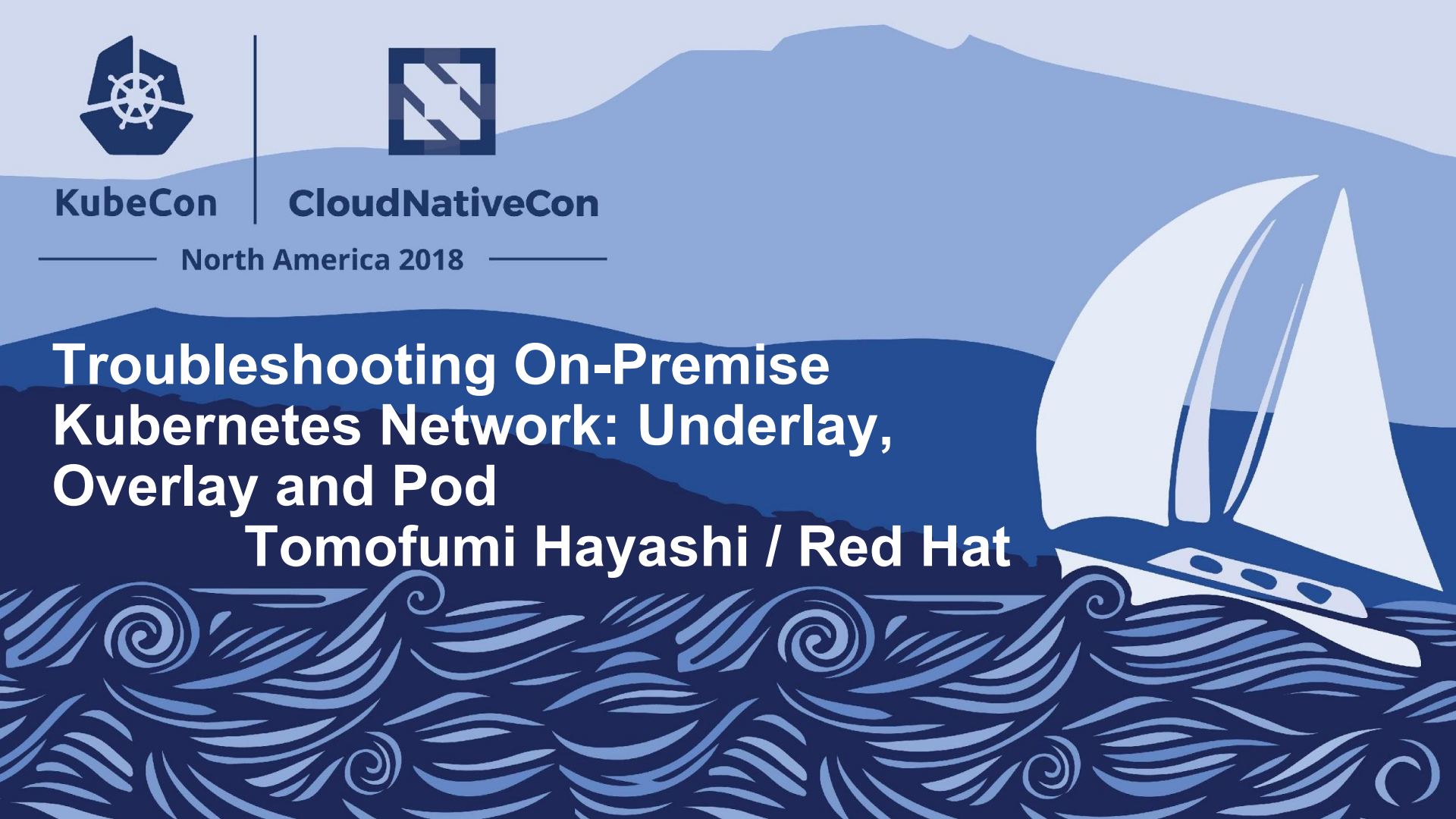


Troubleshooting On-Premise Kubernetes Network: Underlay, Overlay and Pod

Tomofumi Hayashi / Red Hat

Agenda

- Why Kubernetes Network is Difficult?
- Kubernetes and Network Setup
- Troubleshooting
  - How to Identify Container Interface?
  - iptables
  - Packet Capture
    - Demo

This slide URL

Why Kubernetes Network is Difficult?

- Many decision at design
- Co-existent multi-layered network
- Packet modification

This slide URL

Why Kubernetes Network is Difficult? (Cont'd)

Need to consider a lot of design decision:

- Host network
  - DC network, Virtual Network (if you launch it as VM, e.g. OpenStack)
- Container network
  - Overlay network v.s. Non-overlay network
  - IP Address Management (i.e. IPAM)
  - Network Policy
  - Bandwidth Management (e.g. Traffic Shaping)
  - ...
- Kubernetes resources
  - Load Balancer Service [y/n]?
  - Network Policy [y/n]?
  - Ingress [y/n]?

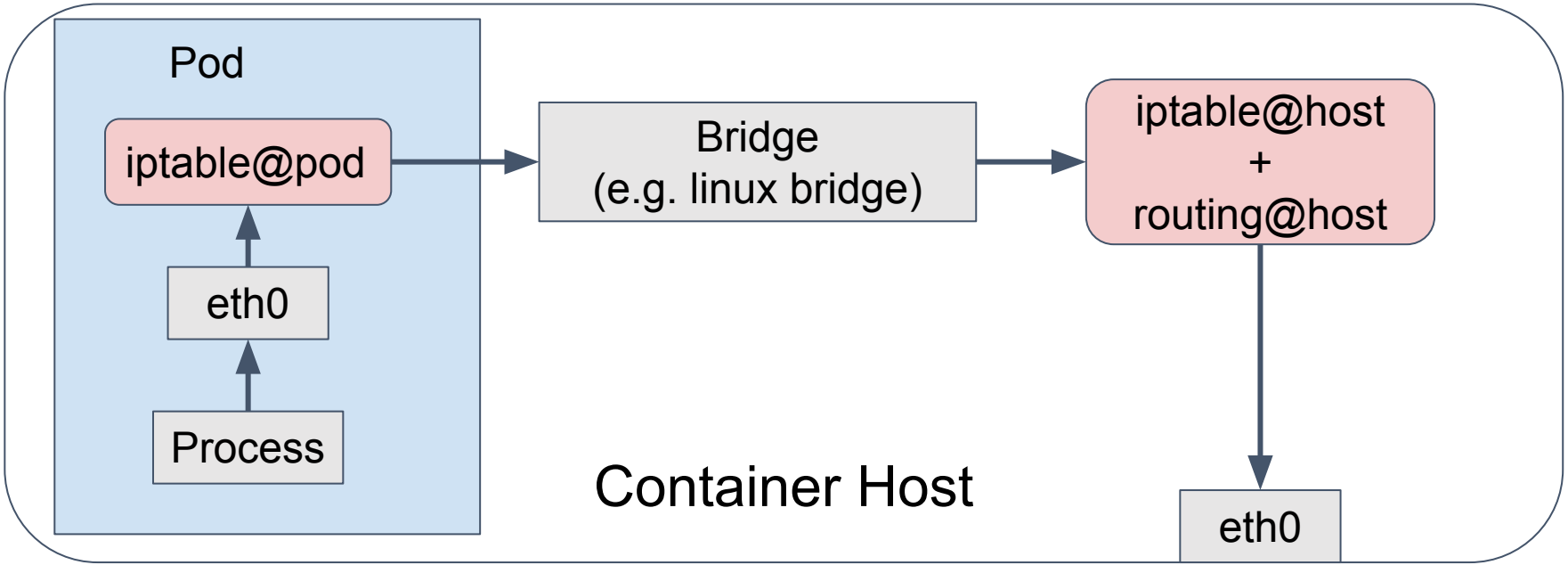
Why Kubernetes Network is Difficult? (Cont'd)

Co-existent Multi-layered Network:

- Host Network
- Container Network (Packet is written everywhere!)
  - Software Forwarding Plane (if exists)
  - iptables rules@host (added by kubernetes, i.e. kube-proxy/kubelet)
  - iptables rules@host (added by container network)
  - iptables rules@pod
- Kubernetes Network Resource
  - Service (iptables or ipvs)
  - LoadBalancer
- Others (bandwidth)

Why Kubernetes Network is Difficult? (Cont'd)

Rewrite packets everywhere!



if ipvs/ovs is used, it could also rewrite packet!

