/udp
7946/tpc/udp
172.32.46.43

Up Next:
Hands-on with Multi-host Overlay Networks

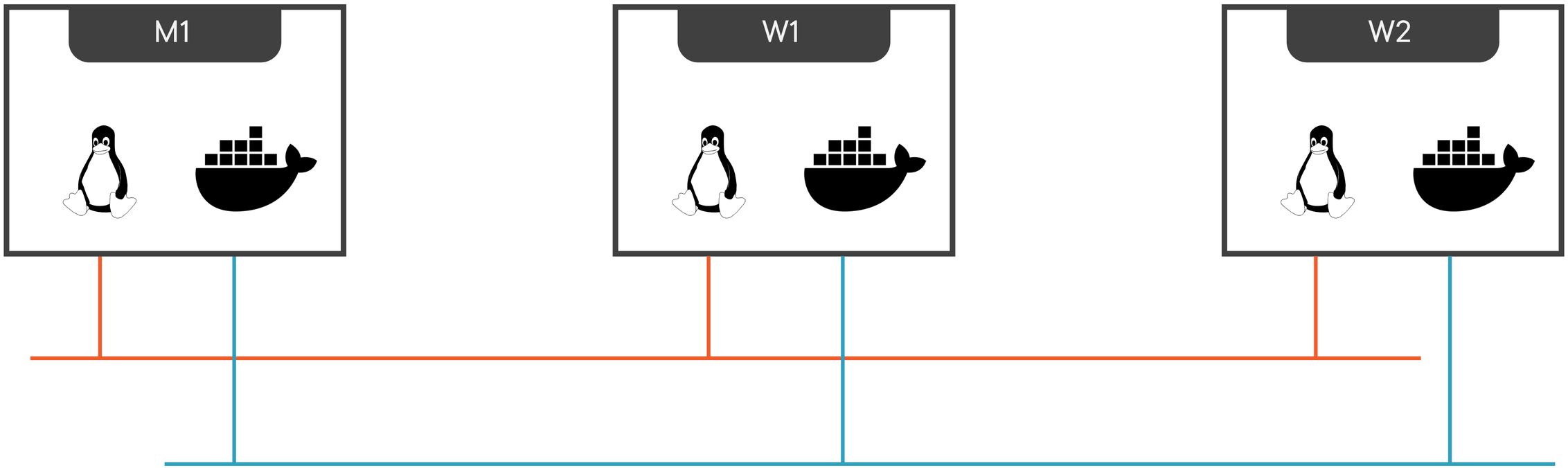
Hands-on with Multi-host Overlay Networks

Demo



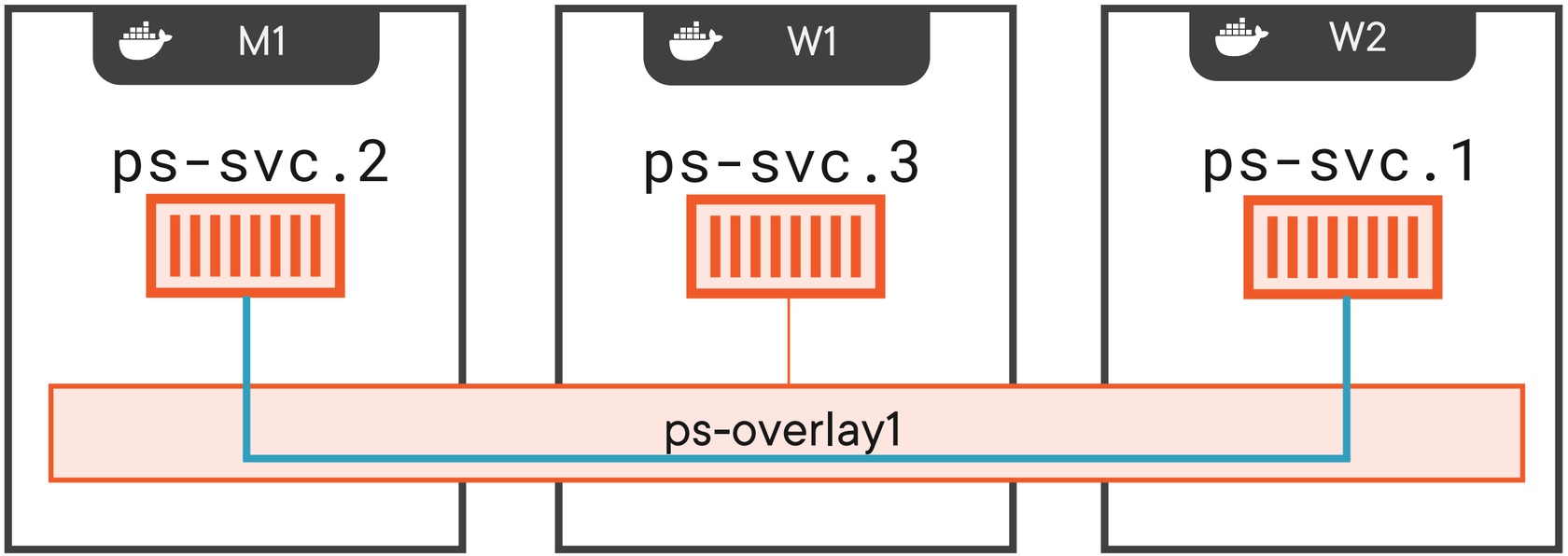
- Lab Config
- Enable Swarm Mode
- Test Some Overlays
- Test Swarm Services vs Standalone Containers

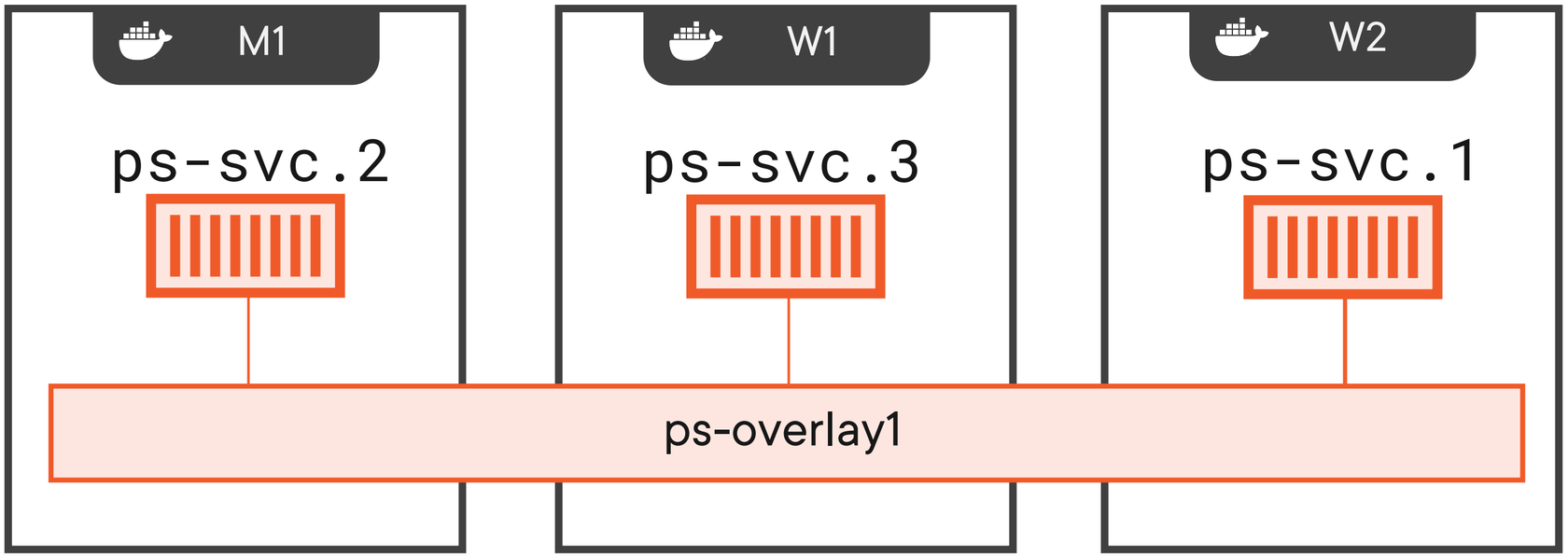
Allowed ports on the private network:
TCP 2377, UDP 4789, TCP/UDP 7946

