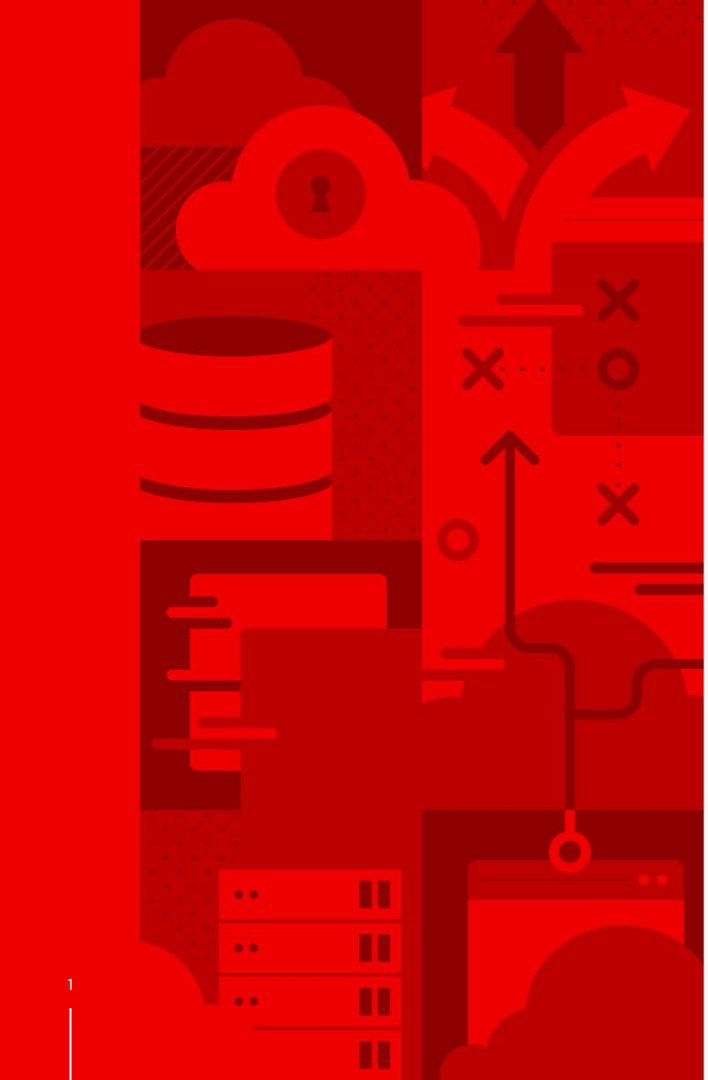


Weaving the Fabric

EVPN overlays for multi cluster KubeVirt deployments

FOSDEM 2026

Federico Paolinelli - Red Hat
Miguel Duarte - Red Hat

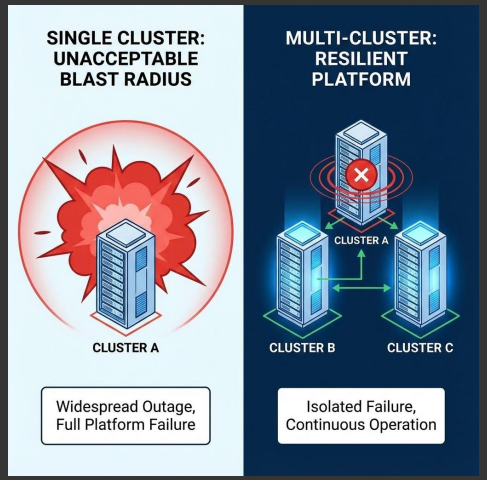
Agenda

- Motivation
- EVPN
- Implementation
- Demos
- Conclusions

Motivation

Legacy applications requirements

Platform resiliency



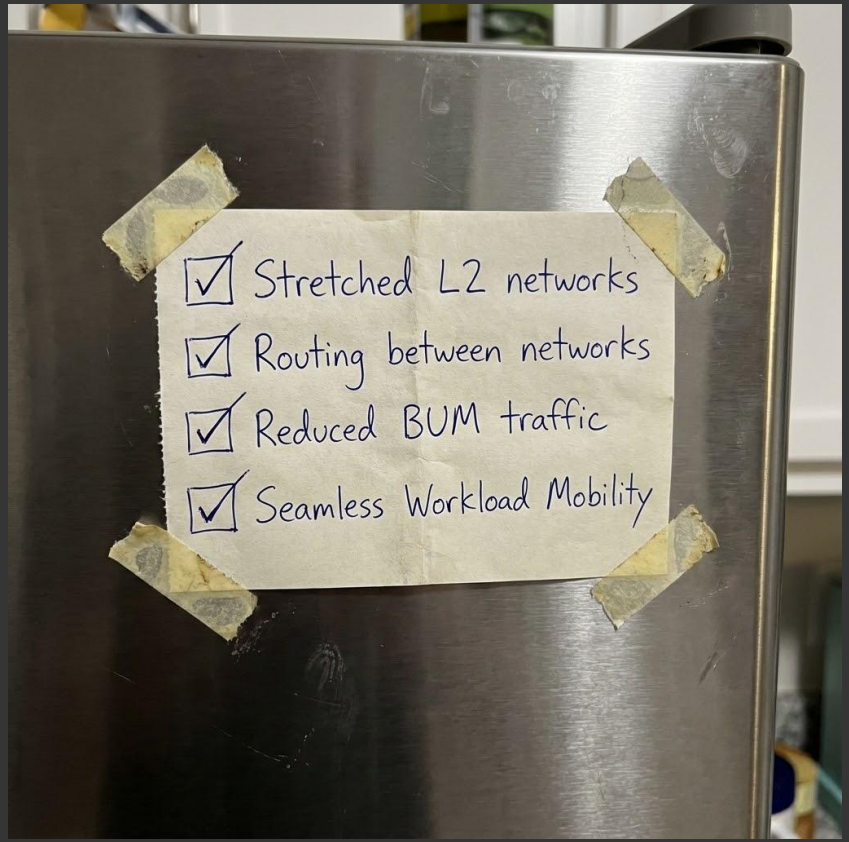
Disaster recovery
Scaling
Hybrid cloud

Goals

- Stretched L2 network across the fabric
  - Cross cluster live migration
  - Resource pooling across multiple clusters

