

# Turning your Kubernetes nodes into a Provider Edge Router

---

Federico Paolinelli - Red Hat

Miguel Duarte - Red Hat

# Agenda

---

- Why we want a router running on kubernetes nodes ?
- What's OpenPERRouter? (it's a router that you can run on your kuberentes nodes)
- How to integrate OpenPERRouter with your cluster ?
- What's ahead ?

Why a router on a kubernetes  
node?

---



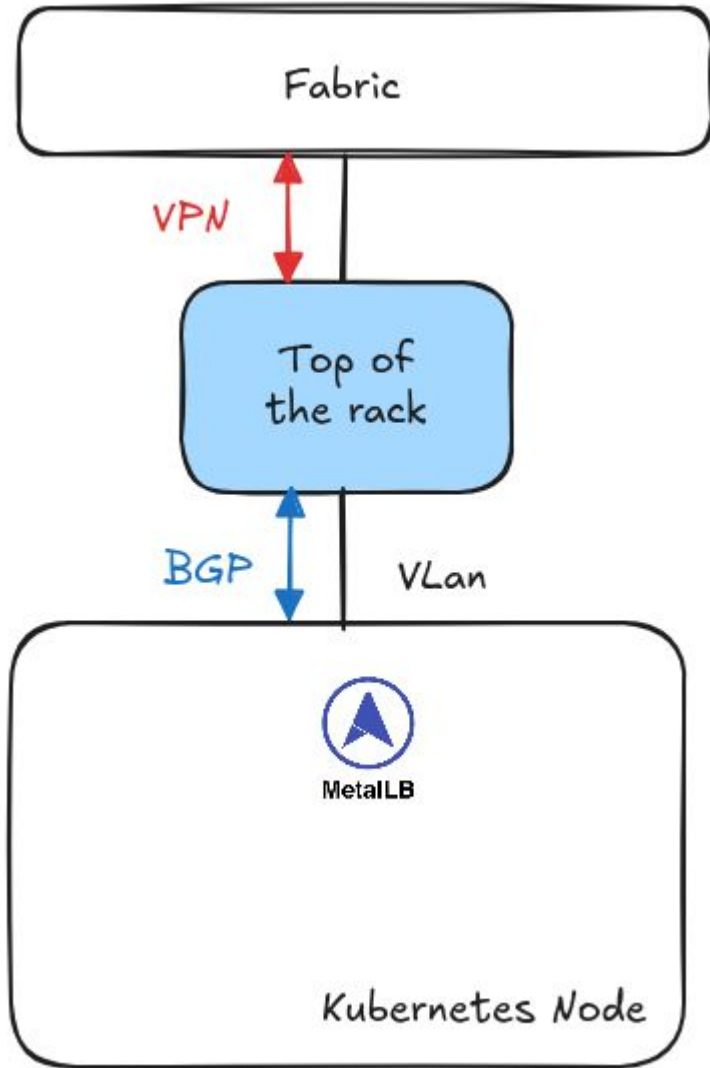
The magic  
land of cloud  
providers



The dark reign  
of bare metal

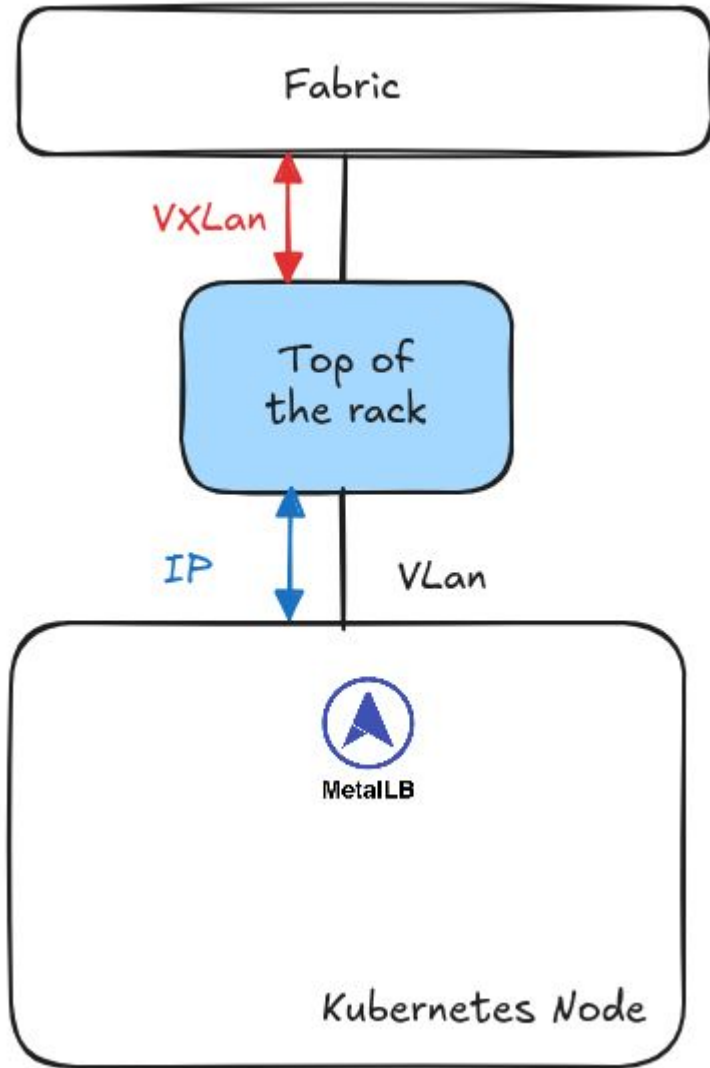
How is the fabric configured?

# The Control Plane



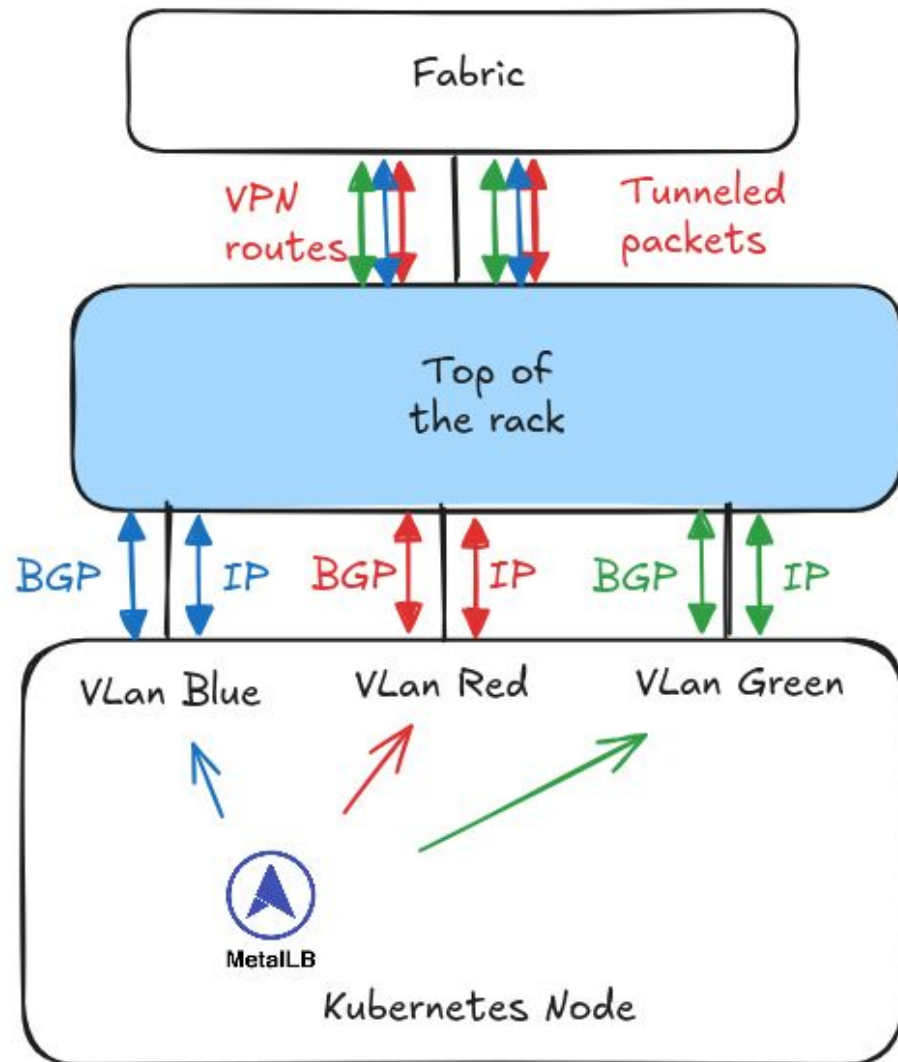
- BGP Speakers running on the node (ie MetalLB, FRR-K8s)
- BGP Routes are translated to the corresponding VPN Layer 3 routes
- The fabric needs to be configured!

# The Data Plane



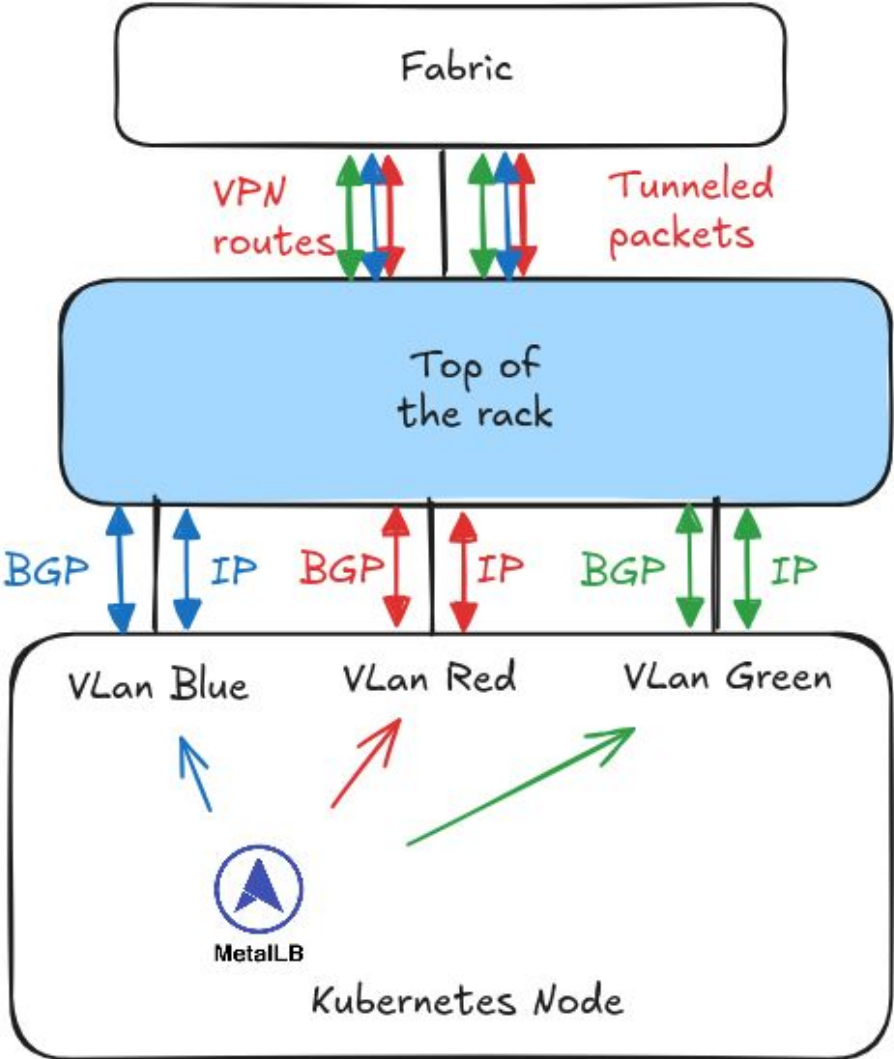
- The Leaves are configured to map each port / vlan to a tunnel
- The traffic is then purely routed across the whole fabric

# From the host point of view



- Multiple interfaces
- A BGP session over each interface
- Each interface is the entry point to / from a given network

# From the host point of view



Who configures the TOR?