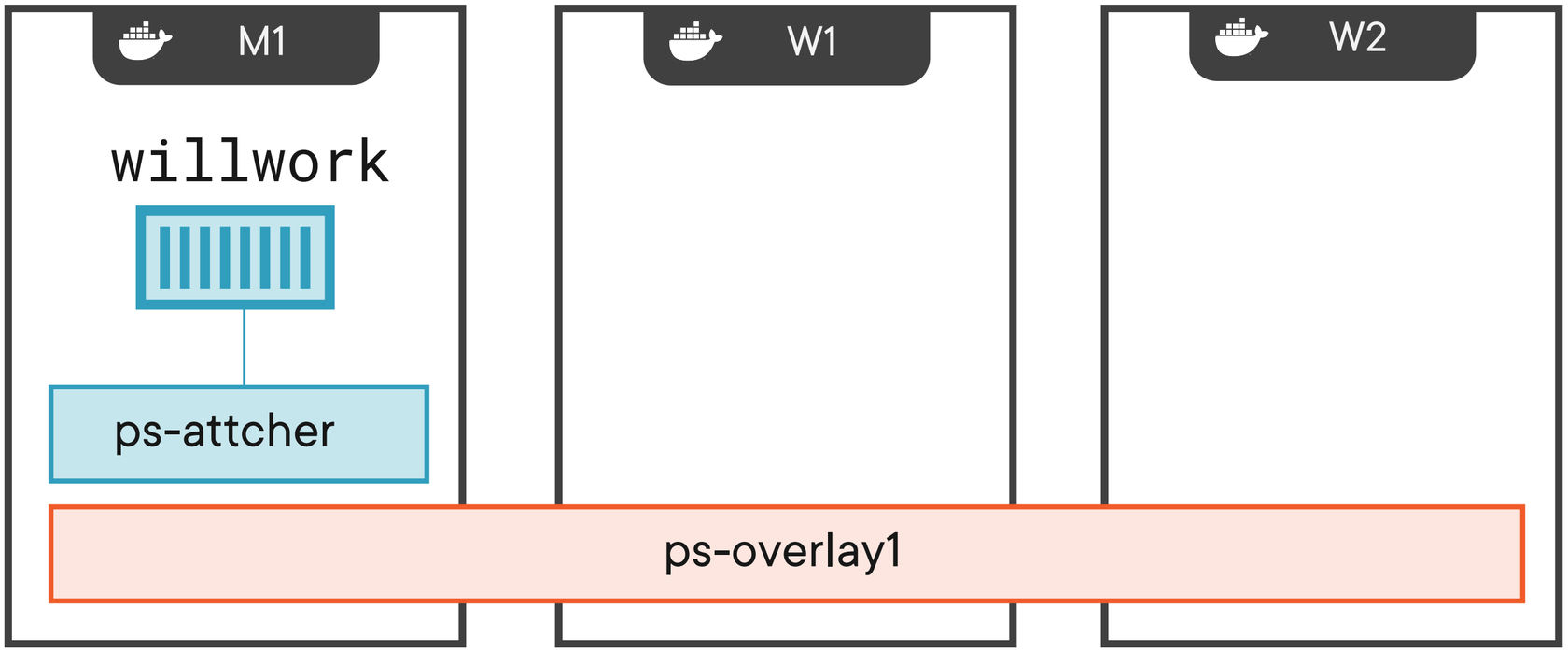


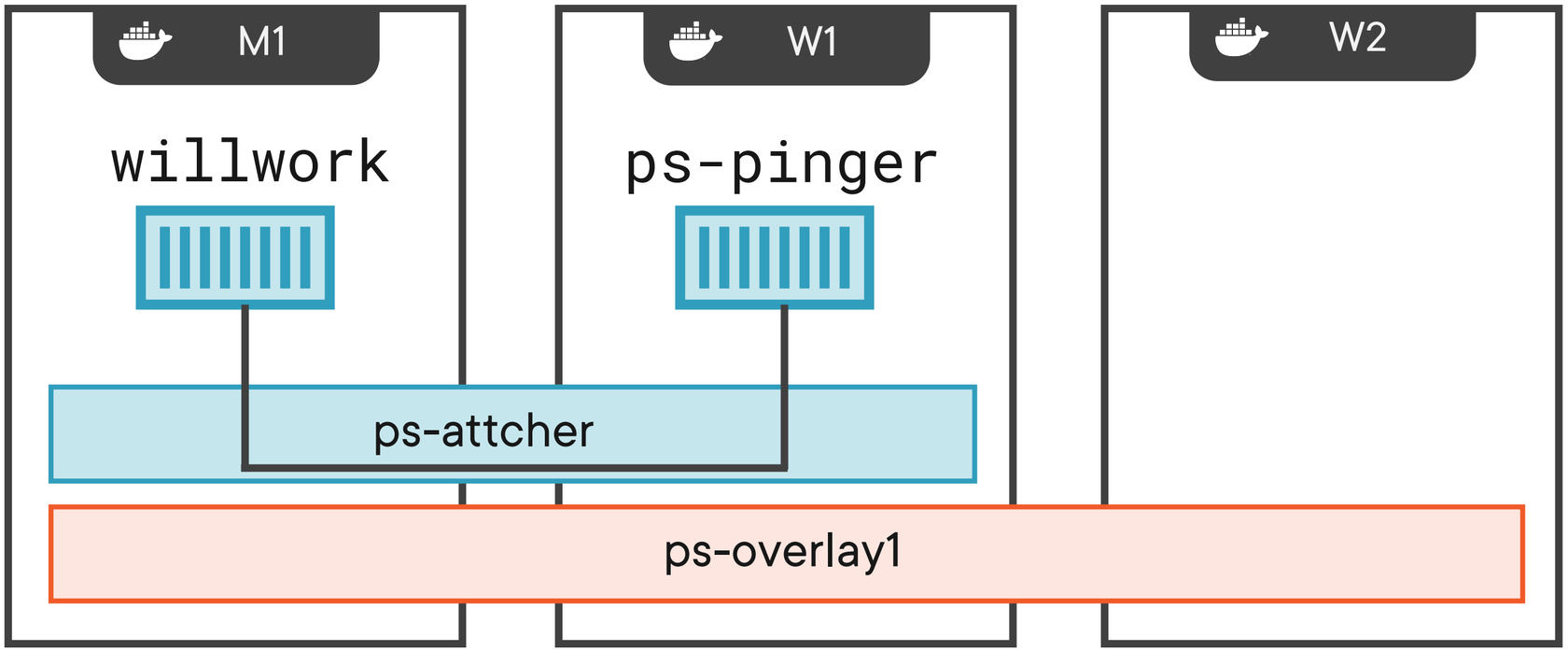
Private IPs. Used for Swarm and overlay networks.

