KubeCon | CloudNativeCon

North America 2018

Connecting and testing virtual network topologies on Kubernetes

Rich Renner & Gage Orsburn

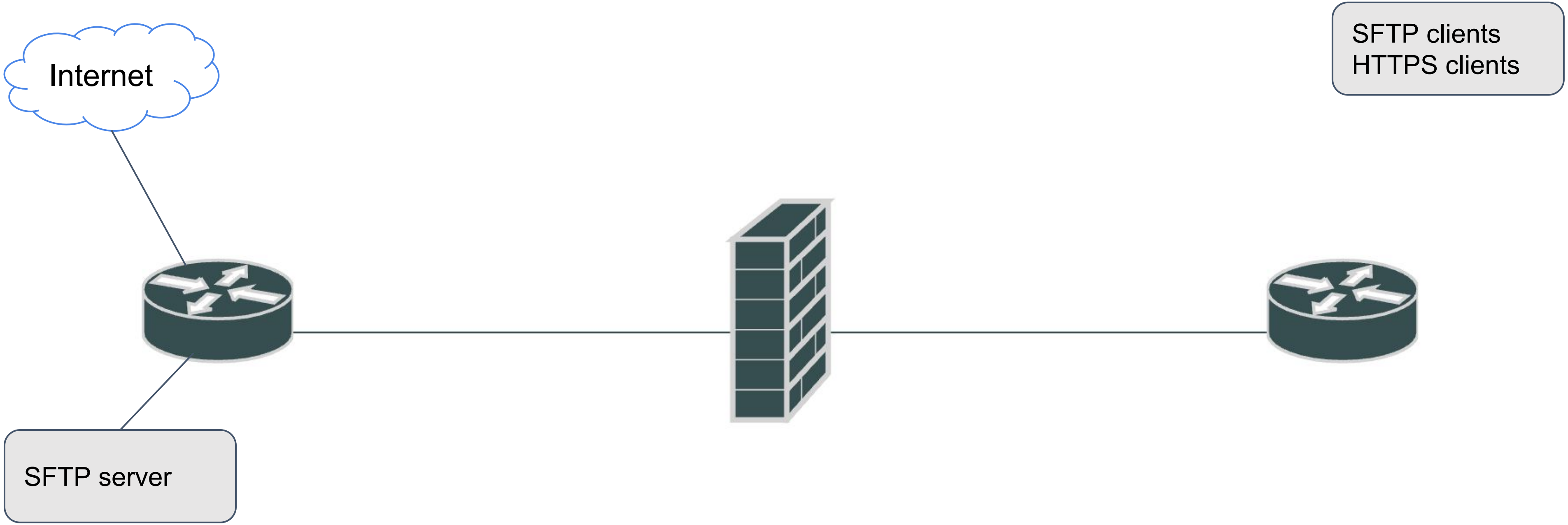
Agenda

1. Problem
2. Goals
3. Challenges
4. KubeVirt
5. Multiple interfaces
6. Writing an OpenVSwitch CNI
7. Network CI/CD Workflow

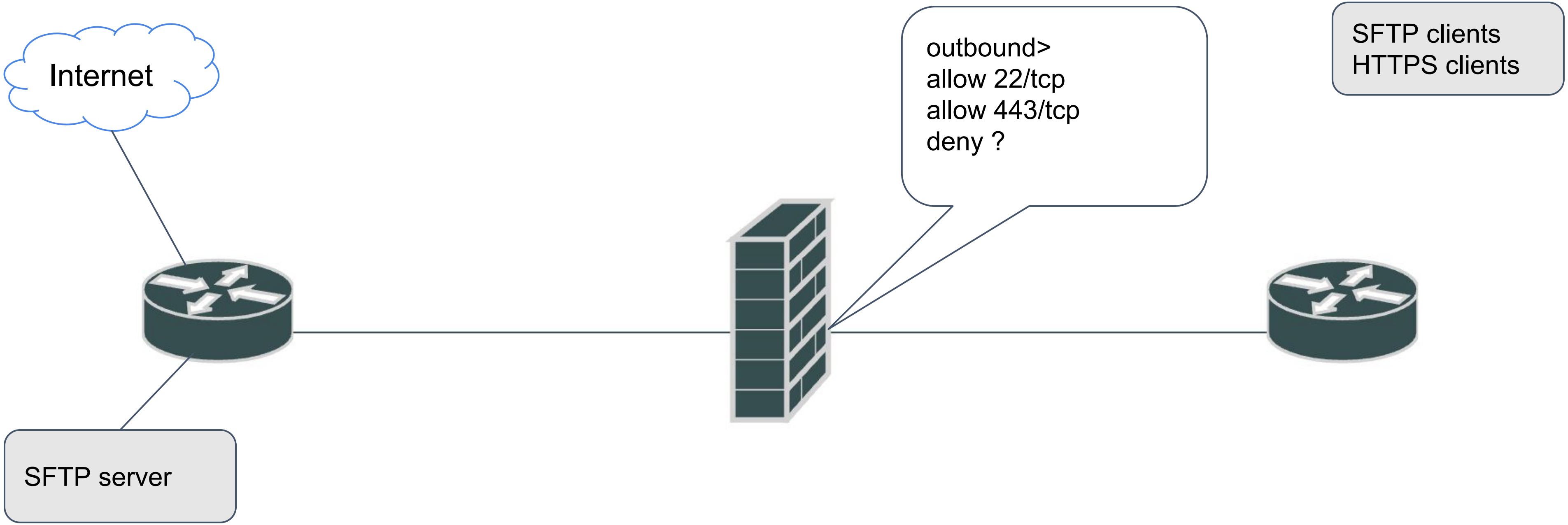
Problem

- Network infrastructure is increasingly business critical and is routinely untested
- Every untested change to the network presents instability and outage potential

Access Control List (ACL) Policy



ACL Policy:



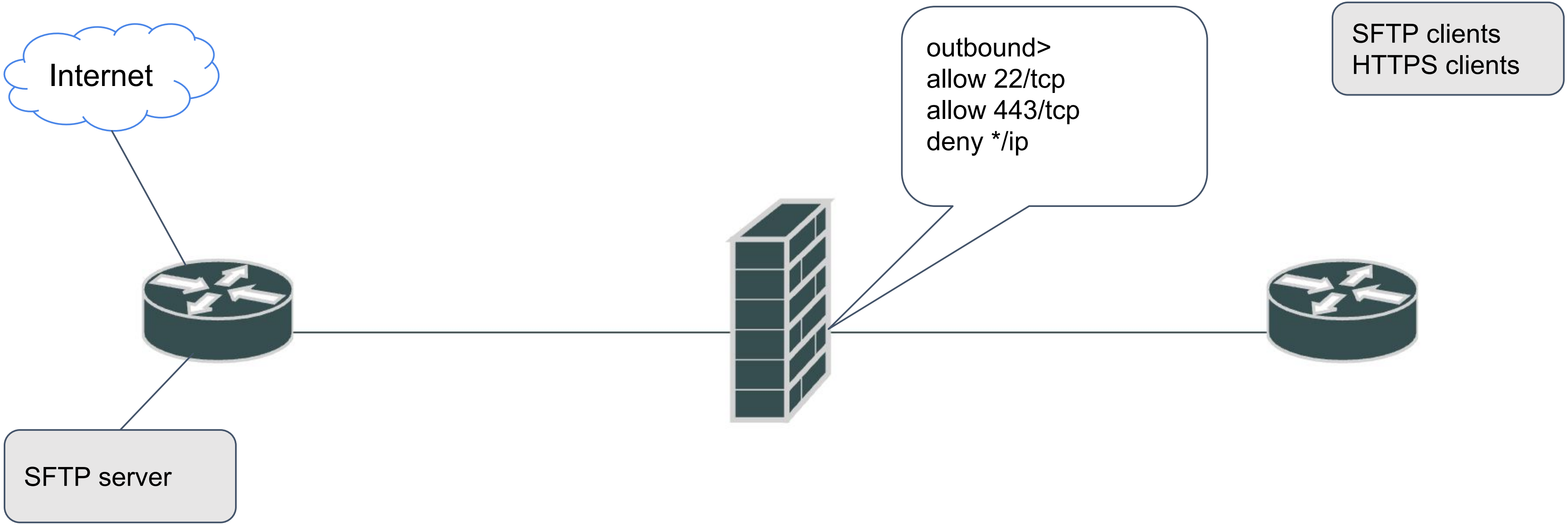
Internet

outbound>
allow 22/tcp
allow 443/tcp
deny ?

SFTP clients
HTTPS clients

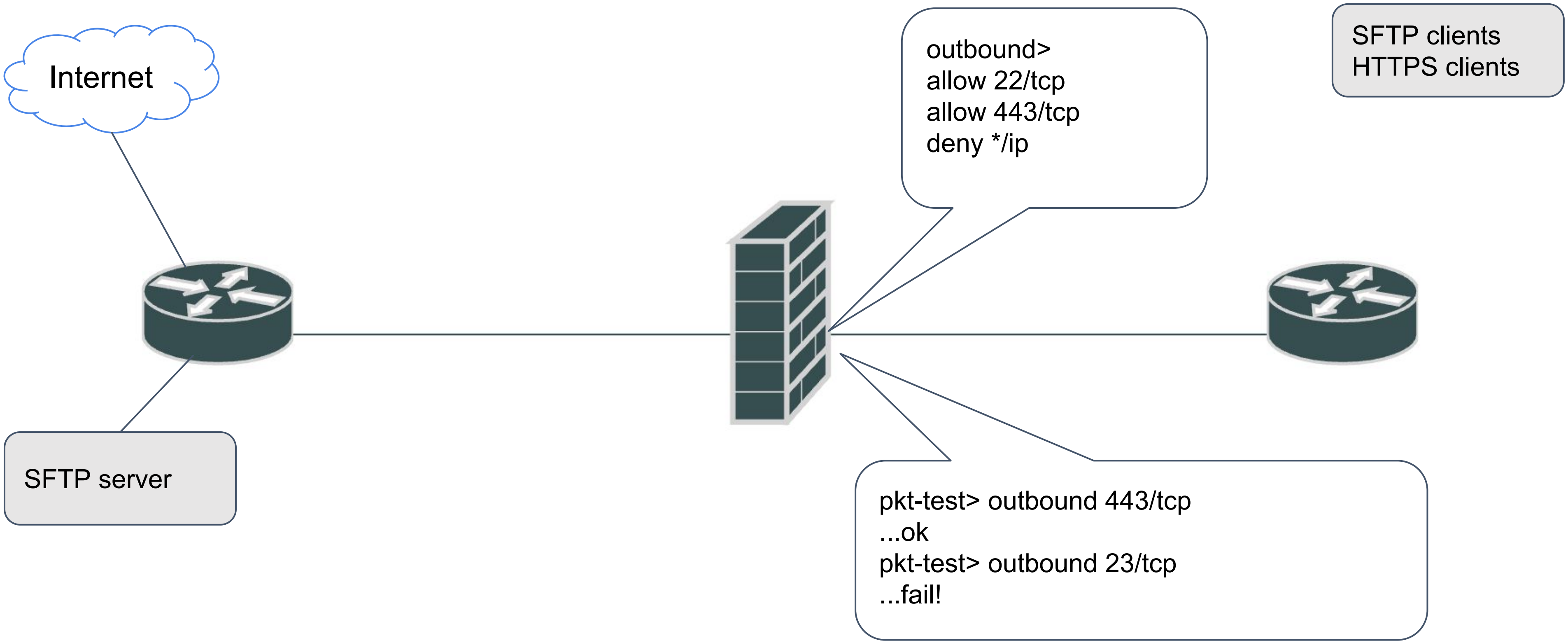
SFTP server

ACL Policy:



outbound>
allow 22/tcp
allow 443/tcp
deny */ip

ACL Policy:



Internet

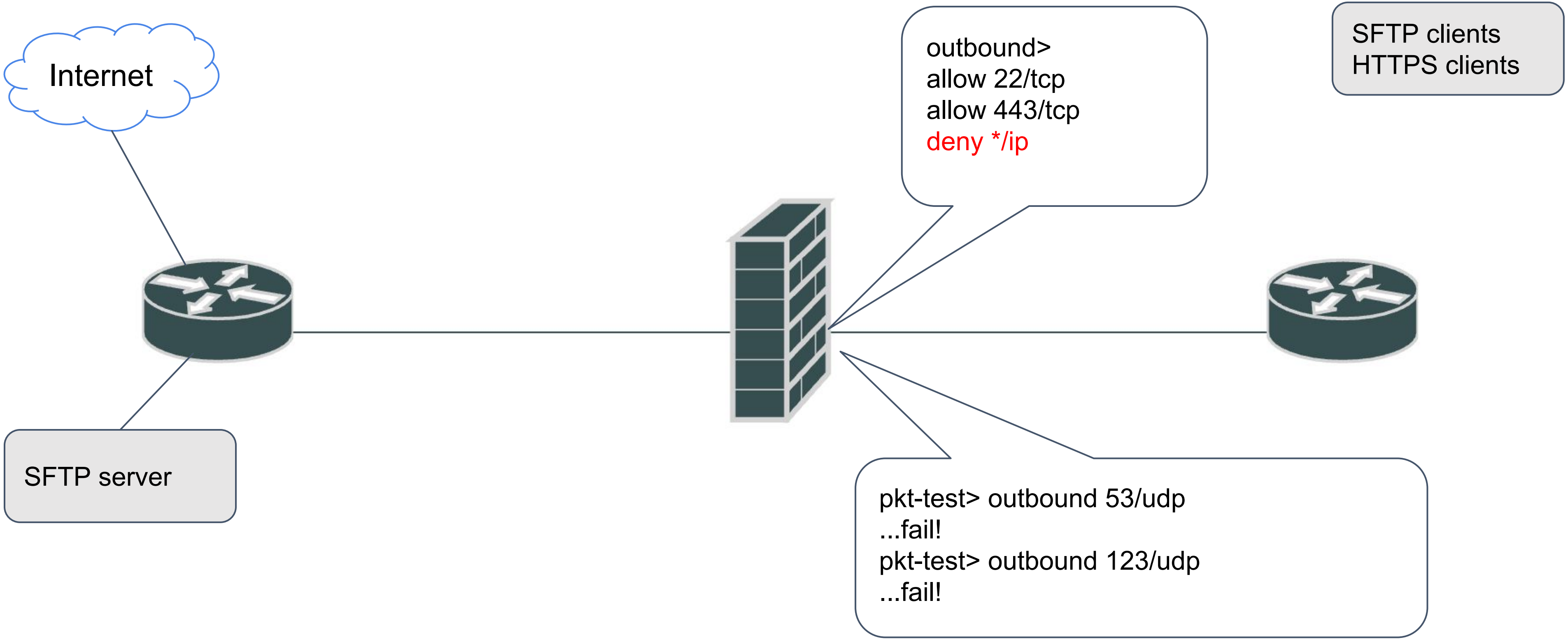
SFTP server

SFTP clients
HTTPS clients

```
outbound>
allow 22/tcp
allow 443/tcp
deny */ip
```

```
pkt-test> outbound 443/tcp
...ok
pkt-test> outbound 23/tcp
...fail!
```

ACL Policy:



Internet

SFTP server

outbound>

allow 22/tcp

allow 443/tcp

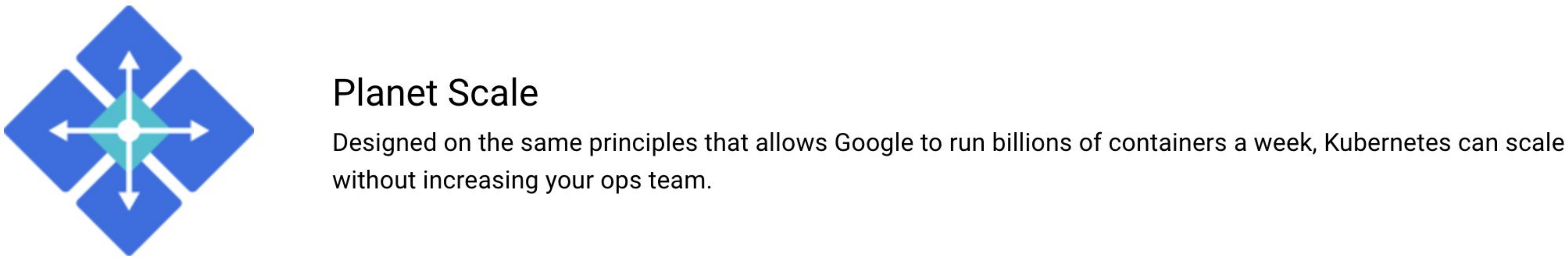
deny */ip

SFTP clients
HTTPS clients

```
pkt-test> outbound 53/udp
...fail!
pkt-test> outbound 123/udp
...fail!
```

Problem

- Scaling to production network size - *"My network has 300 devices. My network emulation software supports 20 devices.."*



Planet Scale

Designed on the same principles that allows Google to run billions of containers a week, Kubernetes can scale without increasing your ops team.

https://kubernetes.io/

Goals

- Provide a virtual testing platform which minimizes the risks of network changes

- Bring CI/CD principles and advantages to the last “infrastructure” frontier

Challenges

- Difficult to containerize network appliances direct from proprietary vendors
- Lack of tooling and solutions in the problem domain
- Distributed connectivity of multiple network interfaces needed

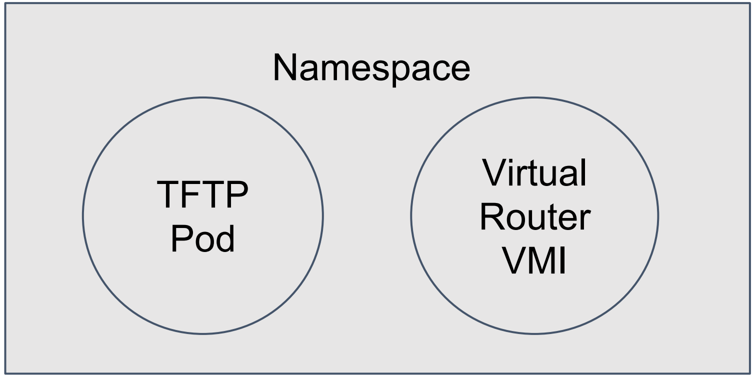
KubeVirt

- What is KubeVirt?
  - https://github.com/kubevirt/kubevirt
  - https://github.com/kubevirt/demo
- Converged infrastructure
- “Containers are cool, but my vendor for X doesn’t think so…”

KubeVirt

- Containerized Data Importer (CDI)
- Bootstrapping base configuration via DHCP or ZTP

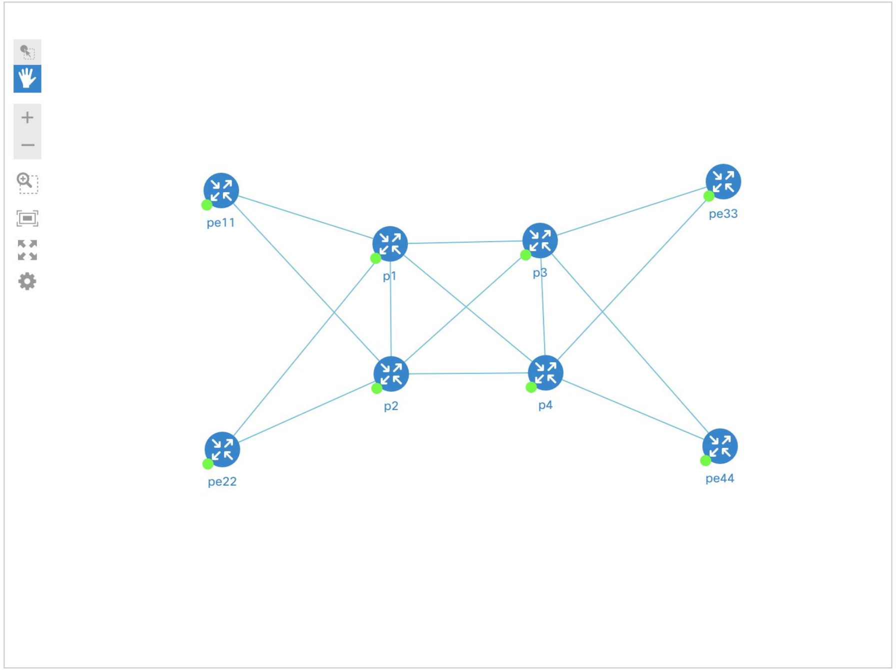
```
interfaces:
 - bridge: {}
   name: default
   dhcpOptions:
     tftpServerName: tftp.default.svc
```



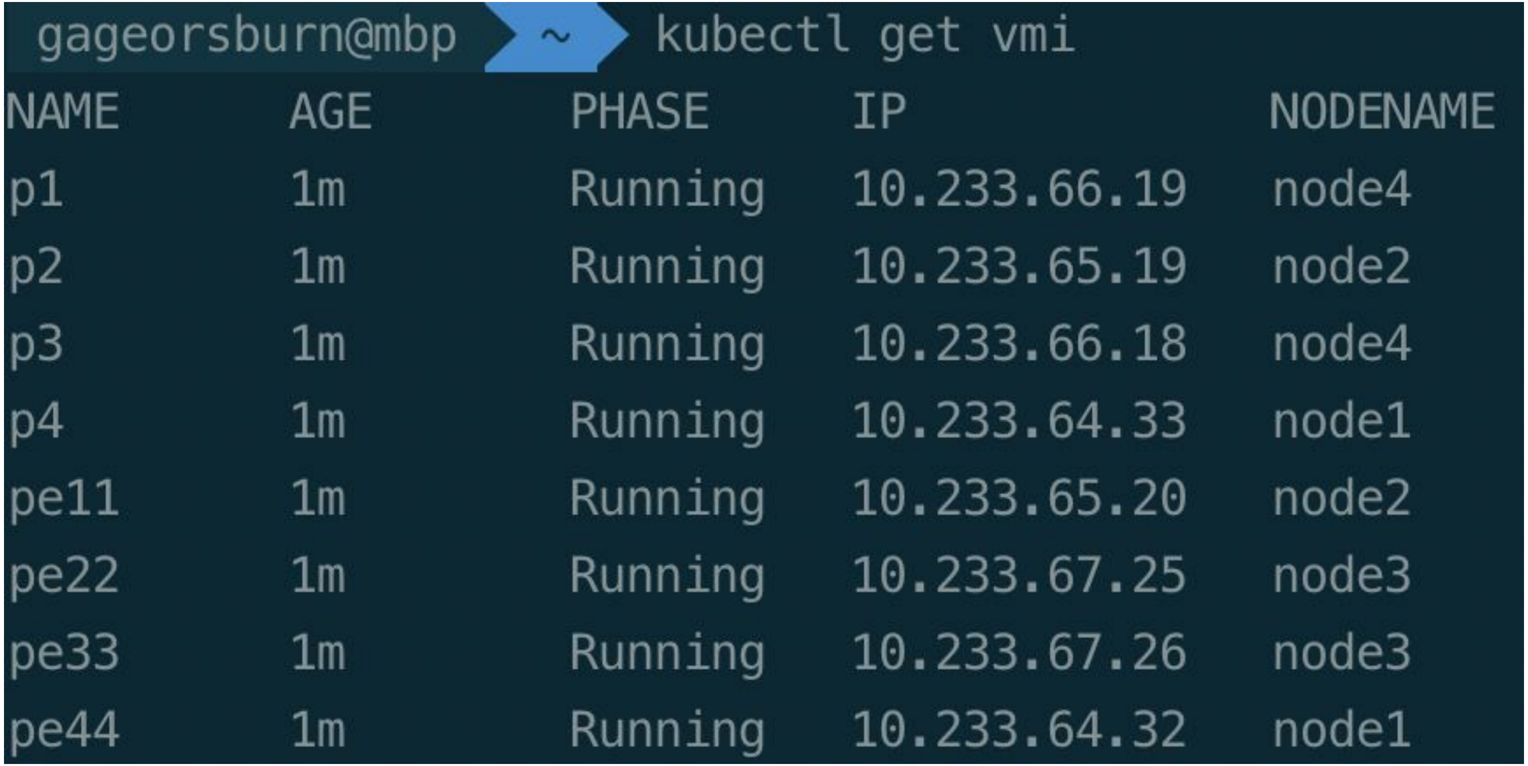
Running VM Workloads Side by Side with Container Workloads - Sebastian Scheele, Loodse
Thursday 10:50 am - 11:25 am

Virtual Topology

service-provider - c6b610f8-bc91-4e58-b68e-cbd75bed31e5

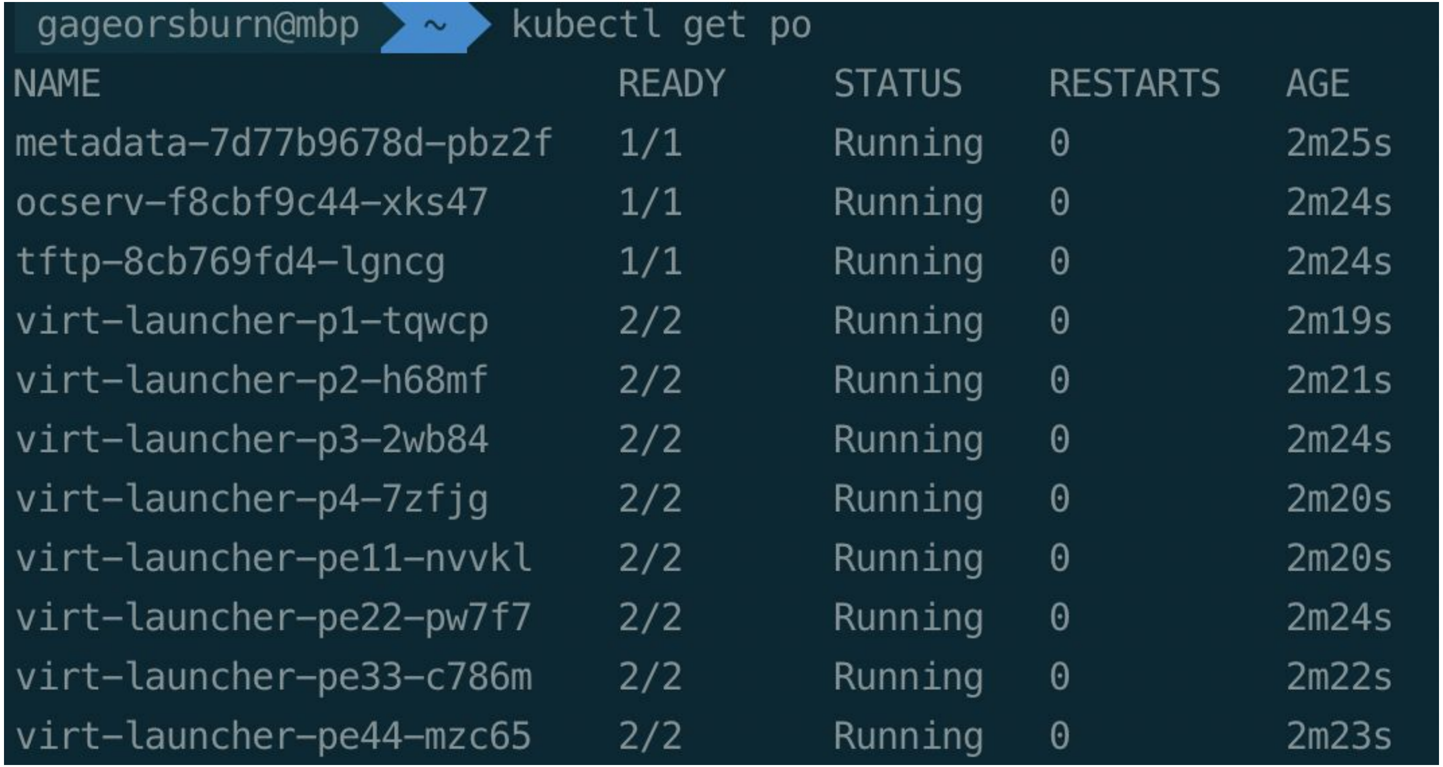


KubeVirt VMIs



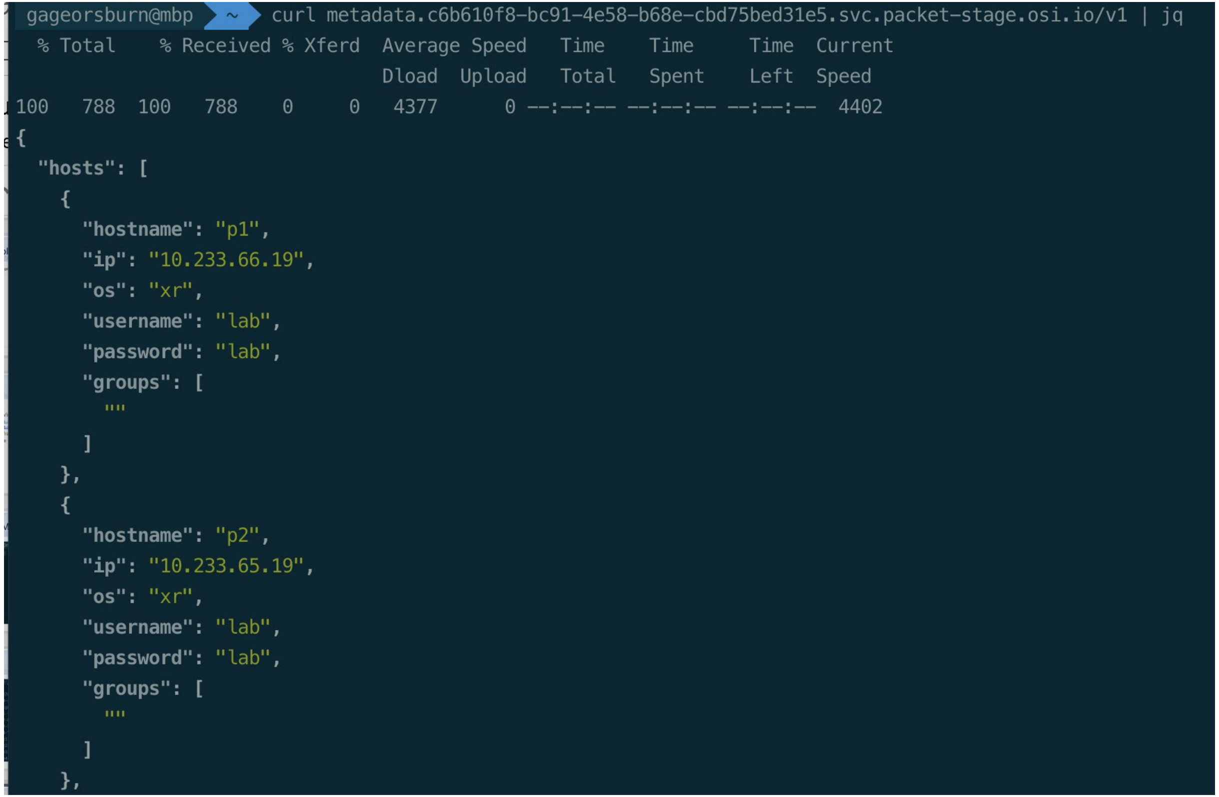
```
gageorsburn@mbp  ~  kubectl get vmi
NAME      AGE       PHASE     IP             NODENAME
p1        1m        Running   10.233.66.19   node4
p2        1m        Running   10.233.65.19   node2
p3        1m        Running   10.233.66.18   node4
p4        1m        Running   10.233.64.33   node1
pe11      1m        Running   10.233.65.20   node2
pe22      1m        Running   10.233.67.25   node3
pe33      1m        Running   10.233.67.26   node3
pe44      1m        Running   10.233.64.32   node1
```