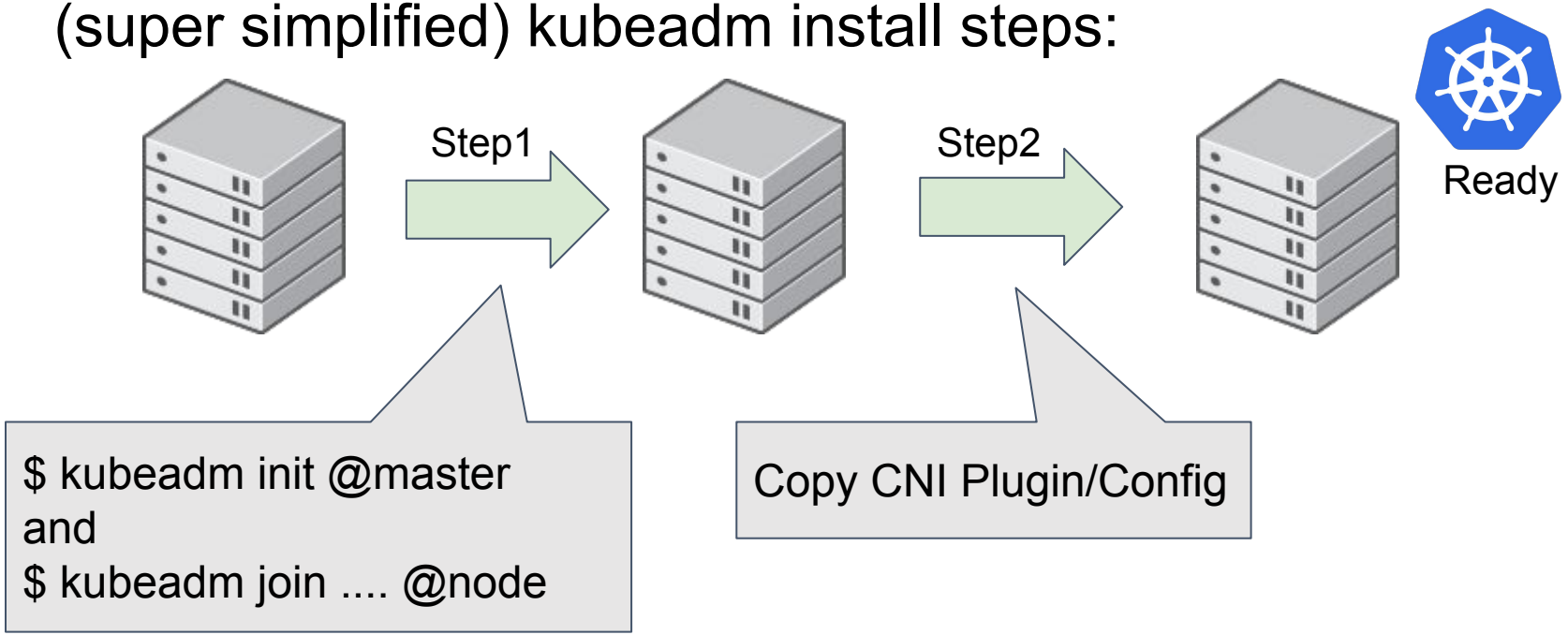
Kubernetes and Network Setup

Purpose

- 1st Step of On-premise Kubernetes
- For troubleshooting at setup
- Clarify when container network is ready

Kubernetes and Network Setup (by kubeadm)

(super simplified) kubeadm install steps:



Kubernetes and Network Setup (by kubeadm) (Cont'd)

Step1) "kubeadm init" and "kubeadm join ..."

```
[centos@kube-master ~]$ kubectl get node
NAME          STATUS     ROLES    AGE     VERSION
kube-master   NotReady   master   1m      v1.11.4
kube-node-1   NotReady   <none>   1m      v1.11.4
kube-node-2   NotReady   <none>   1m      v1.11.4
kube-node-3   NotReady   <none>   1m      v1.11.4
```

Node is registered, but "NotReady"

Kubernetes and Network Setup (by kubeadm) (Cont'd)

Step1) "kubeadm init" and "kubeadm join ..."

```
[centos@kube-master ~]$ kubectl get pod --all-namespaces
NAMESPACE     NAME                                   READY   STATUS    RESTARTS   AGE
kube-system   coredns-78fcdf6894-fq82f               0/1     Pending   0          4m
kube-system   coredns-78fcdf6894-trqrn               0/1     Pending   0          4m
kube-system   etcd-kube-master                       1/1     Running   0          3m
kube-system   kube-apiserver-kube-master             1/1     Running   0          3m
kube-system   kube-controller-manager-kube-master    1/1     Running   0          3m
kube-system   kube-proxy-dv88s                       1/1     Running   0          4m
(snip)
```

k8s pods are Running, but coredns pod is pending

Kubernetes and Network Setup (by kubeadm) (Cont'd)

Step1) "kubeadm init" and "kubeadm join ..."

```
[centos@kube-master ~]$ sudo iptables-save
# Generated by iptables-save v1.4.21 on Wed Nov 21 09:47:38 2018
*nat
<snip>
-A KUBE-MARK-DROP -j MARK --set-xmark 0x8000/0x8000
-A KUBE-MARK-MASQ -j MARK --set-xmark 0x4000/0x4000
-A KUBE-POSTROUTING -m comment --comment "kubernetes service traffic requiring SNAT" -m mark --mark 0x4000/0x4000 -j MASQUERADE
<snip>
COMMIT
# Completed on Wed Nov 21 09:47:38 2018
```

Kubernetes adds some iptables rule

Kubernetes and Network Setup (by kubeadm) (Cont'd)

Step1) "kubeadm init" and "kubeadm join ..."

Status:

- Kube-master registers all nodes by "kubeadm join" but "NotReady"
- Setup all kubernetes pods (including CoreDNS)
  - Kubelets puts CoreDNS/kube-dns "pending" state to wait container network ready
- Kubernetes adds initial iptables rule

Host network => OK
Container Network => NotReady

Kubernetes and Network Setup (by kubeadm) (Cont'd)

Step2) CNI Plugin and Config Files

```
[centos@kube-master ~]$ kubectl get nodes
NAME          STATUS    ROLES     AGE       VERSION
kube-master   Ready     master    14m       v1.11.4
kube-node-1   Ready     <none>    14m       v1.11.4
kube-node-2   Ready     <none>    14m       v1.11.4
kube-node-3   Ready     <none>    14m       v1.11.4
```

All nodes are ready!