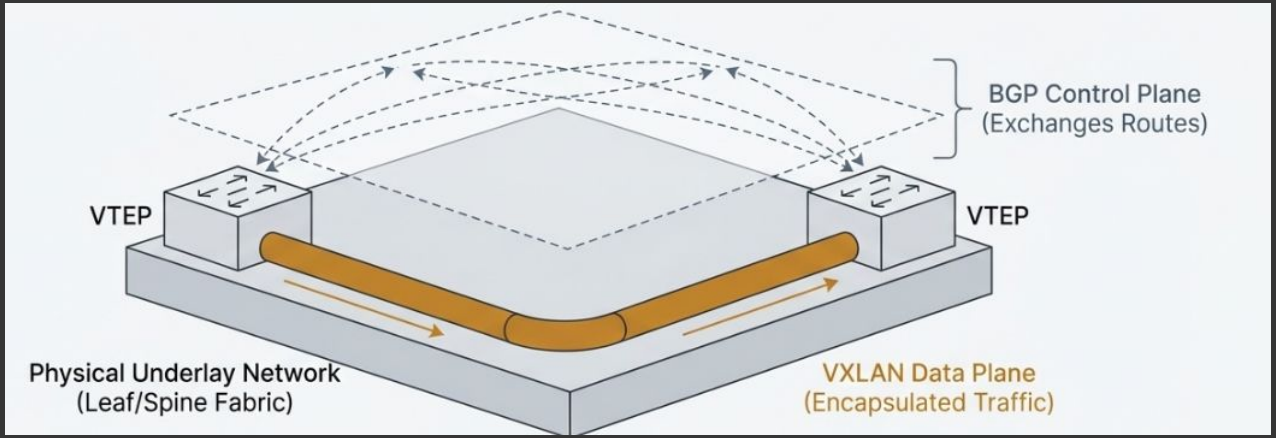
- Routing between networks
  - Traffic segregation
  - Direct routed ingress to VMs
    - No need to expose services
    - No NAT

EVPN


Stretched L2 networks
Routing between networks
Reduced BUM traffic
Seamless Workload Mobility

EVPN

- Ethernet VPN
- Control plane: BGP
- Data plane: **VXLAN** / MPLS / SRv6 / ...

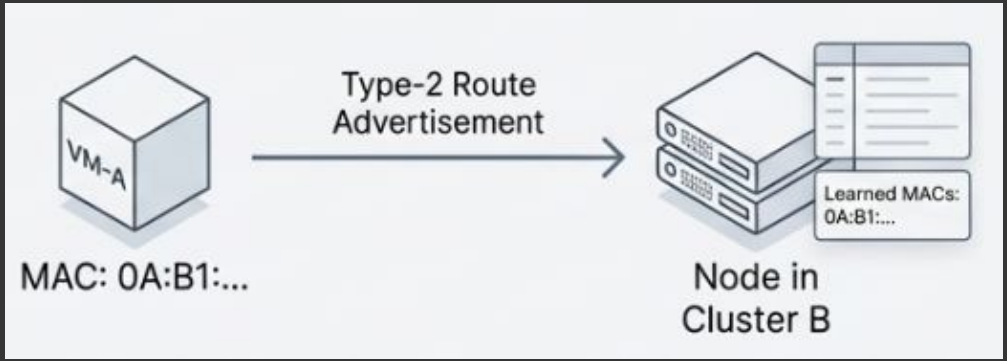


EVPN control plane

- MP-BGP
- Different route types
  - Type 2 routes
  - Type 5 routes

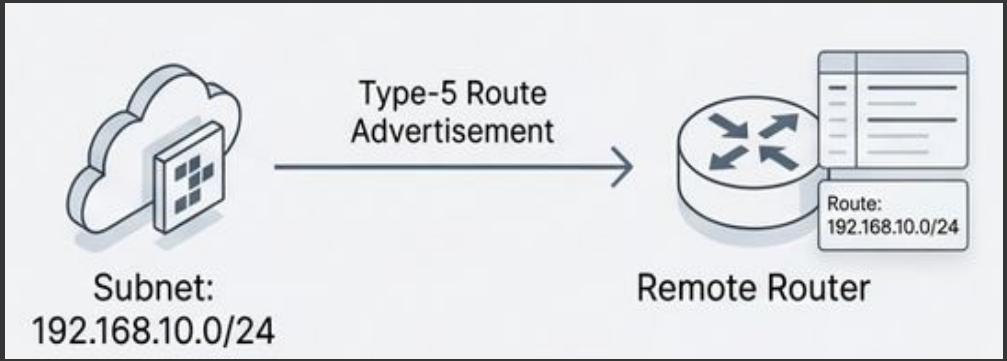
EVPN - Stretching a network (Layer 2)

- Type 2 routes: advertise VM MAC / IP address



EVPN - Routing between networks (Layer 3)

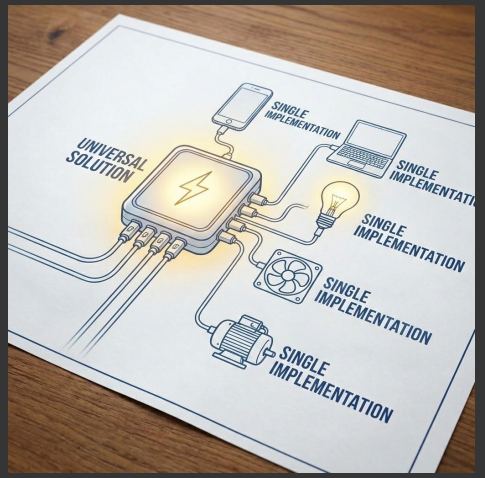
- Type 5 routes: advertise entire prefixes



Implementation

Design principles

Integrate existing solutions

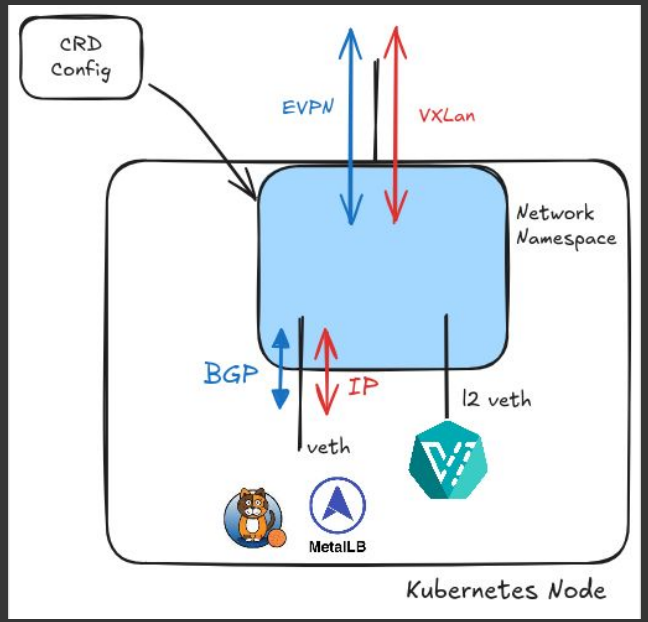


Work with multiple CNI plugins

Building blocks:

