

Writing a CNI plugin from scratch

Eran Yanay



Who am I

Eran Yanay

Twistlock

R&D Team Lead

eran@twistlock.com

```
kubectl apply -f  
"https://cloud.weave.works/k8s/net?k8s-version  
=$(kubectl version | base64 | tr -d '\n')"
```

Agenda

Writing a CNI (Container Network Interface) plugin from scratch, using only bash

- What is CNI?
- How CNI plugin works?
- What a CNI plugin is made of?
- How a CNI plugin is being used in K8s?
- How a CNI plugin is executed?
- Anatomy of pod networking
- Live demo

What is CNI?

- CNI stands for Container Networking Interface
- An interface between container runtime and the network implementation
- Configures the network interfaces and routes
- Concerns itself only with network connectivity
- <https://github.com/container networking/cni/blob/spec-v0.4.0/SPEC.md>

How CNI plugin works (in k8s)

- A CNI binary

Handles connectivity - configures the network interface of the pod

- A daemon

Handles reachability - manages routings across the cluster

What a CNI plugin is made of?

```
# cat /etc/cni/net.d/10-my-cni-demo.conf
{
  "cniVersion": "0.3.1",
  "name": "my-cni-demo",
  "type": "my-cni-demo",
  "podcidr": "10.240.0.0/24",
}
```

```
# cat /opt/cni/bin/my-cni-demo
case $CNI_COMMAND in
ADD)
  # Configure networking for a new container
  ;;
DEL)
  # Cleanup when container is stopped
  ;;
GET)

  ;;
VERSION)
  # Get the plugin version
  ;;
esac
```

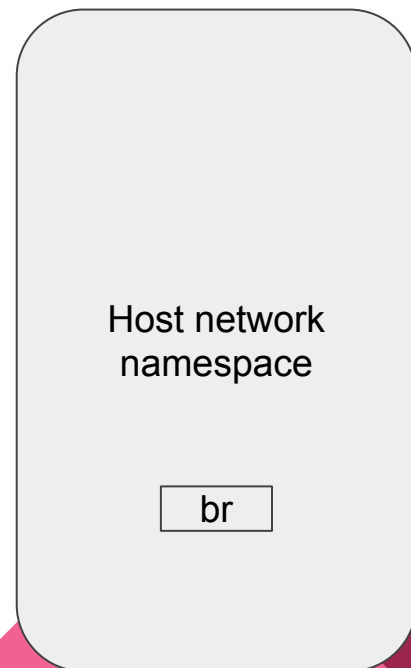
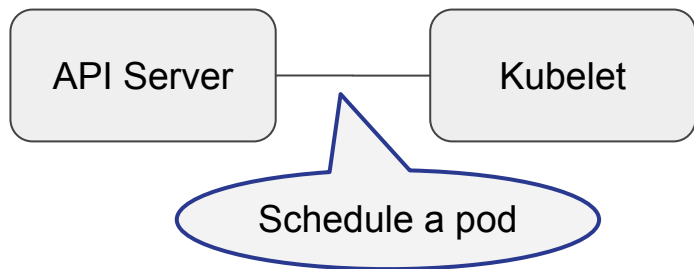
The weave example..

```
kind: DaemonSet
spec:
  containers:
    - name: weave
      command:
        - /home/weave/launch.sh
      image: 'docker.io/weaveworks/weave-kube:2.5.1'
      volumeMounts:
        - name: weavedb
          mountPath: /weavedb
        - name: cni-bin
          mountPath: /host/opt
        - name: cni-bin2
          mountPath: /host/home
        - name: cni-conf
          mountPath: /host/etc
  hostNetwork: true
```