Containers and VMs side by side



```
 gageorsburn@mbp  ~  kubectl get po
NAME                          READY     STATUS    RESTARTS   AGE
metadata-7d77b9678d-pbz2f     1/1       Running   0          2m25s
ocserv-f8cbf9c44-xks47        1/1       Running   0          2m24s
tftp-8cb769fd4-lgncg          1/1       Running   0          2m24s
virt-launcher-p1-tqwcp        2/2       Running   0          2m19s
virt-launcher-p2-h68mf        2/2       Running   0          2m21s
virt-launcher-p3-2wb84        2/2       Running   0          2m24s
virt-launcher-p4-7zfjg        2/2       Running   0          2m20s
virt-launcher-pe11-nvvkl      2/2       Running   0          2m20s
virt-launcher-pe22-pw7f7      2/2       Running   0          2m24s
virt-launcher-pe33-c786m      2/2       Running   0          2m22s
virt-launcher-pe44-mzc65      2/2       Running   0          2m23s
```

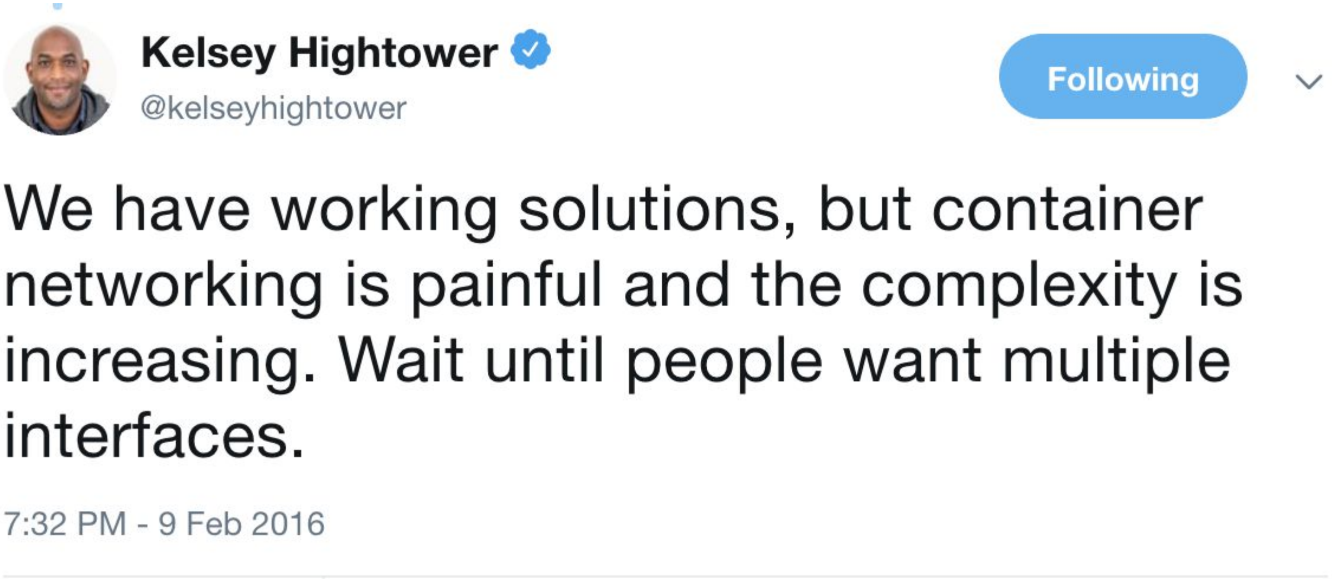
Metadata



```
gageorsburn@mbp  ~  curl metadata.c6b610f8-bc91-4e58-b68e-cbd75bed31e5.svc.packet-stage.osi.io/v1 | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   788  100   788    0     0   4377      0 --:--:-- --:--:-- --:--:--  4402
{
  "hosts": [
    {
      "hostname": "p1",
      "ip": "10.233.66.19",
      "os": "xr",
      "username": "lab",
      "password": "lab",
      "groups": [
        ""
      ]
    },
    {
      "hostname": "p2",
      "ip": "10.233.65.19",
      "os": "xr",
      "username": "lab",
      "password": "lab",
      "groups": [
        ""
      ]
    },
```

Why Multiple Interfaces?

A virtual router with one interface is not very interesting



Kelsey Hightower
@kelseyhightower

We have working solutions, but container networking is painful and the complexity is increasing. Wait until people want multiple interfaces.

7:32 PM - 9 Feb 2016

Why Multiple Interfaces: Multus

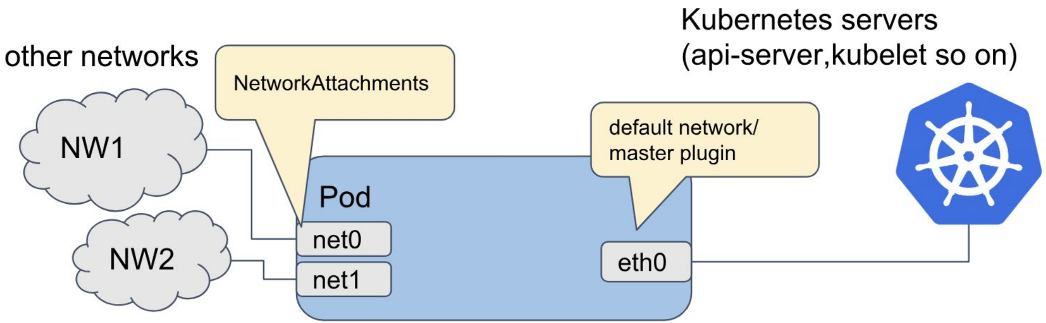
https://github.com/intel/multus-cni

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  name: ovsbr0
spec:
  config: '{ "cniVersion": "0.3.1", "type": "ovs", "bridge": "ovsbr0" }'
```

Why Multiple Interfaces: Multus

```
apiVersion: v1
kind: Pod
metadata:
 name: testpod1
 labels:
   env: test
 annotations:
   k8s.v1.cni.cncf.io/networks: linuxbr1,linuxbr1
```

Annotations use the *Network Attachment Definition* name to indicate how multiple interfaces are plumbed



other networks

NetworkAttachments

Kubernetes servers
(api-server,kubelet so on)

default network/
master plugin

- Specific User Traffic

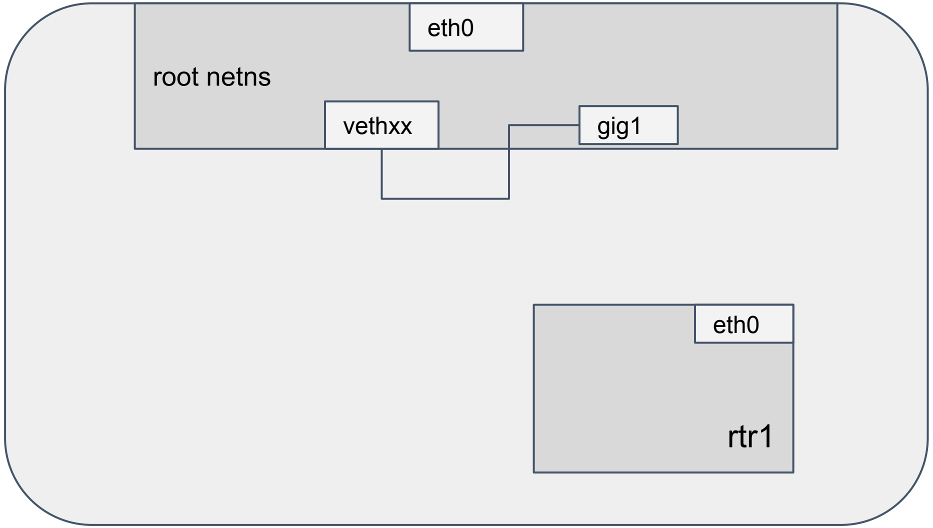
- (Liveness and Readiness) Probes
- Communication between API and Pod

CNI: Container Networking Interface

- Community CNI plugins
- Why not L2 Device Plugins?
- Writing a CNI plugin
- Increased complexity with containerized kubelet

CNI: Community plugins

- host-device
- ptp
- macvlan
- bridge

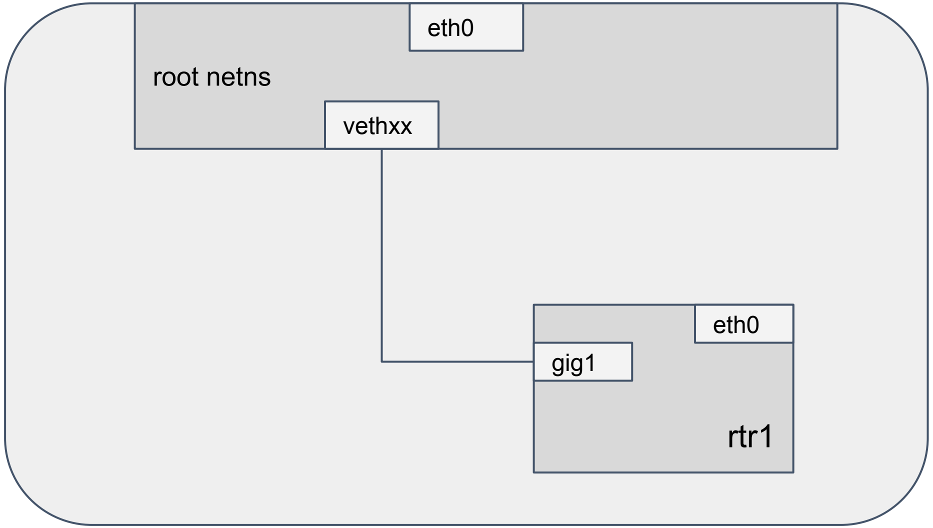


https://github.com/containernetworking/plugins

CNI: Community plugins

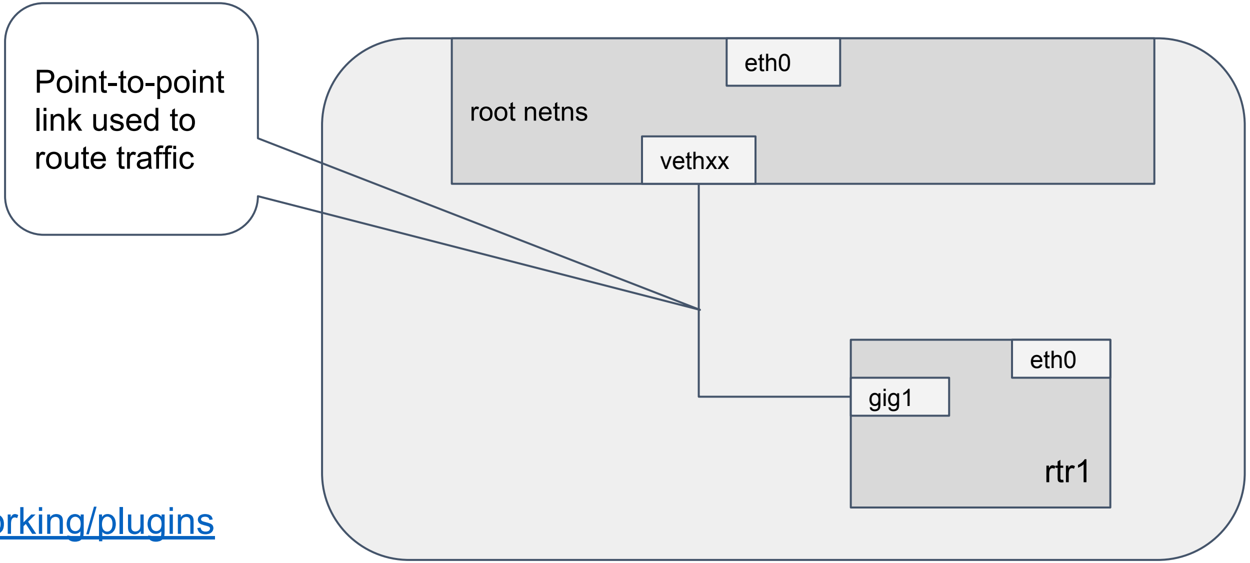
- host-device
- ptp
- macvlan
- bridge

https://github.com/containernetworking/plugins



CNI: Community plugins

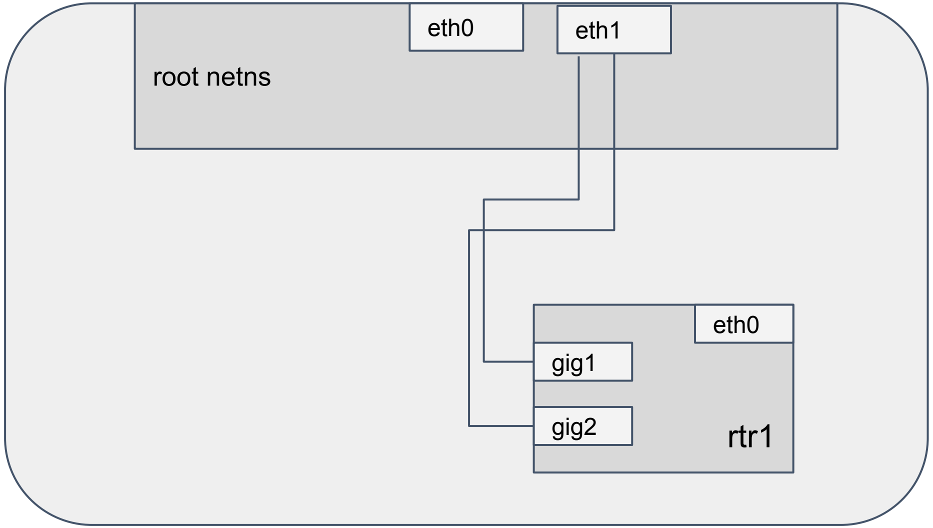
- host-device
- ptp
- macvlan
- bridge



https://github.com/containernetworking/plugins

CNI: Community plugins

- host-device
- ptp
- macvlan
- bridge



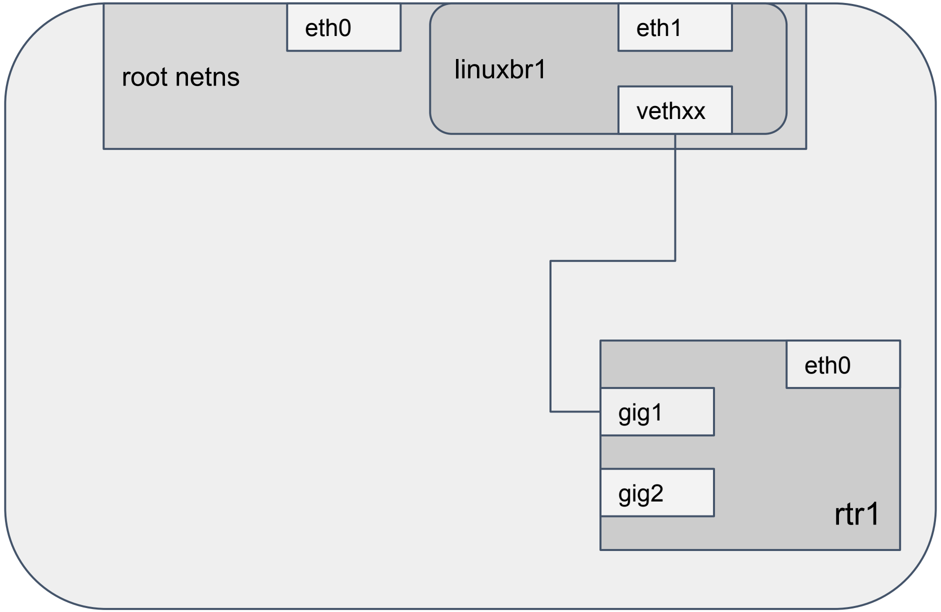
https://github.com/containernetworking/plugins

CNI: Community plugins

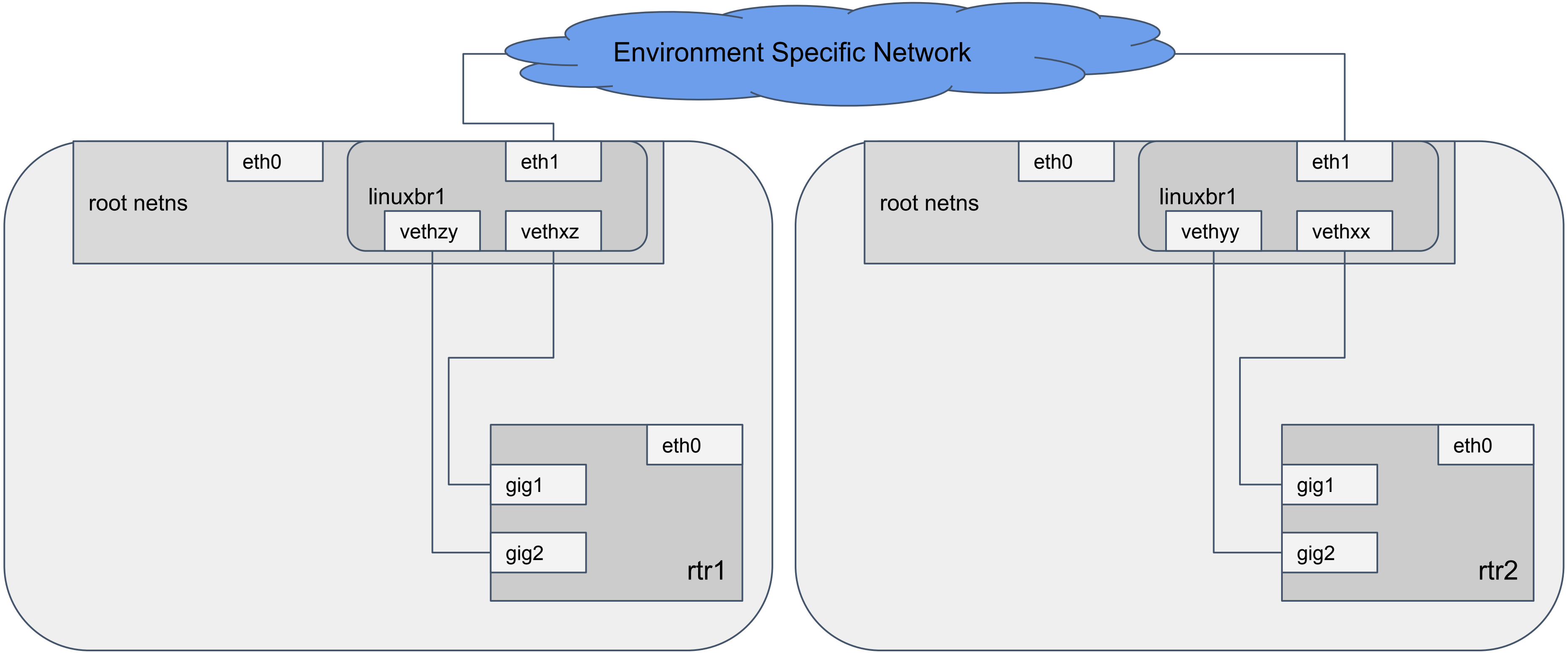
- host-device
- ptp
- macvlan
- bridge