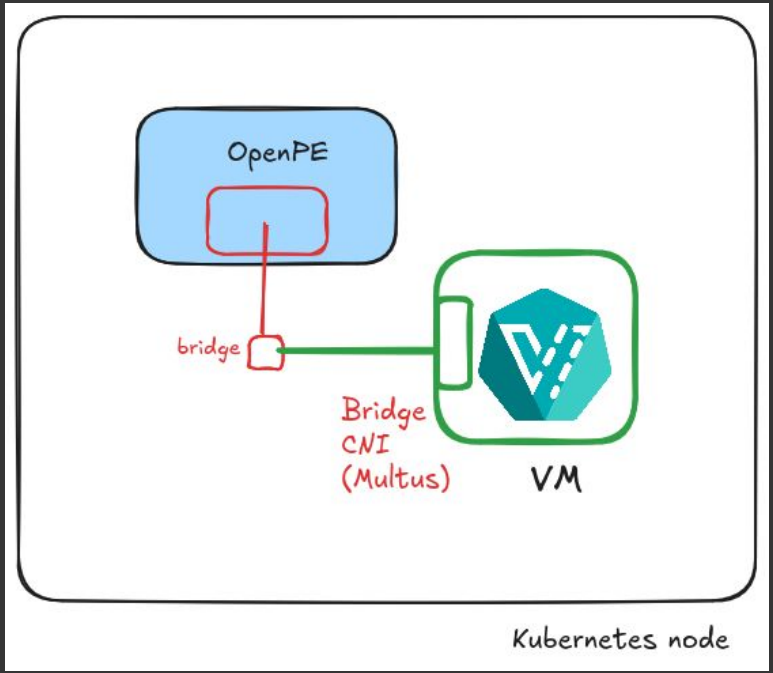
- Kubevirt (kubevirt.io)
- OpenPERouter (openperouter.github.io)
- Any bridge based CNI
  - OVS-CNI
  - bridge-cni

OpenPERouter



- Like a router, but running on our nodes
- L2 overlay exposed to the node as a veth leg

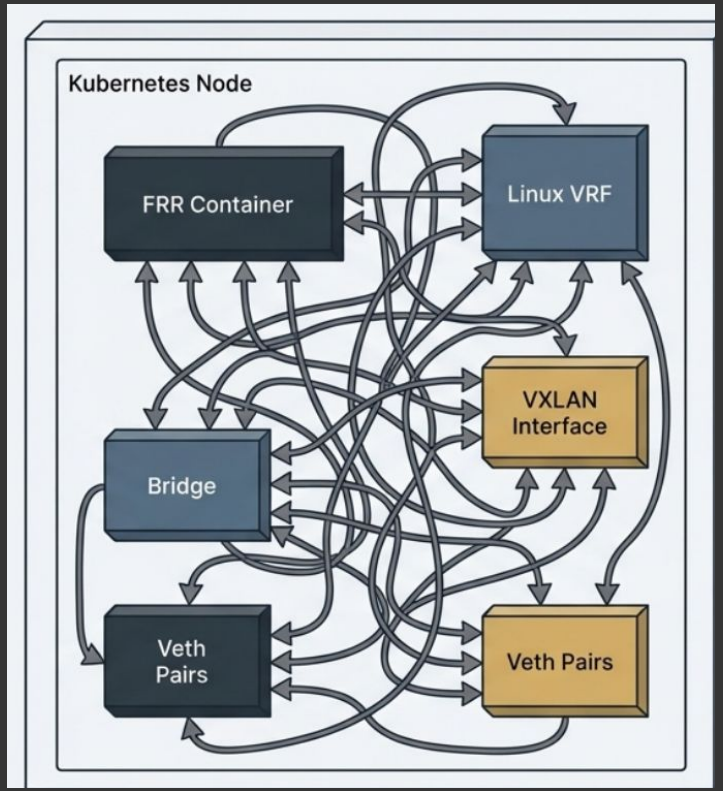
Attaching to the stretched network



- A VM can be plugged to the L2 domain using a bridge on the host
- Distributed Anycast GW
- Live migration support

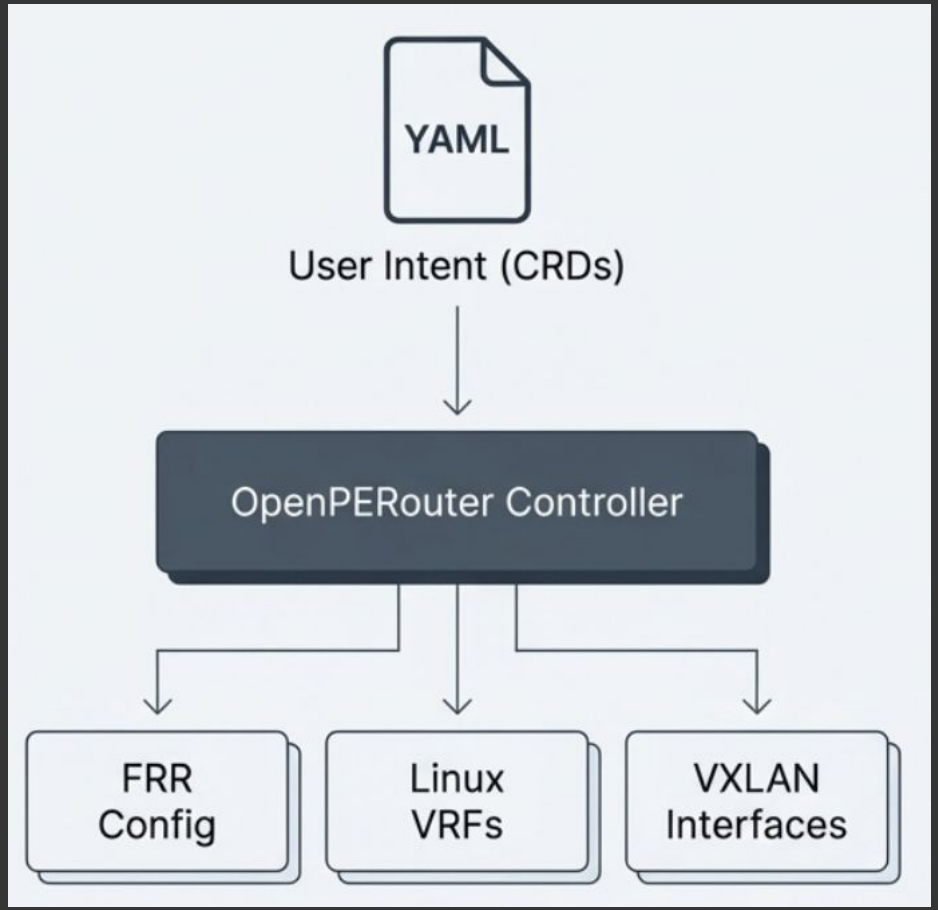
Complexity under the hood

- Integration of low level components
- Manually managing this across a fleet of nodes => complex & error prone
- Need a Kubernetes native approach