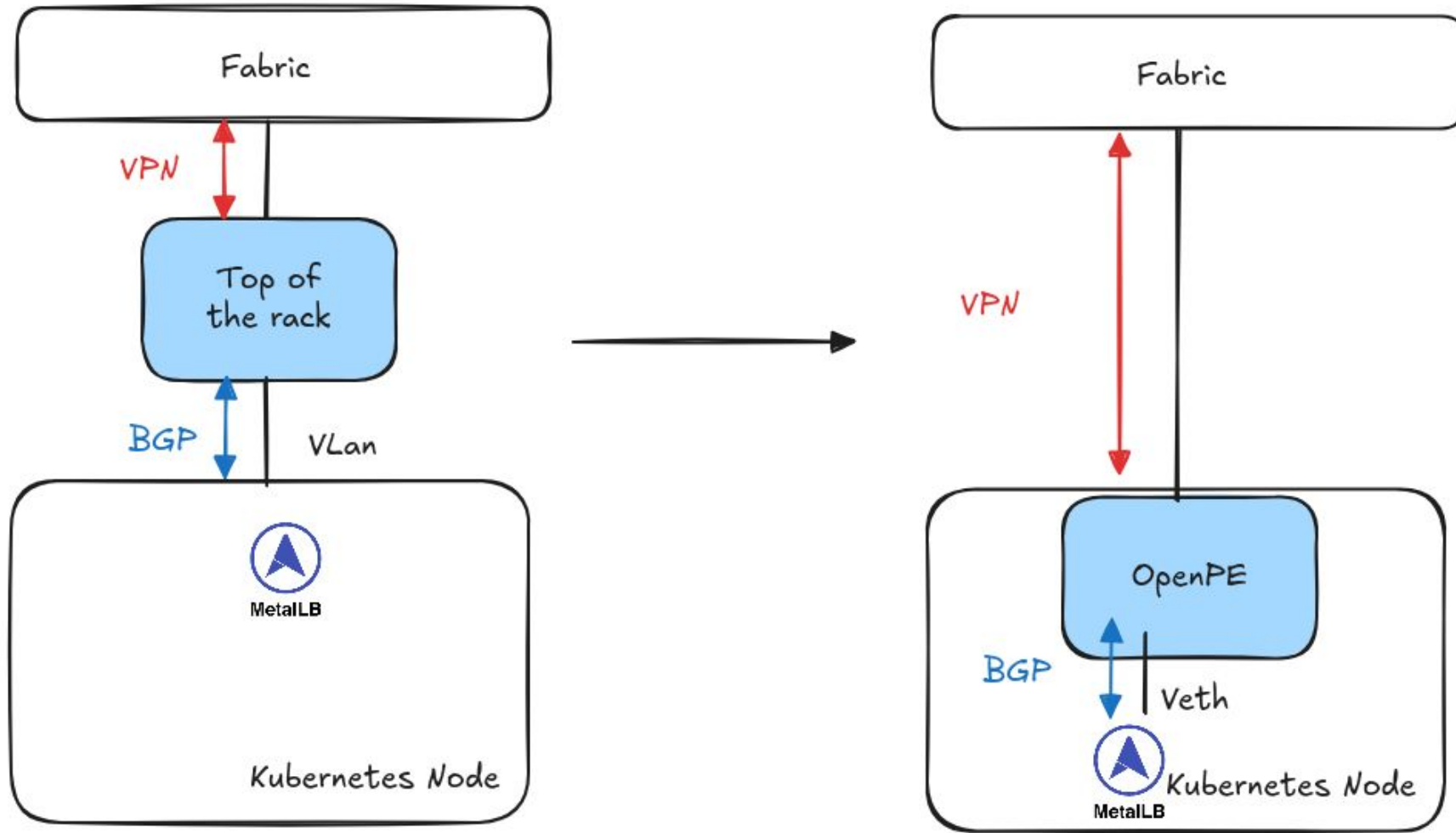
## Tension among Networking folks and “Cloud” folks



- No clear separation of responsibilities
- Reduced agility
- Risks to compromise the whole fabric

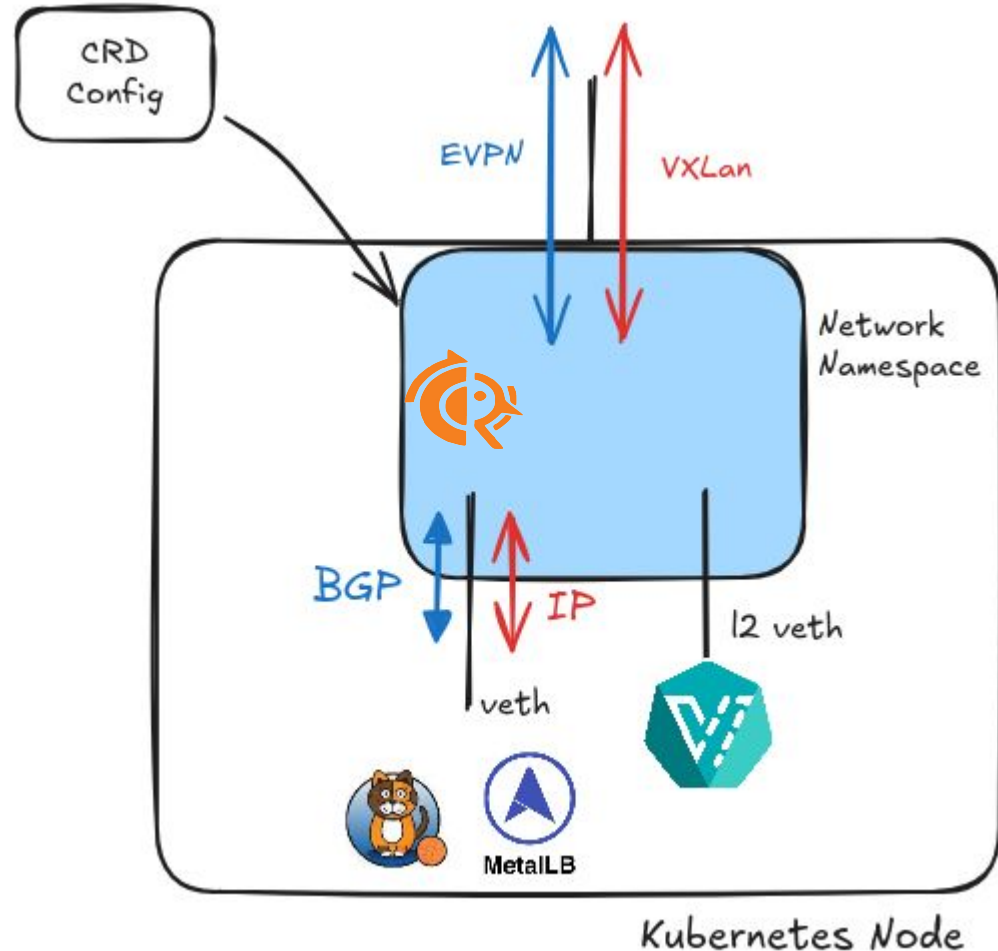
Let's bring the VPN  
termination to the node!

---



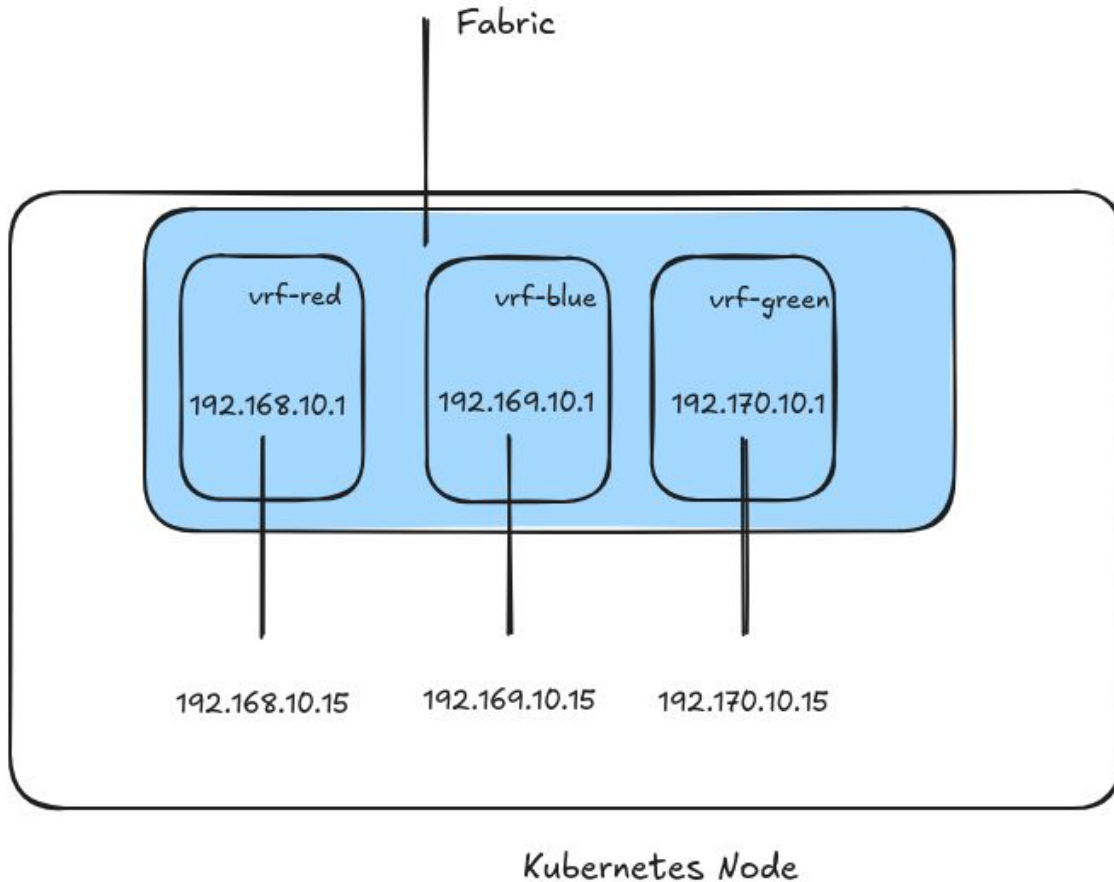
[openrouter.github.io](https://openrouter.github.io)