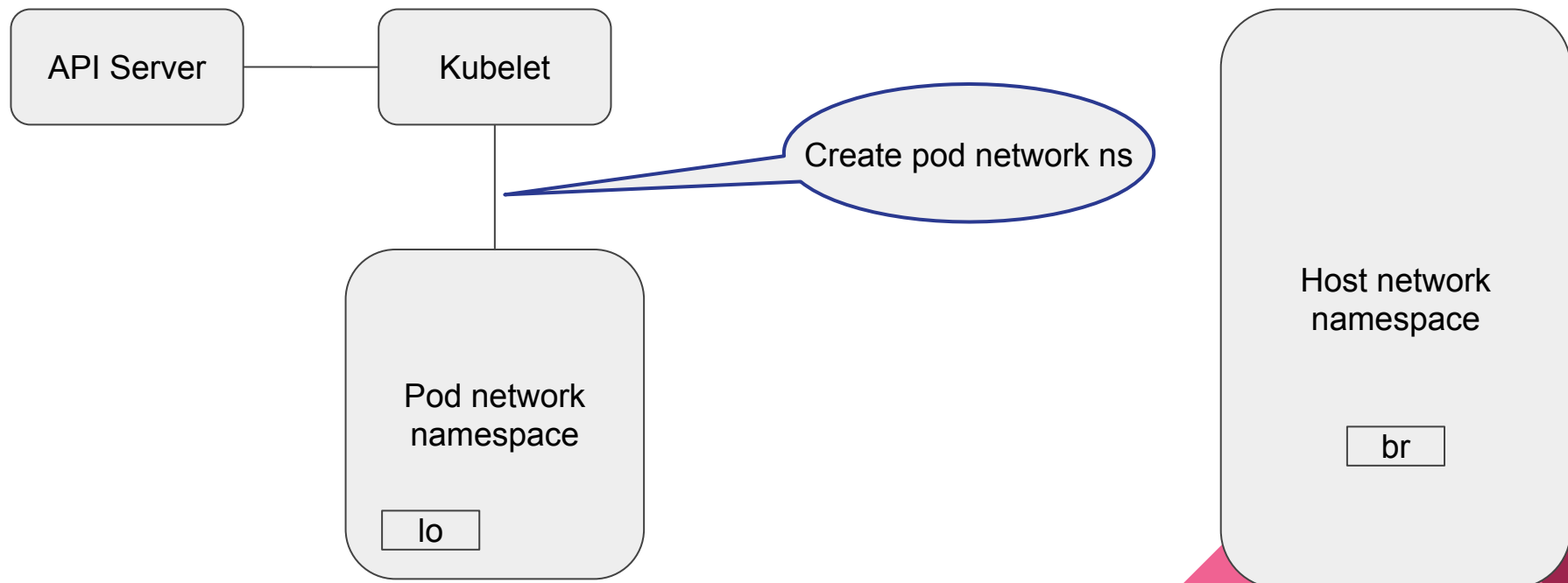

The weave example..

```
$ cat /home/weave/launch.sh
# ... previous non related code ...
# Install CNI plugin binary to typical CNI bin location
# with fall-back to CNI directory used by kube-up on GCI OS
if ! mkdir -p $HOST_ROOT/opt/cni/bin ; then
    if mkdir -p $HOST_ROOT/home/kubernetes/bin ; then
        export WEAVE_CNI_PLUGIN_DIR=$HOST_ROOT/home/kubernetes/bin
    else
        echo "Failed to install the Weave CNI plugin" >&2
        exit 1
    fi
fi
mkdir -p $HOST_ROOT/etc/cni/net.d
export HOST_ROOT
/home/weave/weave --local setup-cni
```

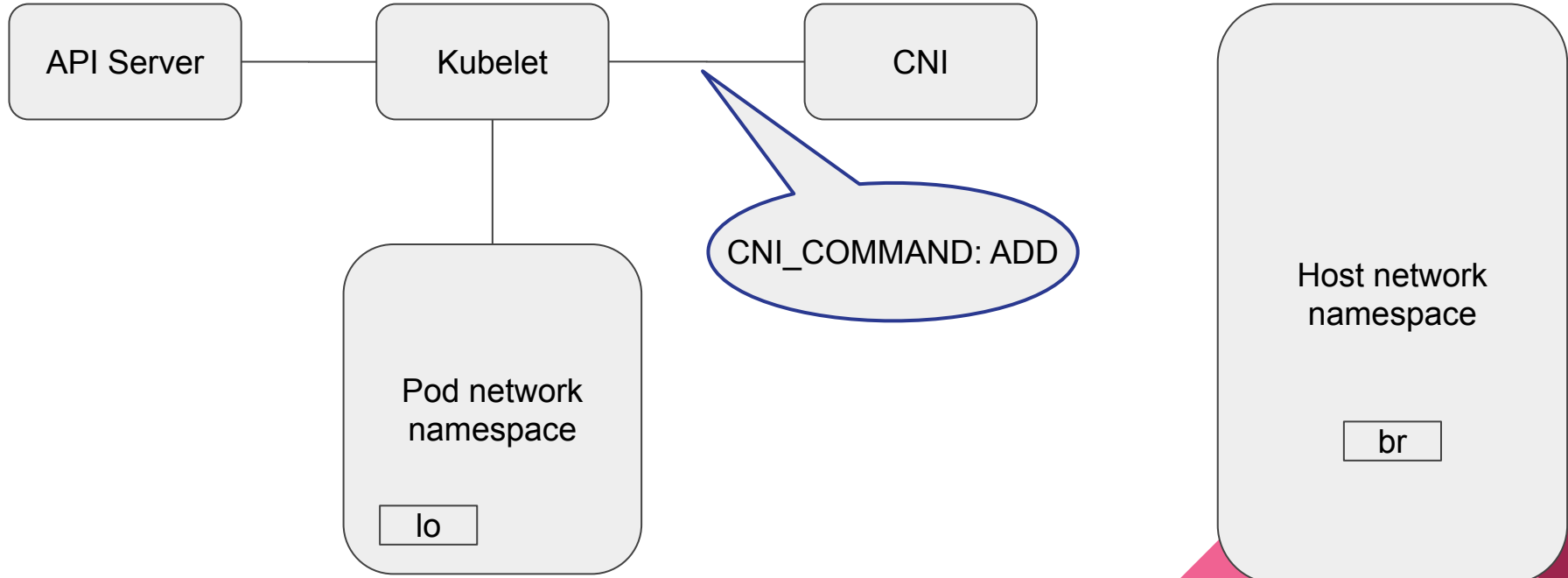
How a CNI plugin is being used in K8s?



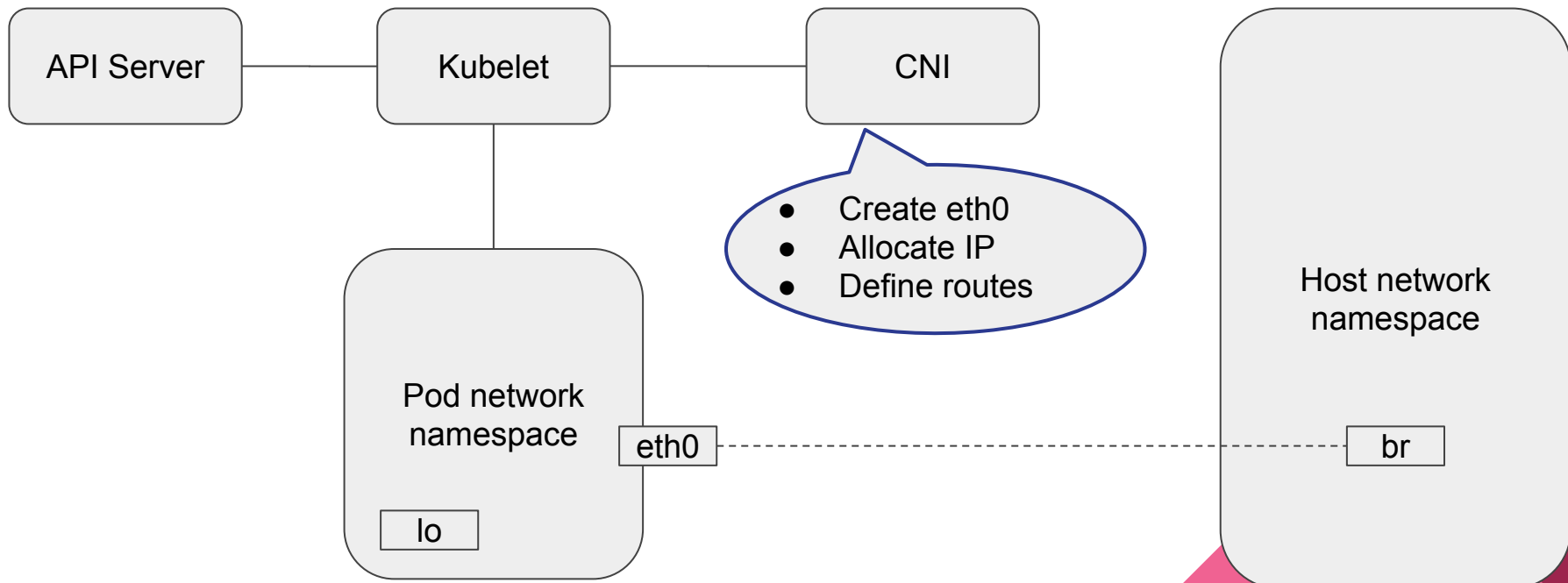
How a CNI plugin is being used in K8s?