```
{
    "cniVersion": "0.3.1",
    "name": "mynet",
    "type": "bridge",
    "bridge": "linuxbr1",
    "ipam": {}
}
```

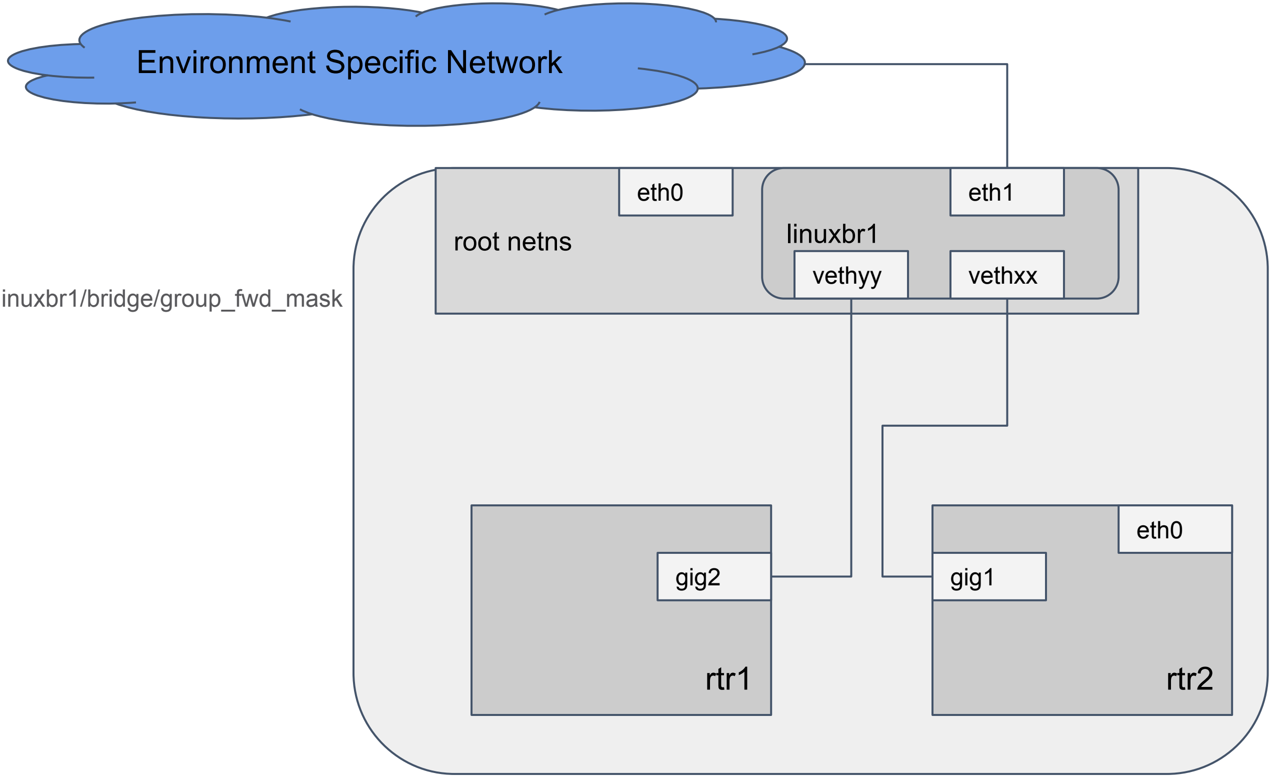
https://github.com/containernetworking/plugins



CNI: Community plugins



CNI: show lldp neighbors



Spanning Tree hack: brctl setageing linuxbr1 0

LLDP forwarding hack: echo 0x4000 > /sys/class/net/linuxbr1/bridge/group_fwd_mask

Optional hacking:
Multicast snooping
General Mac Learning
… Unsolved problem never discovered root cause

CNI: Writing a CNI Plugin

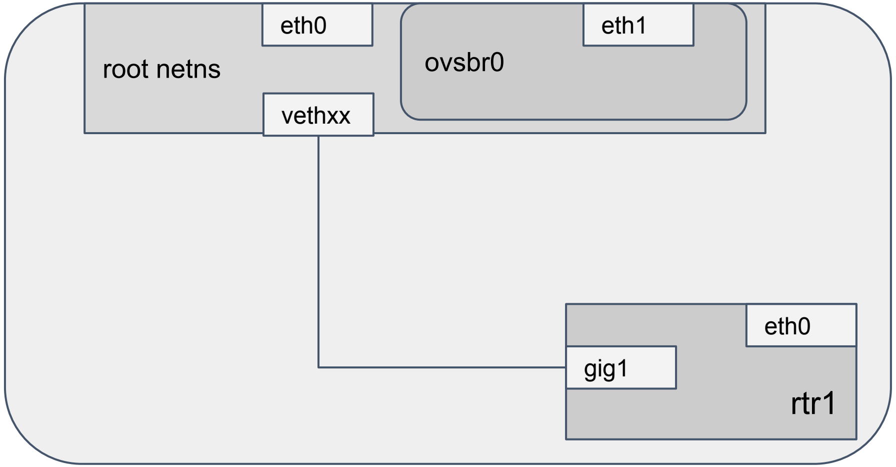
1. A CNI plugin is responsible for inserting a network interface into the container network namespace (e.g. one end of a veth pair)
2. Making any necessary changes on the host (e.g. attaching the other end of the veth into a bridge).
3. ~~It should then assign the IP to the interface and setup the routes consistent with the IP Address Management section by invoking appropriate IPAM plugin.~~

https://github.com/containernetworking/cni

CNI: Step 1

A CNI plugin is responsible for inserting a network interface into the container network namespace (e.g. one end of a veth pair)

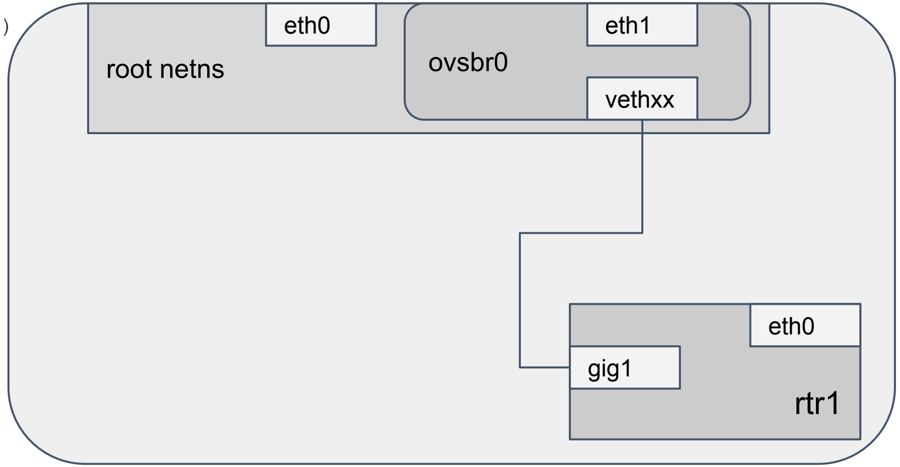
```
// create the veth pair in the container and move host end into host netns
hostVeth, containerVeth, err := ip.SetupVeth(ifName, mtu, hostNS)
if err != nil {
       return err
}
hostIface.Mac = hostVeth.HardwareAddr.String()
contIface.Name = containerVeth.Name
```



CNI: Step 2

Making any necessary changes on the host (e.g. attaching the other end of the veth into a bridge).

```
command := []string{
        "--", "add-port", brName, hostIfaceName,
        "--", "set", "Port", hostIfaceName, fmt.Sprintf("external-ids:contNetns=%s", contNetnsPath),
        "--", "set", "Port", hostIfaceName, fmt.Sprintf("external-ids:contIface=%s", contIfaceName),
        }
output, err := exec.Command("ovs-vsctl", command...).CombinedOutput()
if err != nil {
        return err
}
```



Containerized Kubelet