# OpenPERRouter



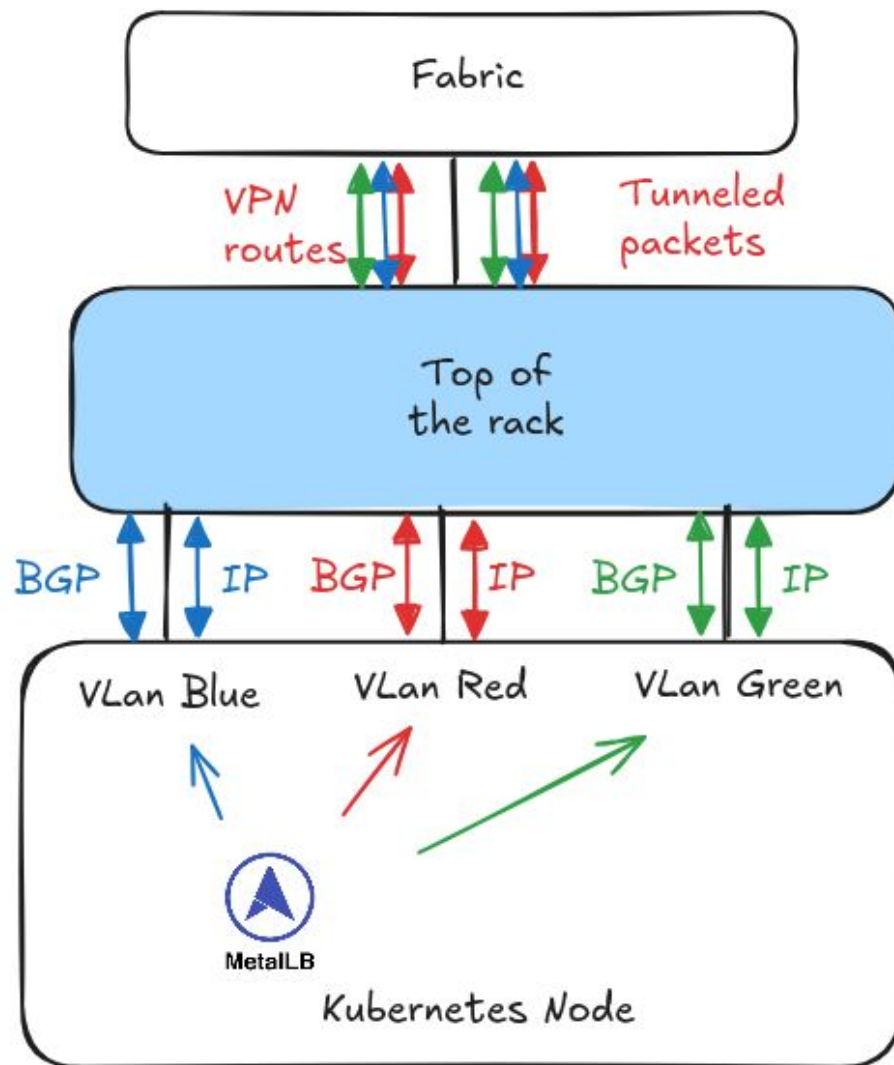
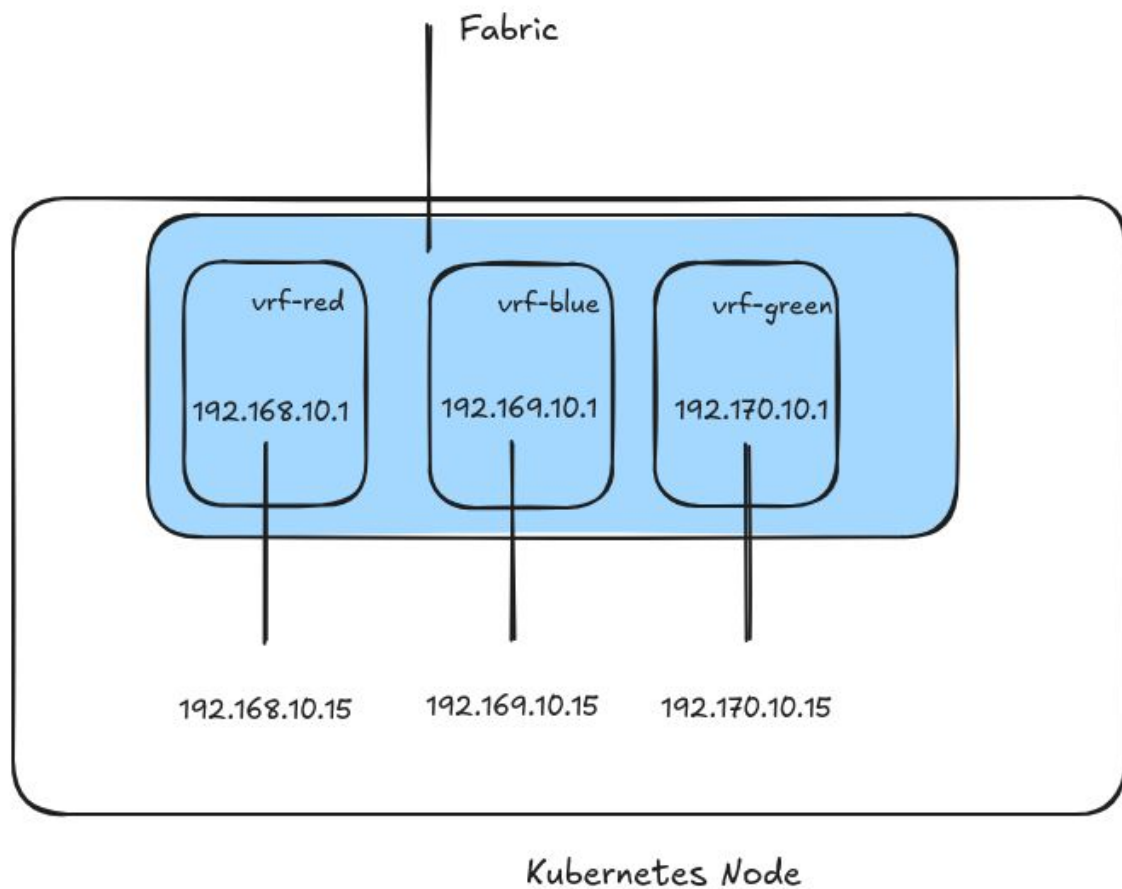
- Runs in a separate network namespace on the host
- Behaves like a router
- Control plane handled by FRR
- If a component can interact with a router, then it can interact with OpenPERRouter
- The configuration is CR driven

# OpenPERRouter



- The VPN tunnels and session creation is dynamic
- The host sees multiple interfaces, and multiple BGP peers

# OpenPERRouter

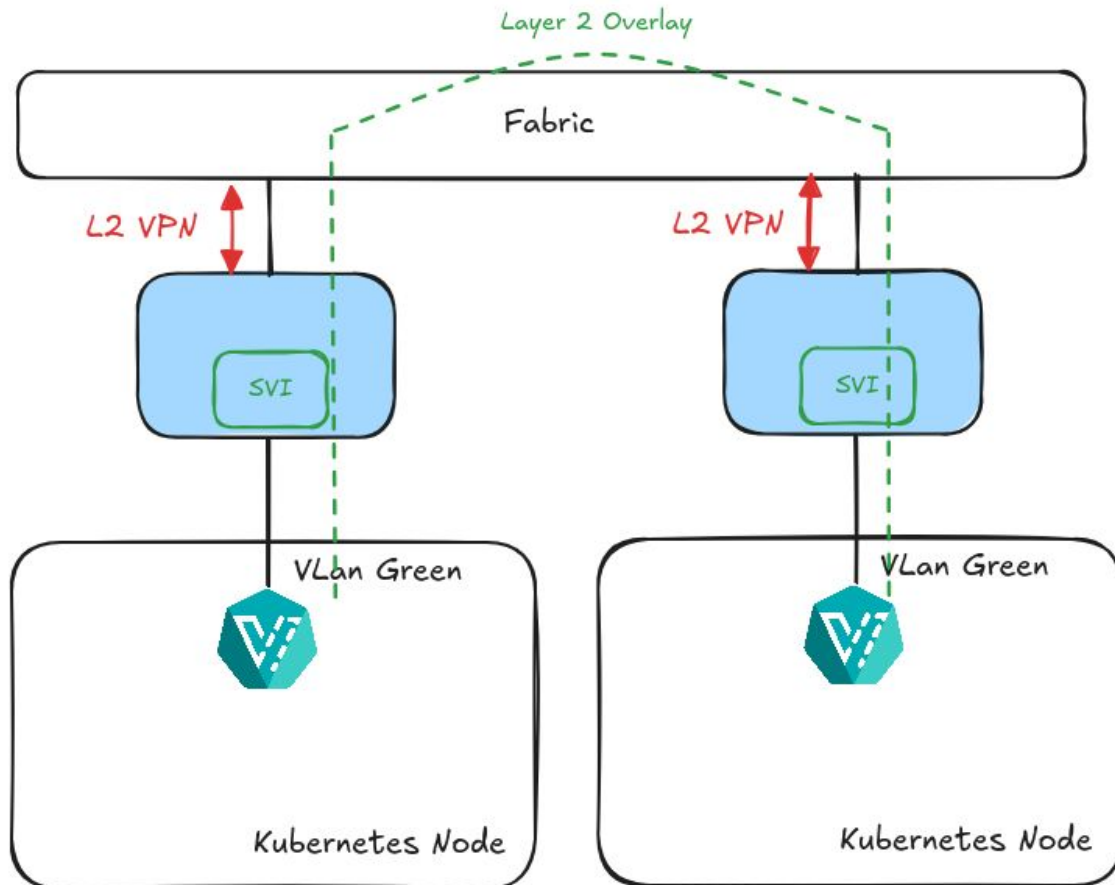


# How about layer 2?

---

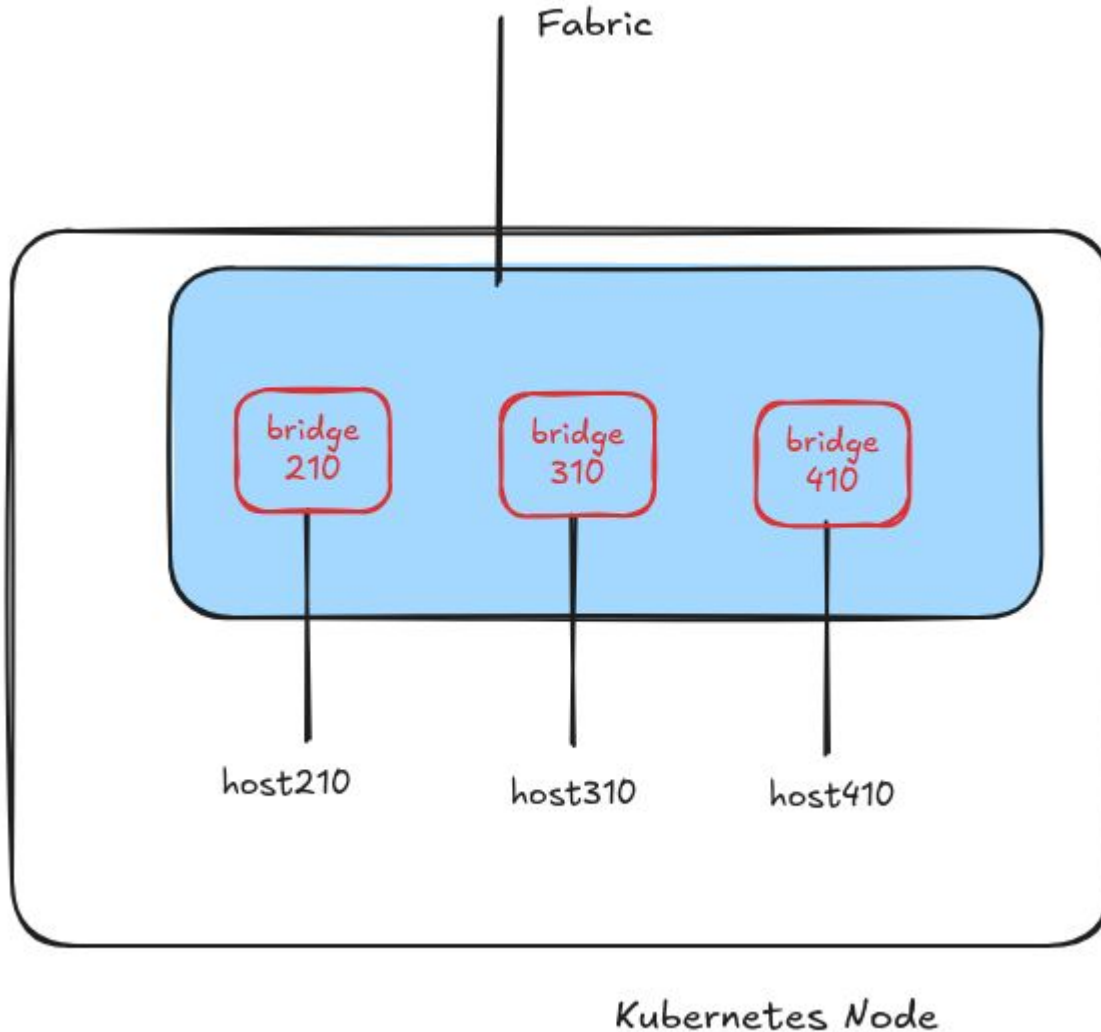


# In a regular datacenter



- Leaves configuration allows stretched layer 2 overlay
- The control plane takes care of MAC reachability

# With Open PE Router

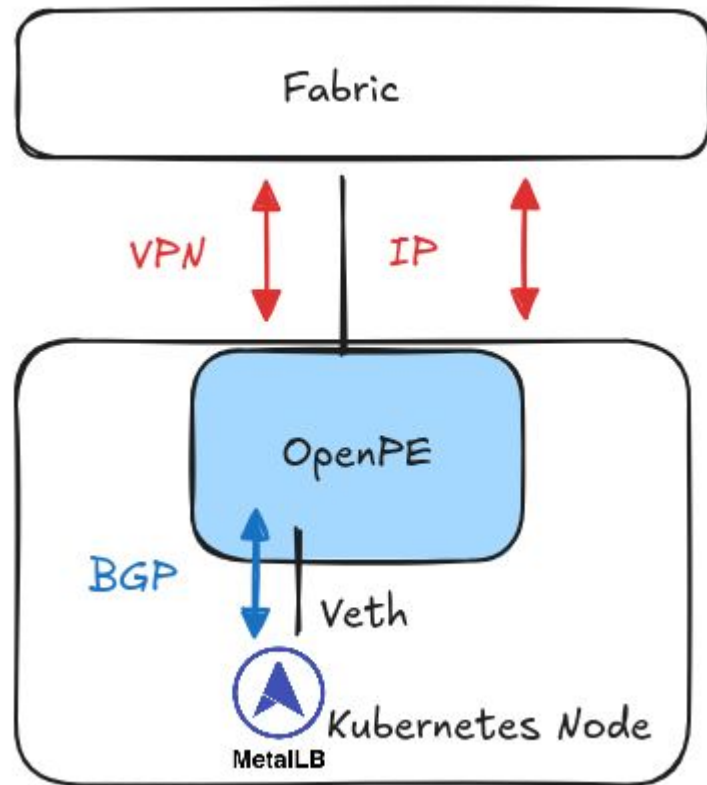


- CR driven
- Each veth is bridged to a Layer 2 Overlay
- The host can plug the veth to any switching mechanism

The fabric becomes dumber

---

# The fabric



- No tunnel is configured at fabric level
- Leaf routers need to relay the VPN routes
- The traffic is purely routed towards the tunnel destination
- No need for extra configuration in case of new networks