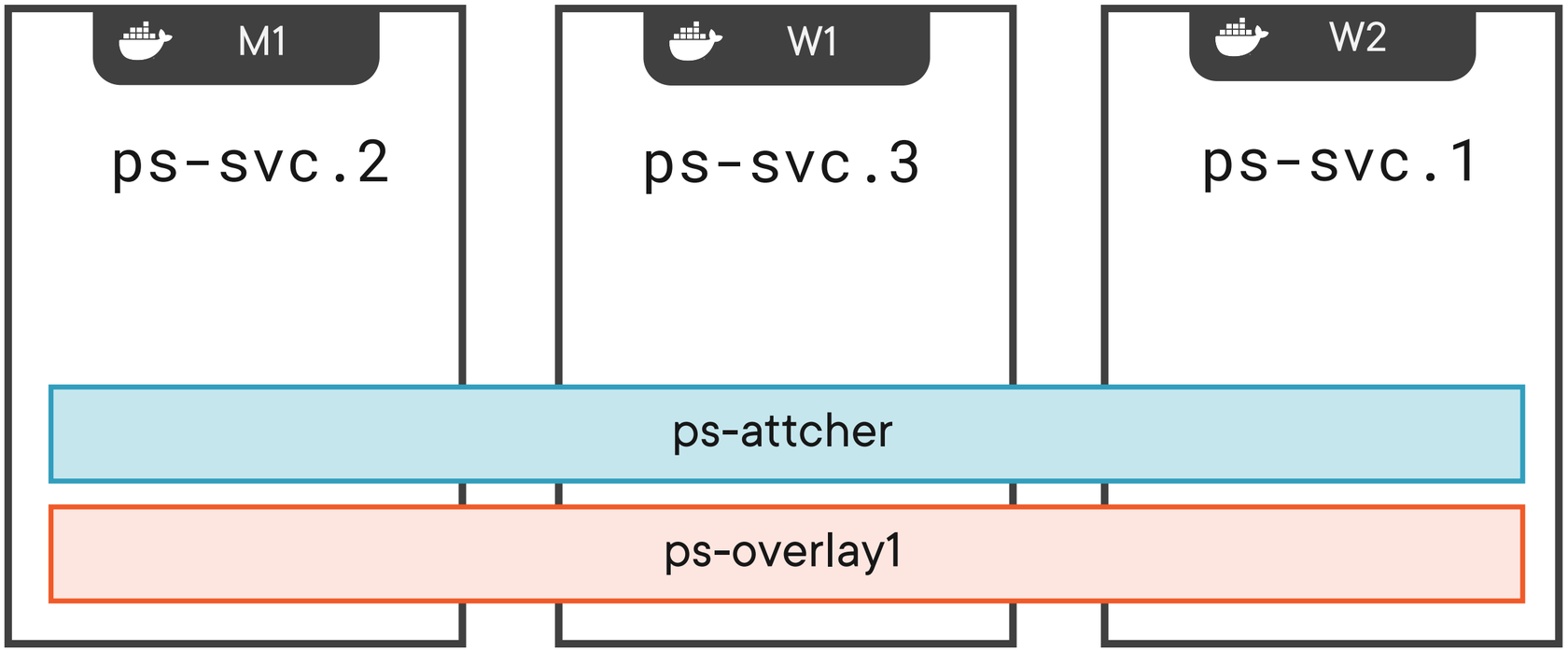
Public IPs. Used for SSH and other management connectivity.


M1
W1
W2
ps-svc.2
ps-svc.3
ps-svc.1
ps-overlay1


M1
W1
W2
ps-svc.2
ps-svc.3
ps-svc.1
ps-overlay1


M1
W1
W2
willwork
ps-attcher
ps-overlay1


M1
W1
W2
willwork
ps-pinger
ps-attcher
ps-overlay1


M1
W1
W2
ps-svc.2
ps-svc.3
ps-svc.1
ps-attcher
ps-overlay1

Up Next:
Connecting to Existing Networks with MACVLAN

Connecting to Existing Networks with MACVLAN

MACVLAN

For apps and containers that need a MAC address on existing networks.