```
func AddBridge(name string) error {
      _, stderr, err := run("ovs-vsctl", "add-br", name)
```

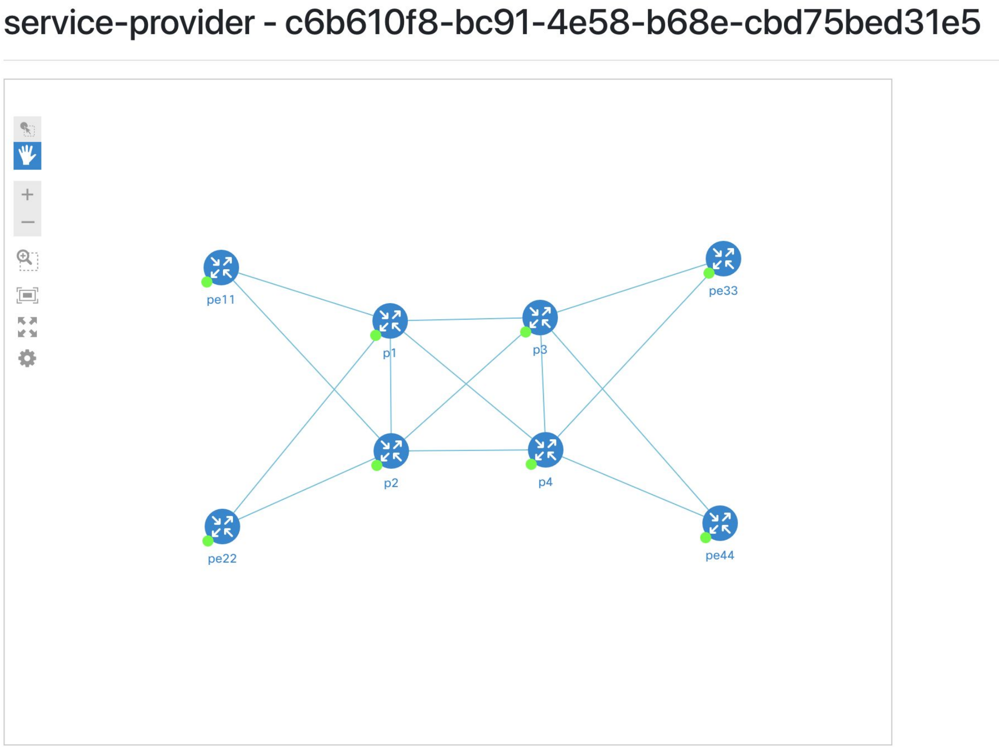
Looking into all repositories on github for

“openvswitch” lang: go

Very few golang libraries for openvswitch implement the ovs spec for transacting ovsdb operations. Wrapping ovs-vsctl binary.

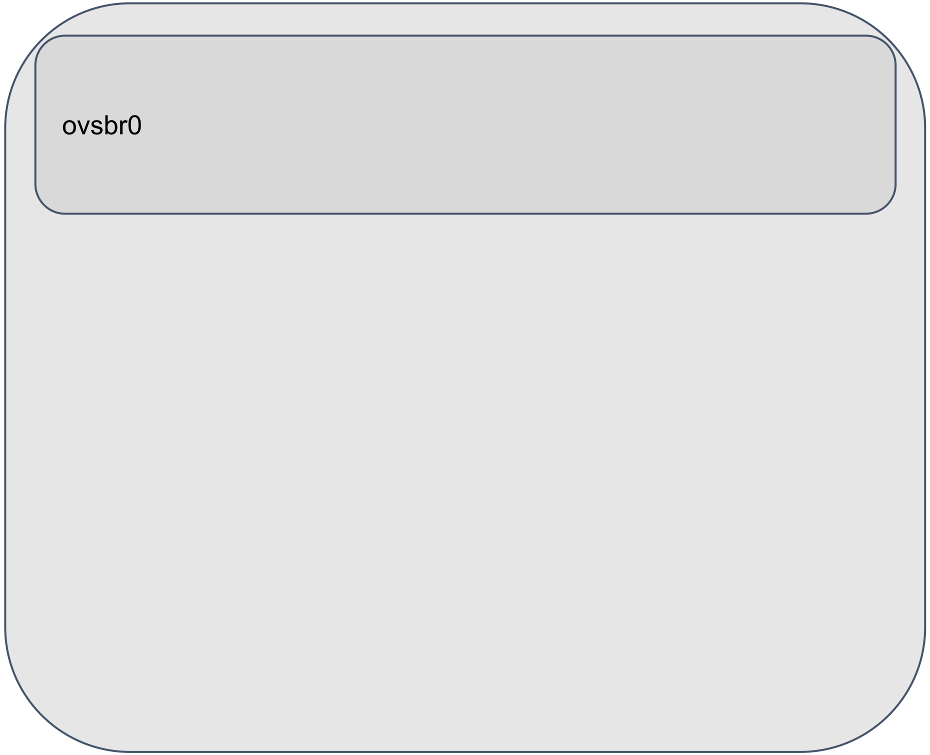
Production hosts using atomic / coreos

Network topology stitching

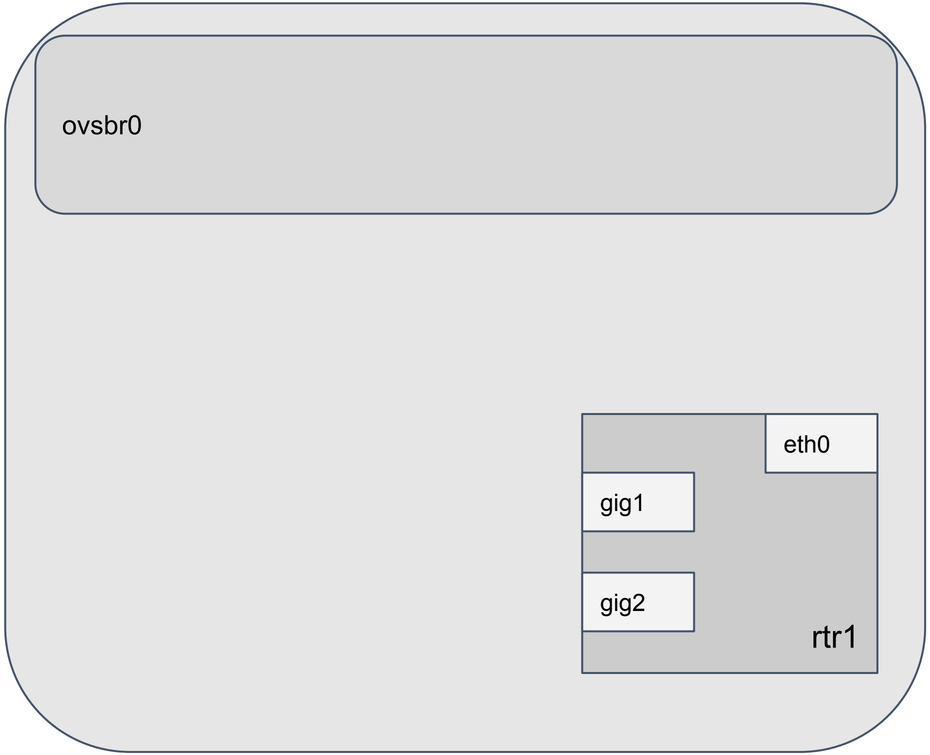


service-provider - c6b610f8-bc91-4e58-b68e-cbd75bed31e5

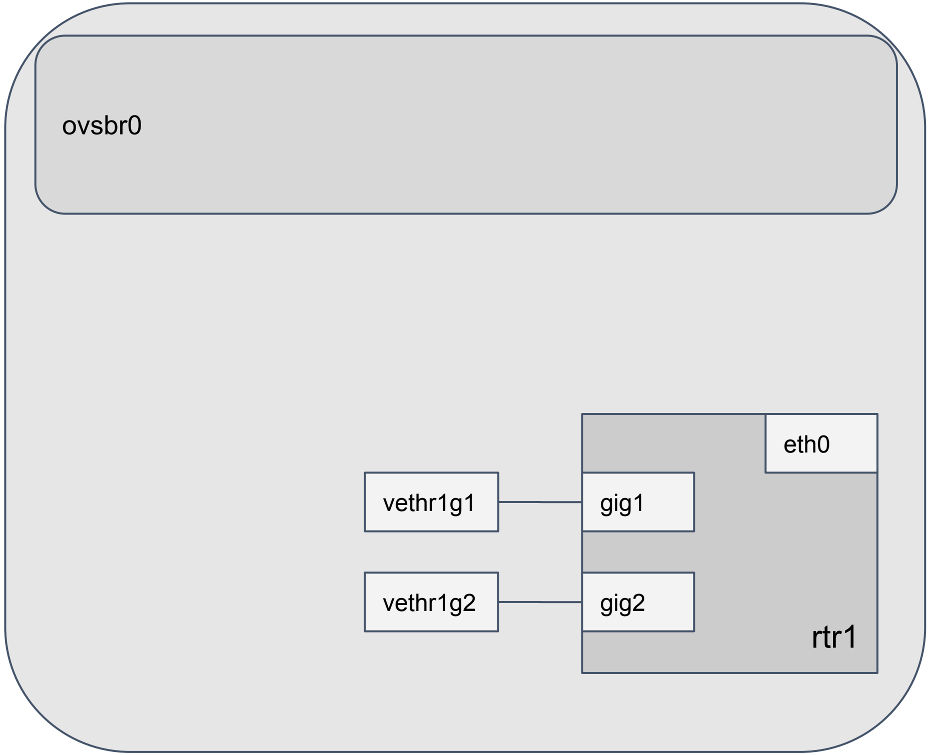
Local Node Programming



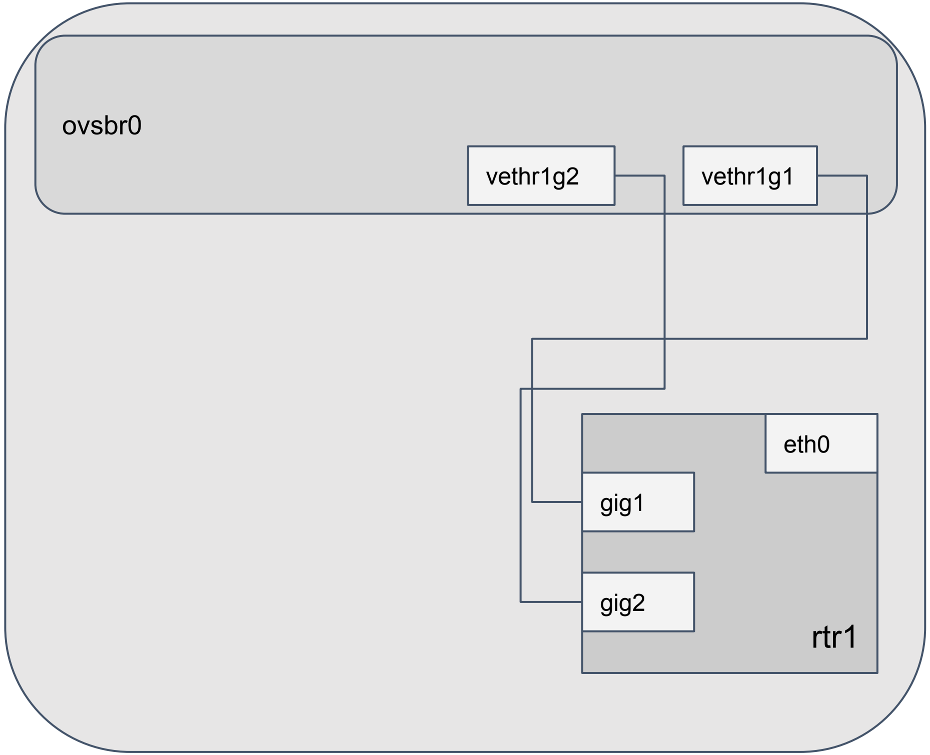
Local Node Programming



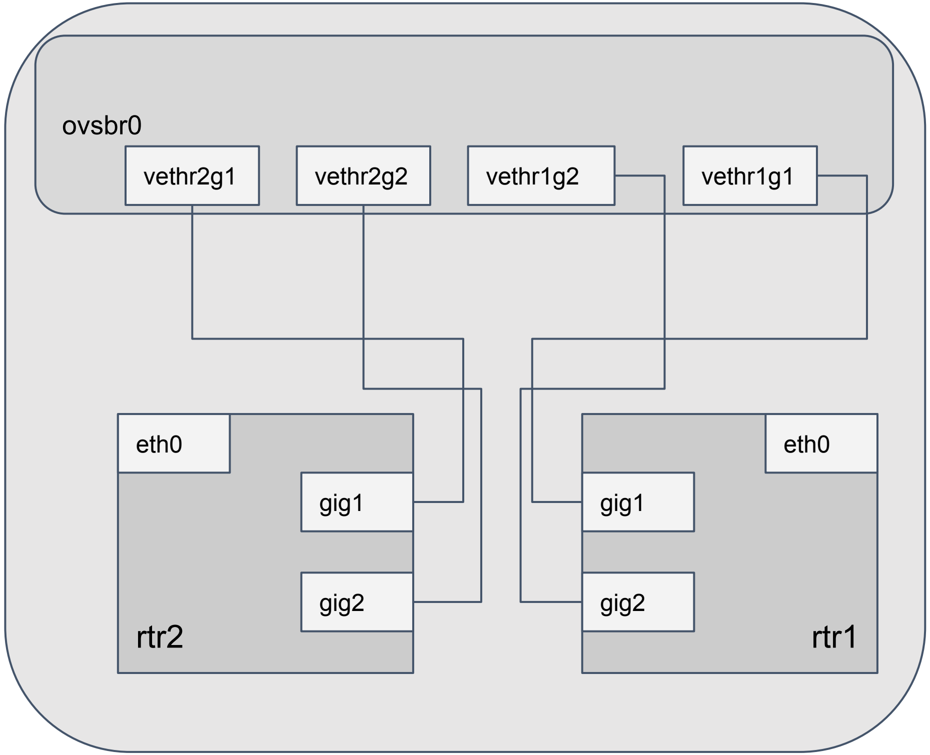
Local Node Programming



Local Node Programming



Local Node Programming



Local Node Programming

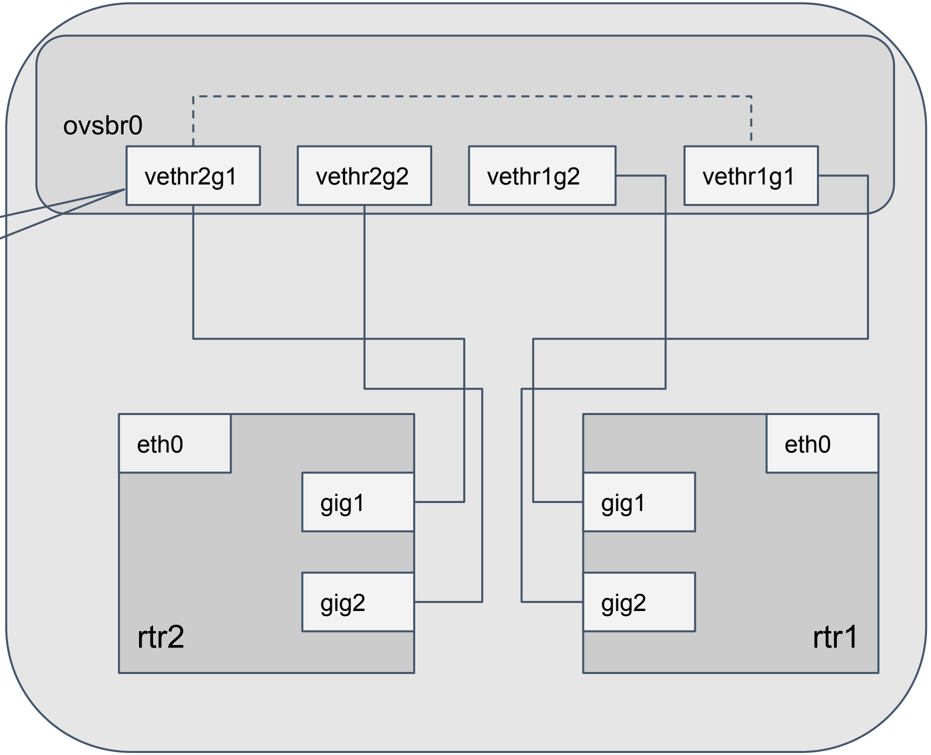
```
ovs-vsctl  set Bridge ovsbr0 fail-mode=secure

ovs-ofctl  add-flow ovsbr0 “table=0, in_port=21,actions=11”
```

Openvswitch allocates an internal port id when an interface is added.

vethr2g1 == 21

vethr1g1 == 11



Local Node Programming

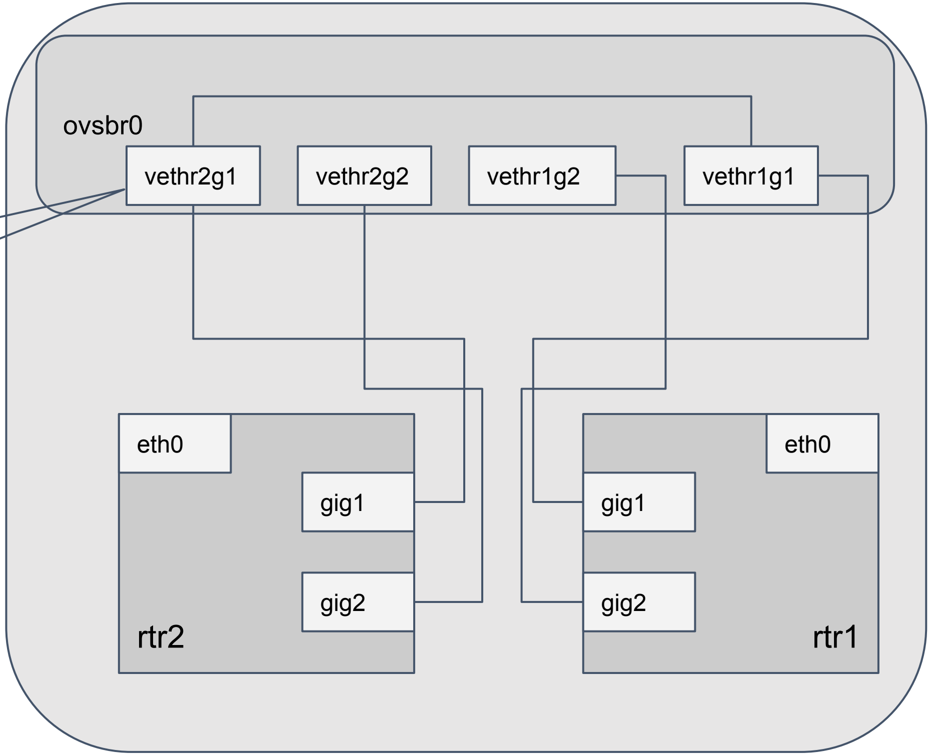
```
ovs-vsctl  set Bridge ovsbr0 fail-mode=secure

ovs-ofctl  add-flow ovsbr0 “table=0, in_port=21,actions=11”
ovs-ofctl  add-flow ovsbr0 “table=0, in_port=11,actions=21”
```

Openvswitch allocates an internal port id when an interface is added.

vethr2g1 == 21

vethr1g1 == 11



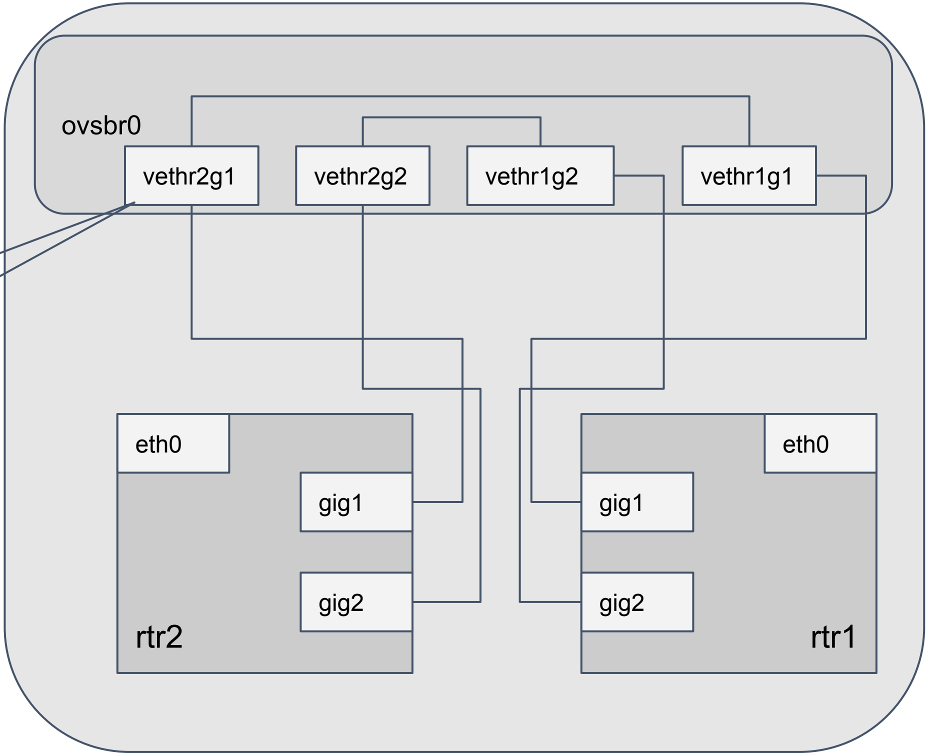
Local Node Programming