Kubernetes and Network Setup (by kubeadm) (Cont'd)

Step2) CNI Plugin and Config Files

```
[centos@kube-master ~]$ kubectl get pod --all-namespaces
NAMESPACE     NAME                                   READY   STATUS    RESTARTS   AGE
kube-system   coredns-78fcdf6894-fq82f               1/1     Running   0          14m
kube-system   coredns-78fcdf6894-trqrn               1/1     Running   0          14m
kube-system   etcd-kube-master                       1/1     Running   0          13m
kube-system   kube-apiserver-kube-master             1/1     Running   0          13m
kube-system   kube-controller-manager-kube-master    1/1     Running   0          13m
kube-system   kube-flannel-ds-amd64-2sxt9            1/1     Running   0          32s
kube-system   kube-flannel-ds-amd64-94qs8            1/1     Running   0          32s
kube-system   kube-flannel-ds-amd64-h9tpb            1/1     Running   0          32s
kube-system   kube-flannel-ds-amd64-pr7g9            1/1     Running   0          32s
kube-system   kube-proxy-dv88s                       1/1     Running   0          14m
(snip)
```

All k8s pods are Running

Kubernetes and Network Setup (by kubeadm) (Cont'd)

Step2) CNI Plugin and Config Files

```
[centos@kube-master ~]$ sudo iptables-save
# Generated by iptables-save v1.4.21 on Wed Nov 21 09:47:38 2018
*nat
<snip>
-A FORWARD -s 10.244.0.0/16 -j ACCEPT
-A FORWARD -d 10.244.0.0/16 -j ACCEPT
<snip>
COMMIT
# Completed on Wed Nov 21 09:47:38 2018
```

Some iptables rules are added by container network

Kubernetes and Network Setup (by kubeadm) (Cont'd)

Step2) CNI Plugin and Config files

Status:

- Kubelet starts to launch CoreDNS/kube-dns pod with CNI
- CNI (and its network components) creates new iptable rules

Host network => OK
Container Network => OK

Kubernetes and Network Setup (by kubeadm) (Cont'd)

Step2-**failed**) CNI Plugin and Config files

Common failure cases:

- (Case A): CNI plugin/config failed
  - Container is failed to create
- (Case B): CNI network is not configured correctly
  - Container could be created
  - But container cannot to reach the network (Readiness Probe failed)

Host network => OK
Container Network => **NG**

Kubernetes and Network Setup (by kubeadm) (Cont'd)

(Case A&B) If container network is not worked correctly....

```
[centos@kube-master ~]$ kubectl get node
NAME          STATUS    ROLES     AGE       VERSION
kube-master   Ready     master    41m       v1.11.4
kube-node-1   Ready     <none>    40m       v1.11.4
kube-node-2   Ready     <none>    40m       v1.11.4
kube-node-3   Ready     <none>    40m       v1.11.4
```

Host network => OK
Container Network => **NG**

Kubernetes and Network Setup (by kubeadm) (Cont'd)

(Case A) CNI plugin/config failed...

```
[centos@kube-master ~]$ kubectl get pod -n=kube-system
NAME                                 READY   STATUS              RESTARTS   AGE
coredns-78fcdf6894-gfpkw             0/1     ContainerCreating   0          38m
coredns-78fcdf6894-mm9sr             0/1     ContainerCreating   0          38m
etcd-kube-master                     1/1     Running             0          38m
kube-apiserver-kube-master           1/1     Running             0          38m
(snip)
kube-proxy-slppt                     1/1     Running             0          38m
kube-proxy-ts769                     1/1     Running             0          38m
kube-scheduler-kube-master           1/1     Running             0          37m
```

Host network => OK
Container Network => **NG**

Kubernetes and Network Setup (by kubeadm) (Cont'd)

(Case A) CNI plugin/config failed...

```
[centos@kube-master ~]$ kubectl describe pod coredns-78fcdf6894-gfpkw -n=kube-system
Name:               coredns-78fcdf6894-gfpkw
Namespace:          kube-system
(snip)
Events:
  Type     Reason                  Age                From                Message
  ----     ------                  ----               ----                -------
(snip)
  Warning  FailedCreatePodSandBox  3m                 kubelet, kube-master  Failed create pod
sandbox: rpc error: code = Unknown desc = [failed to set up sandbox container
```

Host network => OK
Container Network => **NG**

Kubernetes and Network Setup (by kubeadm) (Cont'd)

(Case B): CNI network is not configured correctly...

```
[centos@kube-master ~]$  kubectl get pod -n kube-system
NAME                                 READY    STATUS    RESTARTS   AGE
etcd-kube-master                     1/1      Running   0          3m
kube-apiserver-kube-master           1/1      Running   0          3m
kube-controller-manager-kube-master  1/1      Running   0          3m
kube-dns-86c47599bd-sx5kw            2/3      Running   1          4m
kube-flannel-ds-amd64-c4wb7          1/1      Running   0          1m
kube-flannel-ds-amd64-c8x98          1/1      Running   0          1m
(snip)
kube-proxy-q2czw                     1/1      Running   0          4m
kube-proxy-v2nz2                     1/1      Running   0          4m
```

Host network => OK
Container Network => **NG**

Kubernetes and Network Setup (by kubeadm) (Cont'd)

(Case B): If CNI network is not configured correctly...

```
[centos@kube-master ~]$ kubectl describe pod kube-dns-86c47599bd-sx5kw -n=kube-system
Name:           kube-dns-86c47599bd-sx5kw
(snip)
Events:
  Type     Reason                 Age               From                Message
  ----     ------                 ----              ----                -------
(snip)
  Normal   Started                13s               kubelet, kube-node-3  Started container
  Warning  Unhealthy              7s                kubelet, kube-node-3  Readiness probe
failed: Get http://10.244.1.2:8081/readiness: dial tcp 10.244.1.2:8081:
connect: connection refused
```

Host network => OK
Container Network => **NG**

Troubleshooting

- **How to Identify Container Interface?**
- iptables
- Packet Capture
  - Demo

Troubleshooting: How to Identify Interface?


```
[centos@kube-node-1 ~]$ ip address show dev cni0
5: cni0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP qlen
1000
    link/ether 0a:58:0a:f4:01:01 brd ff:ff:ff:ff:ff:ff
    inet 10.244.1.1/24 scope global cni0
       valid_lft forever preferred_lft forever
    inet6 fe80::a844:2aff:fecf:efb0/64 scope link
       valid_lft forever preferred_lft forever
```

Troubleshooting: How to Identify Interface?

`ip` command with '-d' option (after 'ip') shows the interface types

```
[centos@kube-node-1 ~]$ ip -d address show dev cni0
5: cni0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP qlen
1000
    link/ether 0a:58:0a:f4:01:01 brd ff:ff:ff:ff:ff:ff promiscuity 0
    bridge forward_delay 1500 hello_time 200 max_age 2000 (snip long line)
    inet 10.24
       valid_lft fore
    inet6 fe80::a84
       valid_lft forever preferred_lft forever
```

Hey, this is linux bridge!!