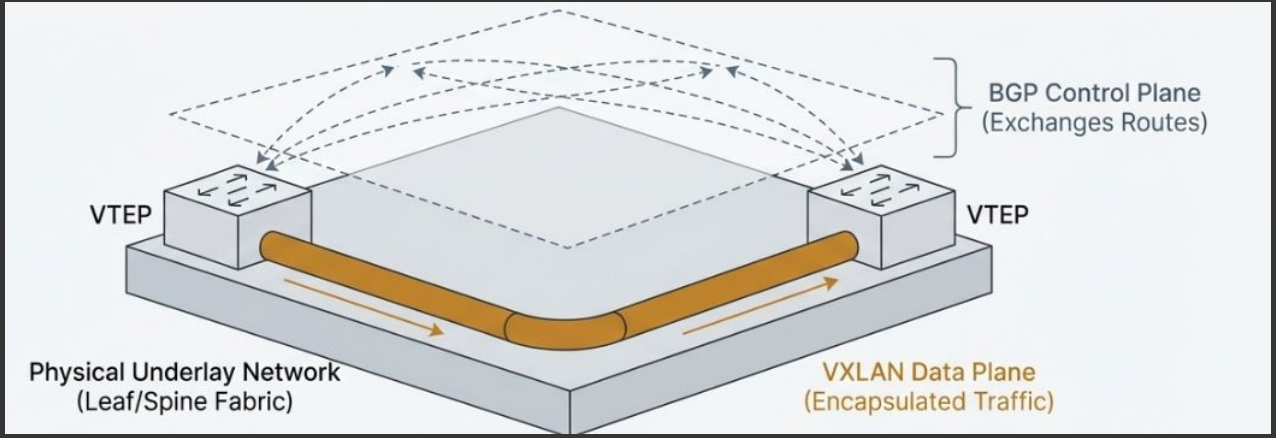
Going Kubernetes native

- Declarative API
- Automated configuration
- Not tied to a particular CNI plugin
  - You just need *a* bridge on the host



Configuring EVPN with OpenPERouter

- VXLAN => configure the tunnel endpoints in the nodes



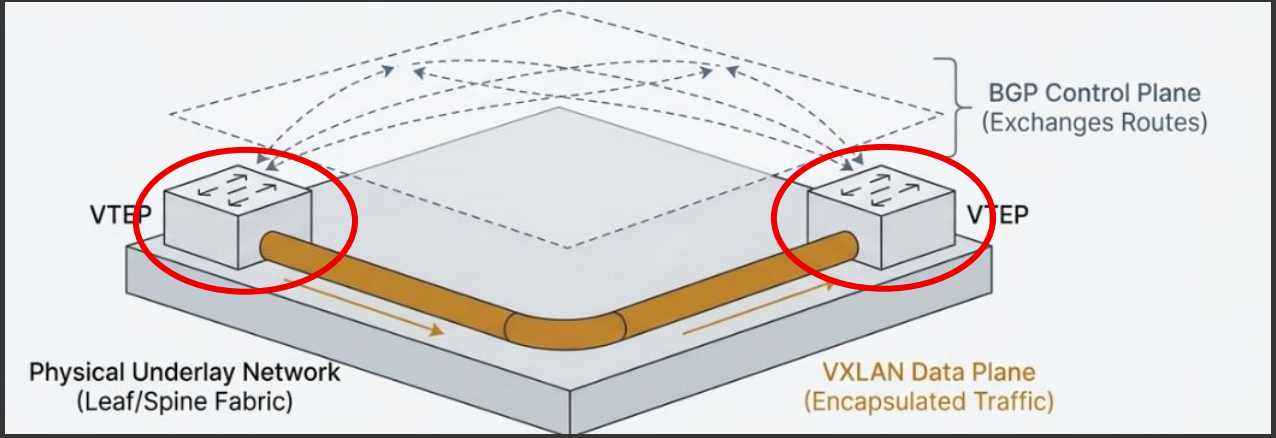
Underlay

Configure the VTEPs on the router nodes

```
apiVersion: openpe.openperouter.github.io/v1alpha1
kind: Underlay
metadata:
  name: underlay
  namespace: openperouter-system
spec:
  asn: 64514
  evpn:
    vtepcidr: 100.65.0.0/24
  nics:
    - toswitch
  neighbors:
    - asn: 64512
      address: 192.168.11.2
```

Configuring EVPN with OpenPERouter

- Data plane: VXLAN
- **`Underlay`** CRD to configure the VTEPs



Configuring EVPN with OpenPERouter

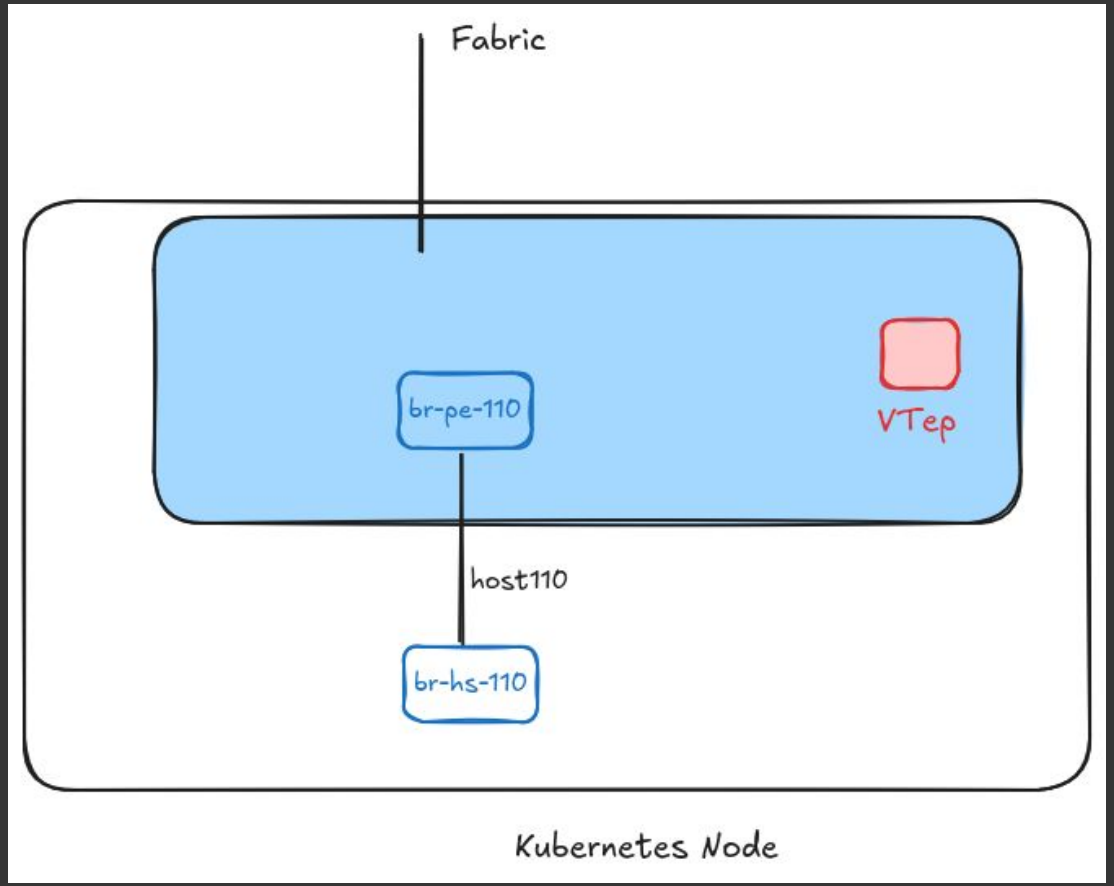
- Creating the MAC VRF by using the **`L2VNI`** CRD
- Stretches the L2 across the fabric

L2VNI

Stretch the L2 network

```
apiVersion: openpe.openperouter.github.io/v1alpha1
kind: L2VNI
metadata:
  name: layer2
  namespace: openperouter-system
spec:
  hostmaster:
    type: linux-bridge
    linux-bridge:
      autocreate: true
  l2gatewayip: 192.170.1.1/24     # ONLY NEEDED FOR L3
  vni: 110
  vrf: red
```