```
ovs-vsctl  set Bridge ovsbr0 fail-mode=secure

ovs-ofctl  add-flow ovsbr0 “table=0, in_port=21,actions=11”
ovs-ofctl  add-flow ovsbr0 “table=0, in_port=11,actions=21”
ovs-ofctl  add-flow ovsbr0 “table=0, in_port=22,actions=12”
ovs-ofctl  add-flow ovsbr0 “table=0, in_port=12,actions=22”
```

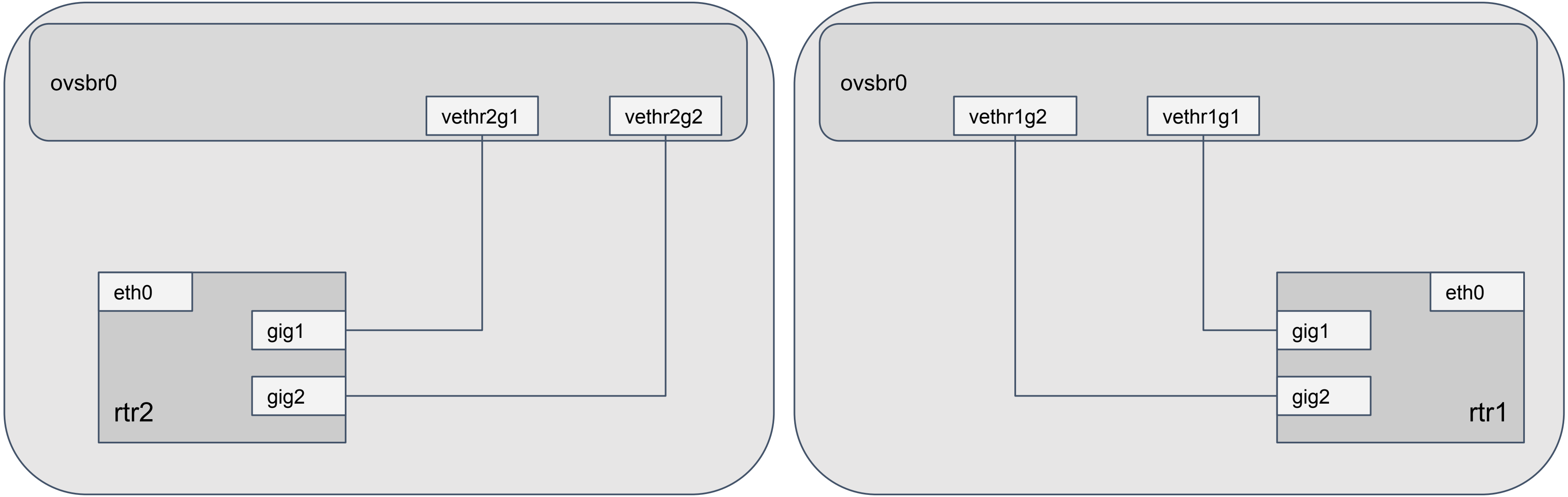
Openvswitch allocates an internal port id when an interface is added.

vethr2g1 == 21
vethr2g2 == 22
vethr1g1 == 11
vethr1g2 == 12

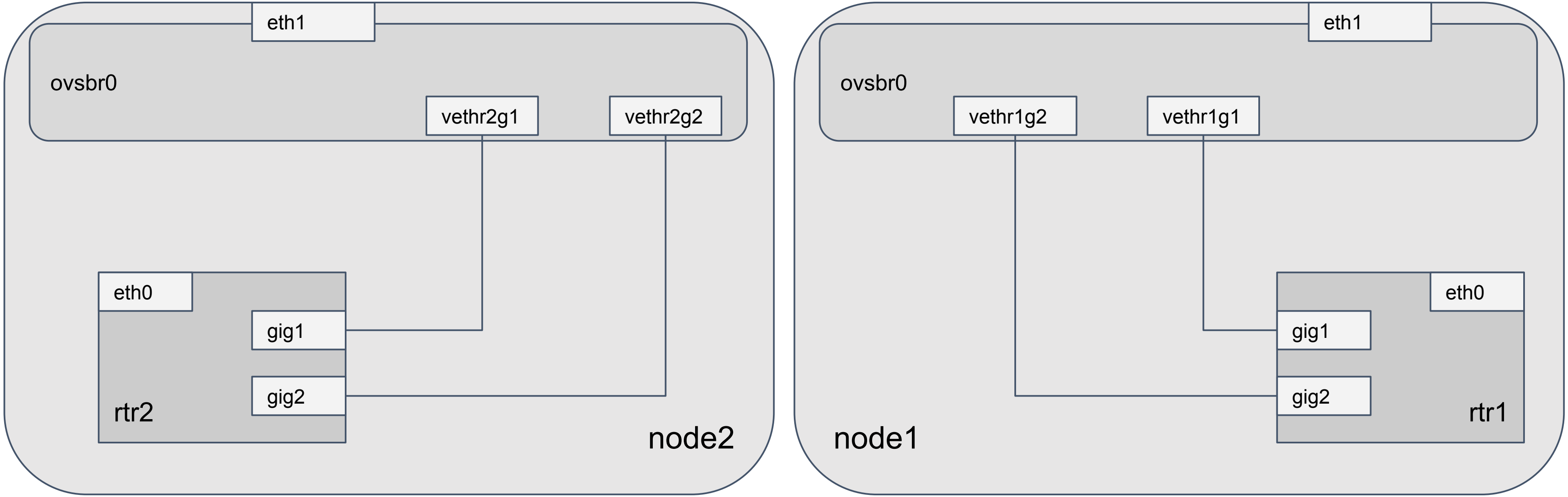


Remote Node Programming

KubeCon | CloudNativeCon North America 2018

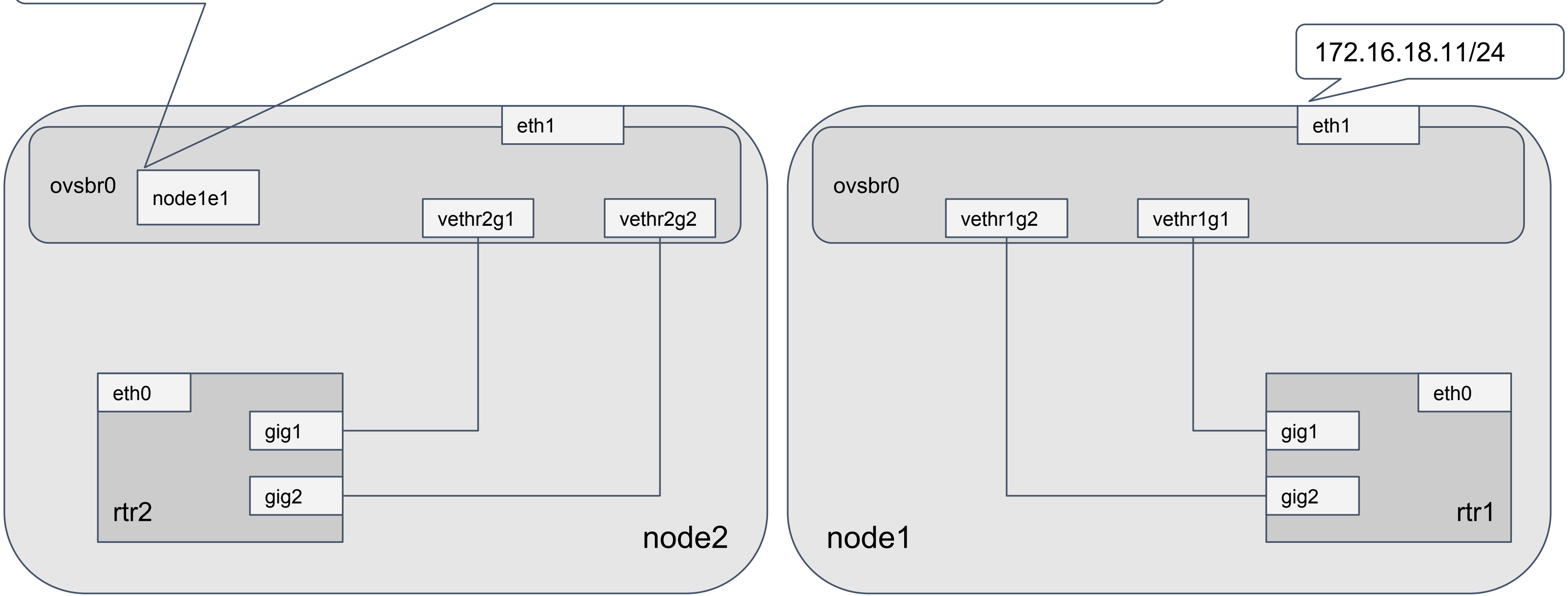


Remote Node Programming



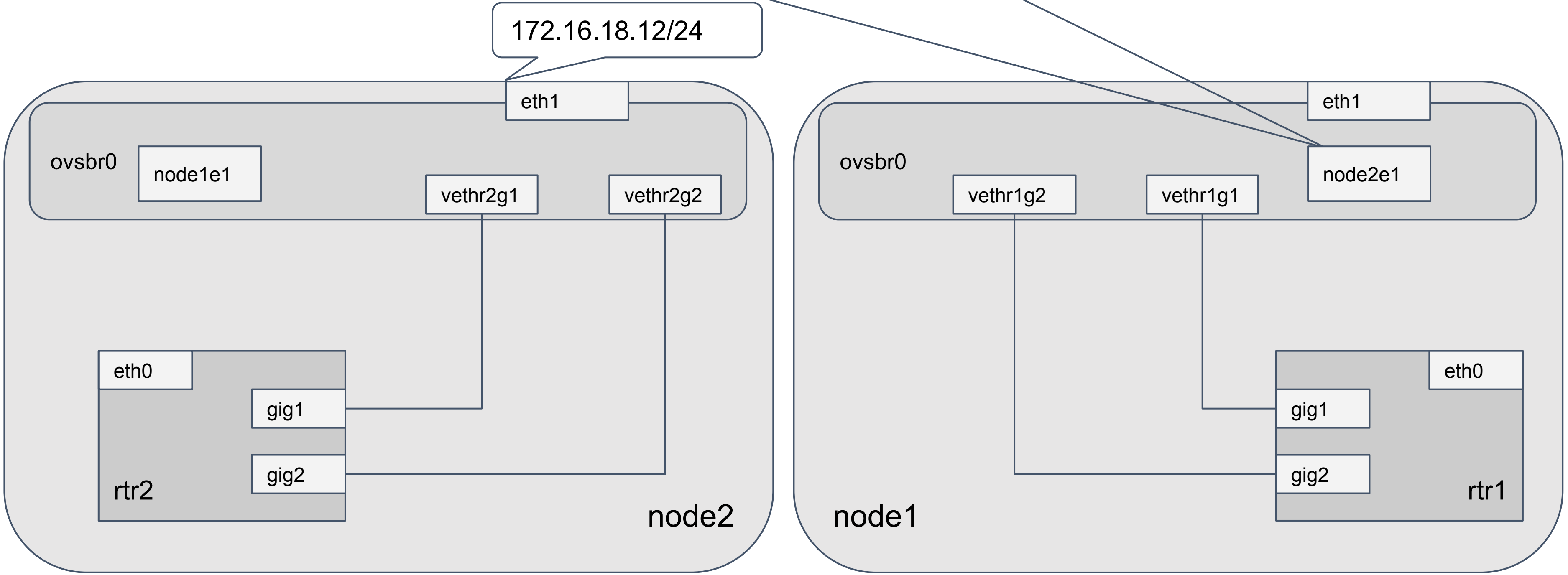
Remote Node Programming