Troubleshooting: How to Identify Interface? (Cont'd)

`ip -d link` command shows the interface types

```
[centos@kube-node-1 ~]$ ip -d address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 promiscuity 0
    inet 127.0.0.1/8 scope host lo
<snip>
6: veth7f8a9d96@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP
    link/ether 5e:8b:8f:82:3f:19 brd ff:ff:ff:ff:ff:ff link-netnsid 0 promiscuity 1
    veth
    bridge_slave state forwarding priority 32 cost 2 hairpin on <snip>
    inet6 fe80::5c8b:8fff:fe82:3f19/64 scope link
       valid_lft forever preferred_lft forever
```

Troubleshooting: How to Identify Interface? (Cont'd)

`ip -d link` command shows the interface types

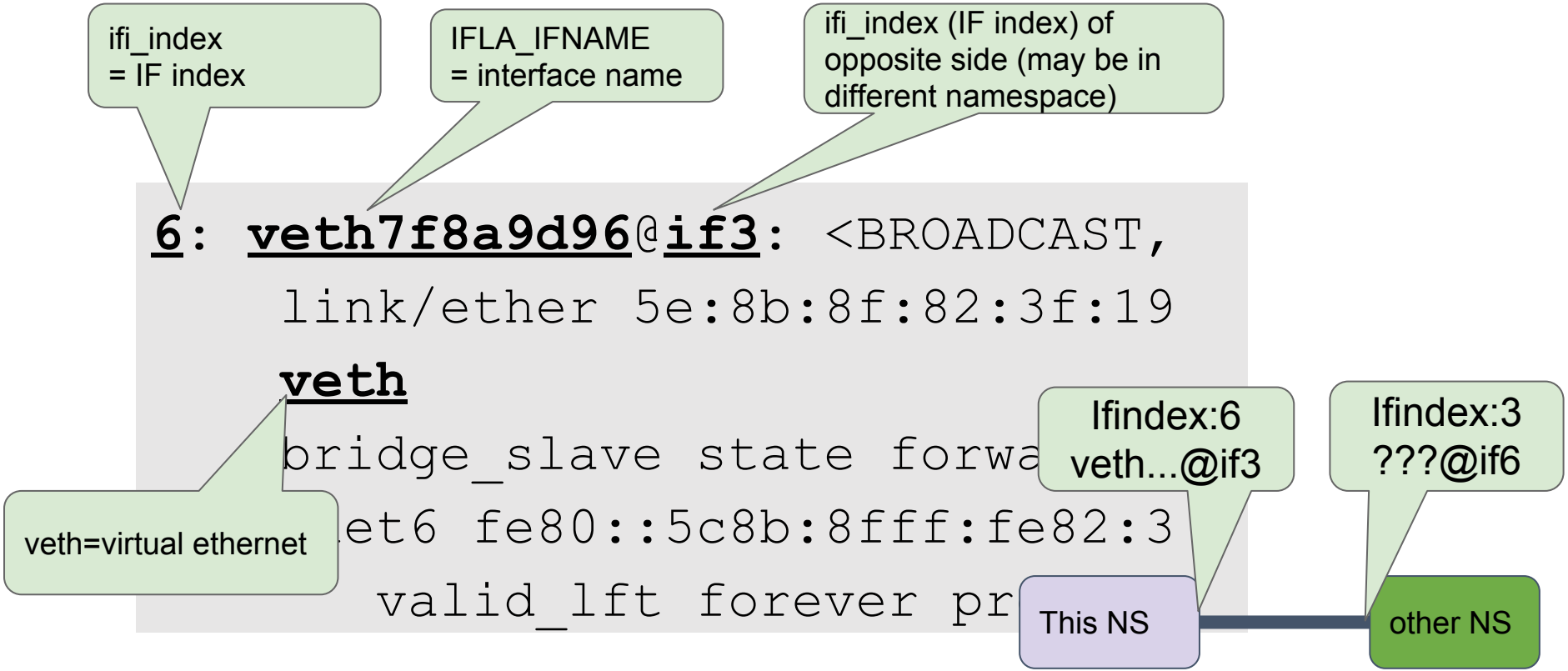
```
[centos@kube-node-1 ~]$ ip -d address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 promiscuity 0
    inet 127.0.0.1/8 scope host lo
<snip>
6: veth7f8a9d96@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0
    link/ether 5e:8b:8f:
    veth
    bridge_slave state forwarding priority 32 cost 2 hairpin on <snip>
    inet6 fe80::5c8b:8fff:fe82:3f19/64 scope link
       valid_lft forever preferred_lft forever
```

Hey, this is veth (i.e. p2p, virtual ethernet device) but what "@if3" means?

Troubleshooting: How to Identify Interface? (Cont'd)

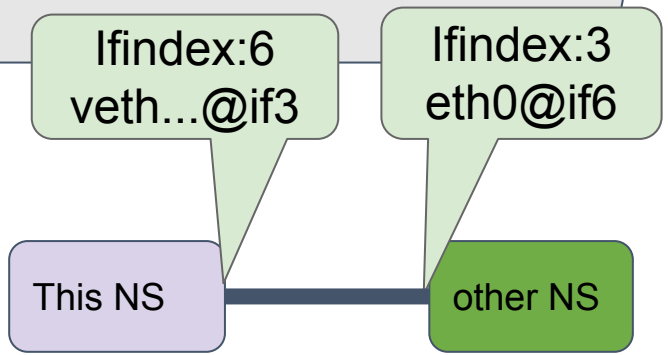

```
6: veth7f8a9d96@if3: <BROADCAST,
    link/ether 5e:8b:8f:82:3f:19
    veth
    bridge_slave state forwardin
    inet6 fe80::5c8b:8fff:fe82:3
       valid_lft forever preferr
```

Troubleshooting: How to Identify Interface? (Cont'd)



Troubleshooting: How to Identify Interface? (Cont'd)

3: **eth0@if6**: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP
link/ether 0a:58:0a:f4:01:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0 promiscuity 0
veth
inet 10.244.1.2/24 scope global eth0
valid_lft forever preferred_lft forever
inet6 fe80::8ce8:f8ff:fe89:4f41/64 scope link tentative dadfailed
valid_lft forever preferred_lft forever



Troubleshooting: How to Identify Interface? (Cont'd)

In case of container without 'ip' command, we can do it with 'nsenter' command and 'ip' command at container host

"sudo nsenter -t <**pid**> -n -- <command>"

```
[centos@kube-node-1 ~]$ sudo nsenter -t 3289 -n -- ip -d address show
(snip)
3: eth0@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state
UP
    link/ether 0a:58:0a:f4:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0 promiscuity 0
    veth
    inet 10.244.0.2/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::7465:88ff:fe63:ae88/64 scope link tentative dadfailed
       valid_lft forever preferred_lft forever
```

Troubleshooting: How to Identify Interface? (Cont'd)

How to use nsenter command:

1. Get "containerID" with "kubectl get pod"

```
[centos@kube-master ~]$ kubectl get pod <pod name> -o json | \
    jq -r .status.containerStatuses[0].containerID
docker://797753ff17005670ee594268c581893b75d1d0e8c0ccd736b5152a5baf65d13e
```

2. Use container runtime command to get PID at corresponding node

```
[centos@kube-node-1 ~]$ docker inspect <containerID (without "docker://")> | grep Pid
or
[centos@kube-node-1 ~]$ crictl inspect <containerID (without :"cri-o://")> | grep pid
```