# Dumber Leaves

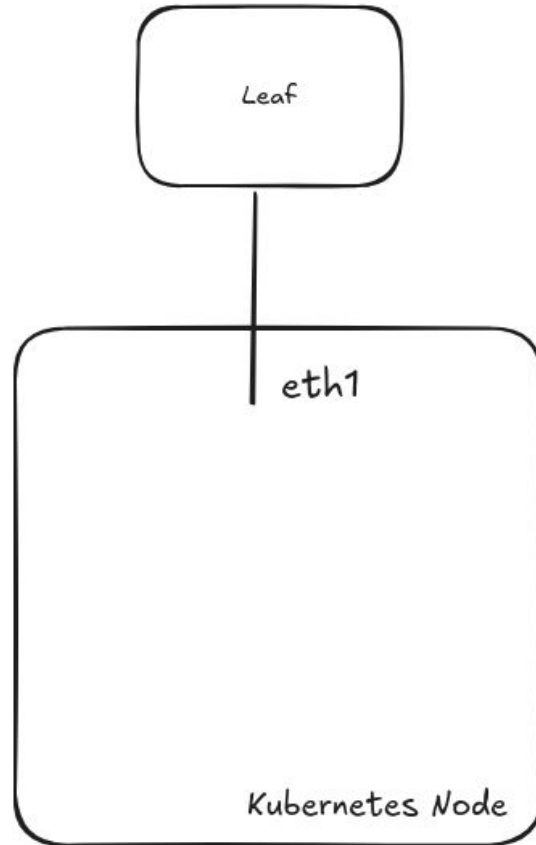


- No need to involve the networking people
- The network configuration is declarative and dynamic
- Very easy to add additional overlays

EVPN - VXLAN

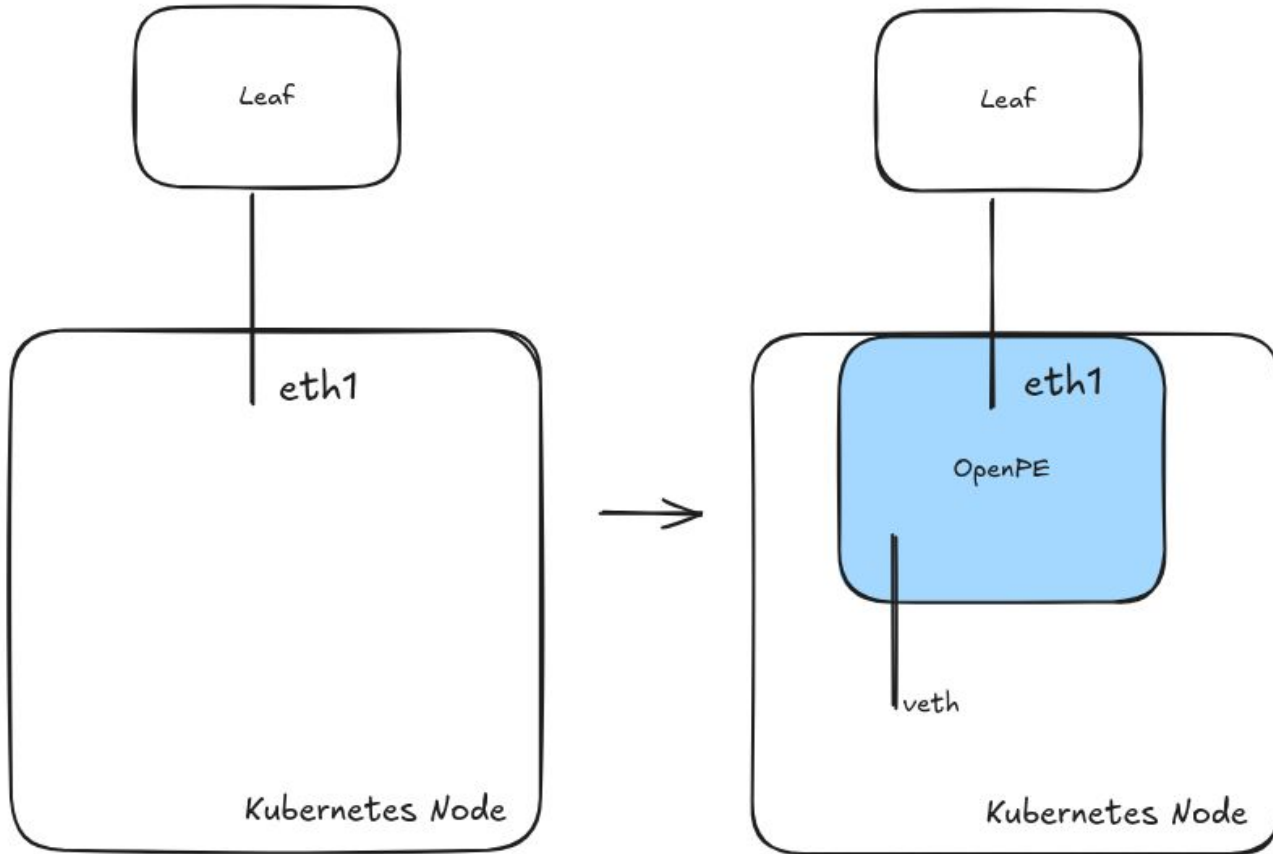
---

# Underlay configuration



- Underlay is the session with the TOR
- We need a session from the network namespace

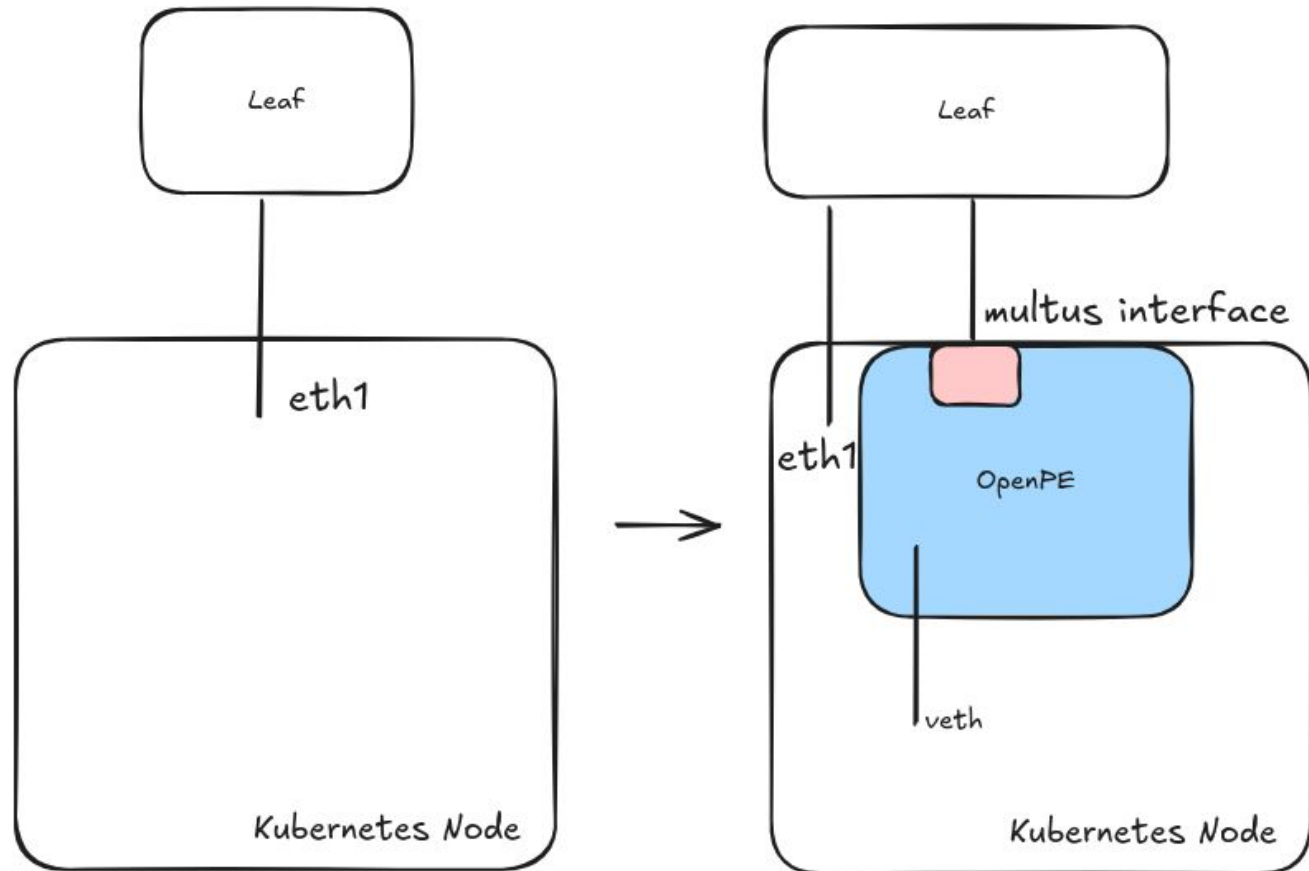
# Underlay configuration



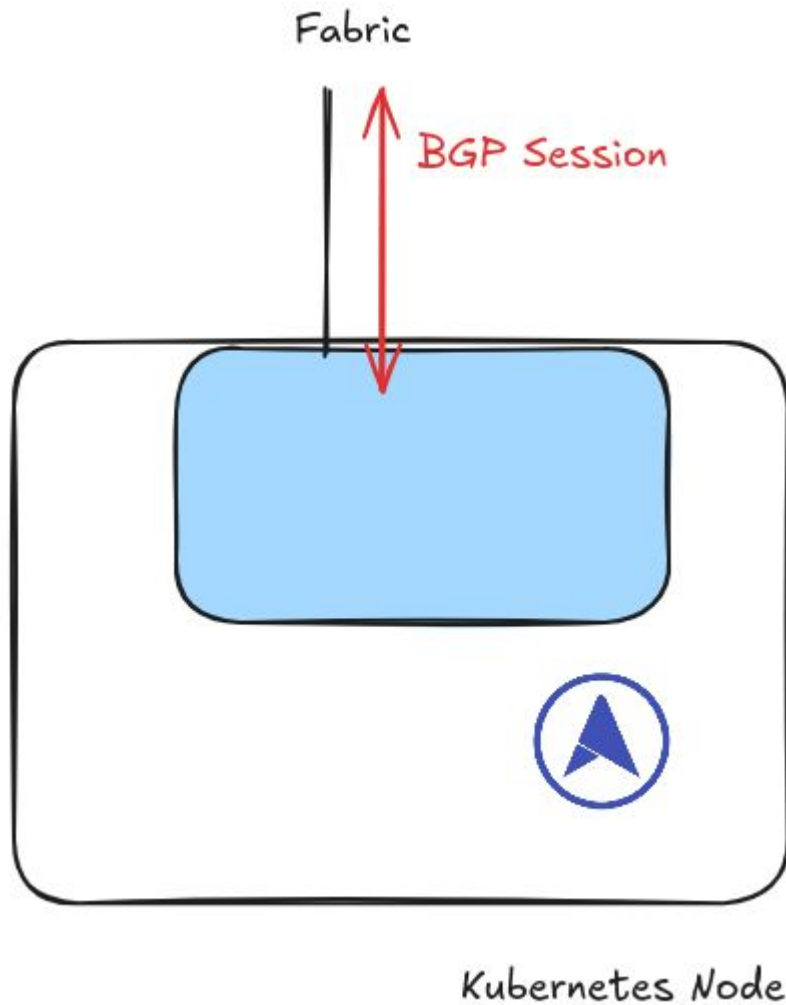
Option 1: move the interface in the network namespace