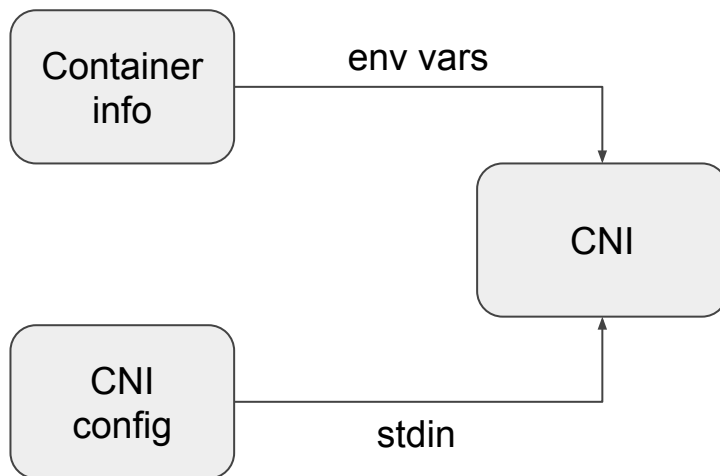
How a CNI plugin is being used in K8s?



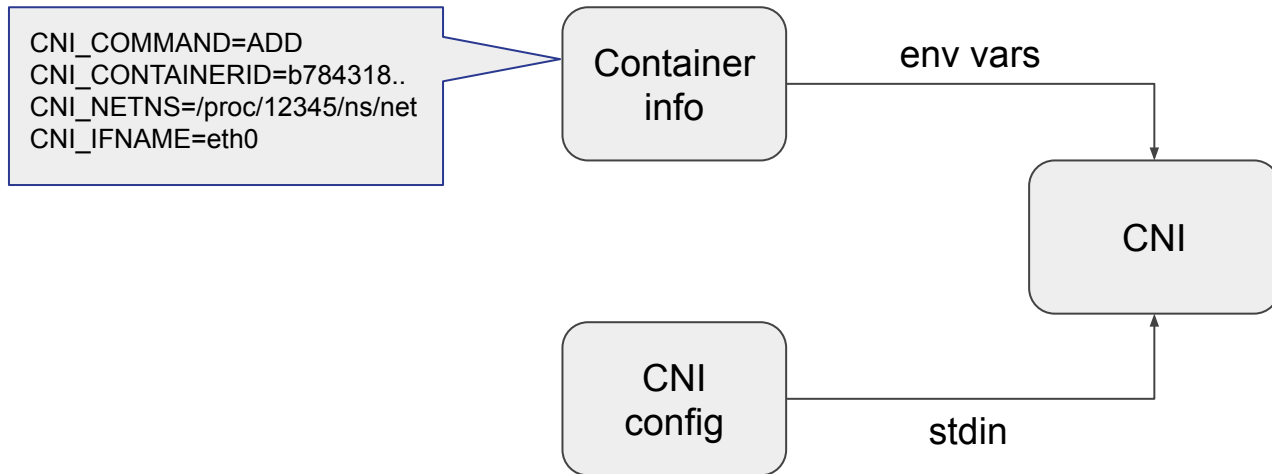
How a CNI plugin is being used in K8s?



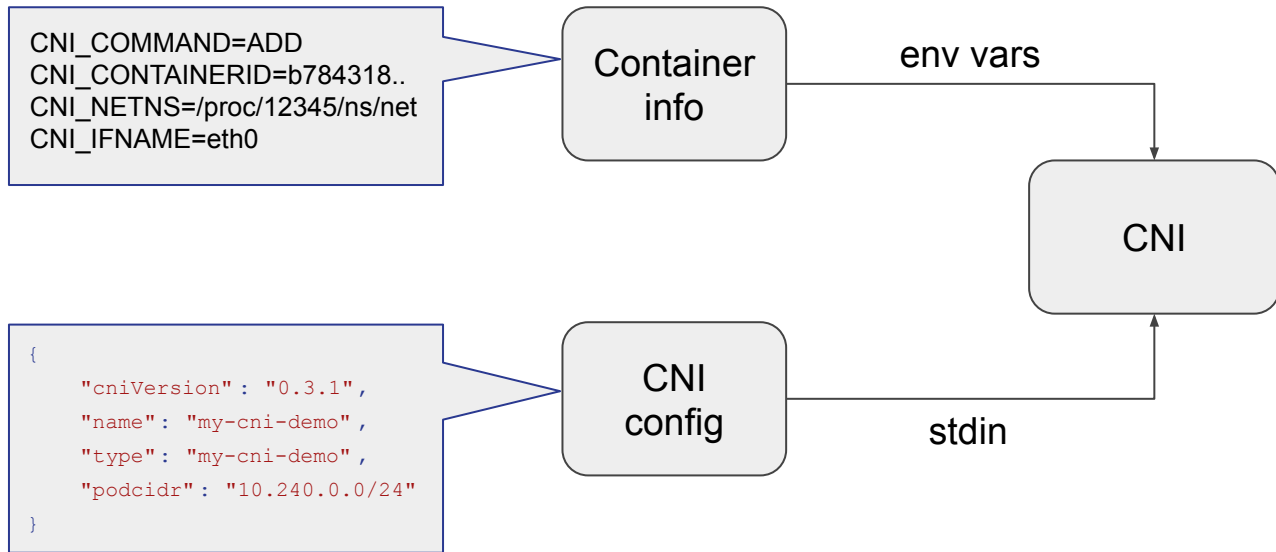
How a CNI plugin is executed?