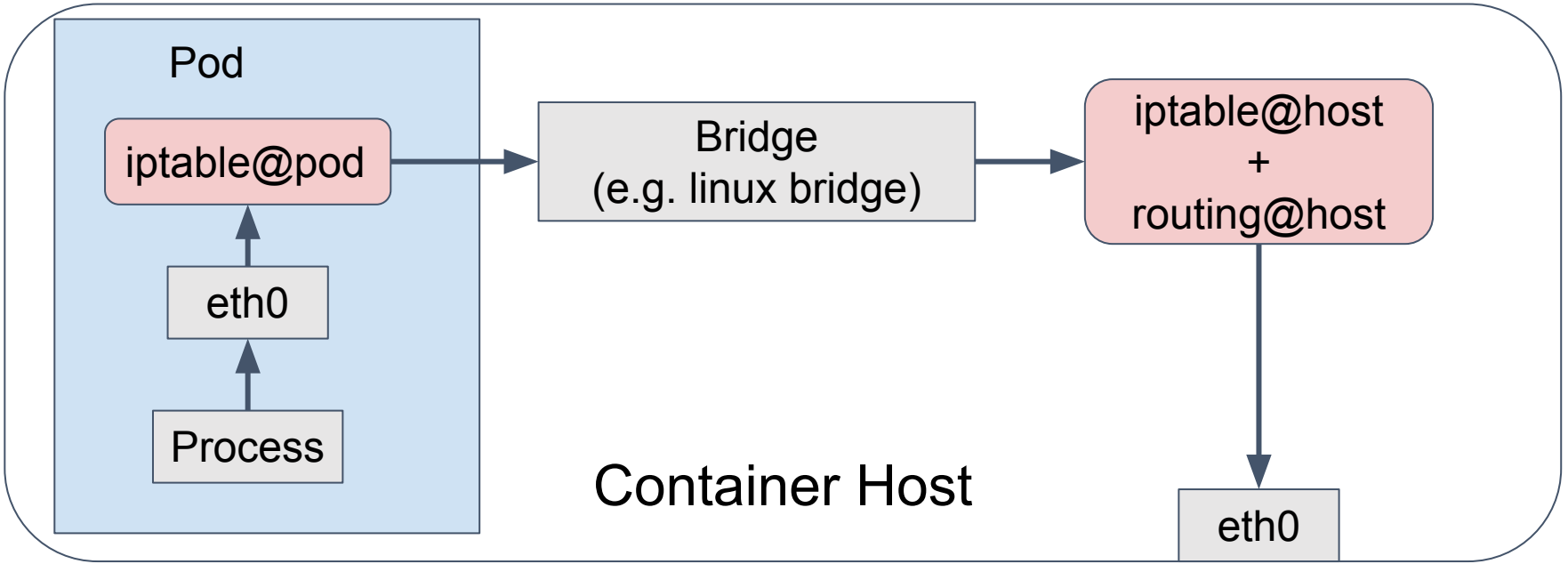
3. Do "nsenter"!

Troubleshooting

- How to Identify Container Interface?
- **iptables**
- Packet Capture
  - Demo

Troubleshooting: iptables

Where does iptables work?



Troubleshooting: iptables

- Q:Who adds iptables?

Troubleshooting: iptables (Cont'd)

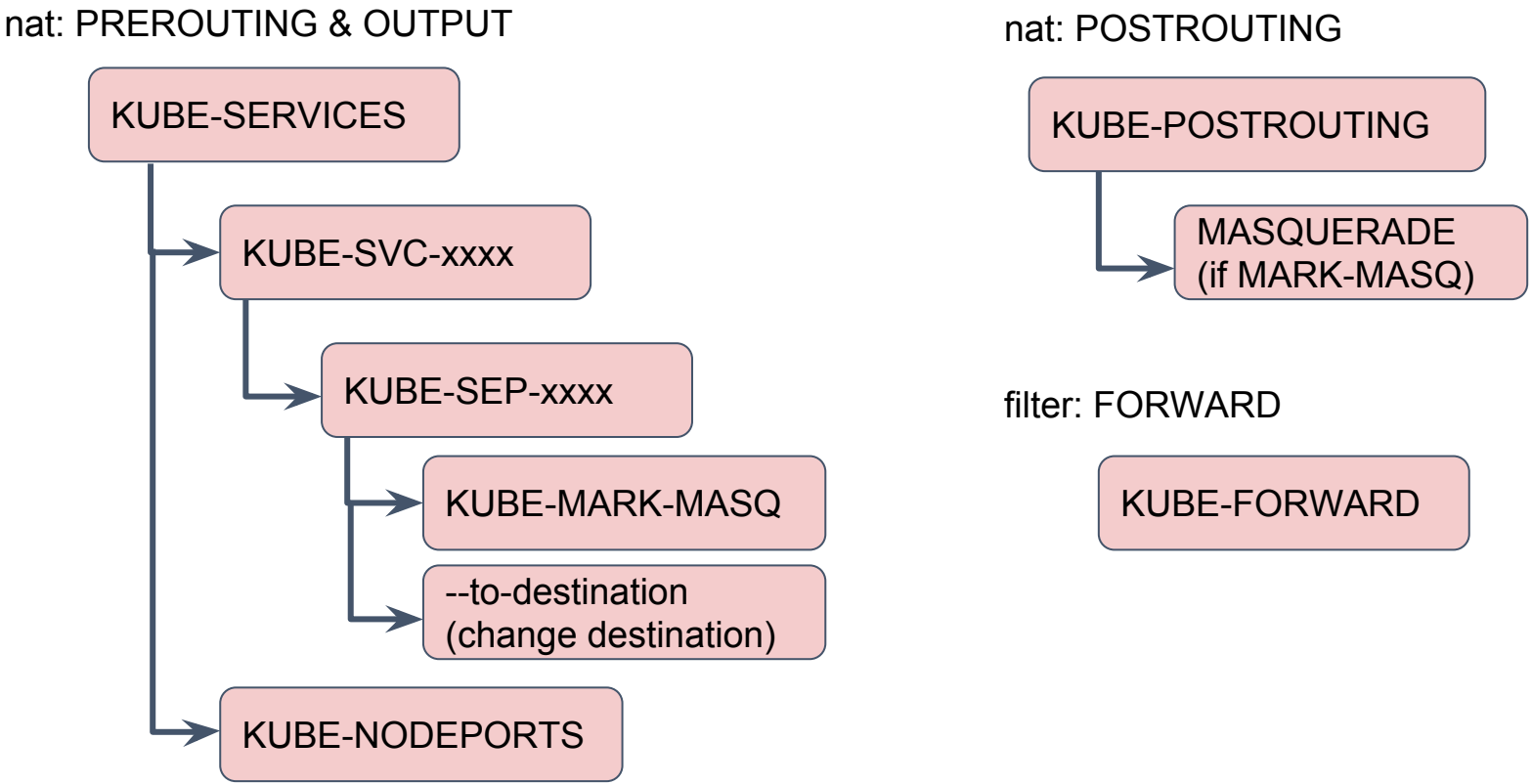
- Q:Who adds iptables?

  → A: (almost) Everything!

Troubleshooting: iptables (Cont'd)

- Q:Who adds iptables?
  - kubelet
    - KUBE-POSTROUTING, KUBE-MARK-MASQ, KUBE-MARK-DROP, KUBE-FIREWALL, KUBE-HOSTPORTS
  - kube-proxy
    - KUBE-POSTROUTING, KUBE-FORWARD, KUBE-SERVICES, KUBE-EXTERNAL-SERVICES, KUBE-NODEPORTS, KUBE-MARK-MASQ, KUBE-MARK-DROP, KUBE-FIREWALL, KUBE-NODE-PORT, KUBE-LOAD-BALANCER
  - CNI Plugins (and related processes)
    - flannel adds a few rules
    - weave adds WEAVE, WEAVE-NPC-xxx
  - Istio
    - ISTIO_OUTPUT, ISTIO_REDIRECT, ISTIO_IN_REDIRECT, ISTIO_INBOUND, ISTIO_DIVERT, ISTIO_TPROXY
  - and so on...