Layer 2 VNI

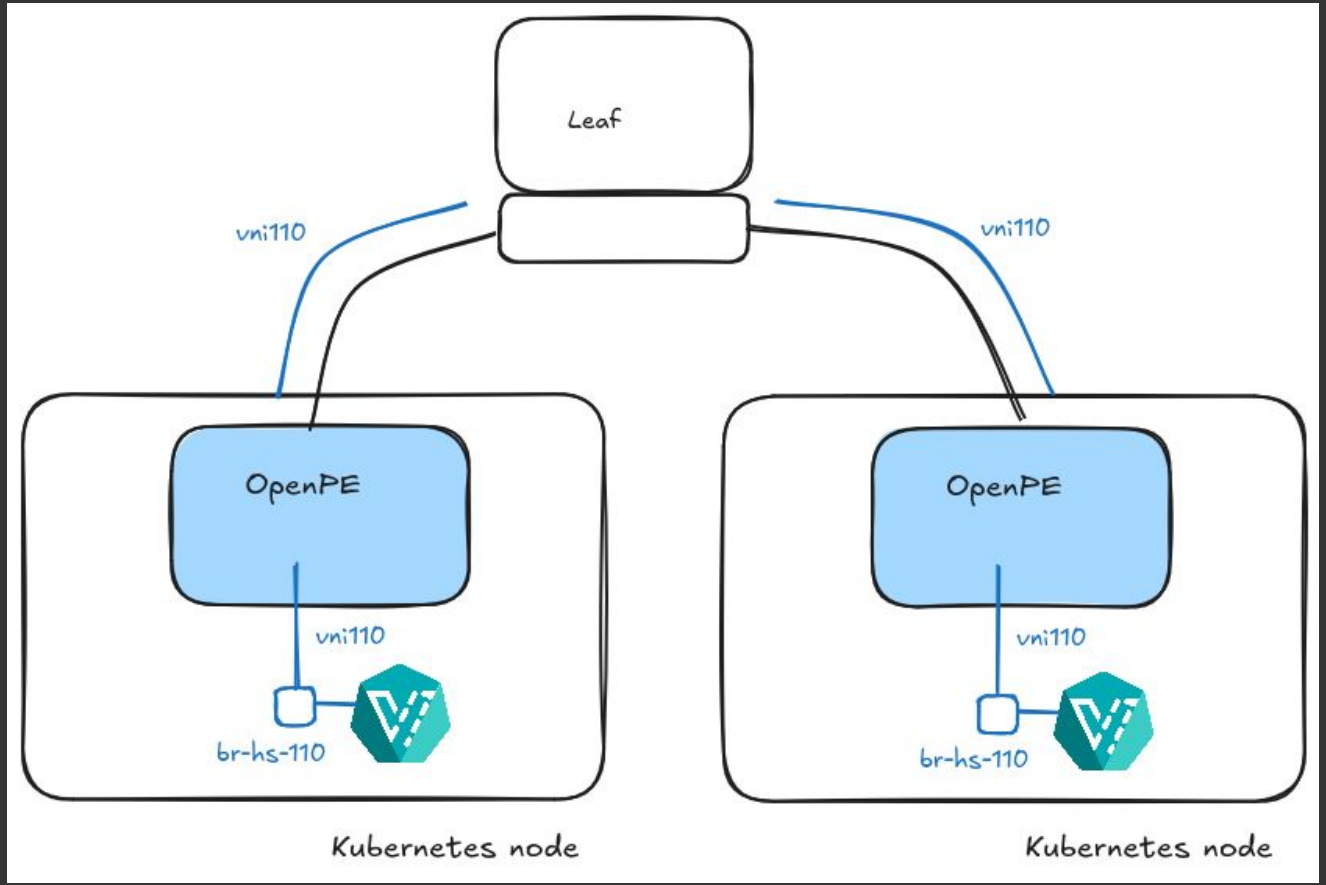


Network Config

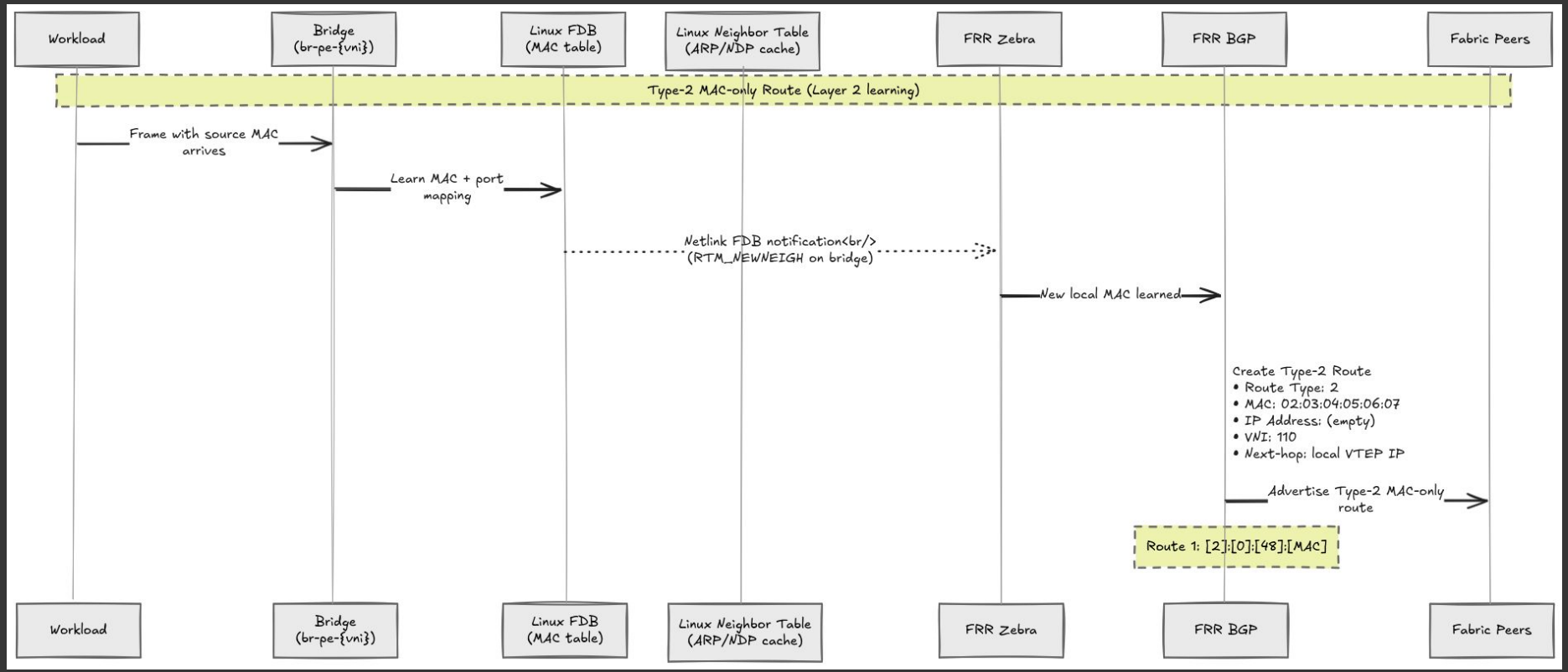
Specifies the attachment configuration

“Bridge” is an example; this works with more CNIs e.g. `ovs-cni`, macvlan, ipvlan, etc.