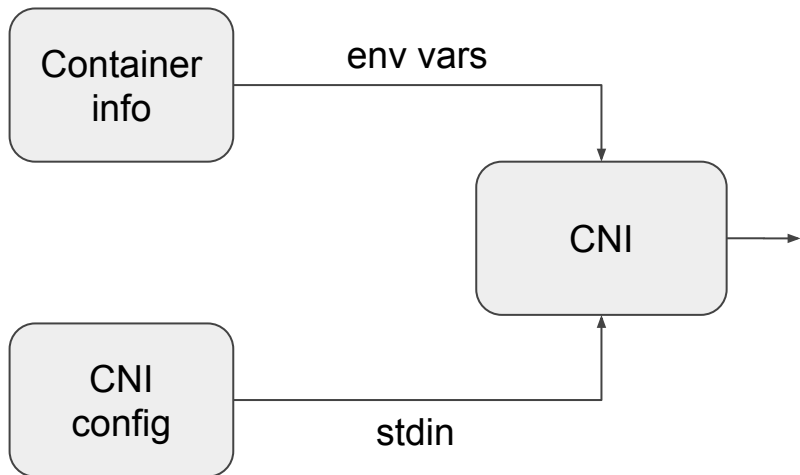
How a CNI plugin is executed?



How a CNI plugin is executed?

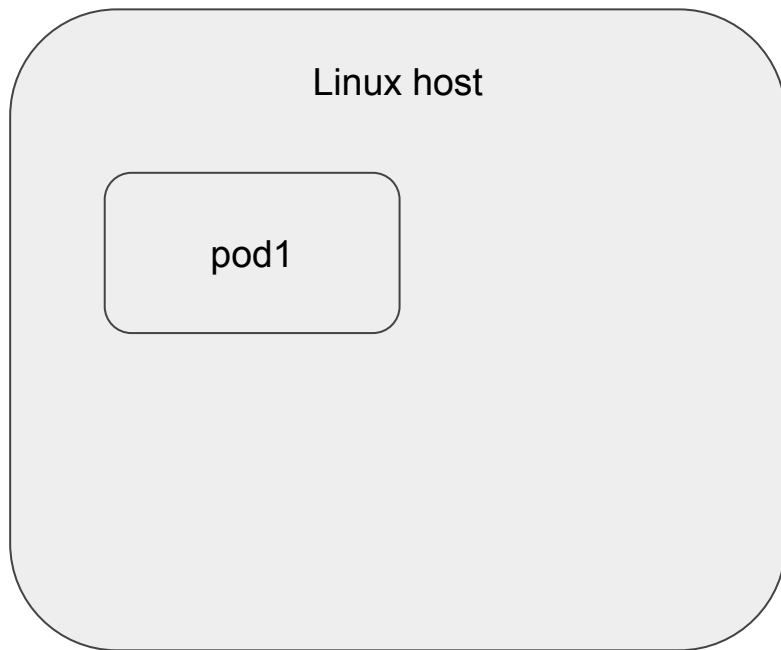


How a CNI plugin is executed?

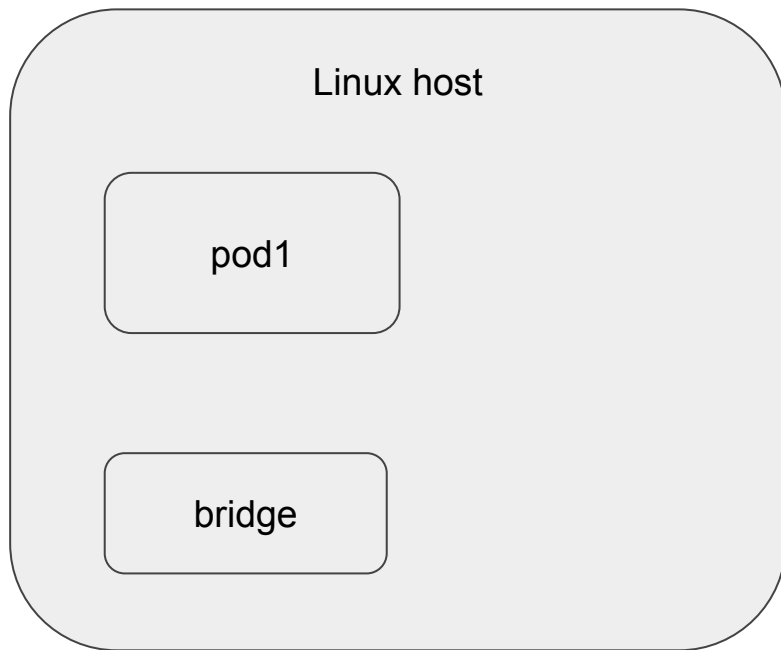


```
{
  "cniVersion": "0.3.1",
  "interfaces": [
    {
      "name": "eth0",
      "mac": "ce:60:4c:b9:3a:06",
      "sandbox": "/proc/15116/ns/net"
    }
  ],
  "ips": [
    {
      "version": "4",
      "address": "10.240.0.6/24",
      "gateway": "10.240.0.1",
      "interface": 0
    }
  ]
}
```