Troubleshooting: iptables (Cont'd): Kubernetes iptables chains (simplify)



Troubleshooting: iptables (Cont'd)

filter: INPUT

- KUBE-EXTERNAL-SERVICES
- KUBE-FIREWALL

filter: FORWARD

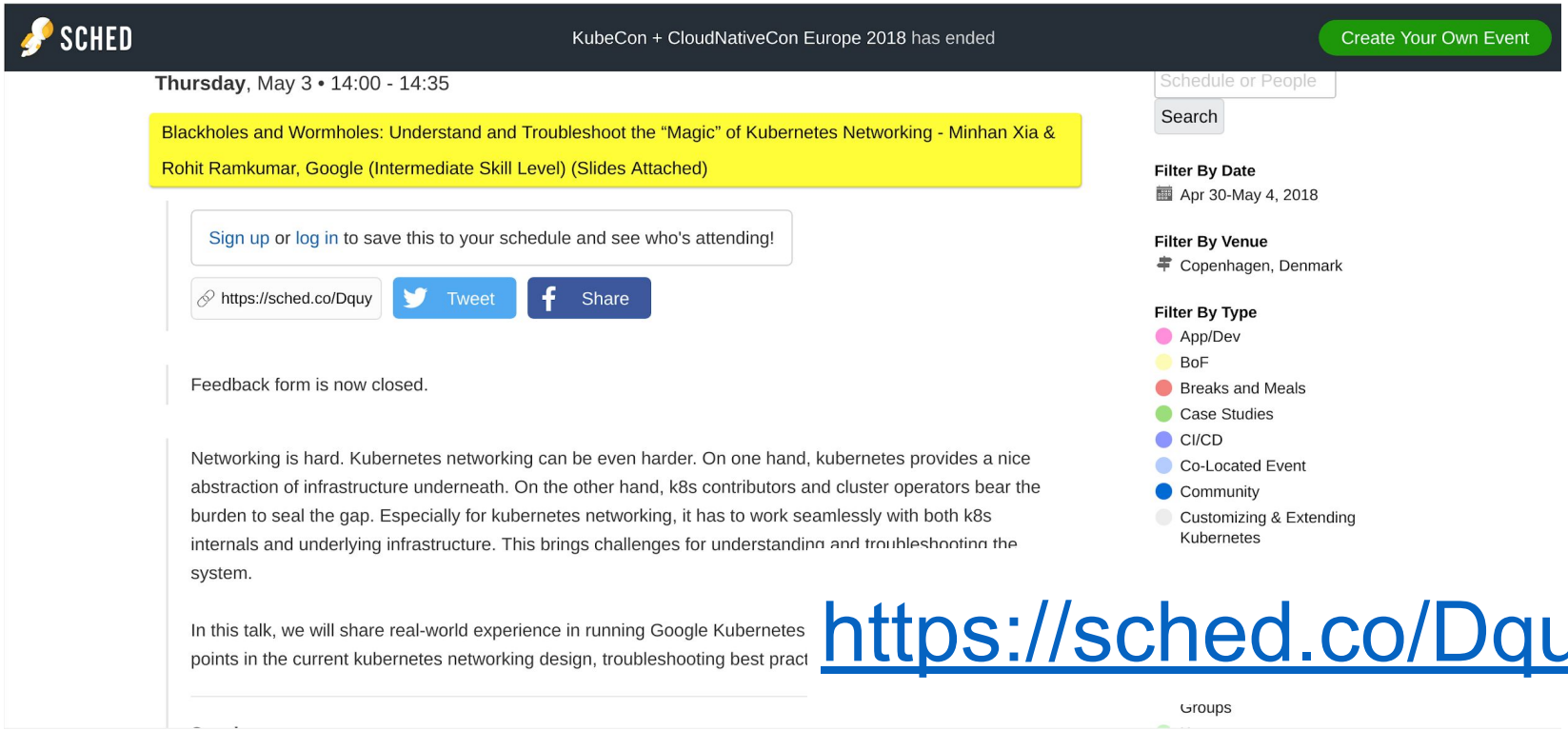
- KUBE-FORWARD

filter: OUTPUT

- KUBE-SERVICES
- KUBE-OUTPUT
- KUBE-FIREWALL

Minhan/Rohit does great talk @KubeCon EU2018

KubeCon | CloudNativeCon North America 2018



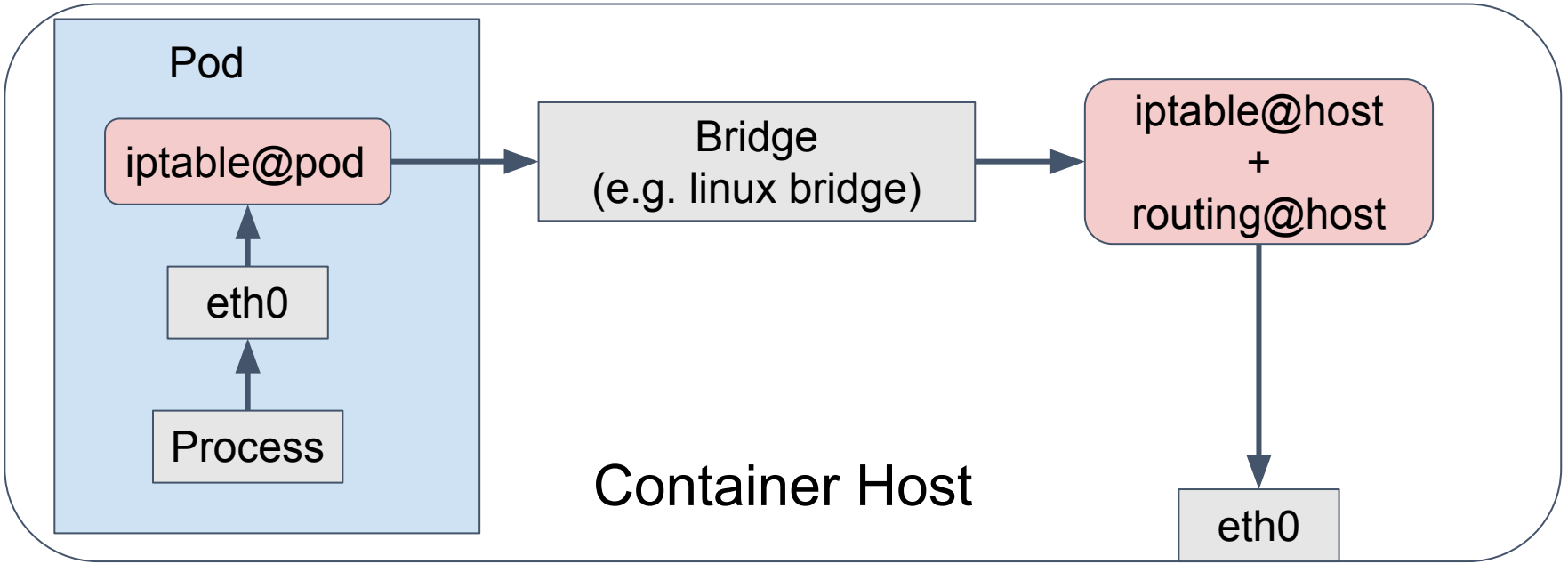
https://sched.co/Dquy

Troubleshooting

- How to Identify Container Interface?
- iptables
- **Packet Capture**
  - Demo

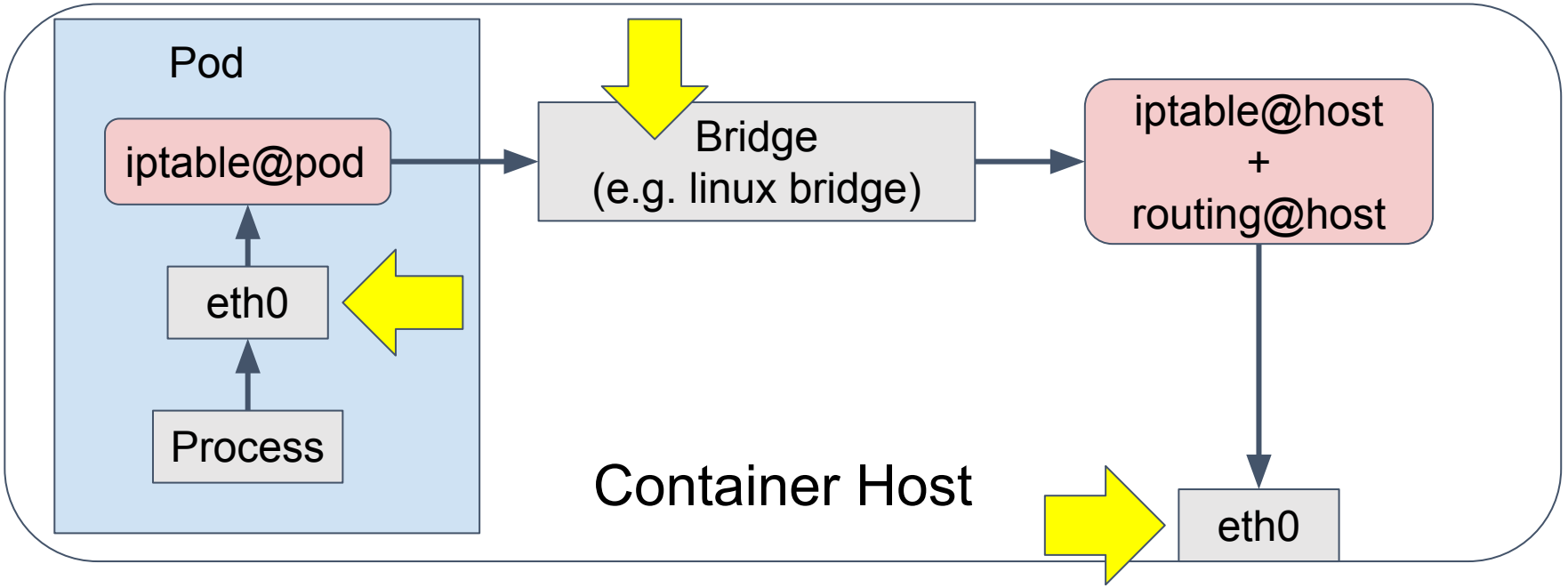