```
ovs-vsctl add-port ovsbr0 node1e1
ovs-vsctl set Interface node1e1 type=vxlan options:remote_ip=172.16.18.11 options:key=flow
```



Remote Node Programming

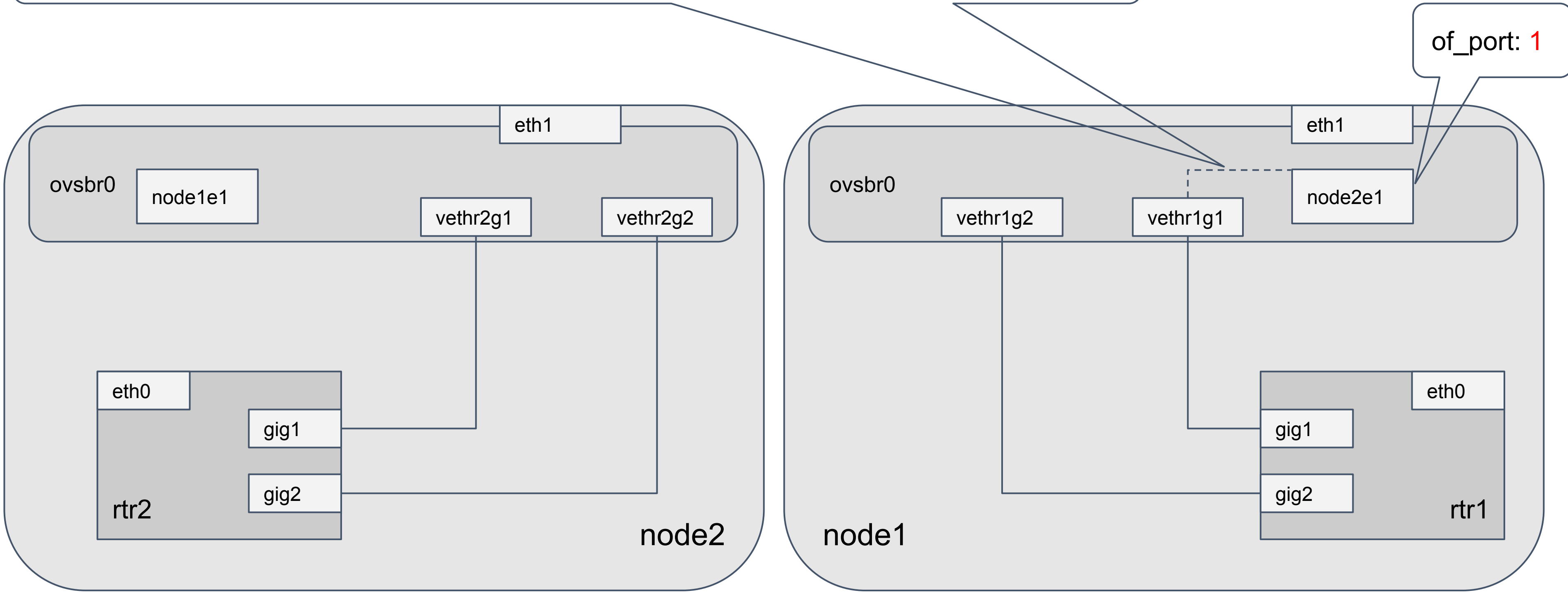
```
ovs-vsctl add-port ovsbr0 node2e1
ovs-vsctl set Interface node2e1 type=vxlan options:remote_ip=172.16.18.12 options:key=flow
```



Remote Node Programming

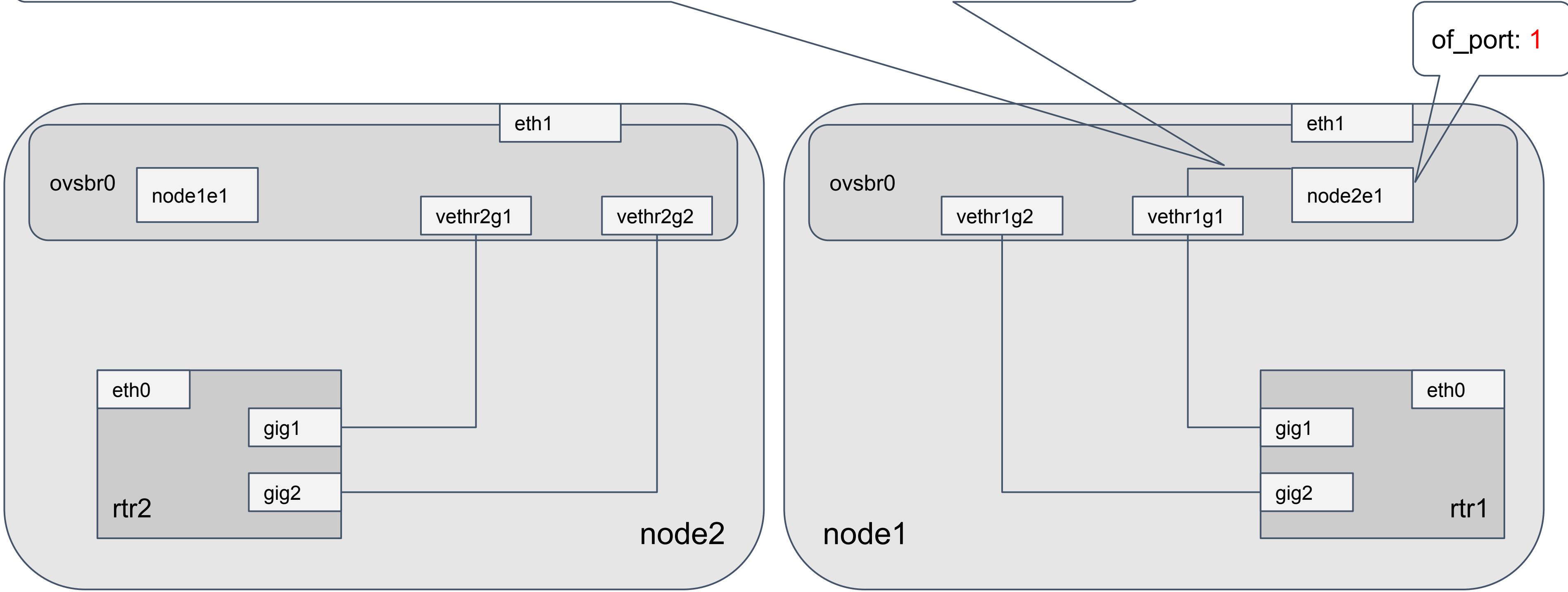
```
ovs-ofctl  add-flow ovsbr0 “table=0, in_port=11,actions=set_tunnel:20001,1”
```

of_port: 1



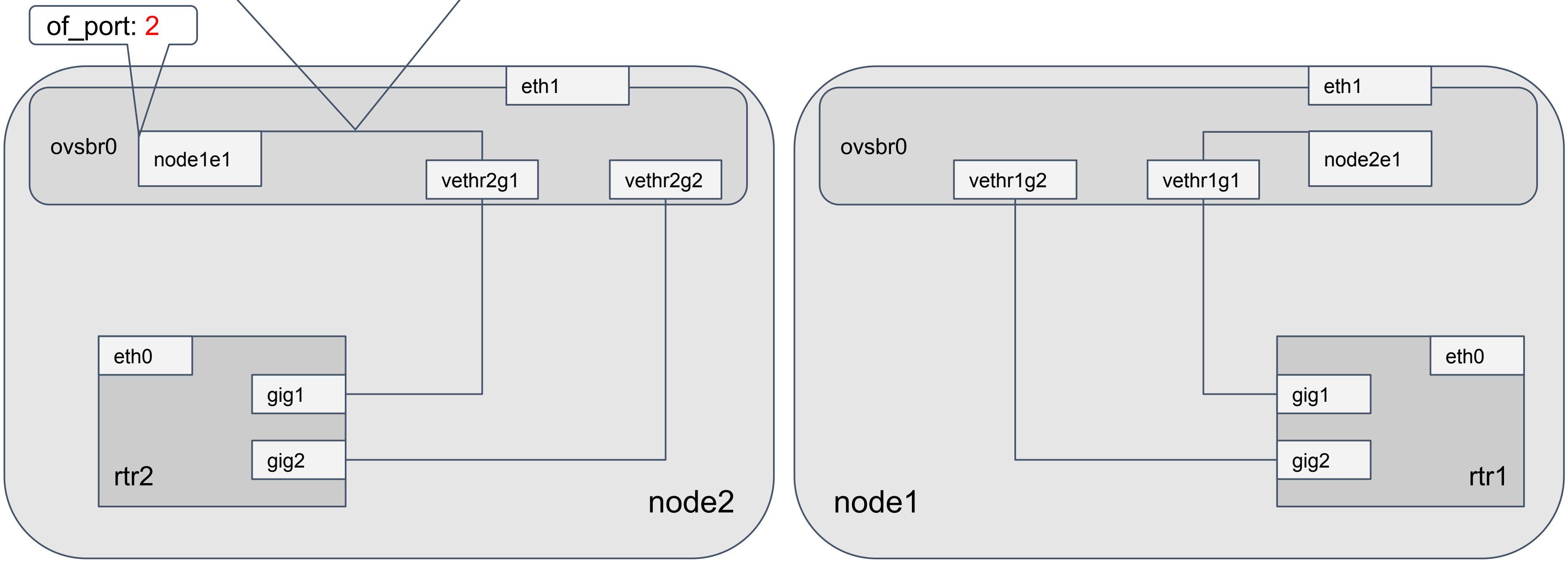
Remote Node Programming

```
ovs-ofctl  add-flow ovsbr0 “table=0, in_port=11 actions=set_tunnel:20001,1”
ovs-ofctl  add-flow ovsbr0 “table=0, in_port=1,tun_id=20001 actions=11”
```



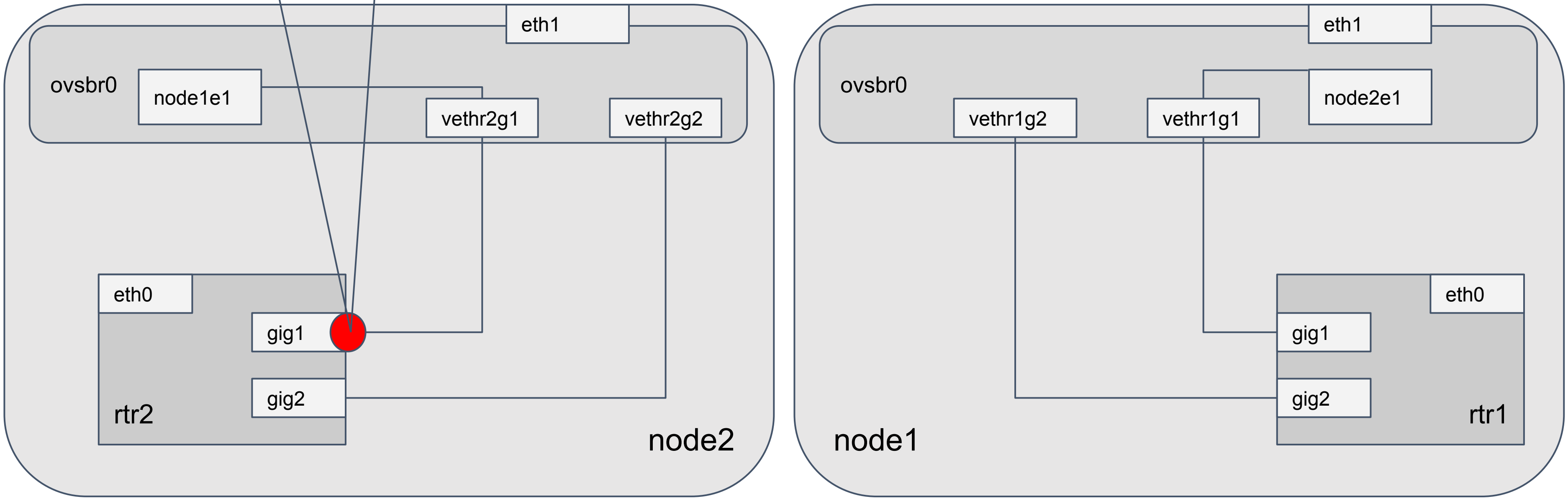
Remote Node Programming