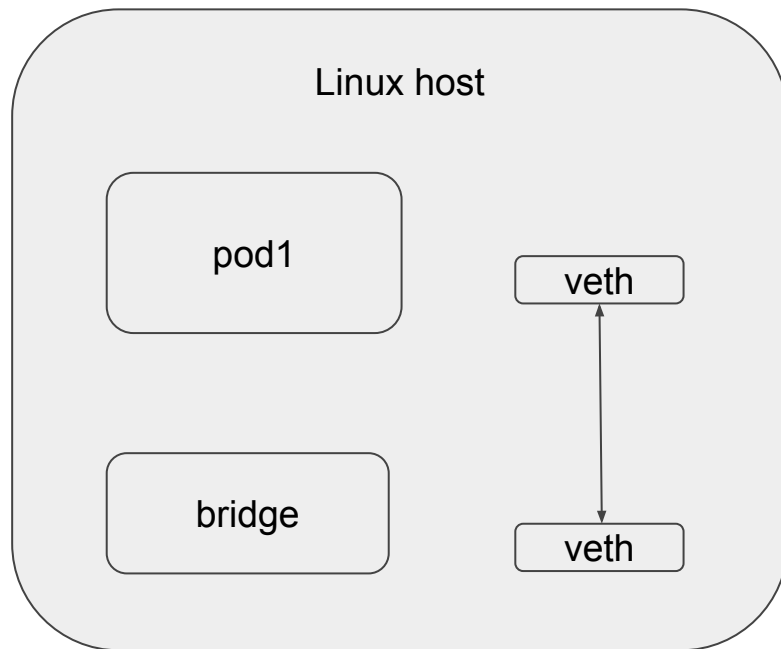
Anatomy of pod networking



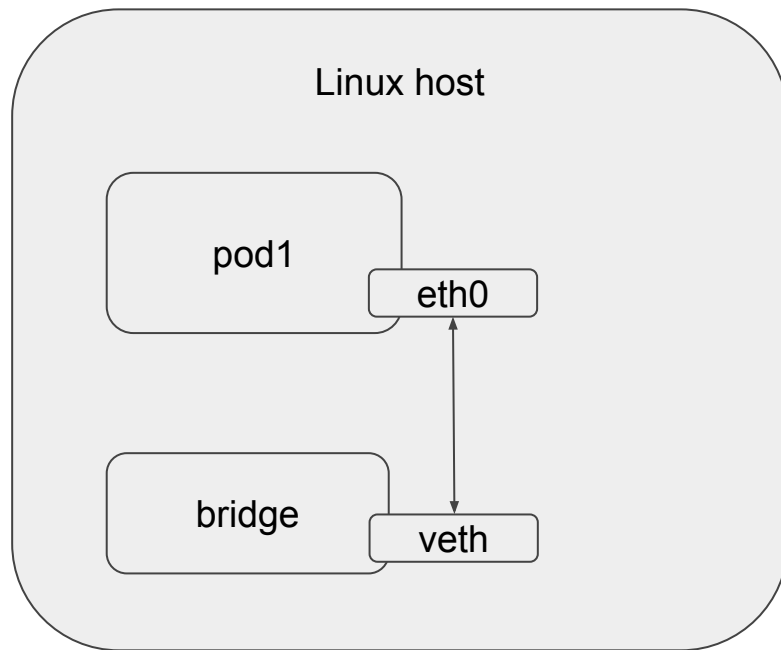
Anatomy of pod networking



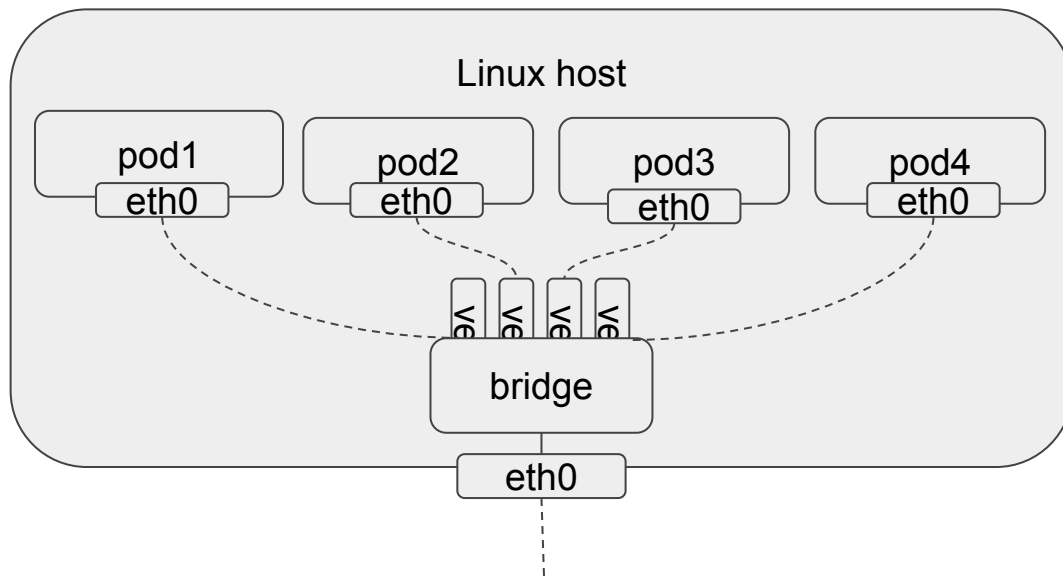
Anatomy of pod networking



Anatomy of pod networking

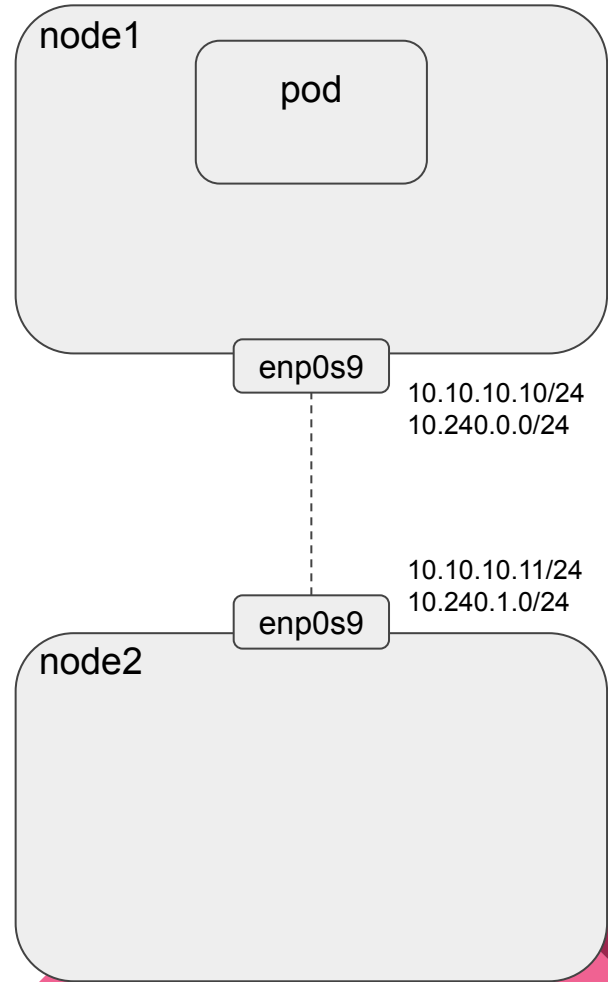


Anatomy of pod networking

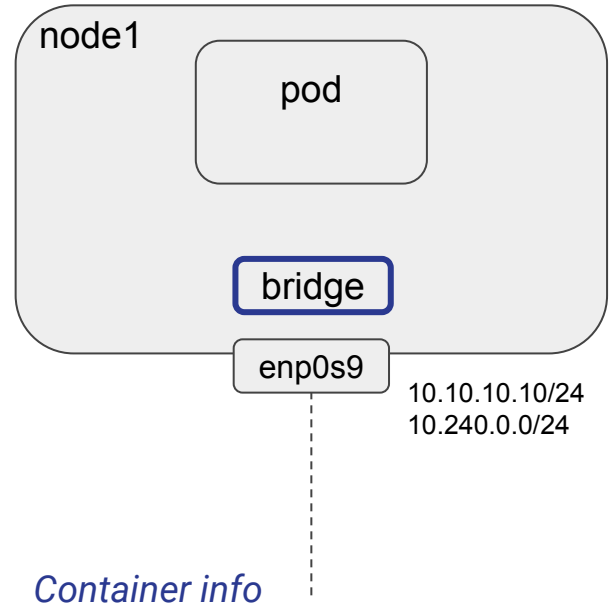


```
# cat /etc/cni/net.d/10-my-cni-demo.conf
{
  "cniVersion": "0.3.1",
  "name": "my-cni-demo",
  "type": "my-cni-demo",
  "podcidr": "10.240.0.0/24",
}

# cat /opt/cni/bin/my-cni-demo
case $CNI_COMMAND in
ADD)
  ;; # Configure networking
DEL)
  ;; # Cleanup
GET)
  ;;
VERSION)
  ;; # Print plugin version