Troubleshooting: Packet Capture

Where can we capture the packet?



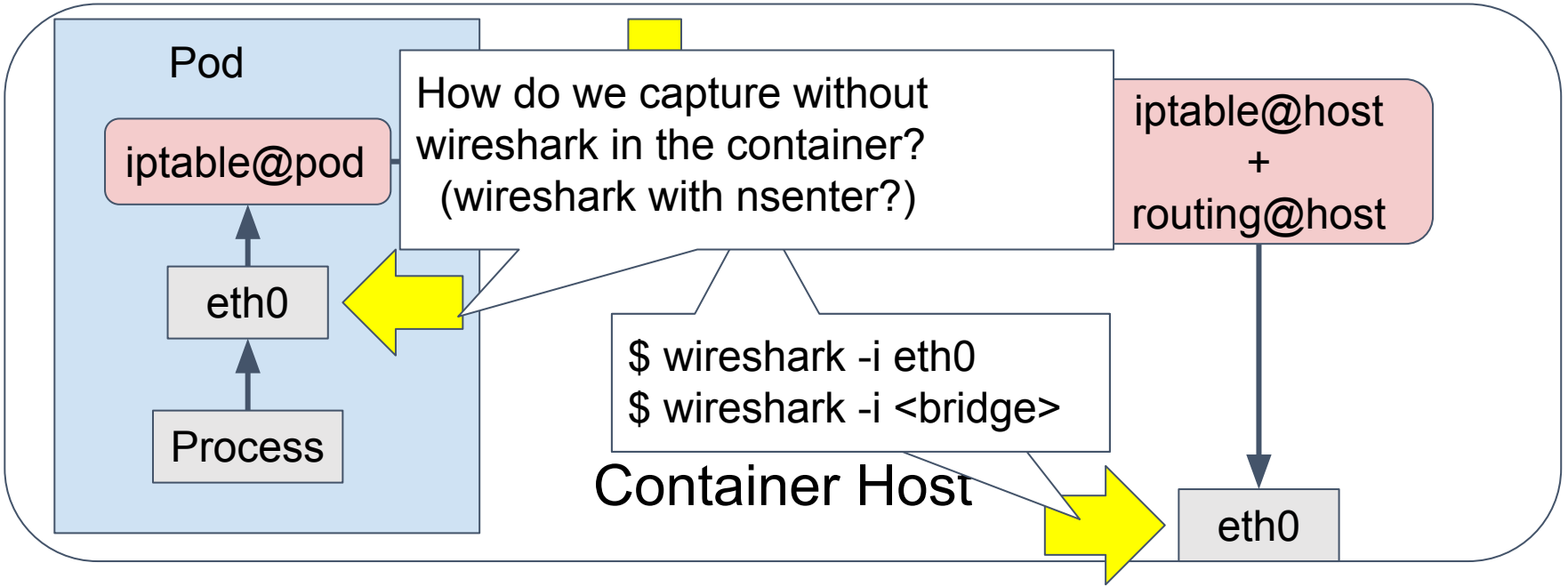
Troubleshooting: Packet Capture (Cont'd)

Where can we capture the packet?



Troubleshooting: Packet Capture (Cont'd)

Where can we capture the packet?



Troubleshooting: Packet Capture (Cont'd)

Kokotap: Tools for kubernetes pod network tapping

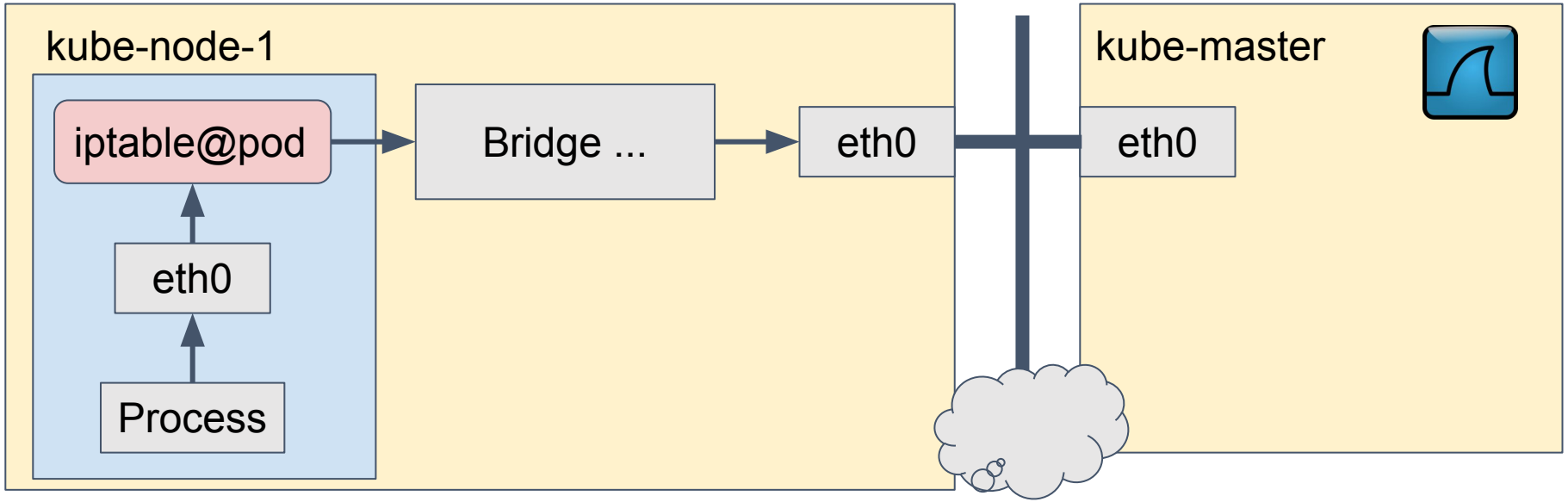
https://github.com/redhat-nfvpe/kokotap

```
$ kokotap --pod=POD --vxlan-id=VXLAN-ID \
          --dest-node=DEST-NODE \
          --mirrortype={ingress,egress,both}
```

