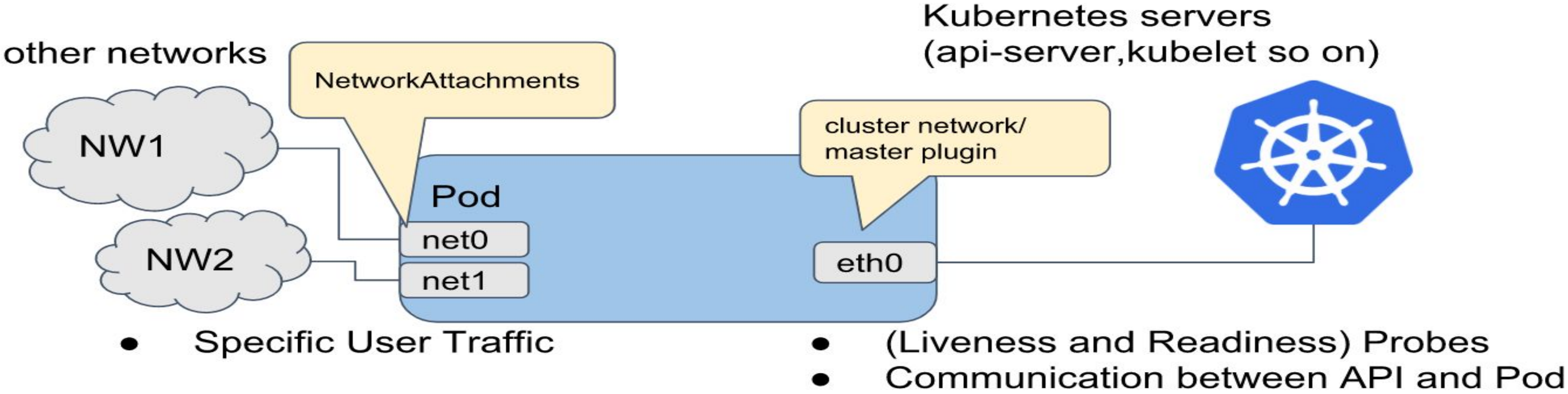
MULTUS

Features

- ***CNI Meta Plugin***
- ***Utilizes other CNI plugins like ipvlan, macvlan, ptp etc***
- ***POD multiple interfaces***
- ***Hardware acceleration (DPDK, SR-IOV)***



- Specific User Traffic

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

Native CNI Plugins

https://www.cni.dev/plugins/

Main CNIs:

- bridge plugin
- host-device
- ipvlan plugin
- macvlan plugin
- ptp plugin
- win-bridge plugin
- win-overlay plugin

Native CNI Plugins

https://www.cni.dev/plugins/

Meta plugins:

- bandwidth plugin
- firewall plugin
- flannel plugin
- Port-mapping plugin
- Source based routing plugin
- tuning plugin
- vrf plugin

IPAM:

- dhcp plugin
- host-local IP address management plugin
- static IP address management plugin

Lab_1- Installation

Check the CNI plugins directory:

```
ls -ltrh /opt/cni/bin
```

Installation:

```
kubectl apply -f https://github.com/intel/multus-cni/raw/master/images/multus-daemonset.yml
```

Contents after installation - kubeconfig and multus cni-config file:

```
ls -ltrh /etc/cni/net.d/
```

CRD network-attachment-definitions will be created.

```
kubectl get crds
```

Lab_2 - IPVLAN

Add another interface to the worker node and vyos router.

IPVLAN Network Attachment Definition:

```
kubectl create -f https://github.com/infinitydon/kubernetes-on-baremetal/raw/main/multus/ipvlan-net-def-1.yaml
```

Create deployment that will make use of the network definition:

```
kubectl create -f https://github.com/infinitydon/kubernetes-on-baremetal/raw/main/multus/deployment-ipvlan.yaml
```

Lab_2 - IPVLAN

Source routing:

```
echo 200 k8s >> /etc/iproute2/rt_tables
ip rule add from 172.18.0.3 table k8s
ip route add default via 172.18.0.1 table k8s
```

Add some privileged config to the deployment before we can modify the ip config in the POD:

```
        securityContext:
         capabilities:
           add:
           - NET_ADMIN
```

Source routing via the SRB plugin:

```
https://github.com/infinitydon/kubernetes-on-baremetal/raw/main/multus/ipvlan-net-def-1-srb.yaml
```

Lab_3 - WhereAbout IPAM

Host-local IPAM is not suitable when you need a common IP subnet especially when PODs are spread across multiple worker nodes, two of more PODs may be allocated the same IP address because IP pool intelligence is local to each worker node so it is likely that IP conflict will occur in the Layer 2 domain.

One of the solution to this is to use another IPAM solution that is called whereabout.

Installation of whereabout:

```
git clone https://github.com/dougbtv/whereabouts && cd whereabouts
```

```
kubectl apply -f ./doc/daemonset-install.yaml -f ./doc/whereabouts.cni.cncf.io_ippools.yaml
```

Files that are installed: ***daemonset, whereabout binary, config folder and kubeconfig file***

The network attachment definition will need to be adjusted so as to add the whereabout IPAM.

Lab_3 - WhereAbout IPAM

Create network definition attachment with the whereabout IPAM

```
kubectl apply -f https://github.com/infinitydon/kubernetes-on-baremetal/raw/main/multus/ipvlan-net-def-1-srb-whereabout.yaml
```

Create 2 deployment that will use whereabout network attachment:

```
kubectl apply -f https://github.com/infinitydon/kubernetes-on-baremetal/raw/main/multus/deployment-test-whereabout.yaml
```

Lab_4 - Bridge Plugin

Create network definition attachment with the whereabout IPAM

```
kubectl create -f
https://github.com/infinitydon/kubernetes-on-baremetal/raw/main/multus/bridge-net-def-whereabout.yaml
```

Create deployment that will use the bridge network attachment:

```
wget
https://github.com/infinitydon/kubernetes-on-baremetal/raw/main/multus/deployment-test-whereabout.yaml
```

Add enp0s9 to the bridge on the worker node and try to ping vyos router from the pod

```
sudo brctl addif br0 enp0s9
```

This will fail because the bridge traffic goes through iptables/netfilter (FORWARD Chain). There are several ways to resolve this:

Lab_4 - Bridge Plugin

1.) Check kernel settings:

```
sudo sysctl -a | grep net.bridge.bridge-nf-call
```

Remove bridge from iptables:

```
sudo sysctl net.bridge.bridge-nf-call-iptables=0
```

(2.)

```
sudo iptables -I FORWARD -j ACCEPT
```

Add back the kernel setting:

```
sudo sysctl net.bridge.bridge-nf-call-iptables=1
```

Lab_4 - Bridge Plugin

(3.) Allow specific subnets in the forward chain:

```
sudo iptables -D FORWARD -j ACCEPT
```

```
sudo iptables -I FORWARD -s 172.18.0.0/24 -j ACCEPT
```

References

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

https://github.com/dougbtv/whereabouts

https://www.cni.dev/plugins/