```
ovs-ofctl  add-flow ovsbr0 “table=0, in_port=21 actions=set_tunnel:20001,2”
ovs-ofctl  add-flow ovsbr0 “table=0, in_port=2,tun_id=20001 actions=21”
```



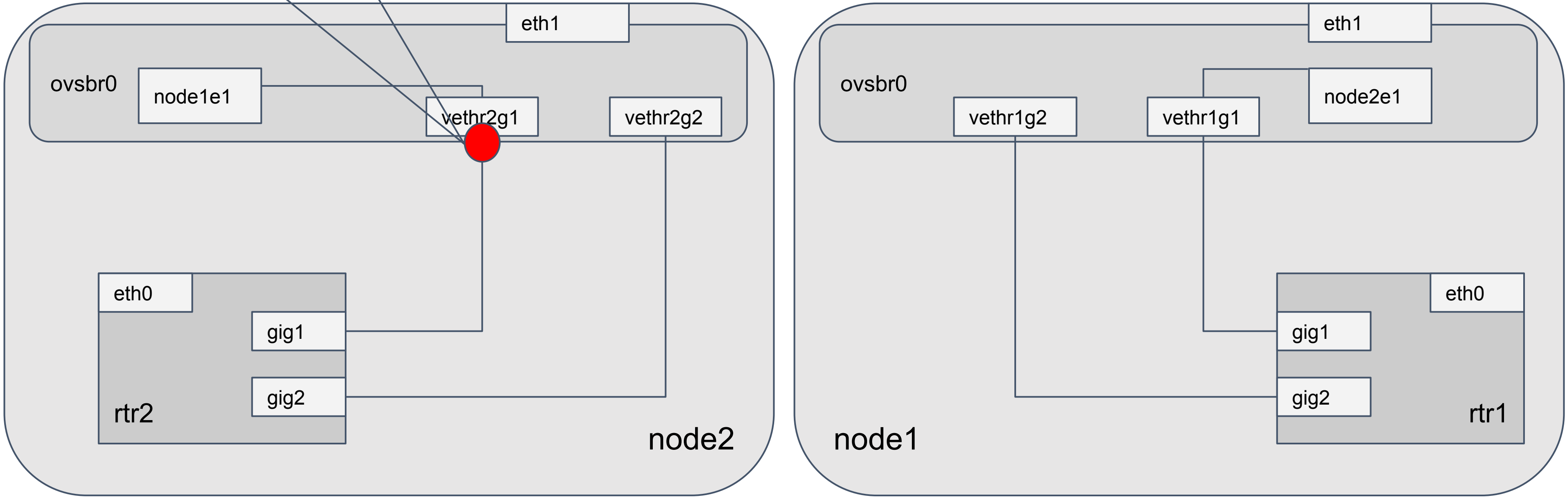
Remote Node Programming

Fwd: Standard Ethernet fwd
Encap: none
Src: container veth
Dst: host veth



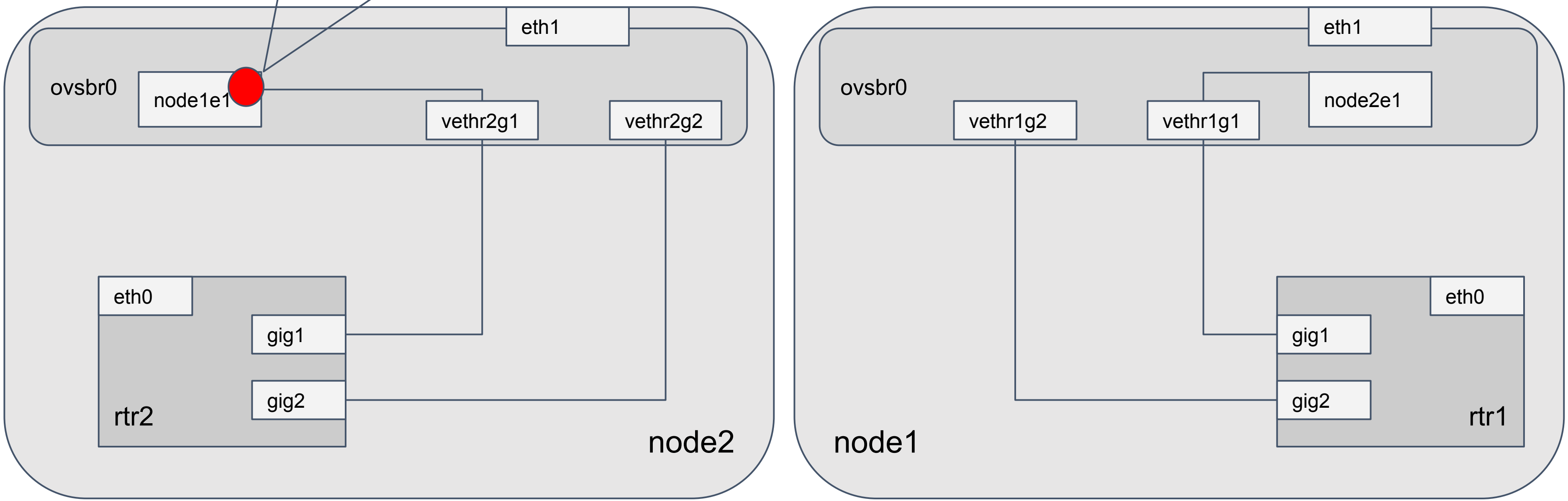
Remote Node Programming

Fwd: Openflow Rule
Encap: none
Src: of_port 21
Dst: of_port 11



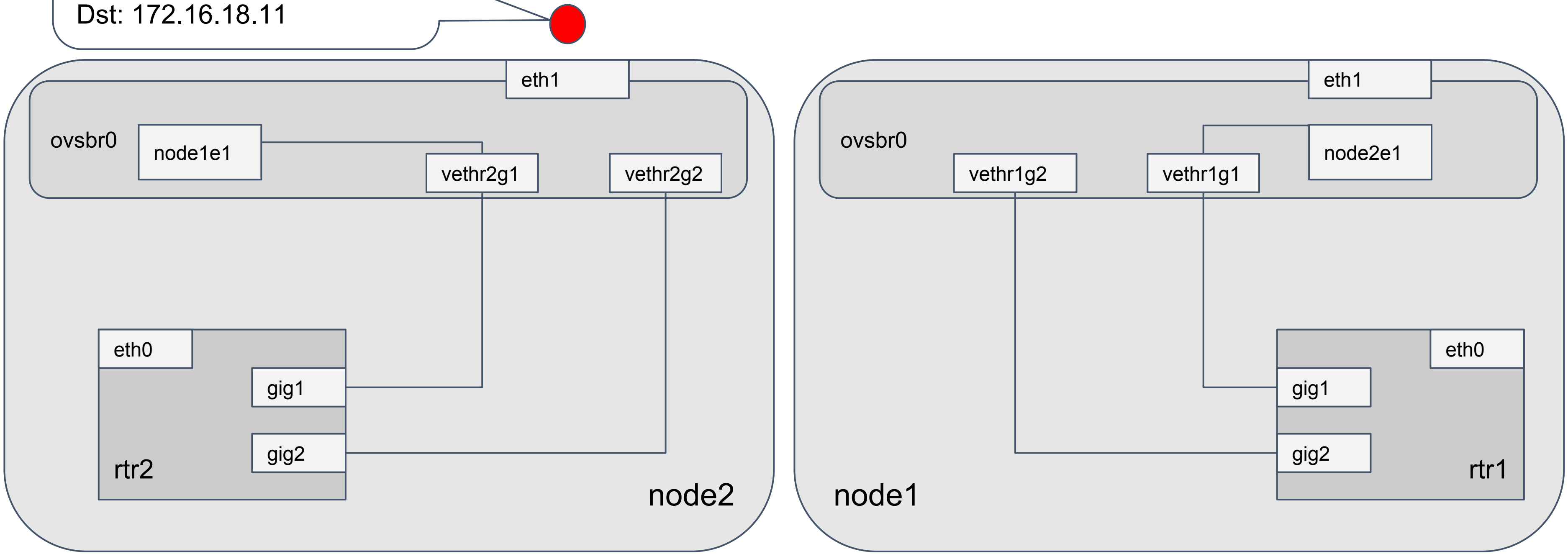
Remote Node Programming

Fwd: Kernel Routing
Encap: vxlan, id 20001
Src: 172.16.18.12
Dst: 172.16.18.11



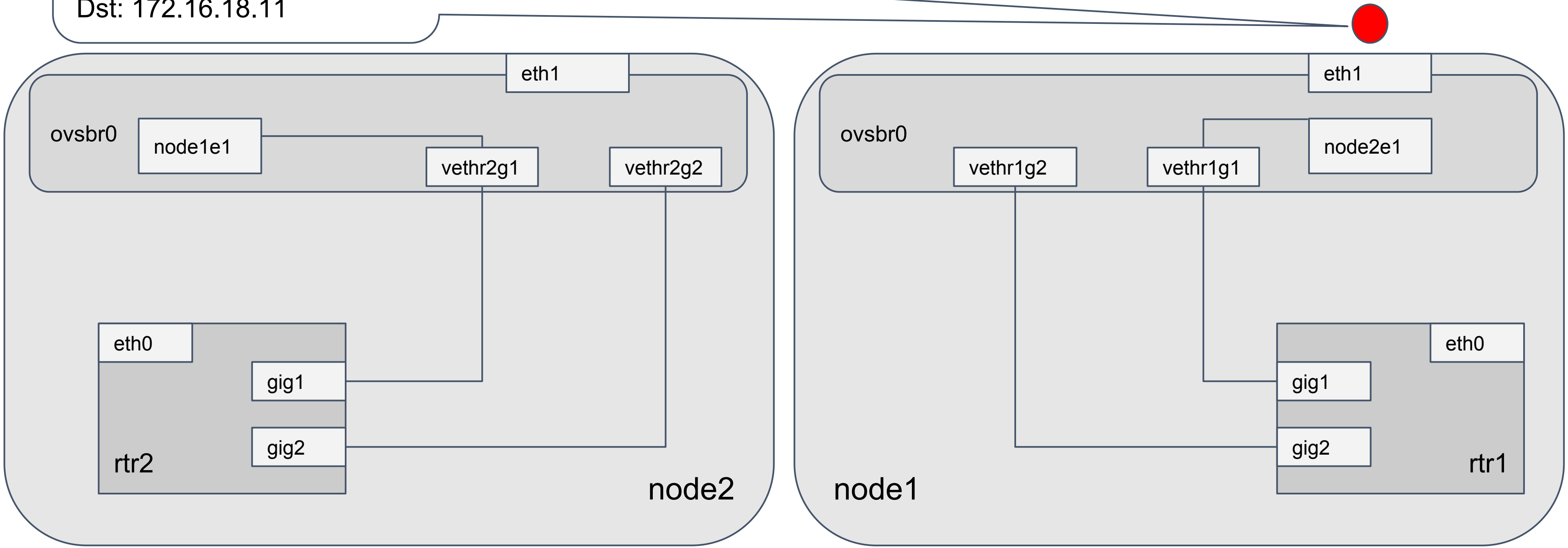
Remote Node Programming

Fwd: Environment Routing
Encap: vxlan, id 20001
Src: 172.16.18.12
Dst: 172.16.18.11



Remote Node Programming

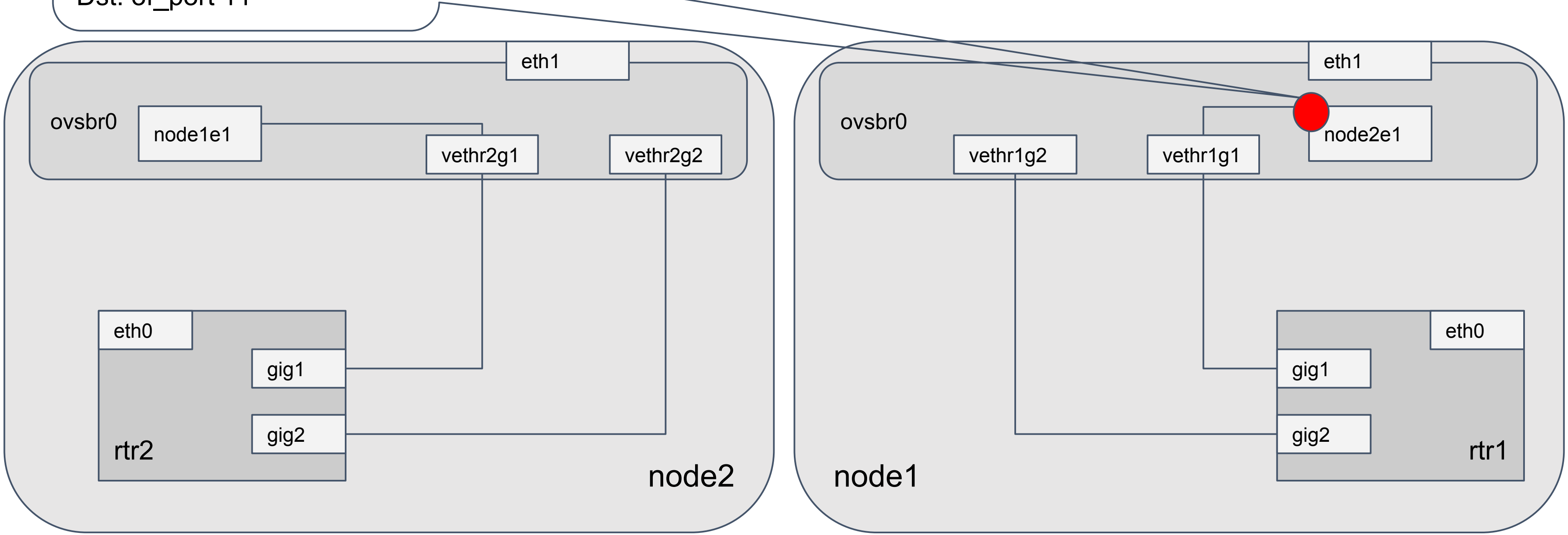
Fwd: Environment Routing
Encap: vxlan, id 20001
Src: 172.16.18.12
Dst: 172.16.18.11



Remote Node Programming

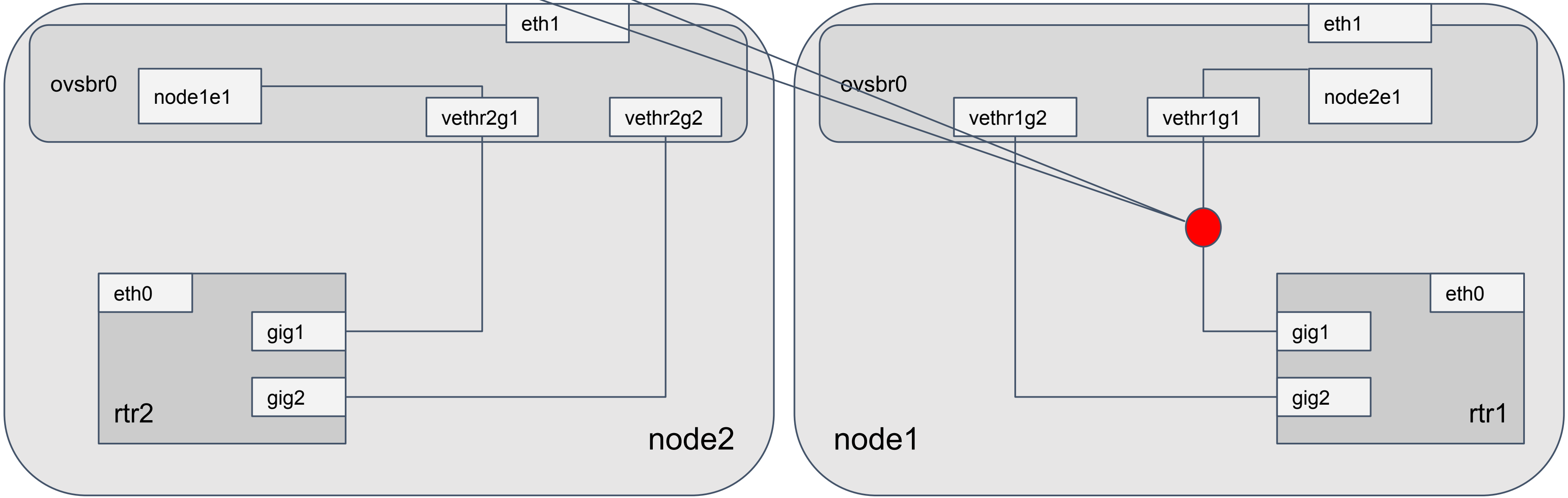
KubeCon | CloudNativeCon North America 2018

Fwd: Openflow Rule
Encap: none
Src: of_port 1
Dst: of_port 11



Remote Node Programming

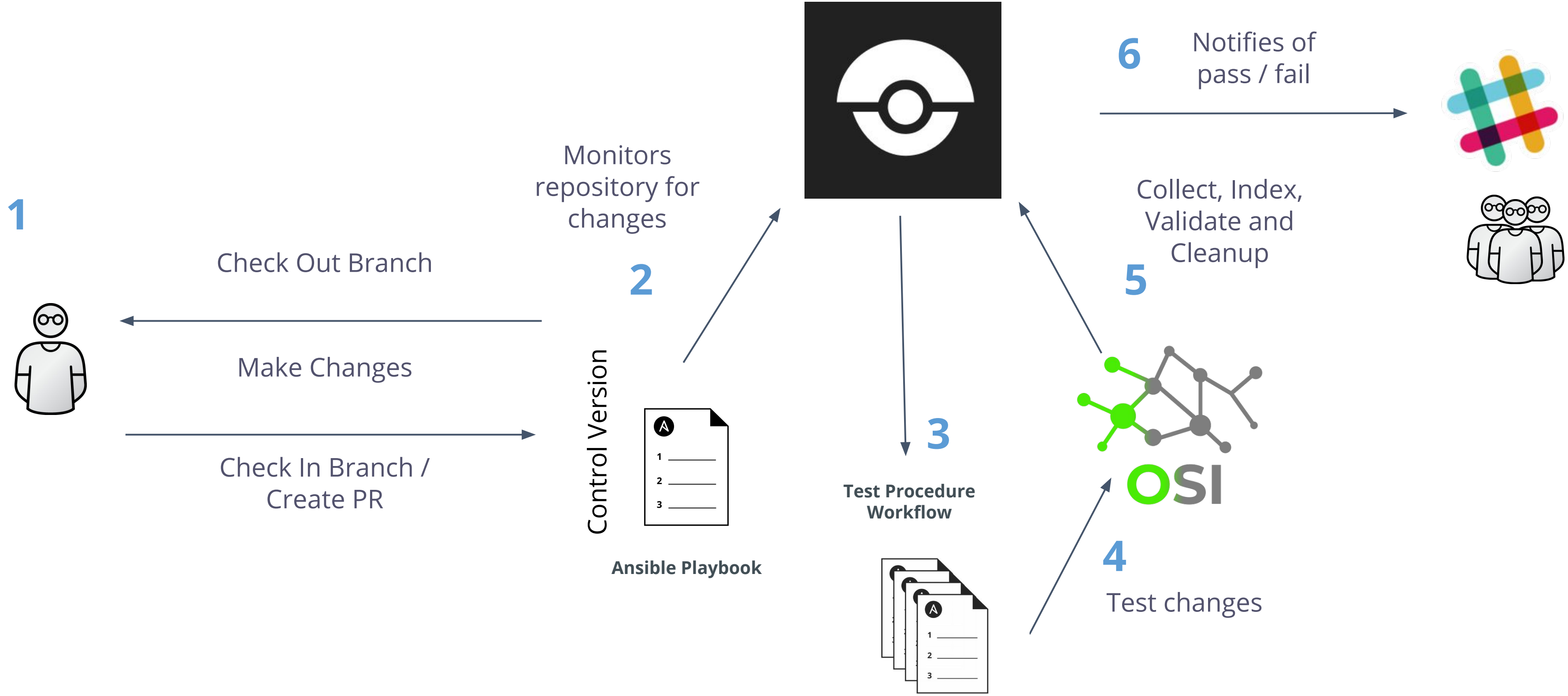
Fwd: Standard Ethernet fwd
Encap: none
Src: host veth
Dst: container veth



Network CI/CD Workflow

1. Define and build virtual network topology
2. Run automated topology provisioning and configuration
3. Perform integration testing to verify baseline and desired state
4. Generate network traffic and initiate testing triggers
5. Verify traffic state results vs. desired state
6. Virtual tear down of topology, freeing up resources until next cycle

Network CI/CD Workflow



1

Check Out Branch

Make Changes

Check In Branch / Create PR

Control Version

Ansible Playbook

Monitors repository for changes

2

3

Test Procedure Workflow

4

Test changes

5

Collect, Index, Validate and Cleanup

6

Notifies of pass / fail

The Witzke story

Best Part? Network engineers don’t need to know Kubernetes.

- Enabling GitOps for network engineers
- Easy mode change windows
- pSIRT API subscription triggering upgrade tests
- Convert brownfield network deployment to infrastructure as code

Other Potential Applications

Testing/Integrating VNF

Network Domain ML/AI applications and testing

Contact us!

Website: https://osi.io
Twitter: @network_ci
OSI YouTube Channel

renner@osi.io
gage@osi.io

OSI
One Source Integrations

KubeCon | CloudNativeCon

North America 2018