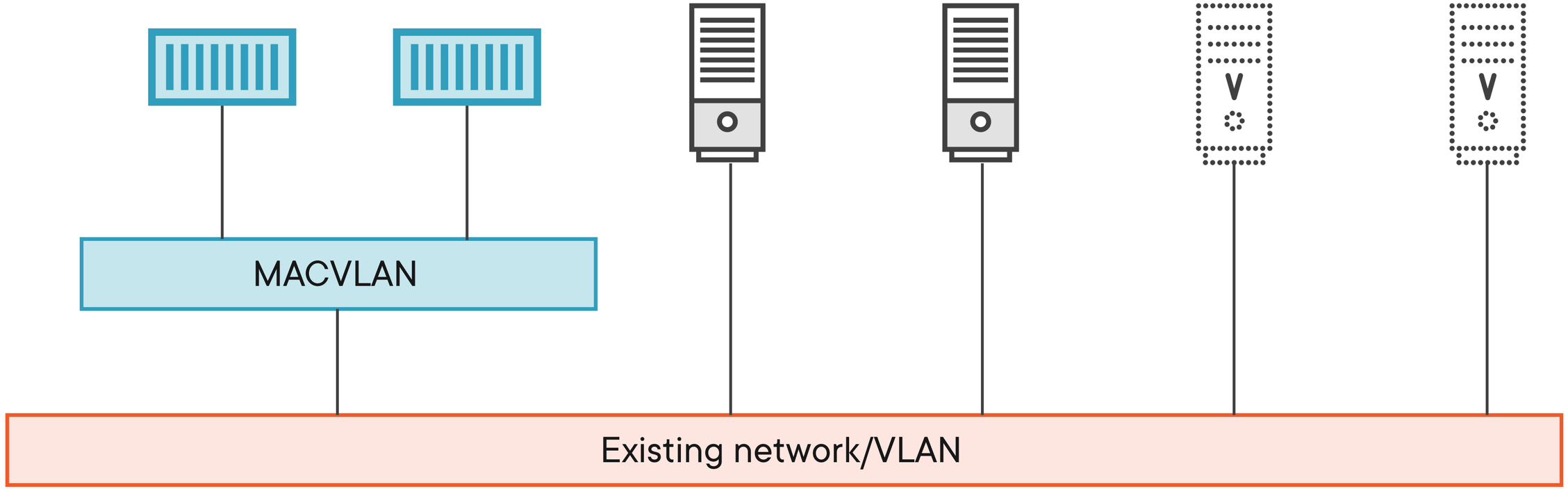
MACVLAN Pre-reqs

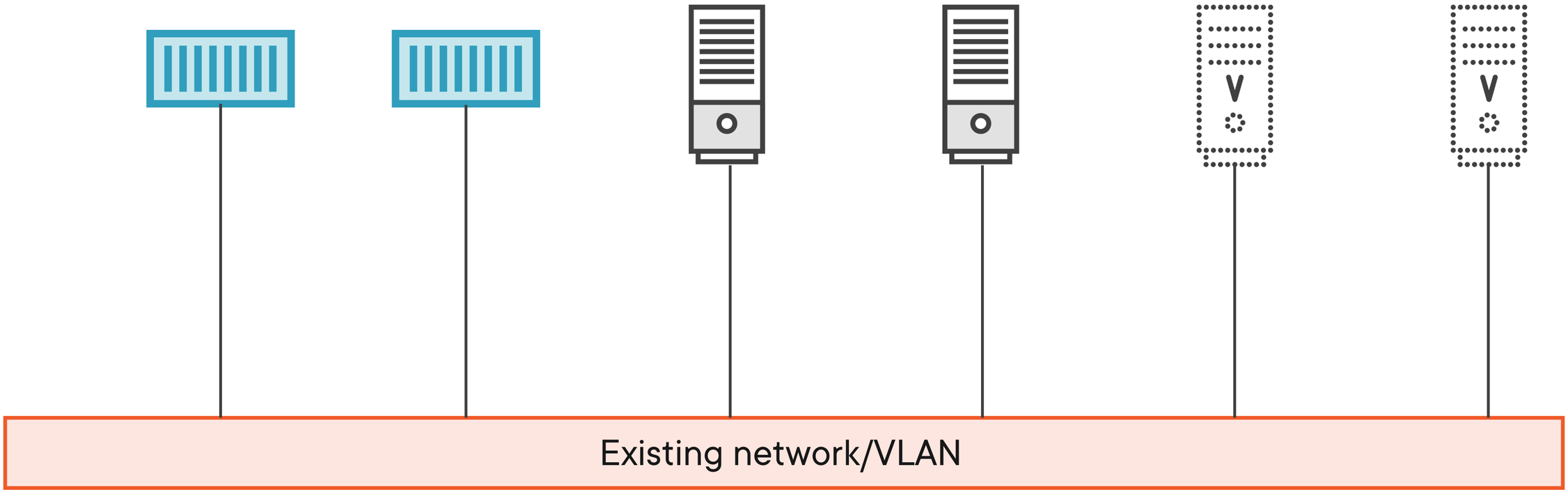
Linux

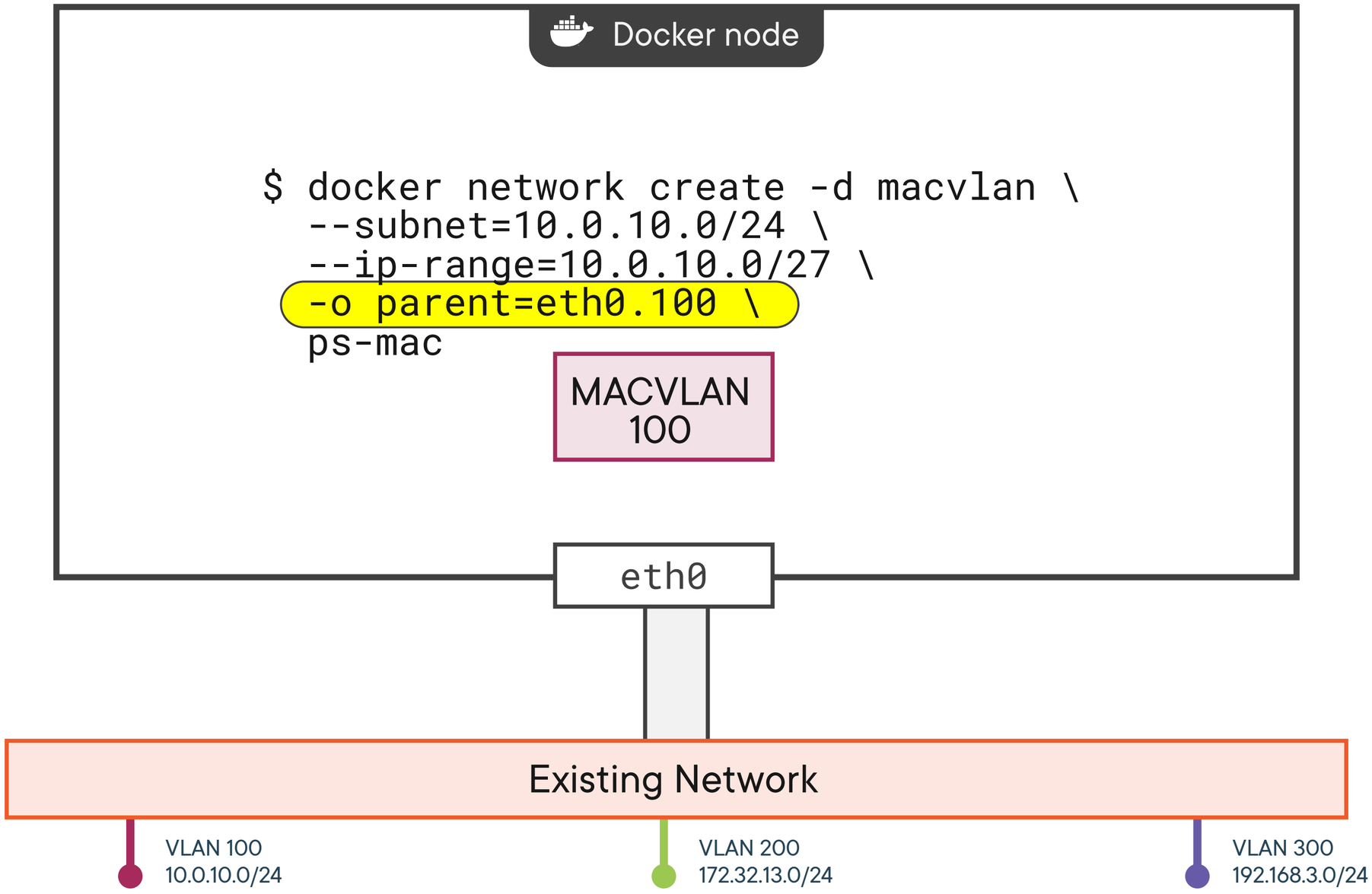
4.x kernel or newer

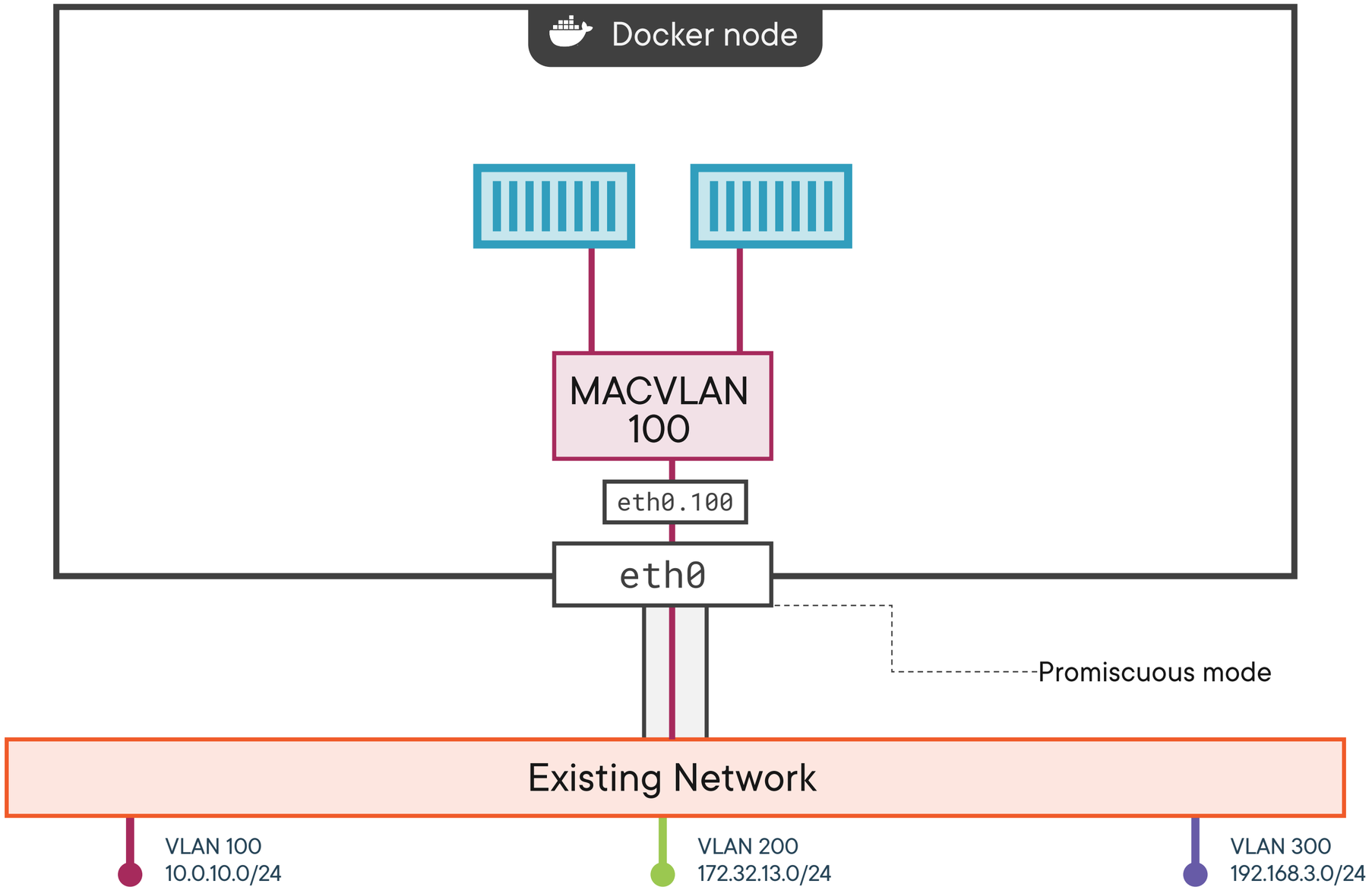
NIC in promiscuous mode

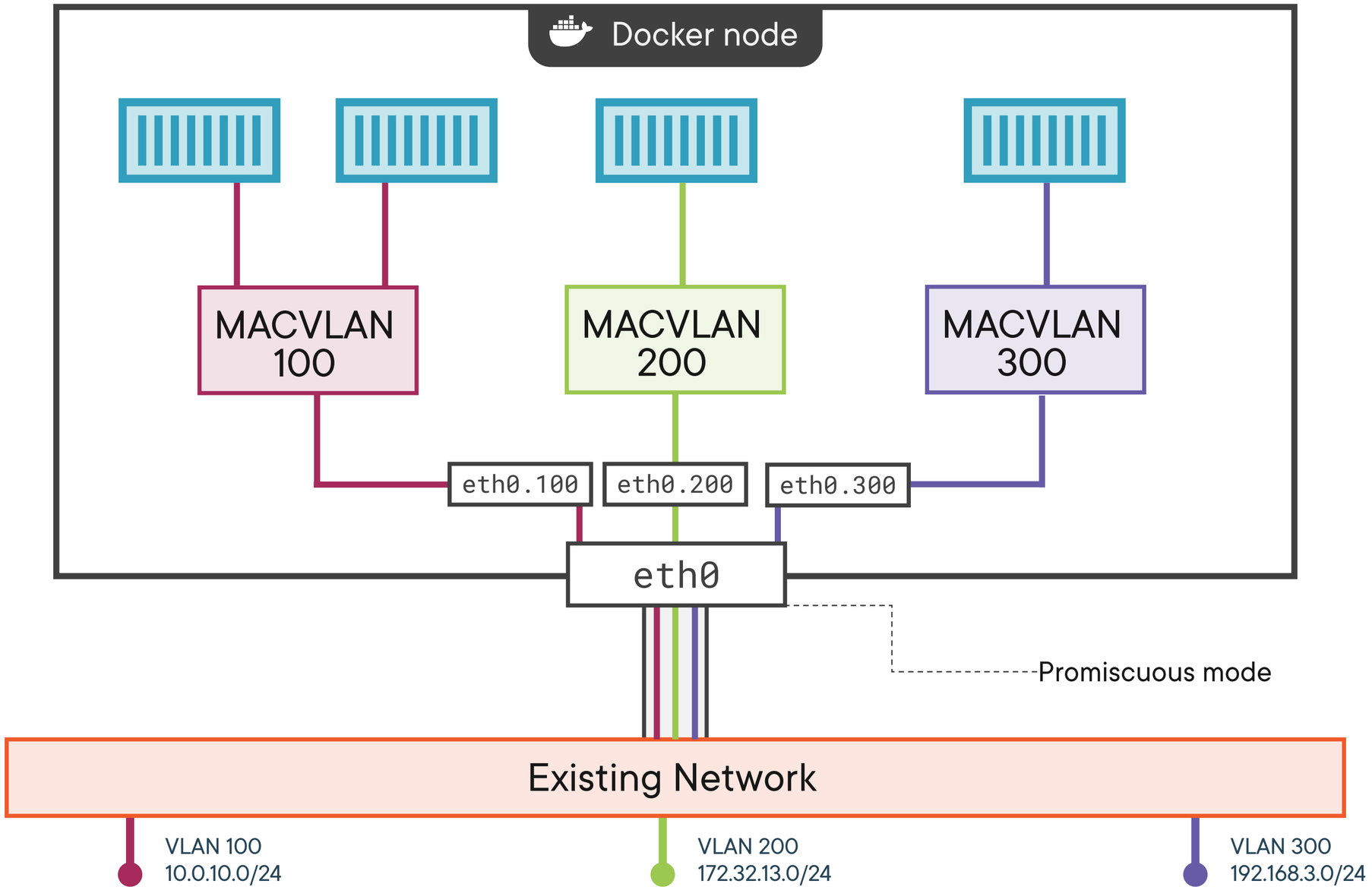
* Doesn't work with Docker Desktop

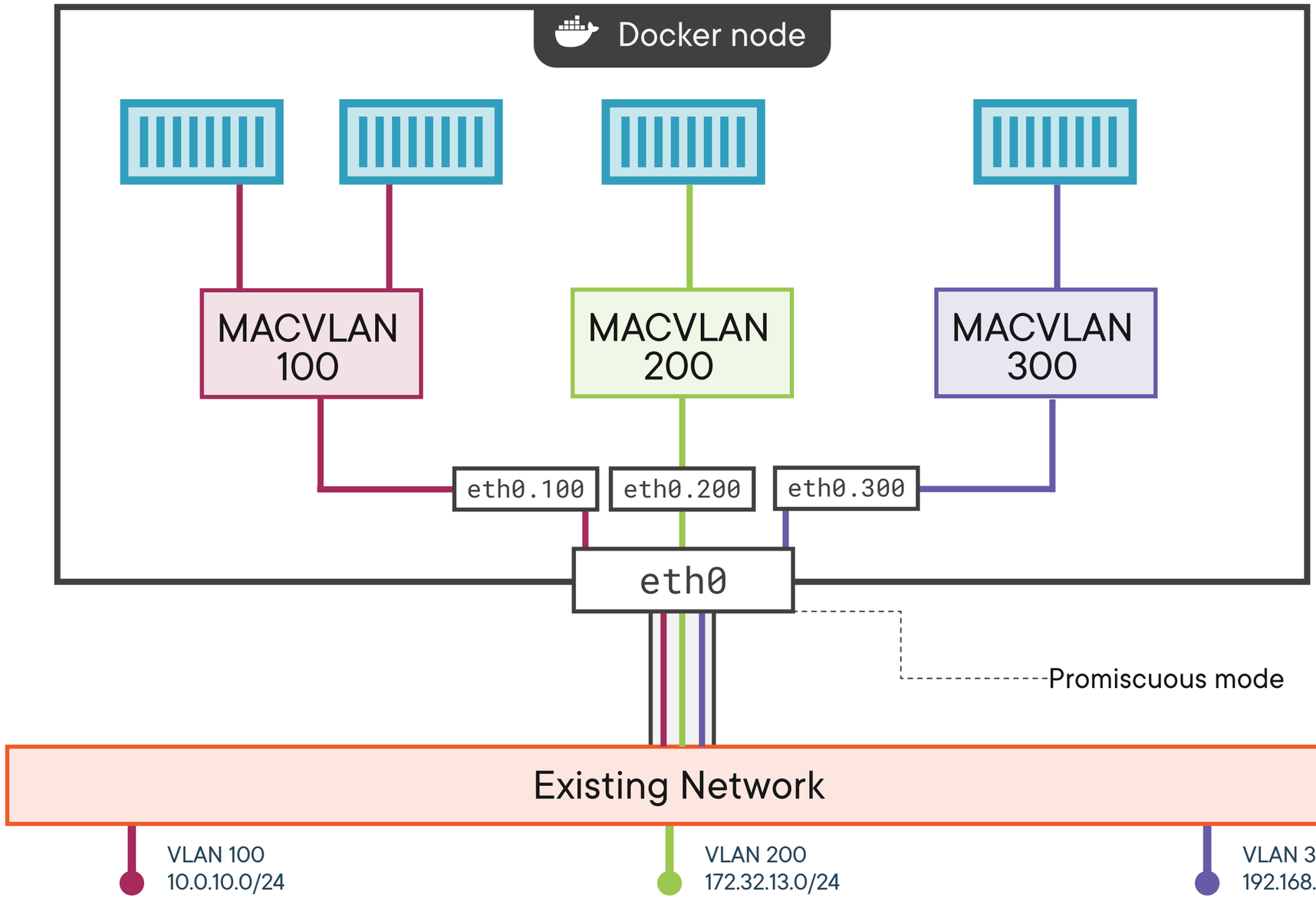

MACVLAN
Existing network/VLAN