# Underlay configuration

Option 2: use a multus interface



# Underlay configuration



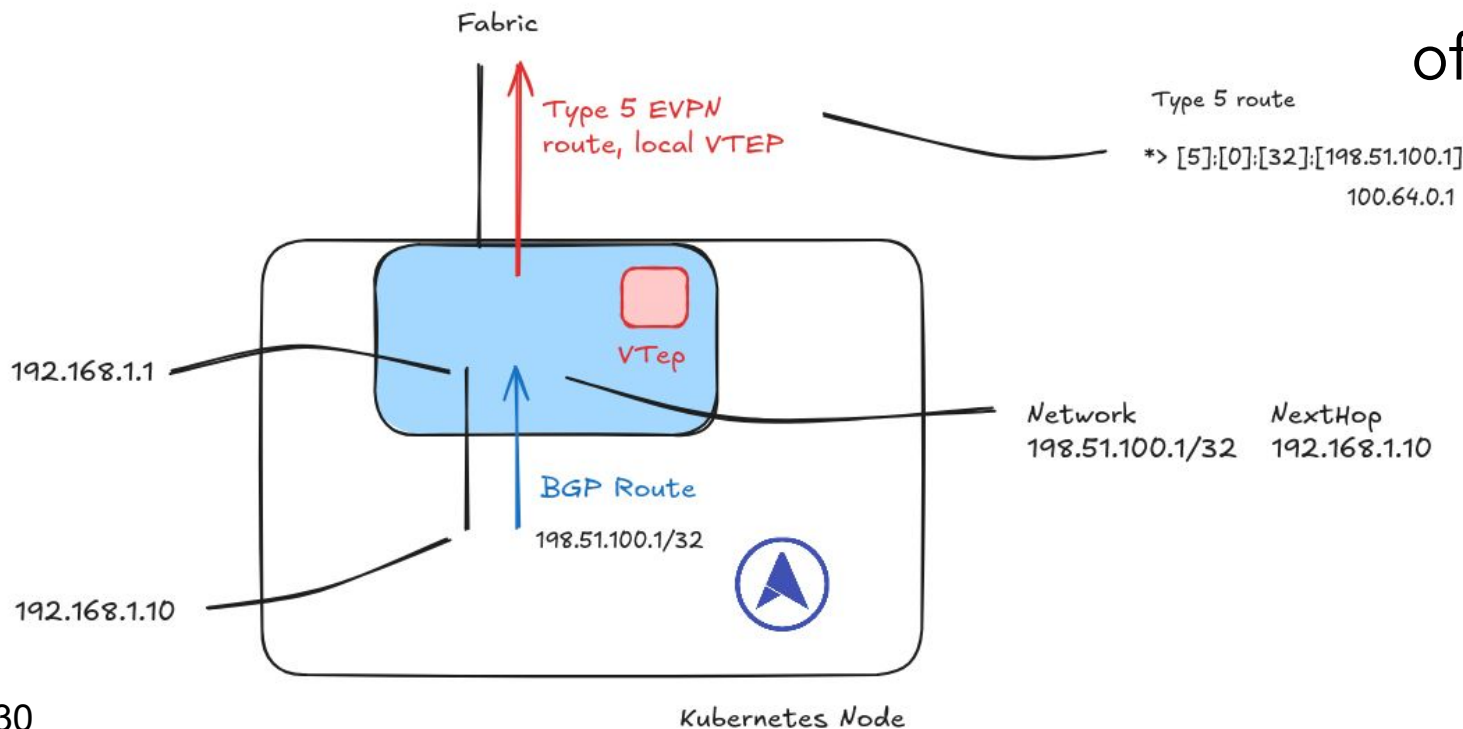
One BGP Session with the TOR

# Underlay configuration

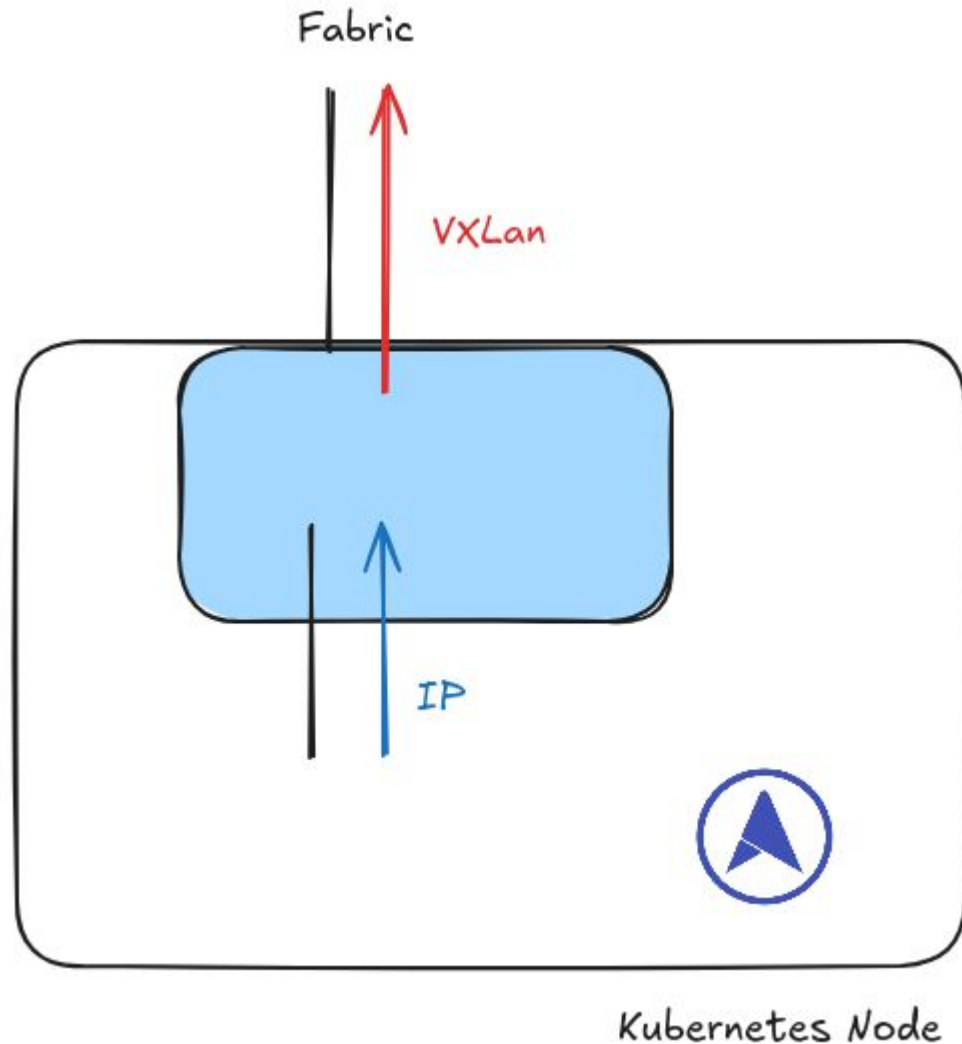
```
apiVersion: openpe.openrouter.github.io/v1alpha1
kind: Underlay
metadata:
  name: underlay
  namespace: openrouter-system
spec:
  asn: 64514
  nics:
    - toswitch
  neighbors:
    - asn: 64512
      address: 192.168.11.2
```

# L3 VPN - L3 VNI

- One veth per VNI
- The L3 routes corresponding to the VNI are propagated to the session of the corresponding veth



# L3 VPN - L3 VNI

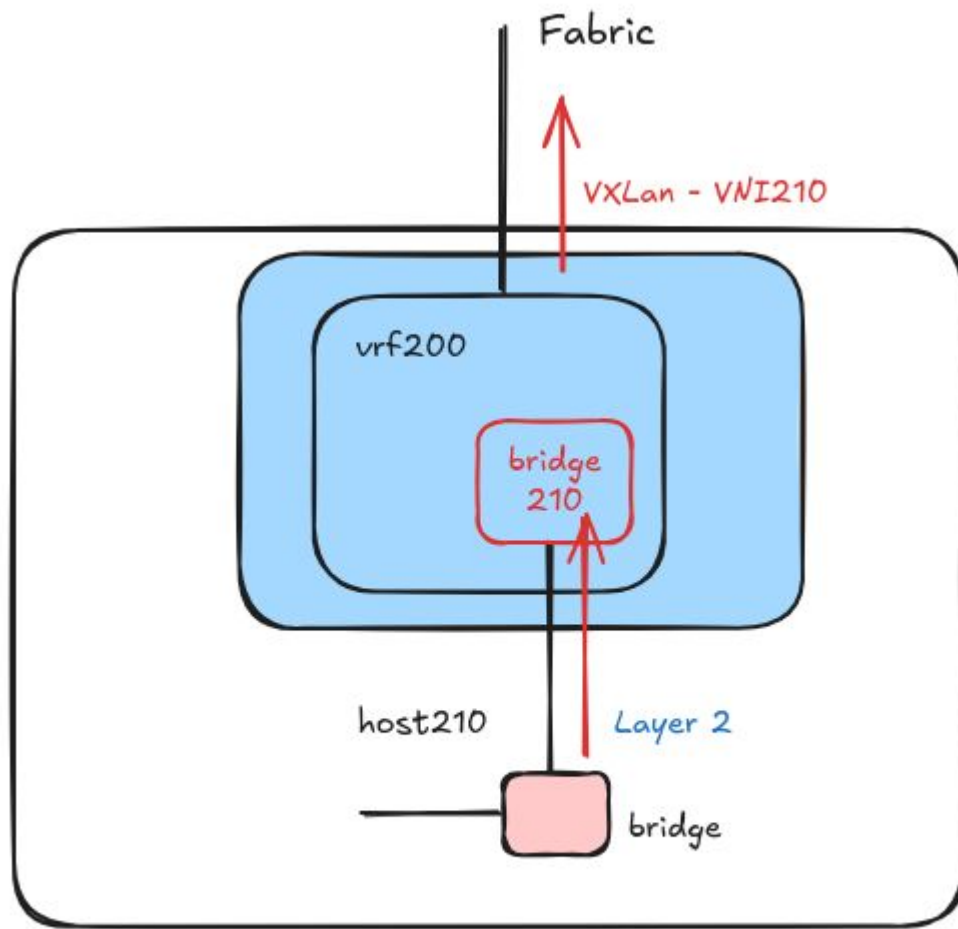


- The traffic coming from the veth is VXLAN encapsulated
- The VXLAN packets are routed to the right destination

# L3 VPN - L3 VNI

```
apiVersion: openpe.openrouter.github.io/v1alpha1
kind: L3VNI
metadata:
  name: blue
  namespace: openrouter-system
spec:
  hostsession:
    asn: 64514
    hostasn: 64515
    localcidr:
      ipv4: 192.169.11.0/24
  vrf: blue
  vni: 200
```

# L2 VPN - L2 VNI



- One veth per VNI
- The veth is connected to the layer 2 domain
- On the host side, it can be plugged to any virtual switch
- Can belong to a layer 3 routing domain associated to a different vni (expose the network)

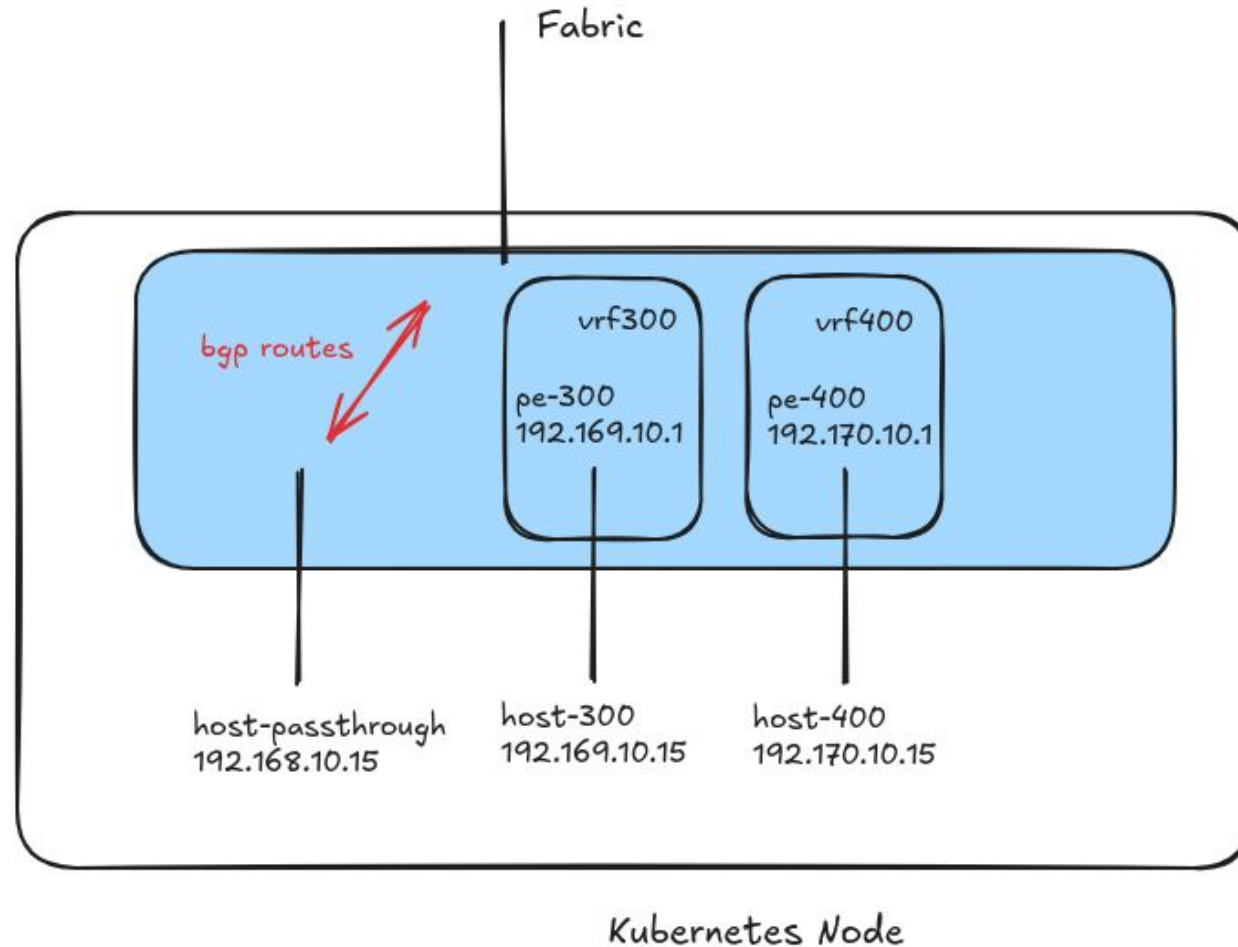


# L2 VPN - L2 VNI

```
apiVersion:  
openpe.openrouter.github.io/v1alpha1  
kind: L2VNI  
metadata:  
  name: l2red  
  namespace: openrouter-system  
spec:  
  vni: 210  
  vrf: red  
  hostmaster:  
    type: bridge  
    autocreate: true
```