esac
```



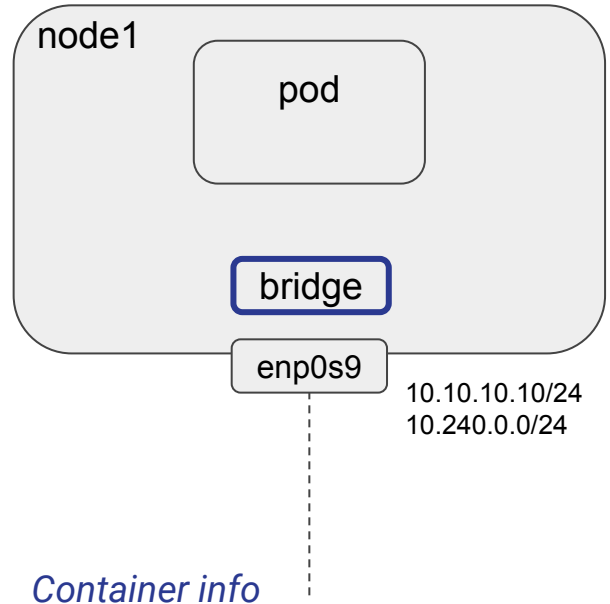
```
case $CNI_COMMAND in
ADD)
  podcidr=$(cat /dev/stdin | jq -r ".podcidr") # 10.240.0.0/24
  podcidr_gw=$(echo $podcidr | sed "s:0/24:1:g") # 10.240.0.1
;;
```



```
CNI_CONTAINERID=b552f9...
CNI_IFNAME=eth0
CNI_COMMAND=ADD
CNI_NETNS=/proc/6137/ns/net
```



```
case $CNI_COMMAND in
ADD)
  podcidr=$(cat /dev/stdin | jq -r ".podcidr") # 10.240.0.0/24
  podcidr_gw=$(echo $podcidr | sed "s:0/24:1:g") # 10.240.0.1
  brctl addbr cni0 # create a new bridge (if doesnt exist), cni0
  ip link set cni0 up
  ip addr add "${podcidr_gw}/24" dev cni0 # assign 10.240.0.1/24 to cni0
;;
```