Existing network/VLAN


Docker node
$ docker network create -d macvlan \
--subnet=10.0.10.0/24 \
--ip-range=10.0.10.0/27 \
-o parent=eth0.100 \
ps-mac
MACVLAN
100
eth0
Existing Network
VLAN 100
10.0.10.0/24
VLAN 200
172.32.13.0/24
VLAN 300
192.168.3.0/24


Docker node
MACVLAN
100
eth0.100
eth0
Promiscuous mode
Existing Network
VLAN 100
10.0.10.0/24
VLAN 200
172.32.13.0/24
VLAN 300
192.168.3.0/24


Docker node
MACVLAN 100
MACVLAN 200
MACVLAN 300
eth0.100
eth0.200
eth0.300
eth0
Promiscuous mode
Existing Network
VLAN 100
10.0.10.0/24
VLAN 200
172.32.13.0/24
VLAN 300
192.168.3.0/24

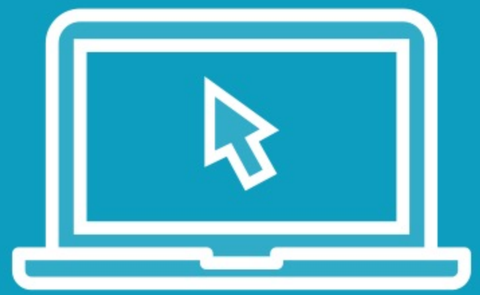


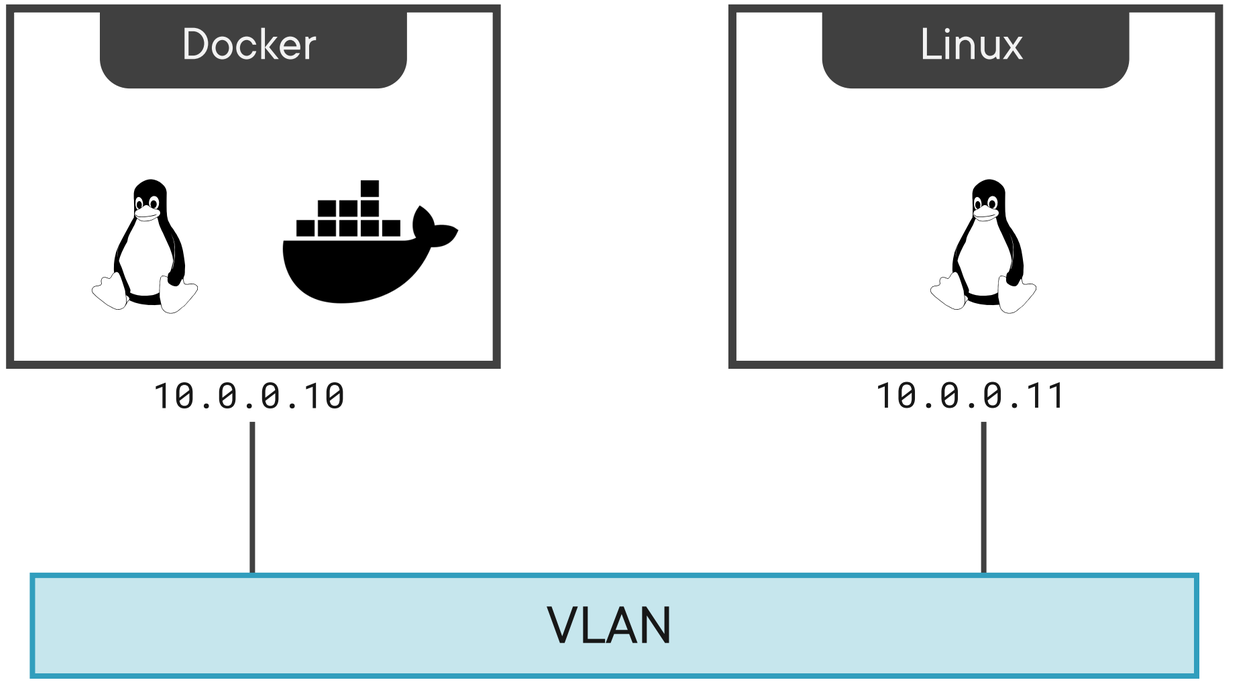

```
$ docker network create -d macvlan \
  --subnet=10.0.10.0/24 \
  --ip-range=10.0.10.0/27 \
  -o parent=eth0.100 \
  ps-mac
```