# Passthrough mode



- The host receives / advertises routes to the fabric
- Direct access without tunneling

Integrating it with existing  
projects

---

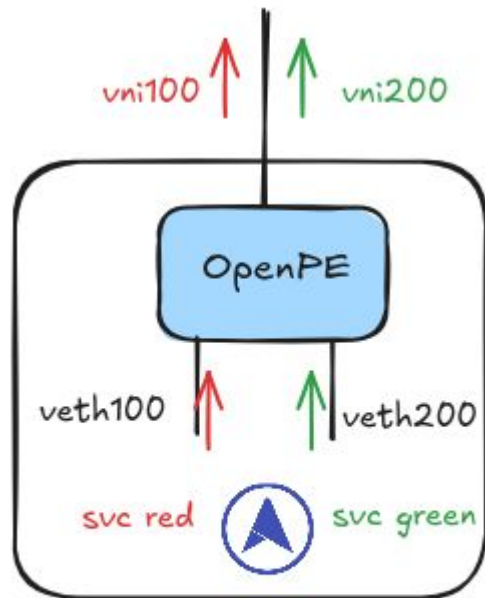
*“If a component can interact with a real router, then it can be enabled to EVPN with Open PE Router”*

# Layer 3 integrations

- MetalLB - to expose LoadBalancer services via EVPN
- Calico - to allow east - west traffic via EVPN

[openrouter.github.io/docs/examples/](https://openrouter.github.io/docs/examples/)

# MetalLB

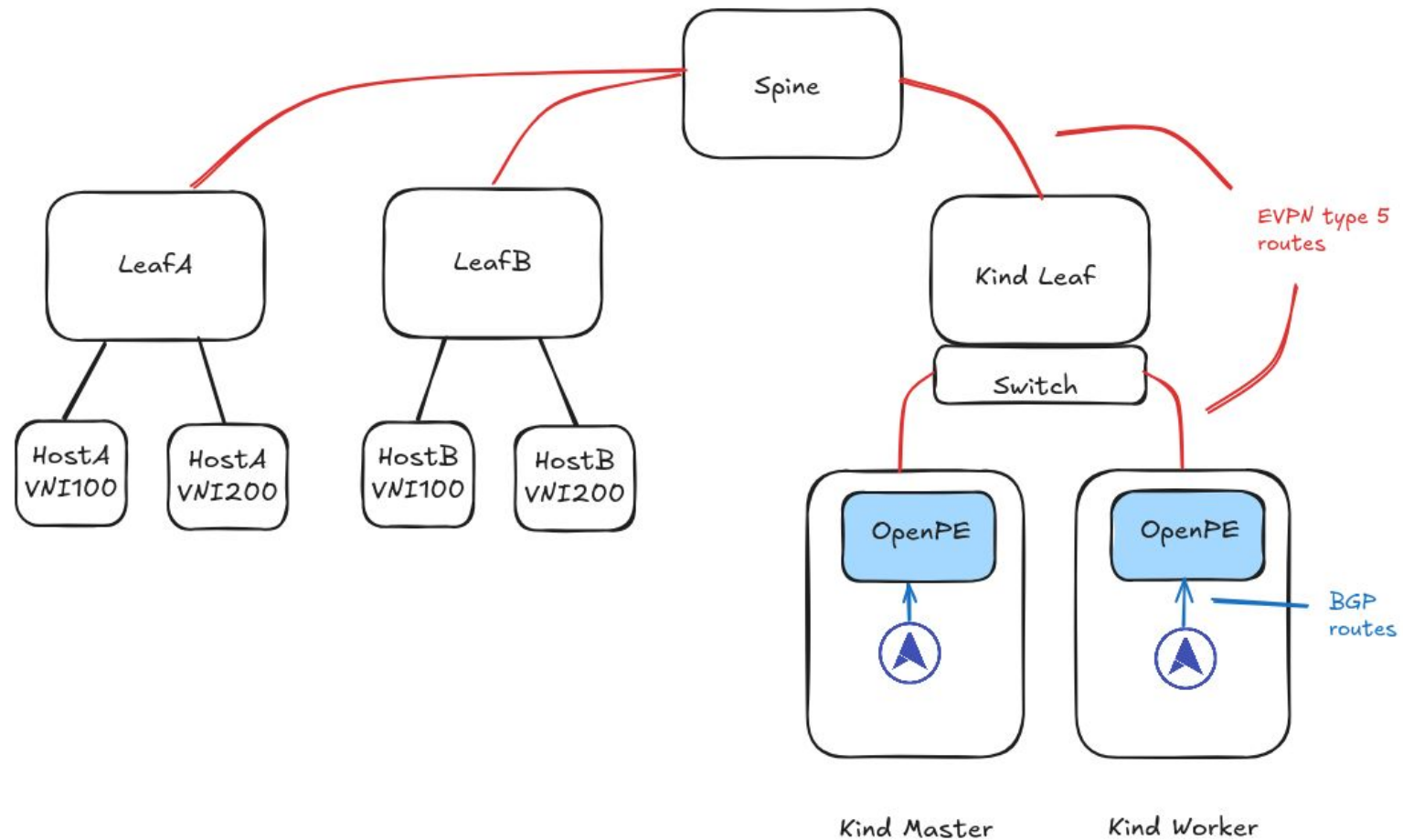


Kind Worker

- One BGP session per veth
- Each veth is associated to a different VNI
- Different services can be associated to different networks