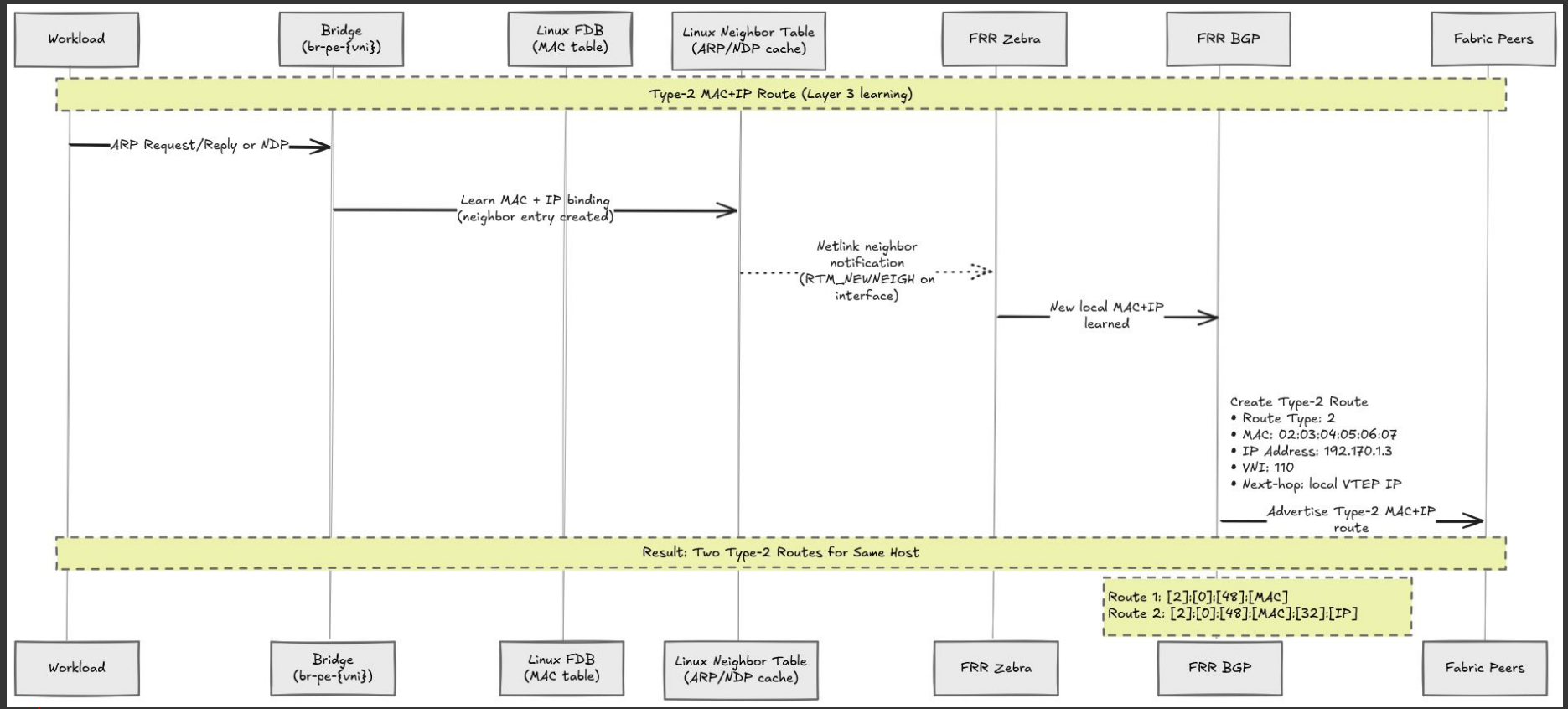
```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  name: evpn
  namespace: default
spec:
  config: |
    {
      "cniVersion": "0.3.1",
      "name": "evpn",
      "type": "bridge",
      "bridge": "br-hs-110",
      "macspoofchk": false,
      "disableContainerInterface": true
    }
```


Leaf
vni110
vni110
OpenPE
OpenPE
vni110
vni110
br-hs-110
br-hs-110
Kubernetes node
Kubernetes node

How FRR advertises type 2 routes

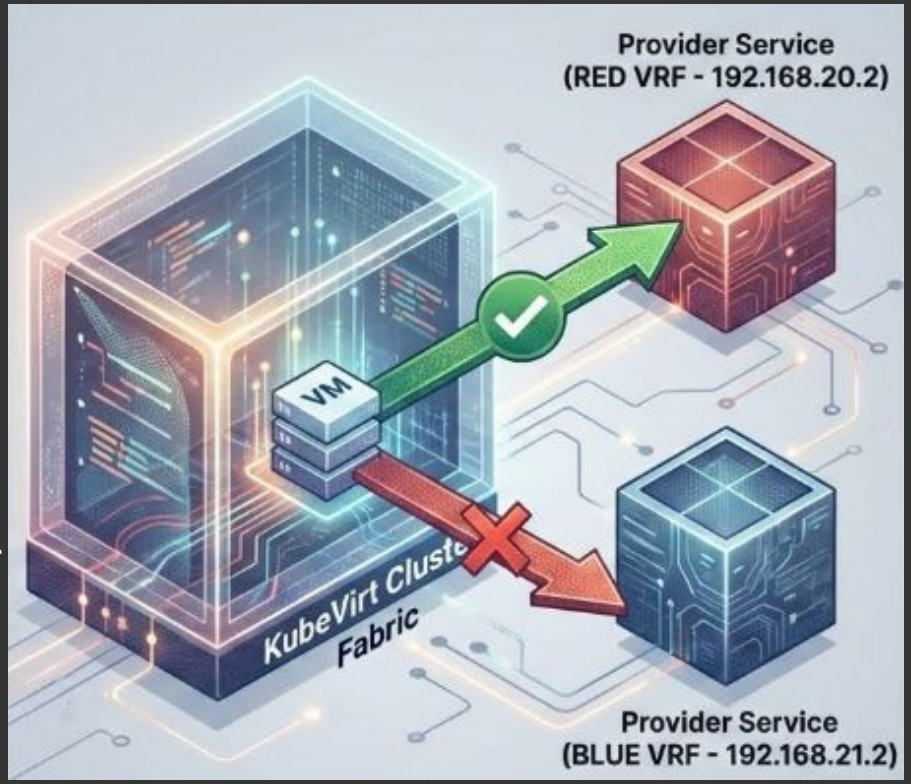


How FRR advertises type 2 routes



Connecting to provider networks

- L3 overlays to import/export routes
  - Access services in provider networks
  - Direct routed ingress into the VMs