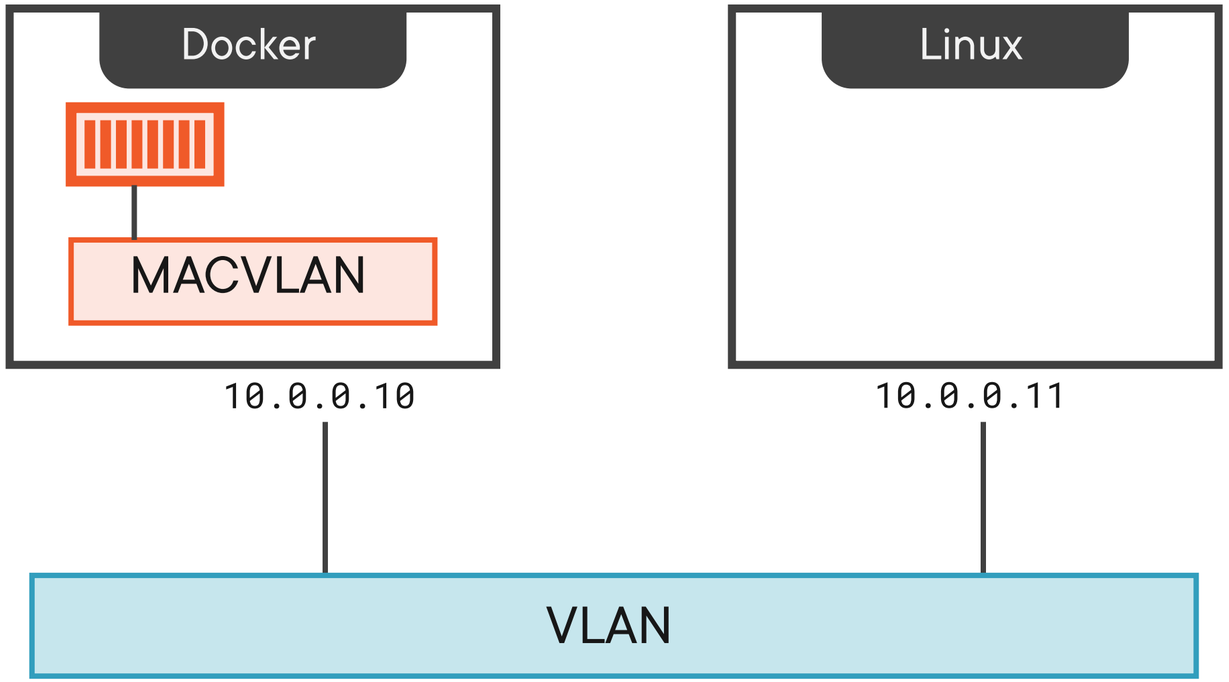
Up Next:
Hands-on with MACVLAN

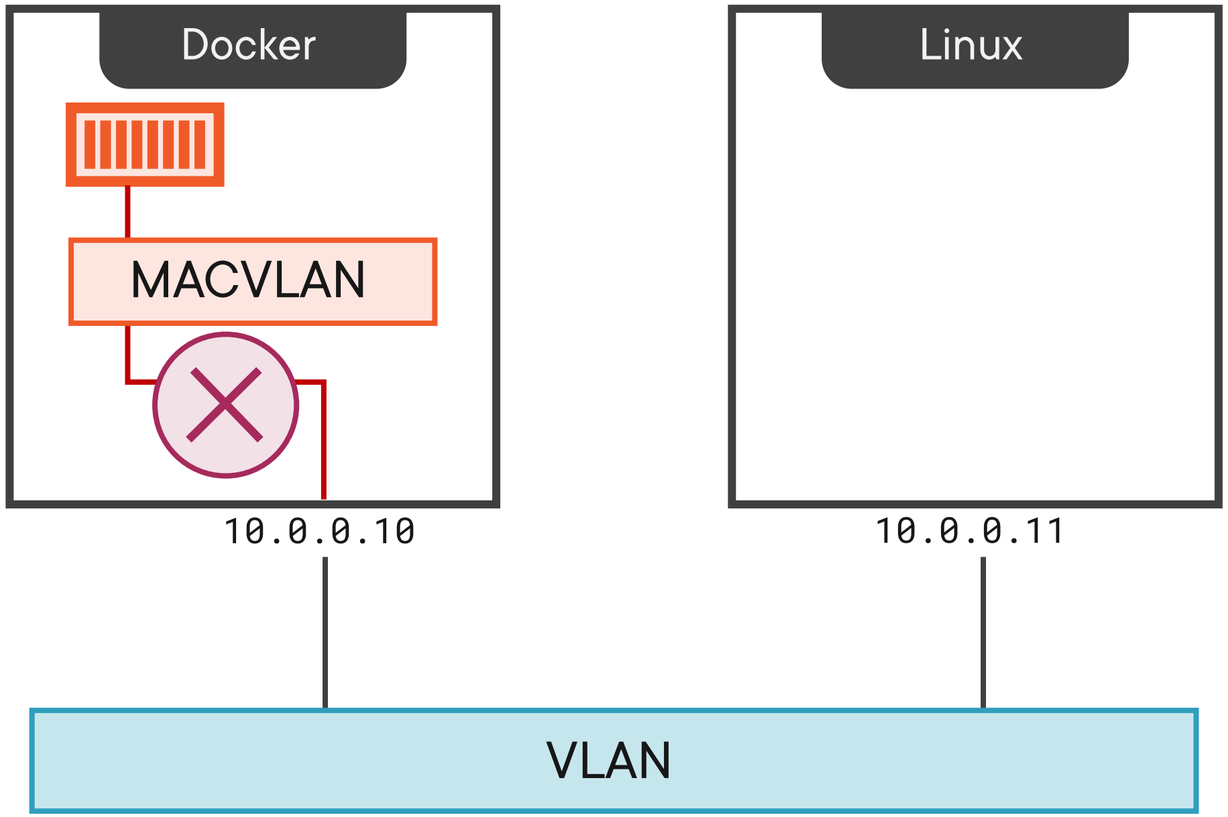
Hands-on with MACVLAN

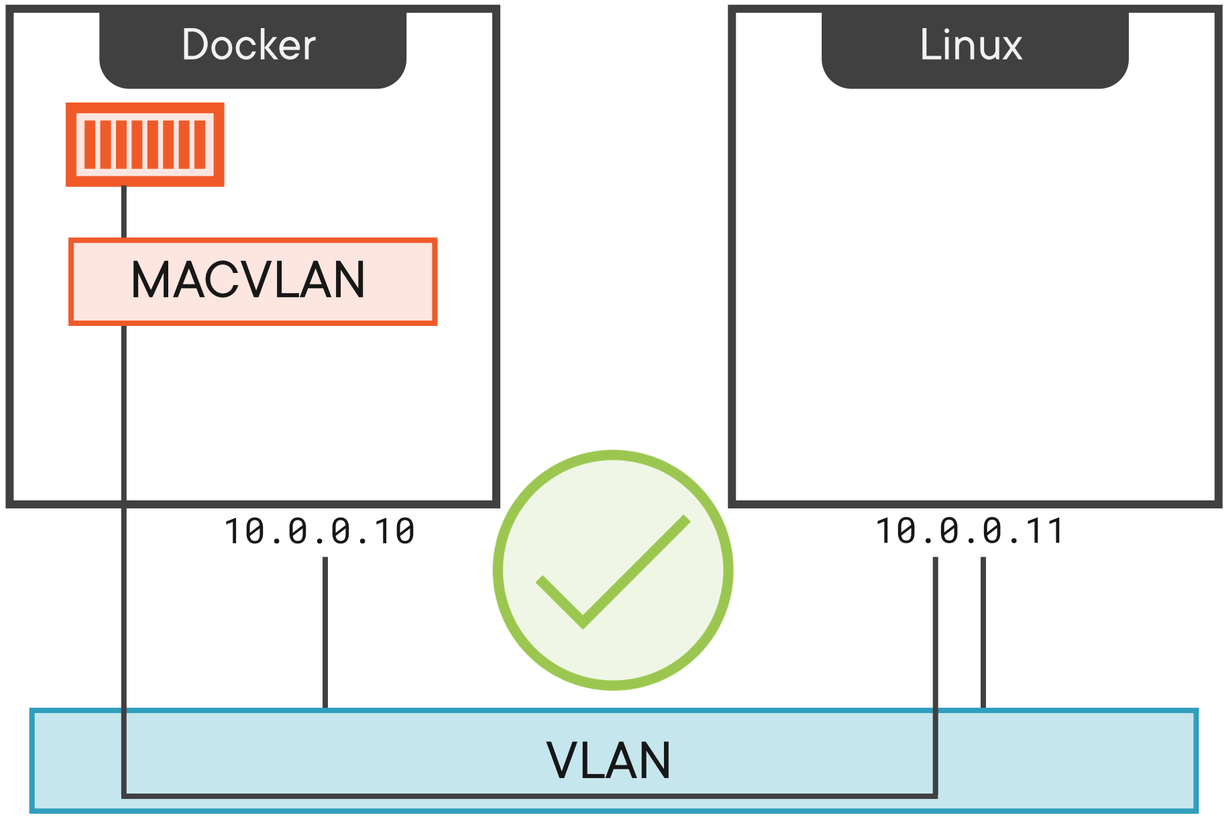
Demo






Docker
MACVLAN
Linux
10.0.0.10
10.0.0.11
VLAN


Docker
MACVLAN
Linux
10.0.0.10
10.0.0.11
VLAN


Docker
MACVLAN
10.0.0.10
Linux
10.0.0.11
VLAN

Clean-up commands...

```
docker container rm test test2 -f

docker network rm ps-mac
```