Troubleshooting: Packet Capture (Cont'd)

How to capture traffic by kokotap

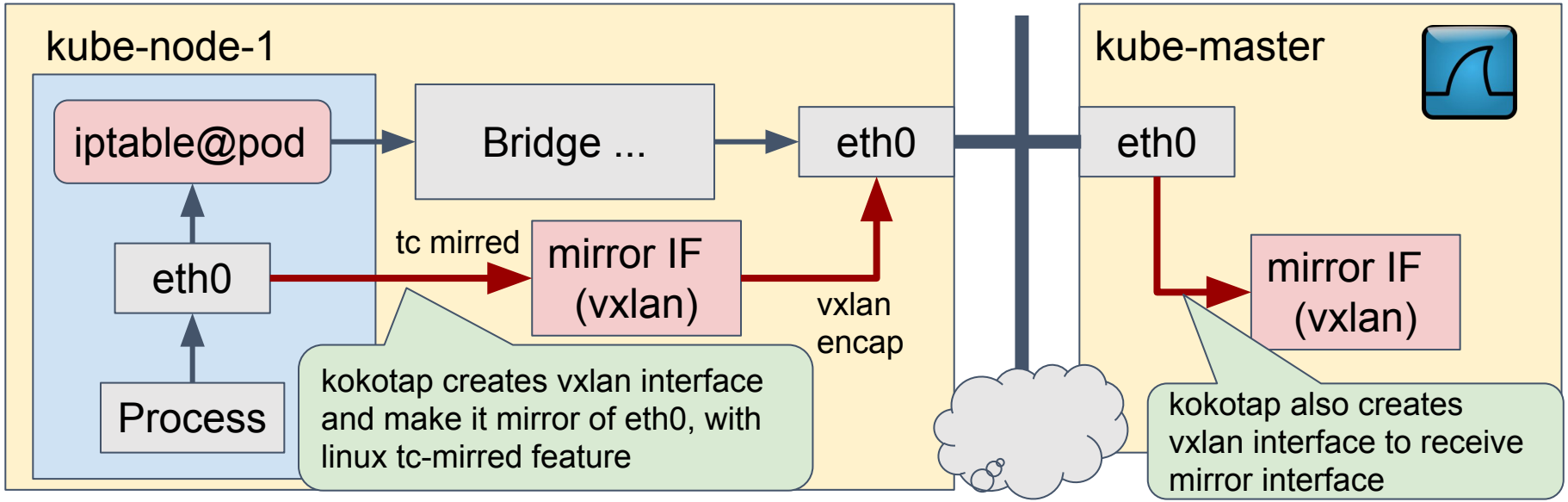
```
$ kokotap --pod=pod1 --vxlan-id=1001 --dest-node=kube-master
          --mirrortype={ingress,egress,both} | kubectl apply -f -
```



Troubleshooting: Packet Capture (Cont'd)

How to capture traffic by kokotap

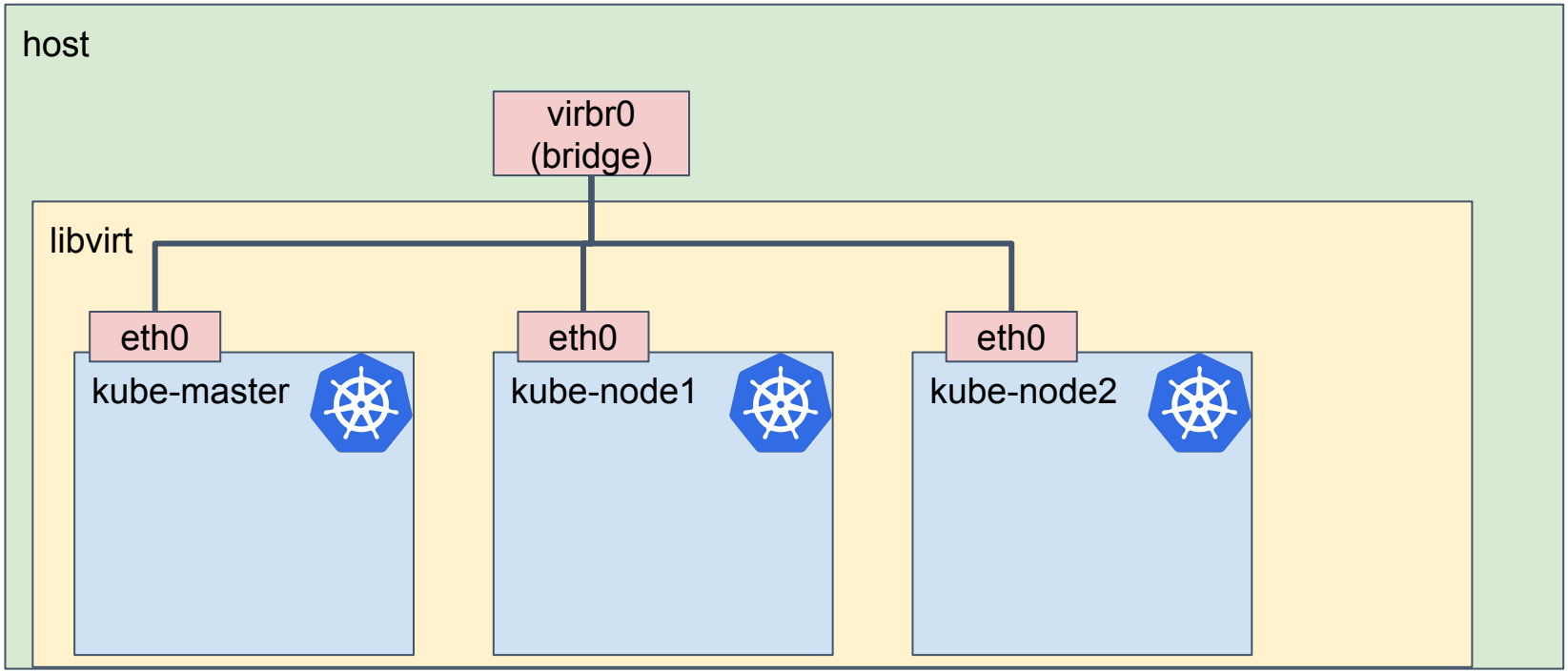
```
$ kokotap --pod=pod1 --vxlan-id=1001 --dest-node=kube-master
          --mirrortype={ingress,egress,both} | kubectl apply -f -
```



Troubleshooting

- How to Identify Container Interface?
- iptables
- Packet Capture
  - **Demo**

Demo


host
virbr0
(bridge)
libvirt
eth0
kube-master
eth0
kube-node1
eth0
kube-node2

Wrap Up

- Why Kubernetes Network is Difficult?
- Kubernetes and Network Setup
- Troubleshooting
  - How to Identify Container Interface?
  - iptables
  - Packet Capture
    - Demo

Thank you! Questions?

Kokotap: http://github.com/redhat-nfvpe/kokotap

Slides available at https://sched.co/GrWr

@s1061123

Slides URL!


KubeCon
CloudNativeCon
North America 2018