Container info

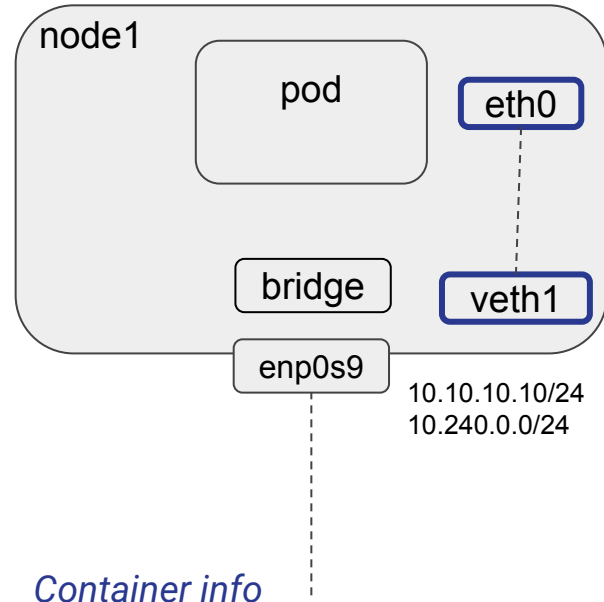
```
CNI_CONTAINERID=b552f9...
CNI_IFNAME=eth0
CNI_COMMAND=ADD
CNI_NETNS=/proc/6137/ns/net
```

```

case $CNI_COMMAND in
ADD)
  podcidr=$(cat /dev/stdin | jq -r ".podcidr") # 10.240.0.0/24
  podcidr_gw=$(echo $podcidr | sed "s:0/24:1:g") # 10.240.0.1
  brctl addbr cni0 # create a new bridge (if doesnt exist), cni0
  ip link set cni0 up
  ip addr add "${podcidr_gw}/24" dev cni0 # assign 10.240.0.1/24 to cni0

  host_ifname="veth$n" # n=1,2,3...
  ip link add $CNI_IFNAME type veth peer name $host_ifname
  ip link set $host_ifname up
;;

```



Container info

```

CNI_CONTAINERID=b552f9...
CNI_IFNAME=eth0
CNI_COMMAND=ADD
CNI_NETNS=/proc/6137/ns/net

```

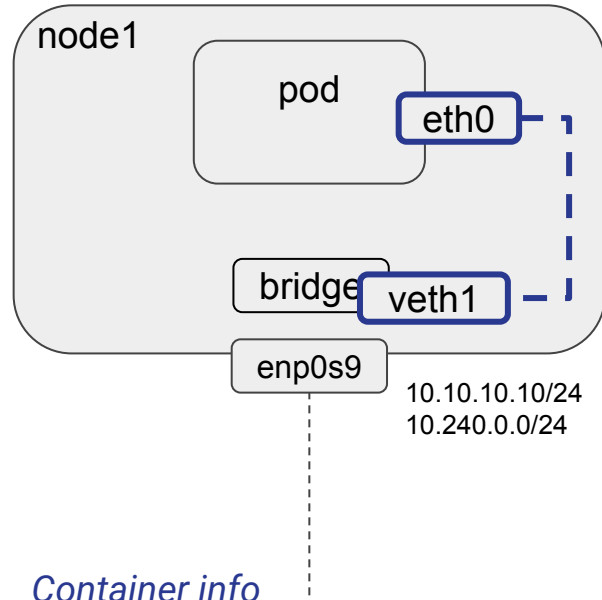
```

case $CNI_COMMAND in
ADD)
  podcidr=$(cat /dev/stdin | jq -r ".podcidr") # 10.240.0.0/24
  podcidr_gw=$(echo $podcidr | sed "s:0/24:1:g") # 10.240.0.1
  brctl addbr cni0 # create a new bridge (if doesnt exist), cni0
  ip link set cni0 up
  ip addr add "${podcidr_gw}/24" dev cni0 # assign 10.240.0.1/24 to cni0

  host_ifname="veth$n" # n=1,2,3...
  ip link add $CNI_IFNAME type veth peer name $host_ifname
  ip link set $host_ifname up

  ip link set $host_ifname master cni0 # connect veth1 to bridge
  ln -sft $CNI_NETNS /var/run/netns/$CNI_CONTAINERID
  ip link set $CNI_IFNAME netns $CNI_CONTAINERID # move eth0 to pod ns
;;

```



Container info

```

CNI_CONTAINERID=b552f9...
CNI_IFNAME=eth0
CNI_COMMAND=ADD
CNI_NETNS=/proc/6137/ns/net

```

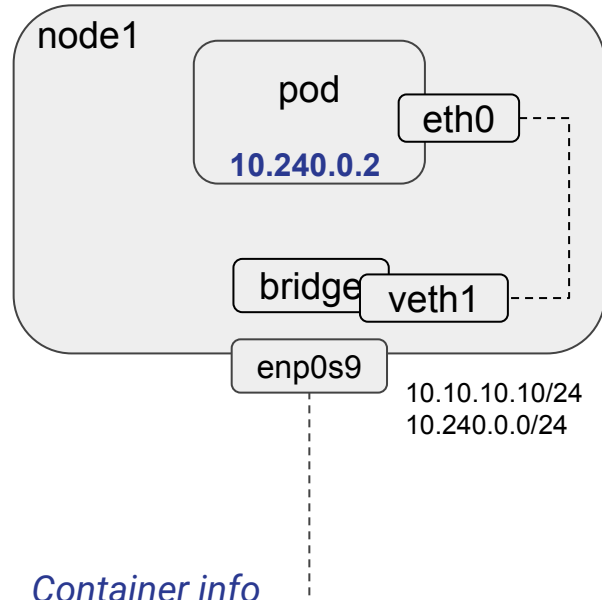
```

case $CNI_COMMAND in
ADD)
  podcidr=$(cat /dev/stdin | jq -r ".podcidr") # 10.240.0.0/24
  podcidr_gw=$(echo $podcidr | sed "s:0/24:1:g") # 10.240.0.1
  brctl addbr cni0 # create a new bridge (if doesnt exist), cni0
  ip link set cni0 up
  ip addr add "${podcidr_gw}/24" dev cni0 # assign 10.240.0.1/24 to cni0

  host_ifname="veth$n" # n=1,2,3...
  ip link add $CNI_IFNAME type veth peer name $host_ifname
  ip link set $host_ifname up

  ip link set $host_ifname master cni0 # connect veth1 to bridge
  ln -sFT $CNI_NETNS /var/run/netns/$CNI_CONTAINERID
  ip link set $CNI_IFNAME netns $CNI_CONTAINERID # move eth0 to pod ns
  # calculate $ip
  ip netns exec $CNI_CONTAINERID ip link set $CNI_IFNAME up
  ip netns exec $CNI_CONTAINERID ip addr add $ip/24 dev $CNI_IFNAME
  ip netns exec $CNI_CONTAINERID ip route add default via $podcidr_gw
dev $CNI_IFNAME