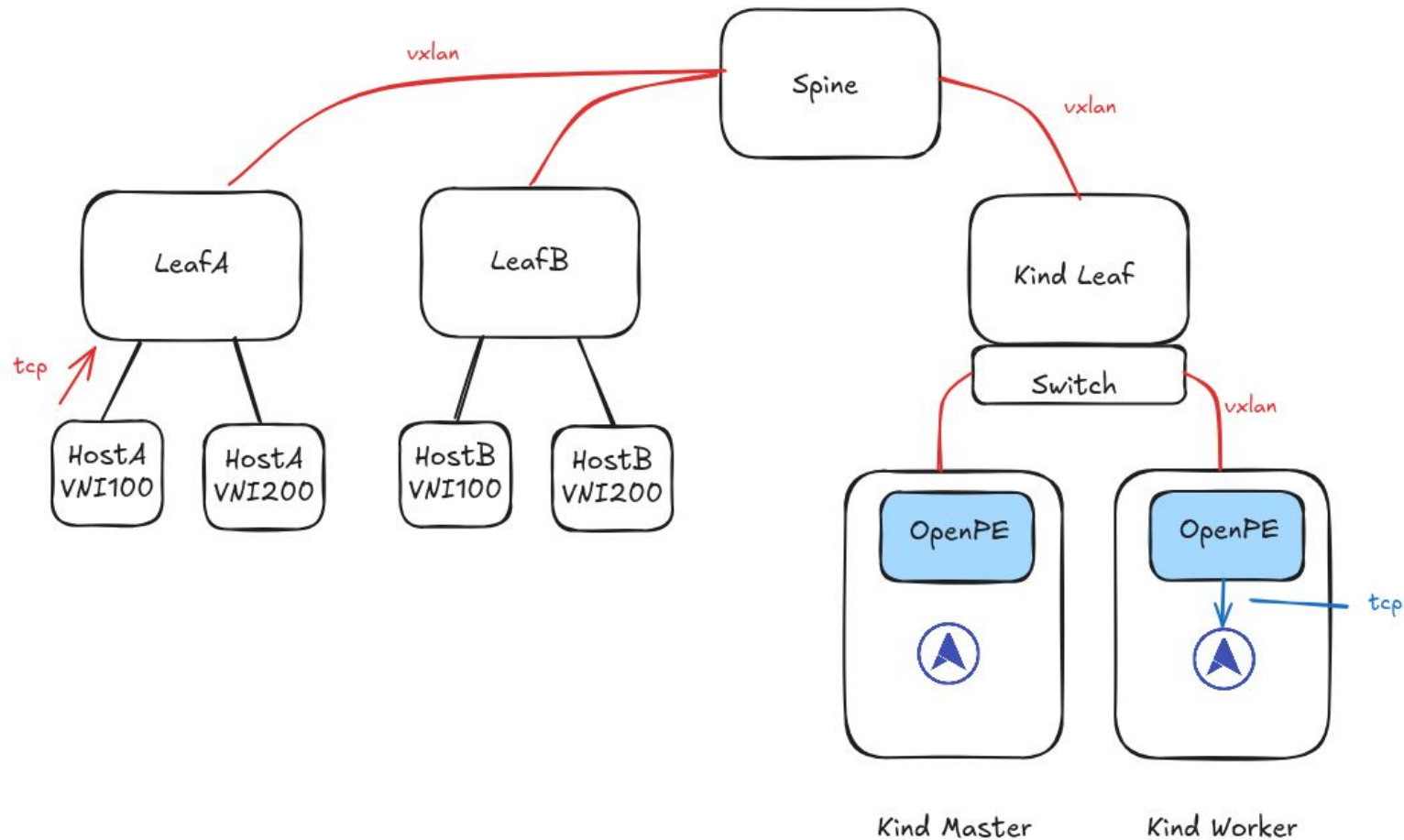
# MetalLB

BGP Routes are translated to EVPN type 5 routes

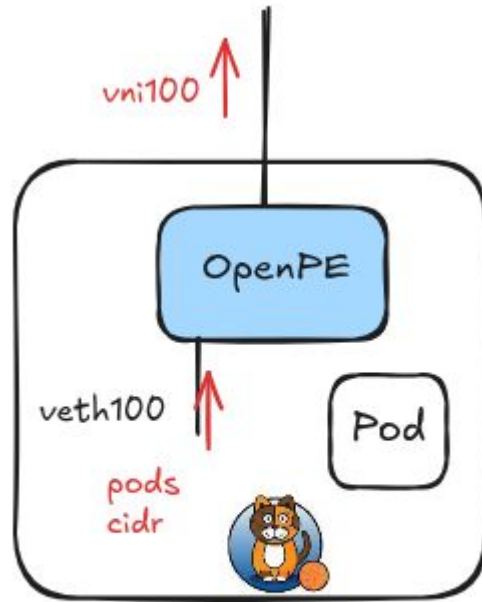


# MetalLB

The traffic entering the veths is vxlan encapsulated / decapsulated



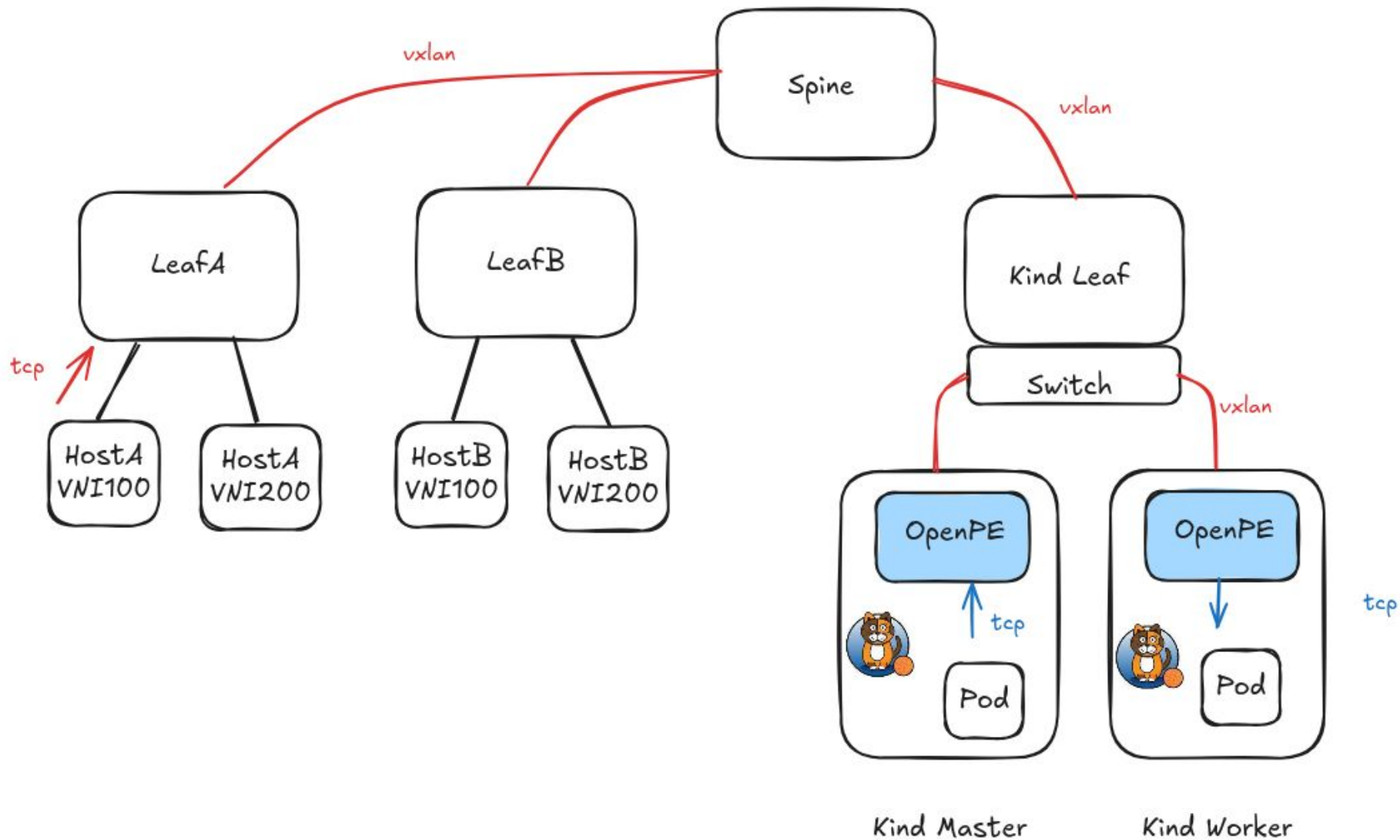
# Calico



Kind Worker

- Calico peered with OpenPERouter via a veth
- Routes are sent / received
- Pod to pod traffic happens through VXLAN
- The pods are routable from the fabric

# Calico



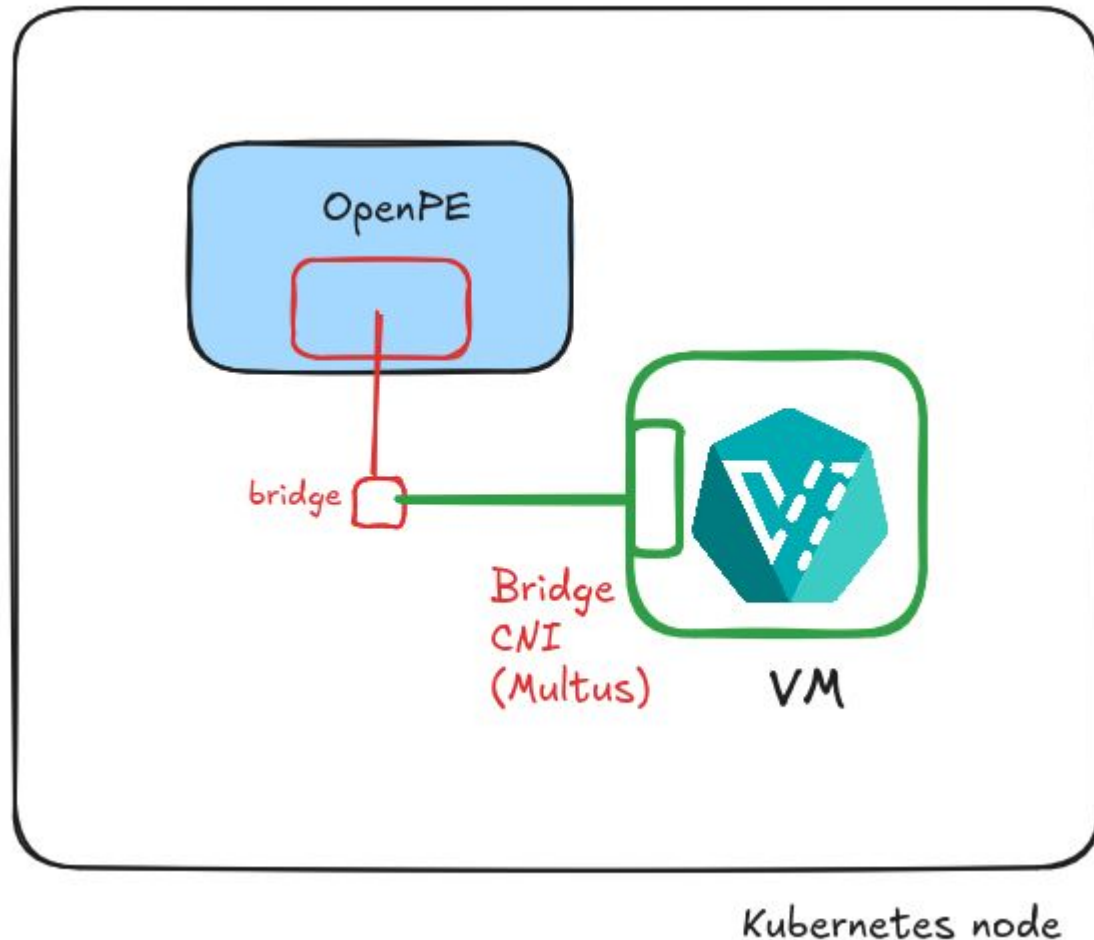
# Layer 2 integrations

- Kubevirt - to connect multiple virtual machines via L2 over a VXLAN tunnel
- Pod to pod layer 2 via secondary interfaces

[openrouter.github.io/docs/examples/](https://openrouter.github.io/docs/examples/)



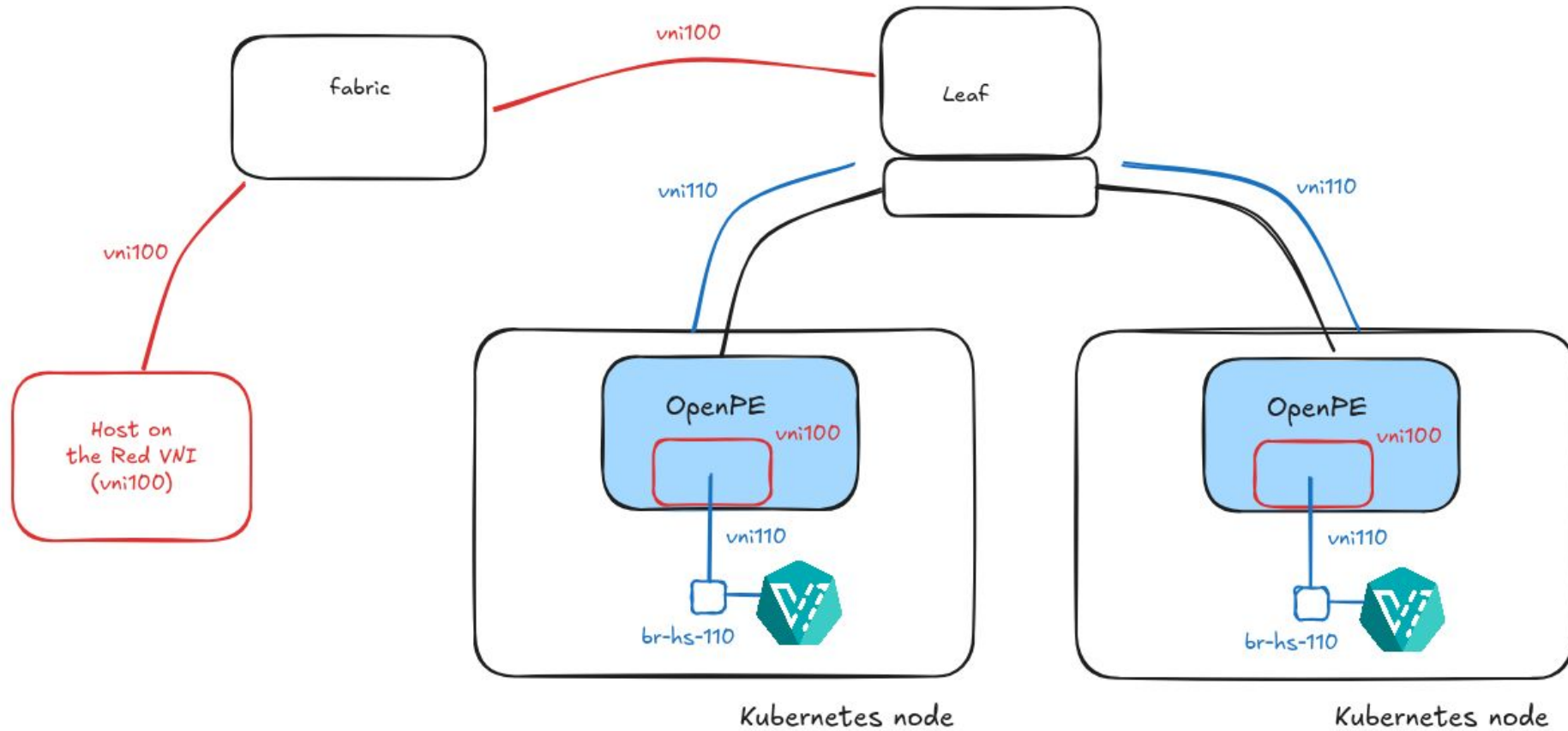
# Kubevirt



- L2 VNI connected to a bridge on the host
- Multiple ways to connect the VM to the L2 overlay (bridge CNI, OVS-CNI, etc)
- Distributed Gateway IP
- Live migration enabled



# Kubevirt



# Demos!

---



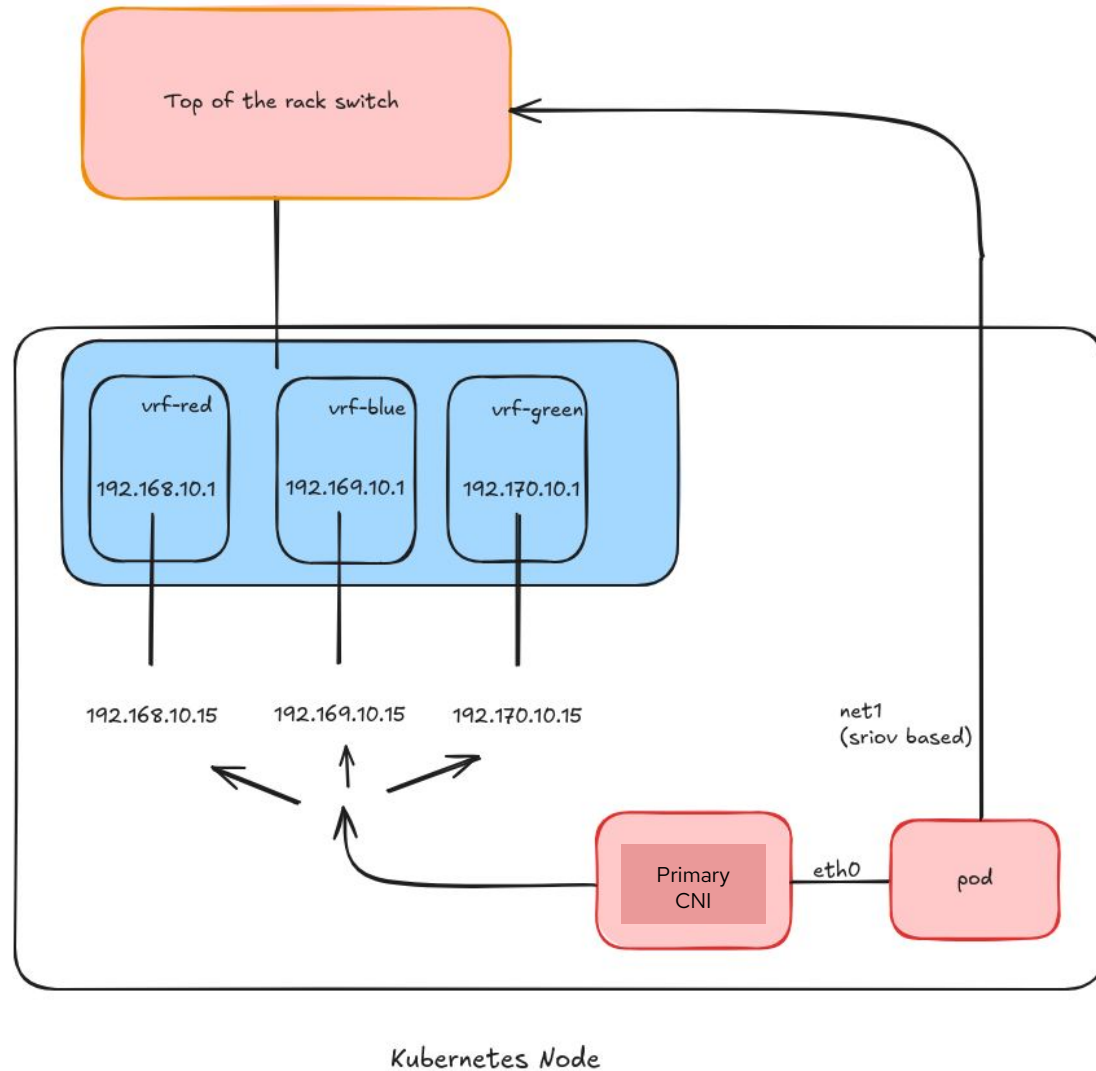
[tinyurl.com/db3atev3](https://tinyurl.com/db3atev3)



What's next?