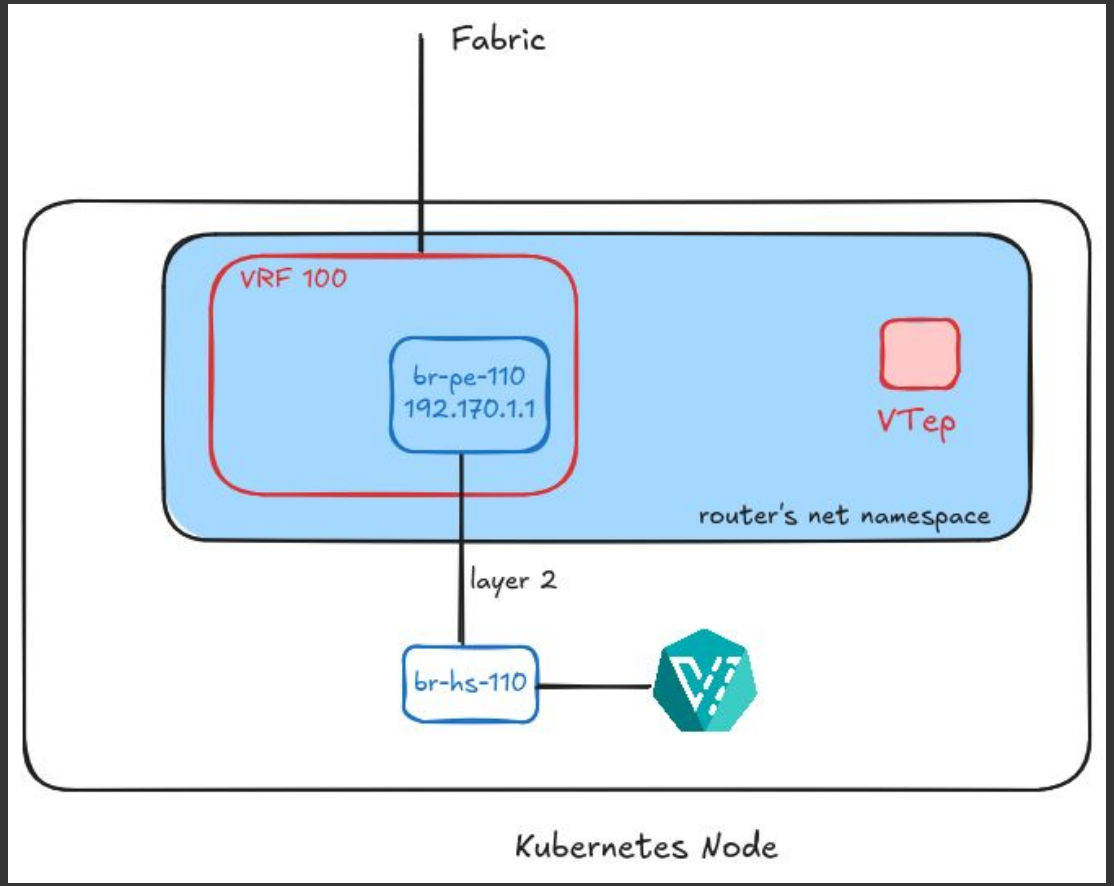


L3 VNI

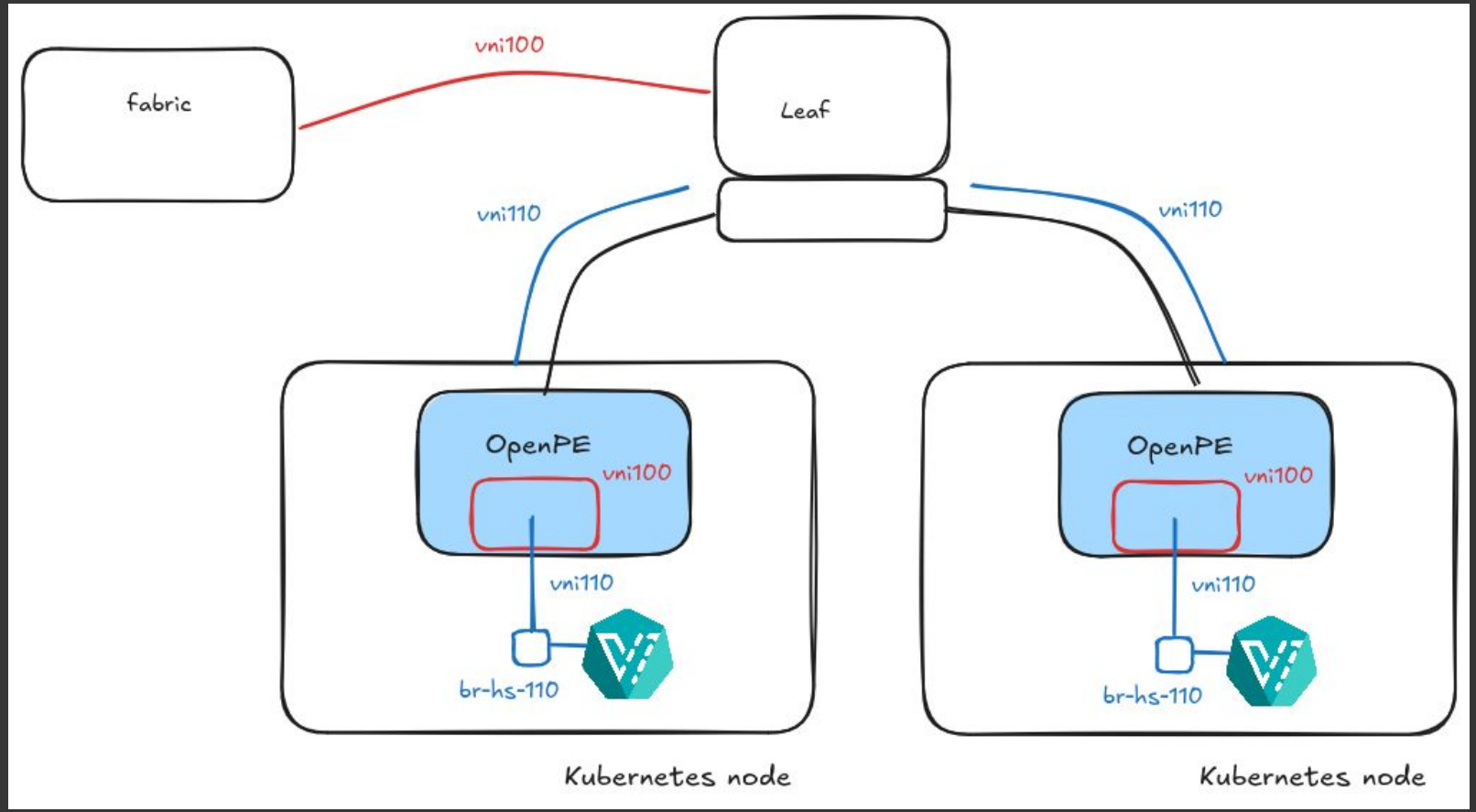
Expose the network

```
apiVersion: openpe.openperouter.github.io/v1alpha1
kind: L3VNI
metadata:
  name: red
  namespace: openperouter-system
spec:
  vni: 100
  vrf: red
```