```



Container info

```

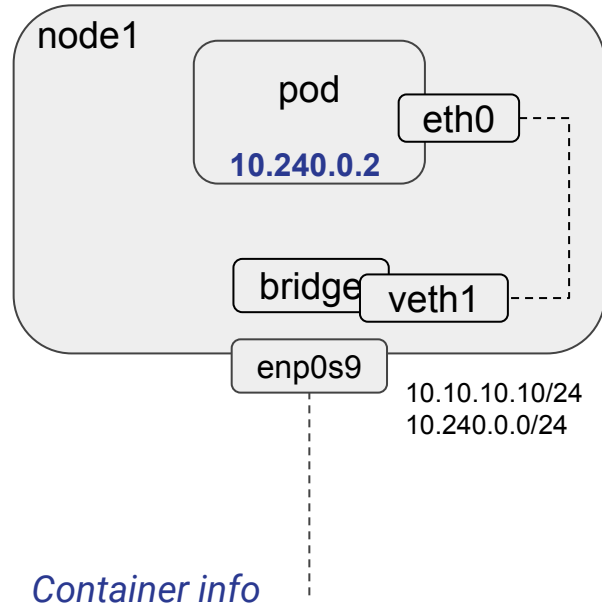
CNI_CONTAINERID=b552f9...
CNI_IFNAME=eth0
CNI_COMMAND=ADD
CNI_NETNS=/proc/6137/ns/net

```

```

case $CNI_COMMAND in
ADD)
  podcidr=$(cat /dev/stdin | jq -r ".podcidr") # 10.240.0.0/24
  podcidr_gw=$(cat /dev/stdin | jq -r ".podcidr_gw") # 10.240.0.1
  brctl addbr $podcidr_gw # create bridge
  ip link set $podcidr_gw up # bring up bridge
  ip addr add $podcidr_gw/24 dev $podcidr_gw # add IP to bridge
  if [ -f /tmp/last_allocated_ip ]; then
    n=`cat /tmp/last_allocated_ip` # get last allocated IP
  else
    n=1 # start with 1
  fi
  host_ifname=$(cat /dev/stdin | jq -r ".host_ifname") # get host interface
  ip=$(echo $podcidr | sed "s:0/24:${(n+1)}:g") # calculate $ip
  echo ${n+1} > /tmp/last_allocated_ip
  ip link set $host_ifname master cni0 # connect veth1 to bridge
  ln -sFT $CNI_NETNS /proc/6137/ns/net
  ip link set $CNI_IFNAME netns $CNI_CONTAINERID # move eth0 to pod ns
  # calculate $ip
  ip netns exec $CNI_CONTAINERID ip link set $CNI_IFNAME up
  ip netns exec $CNI_CONTAINERID ip addr add $ip/24 dev $CNI_IFNAME
  ip netns exec $CNI_CONTAINERID ip route add default via $podcidr_gw
  dev $CNI_IFNAME

```



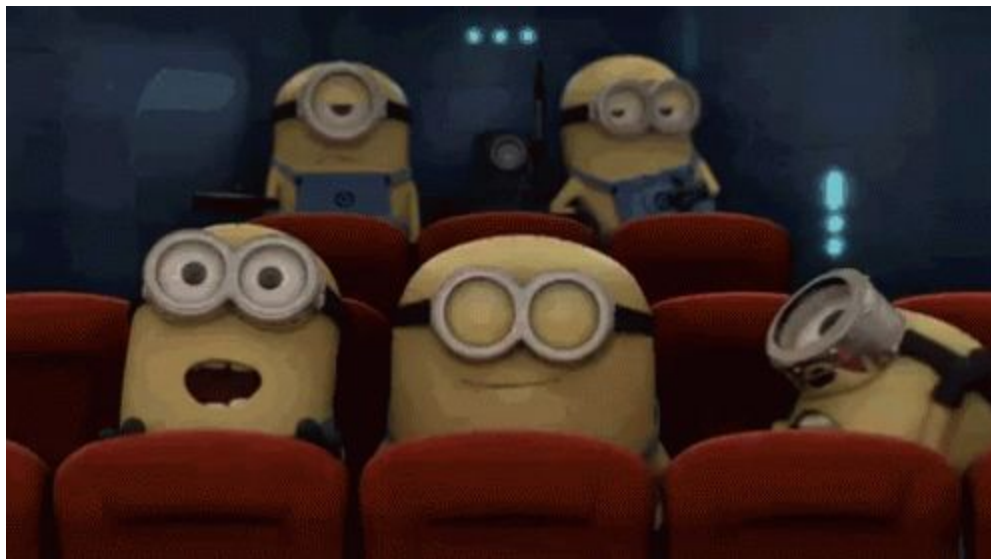
Container info

```

CNI_CONTAINERID=b552f9...
CNI_IFNAME=eth0
CNI_COMMAND=ADD
CNI_NETNS=/proc/6137/ns/net

```

Lets see a demo!



Thank you!

Full implementation is available at <https://github.com/eranyanay/cni-from-scratch>



Twistlock™