Up Next:
Connecting to Existing Networks with IPVLAN

Connecting to Existing Networks with IPVLAN

MACVLAN

Connect to existing networks

Containers get MAC address

Requires promiscuous mode

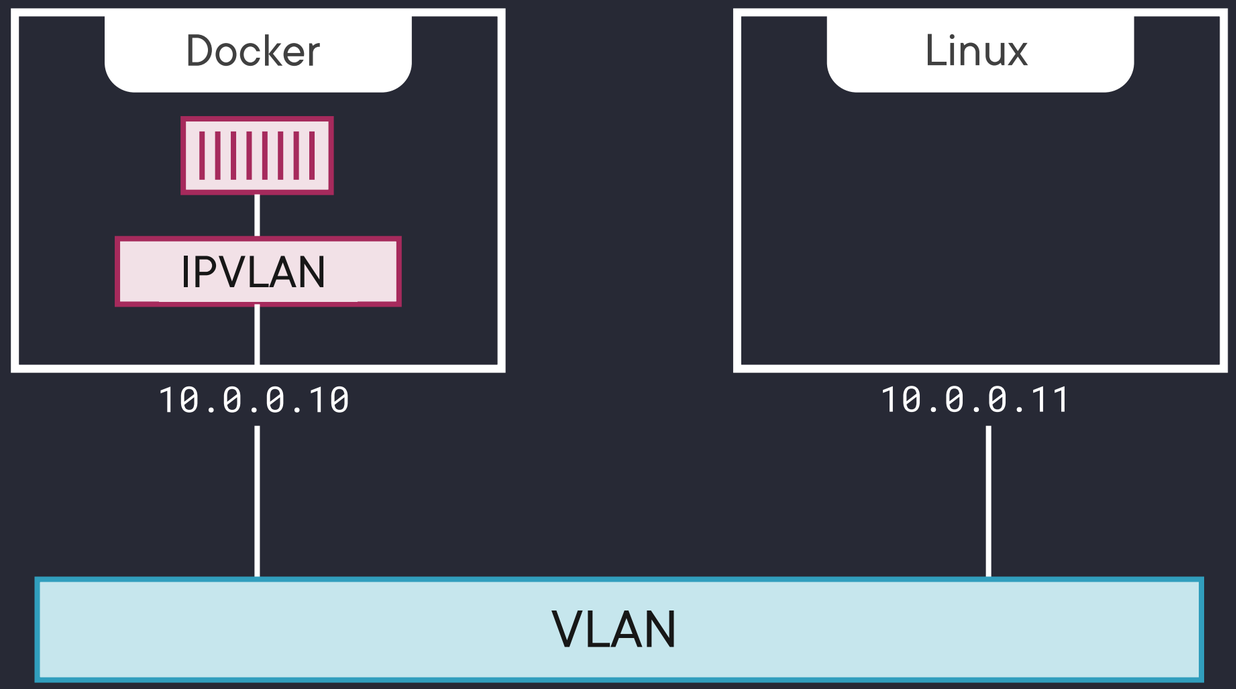
IPVLAN

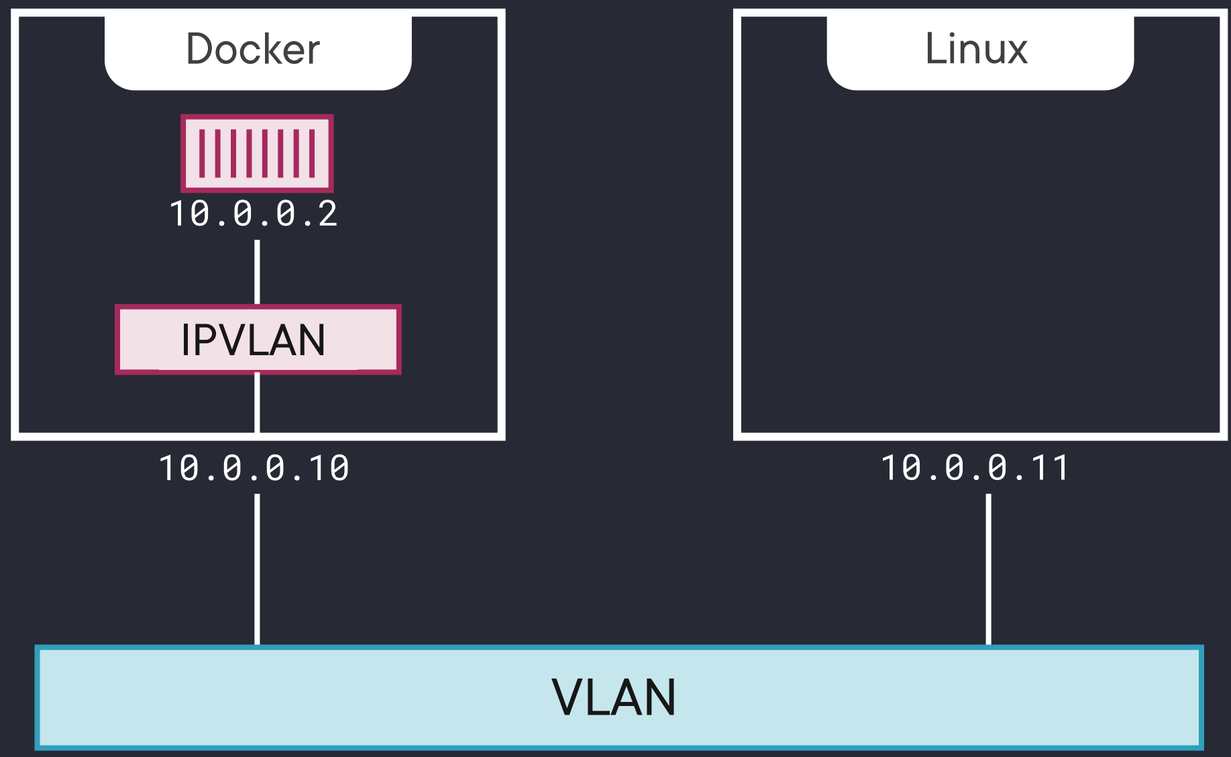
Connect to existing networks

Containers do **NOT** get MAC address

Does **NOT** Require promiscuous mode

IPVLAN may not work with DHCP


Docker
IPVLAN
10.0.0.10
Linux
10.0.0.11
VLAN


Docker
10.0.0.2
IPVLAN
10.0.0.10
Linux
10.0.0.11
VLAN

Recap

Single-host bridge networks

Multi-host overlay networks

Connecting to existing networks with MACVLAN

Connecting to existing networks with IPVLAN

Up Next:
Network Services