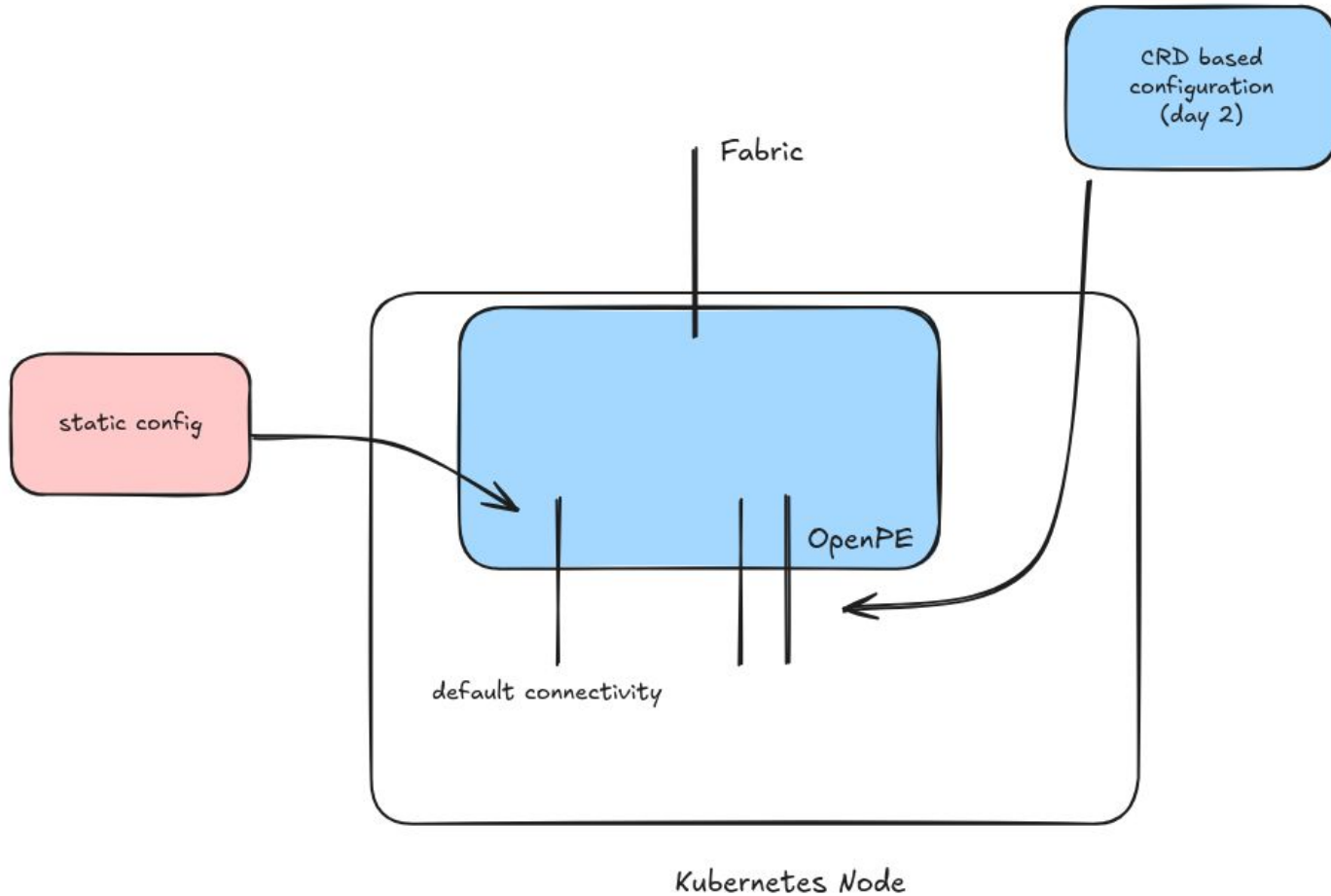
---

# SR-IOV interfaces



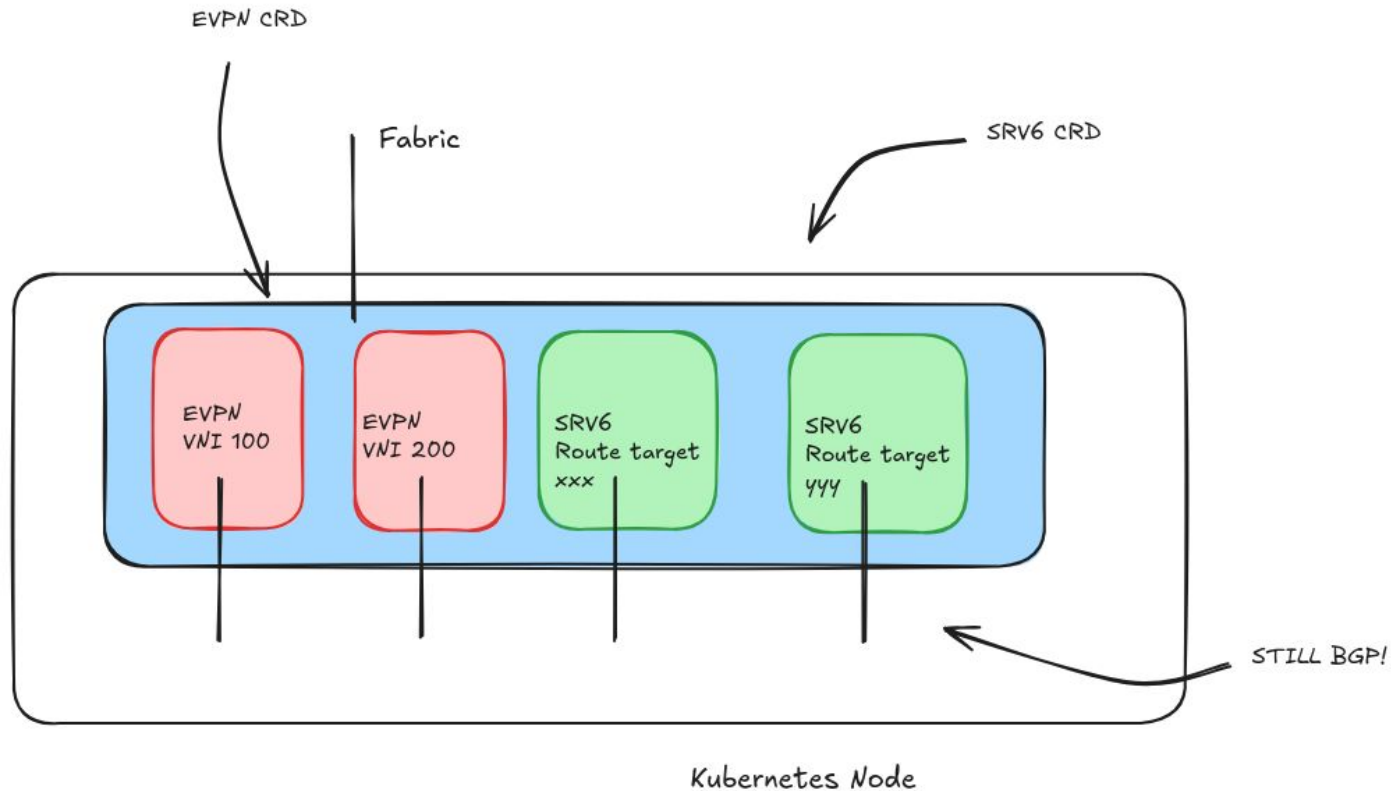
- SR-IOV interfaces bypass the host
- The fabric needs to be configured

# Day 0 mode



- The Open PE Router will start as a systemd unit, providing access to the default gateway
- A static configuration is read at boot time
- Extra VPNs can be created at runtime

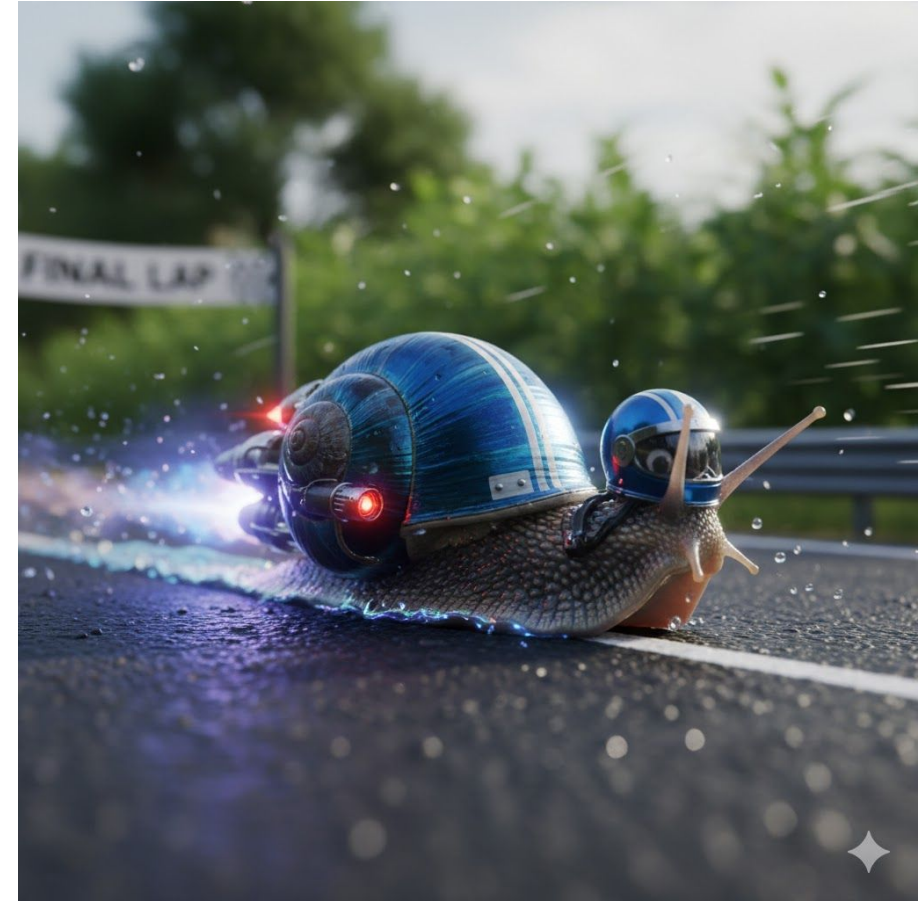
# SRV6



- Just a different VPN type
- No changes on the host side!
- No further integration requirements

# A faster datapath

- The current datapath is kernel based
- SR-IOV interfaces problem
- Kernel speed / determinism



# grout # a graph router based on DPDK



- Grout has FRR integration support
- Long term vision: pluggable datapath, Kernel, Grout, ...

<https://github.com/DPDK/grout>

# Resources

- Upstream project and documentation [openrouter.github.io](https://openrouter.github.io)
- Examples! [openrouter.github.io/docs/examples](https://openrouter.github.io/docs/examples)
- Federico's talk at Fosdem 2025:  
[archive.fosdem.org/2025/schedule/event/fosdem-2025-5246-running-an-evpn-endpoint-in-a-kubernetes-cluster-on-my-laptop-/](https://archive.fosdem.org/2025/schedule/event/fosdem-2025-5246-running-an-evpn-endpoint-in-a-kubernetes-cluster-on-my-laptop-/)

# Wrapping Up

---

# Thanks! Questions?

---

[fpaline@redhat.com](mailto:fpaline@redhat.com)  
[mdbarroso@redhat.com](mailto:mdbarroso@redhat.com)