Layer 3 VNI

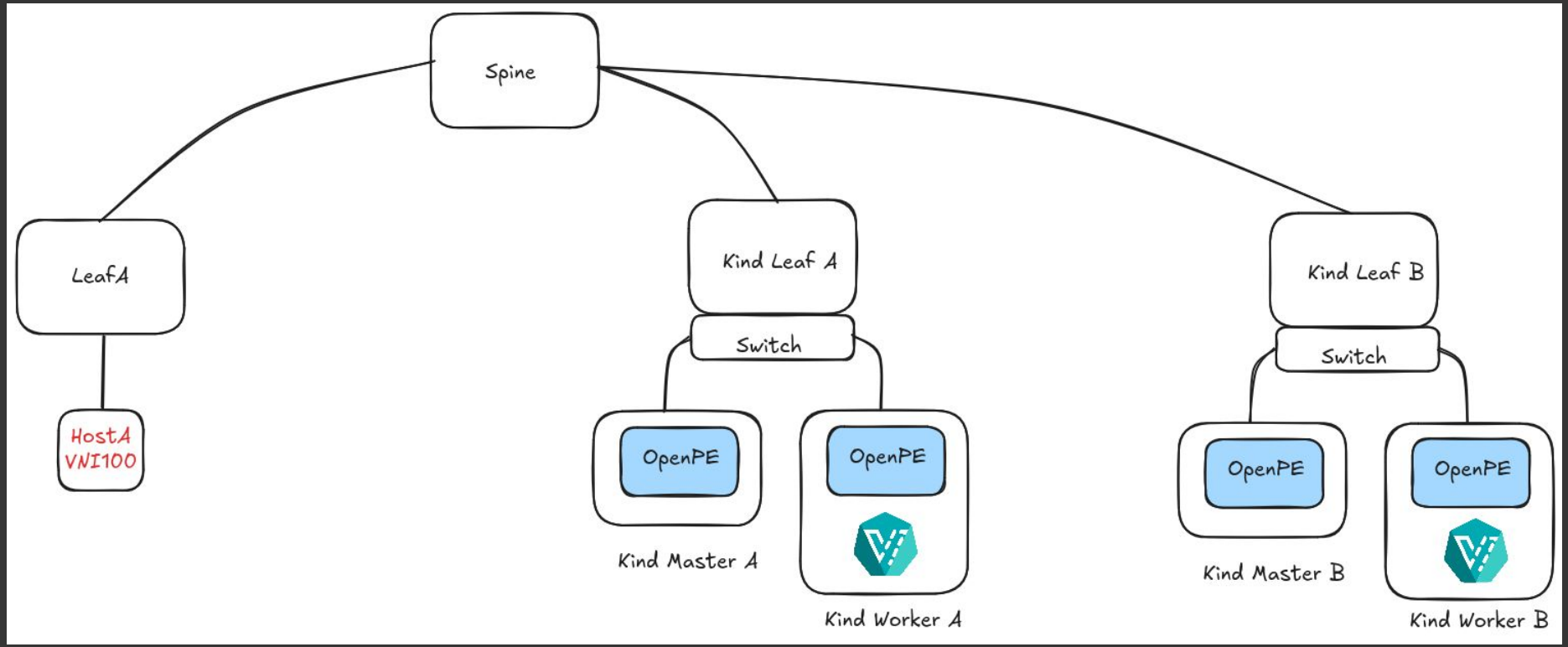

Fabric
VRF 100
br-pe-110
192.170.1.1
VTep
router's net namespace
layer 2
br-hs-110
Kubernetes Node

Layer 3 VNI

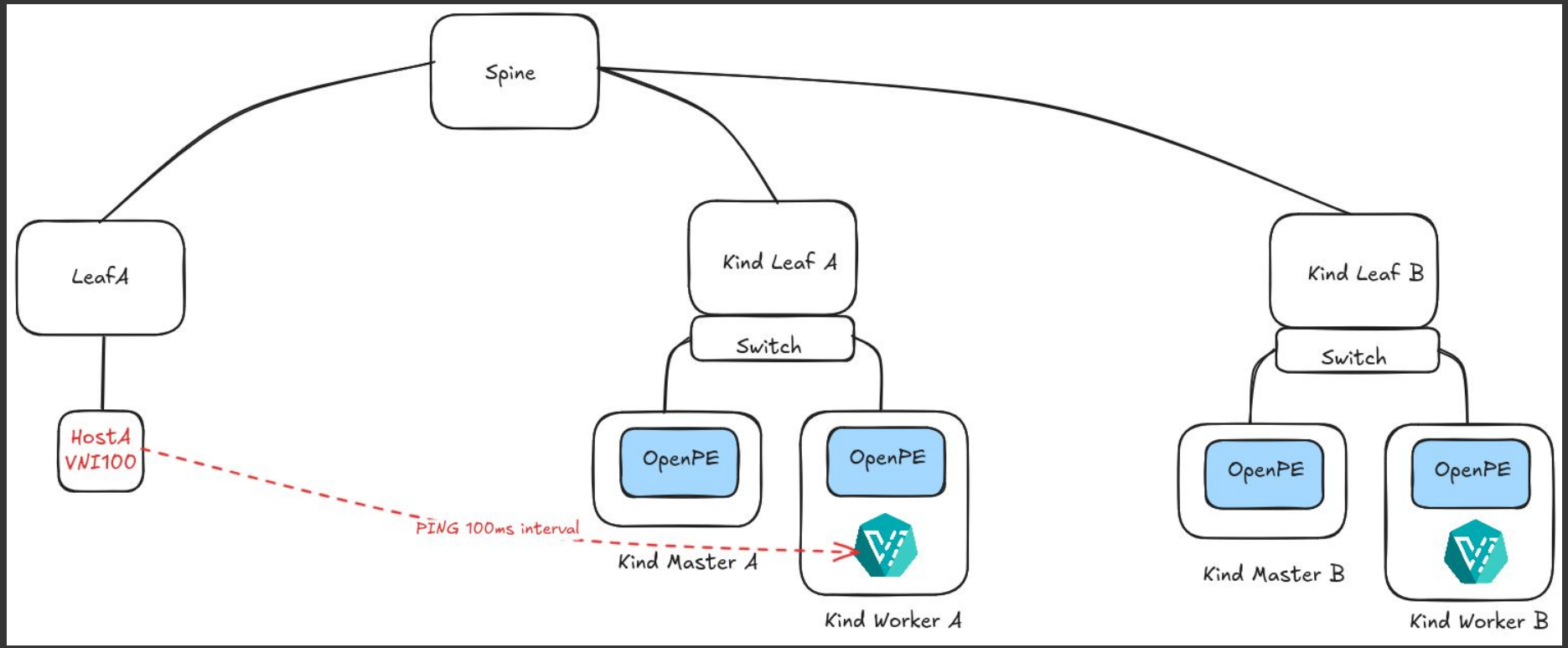

fabric
vni100
Leaf
vni110
vni110
OpenPE
vni100
vni110
br-hs-110
OpenPE
vni100
vni110
br-hs-110
Kubernetes node
Kubernetes node

Demo

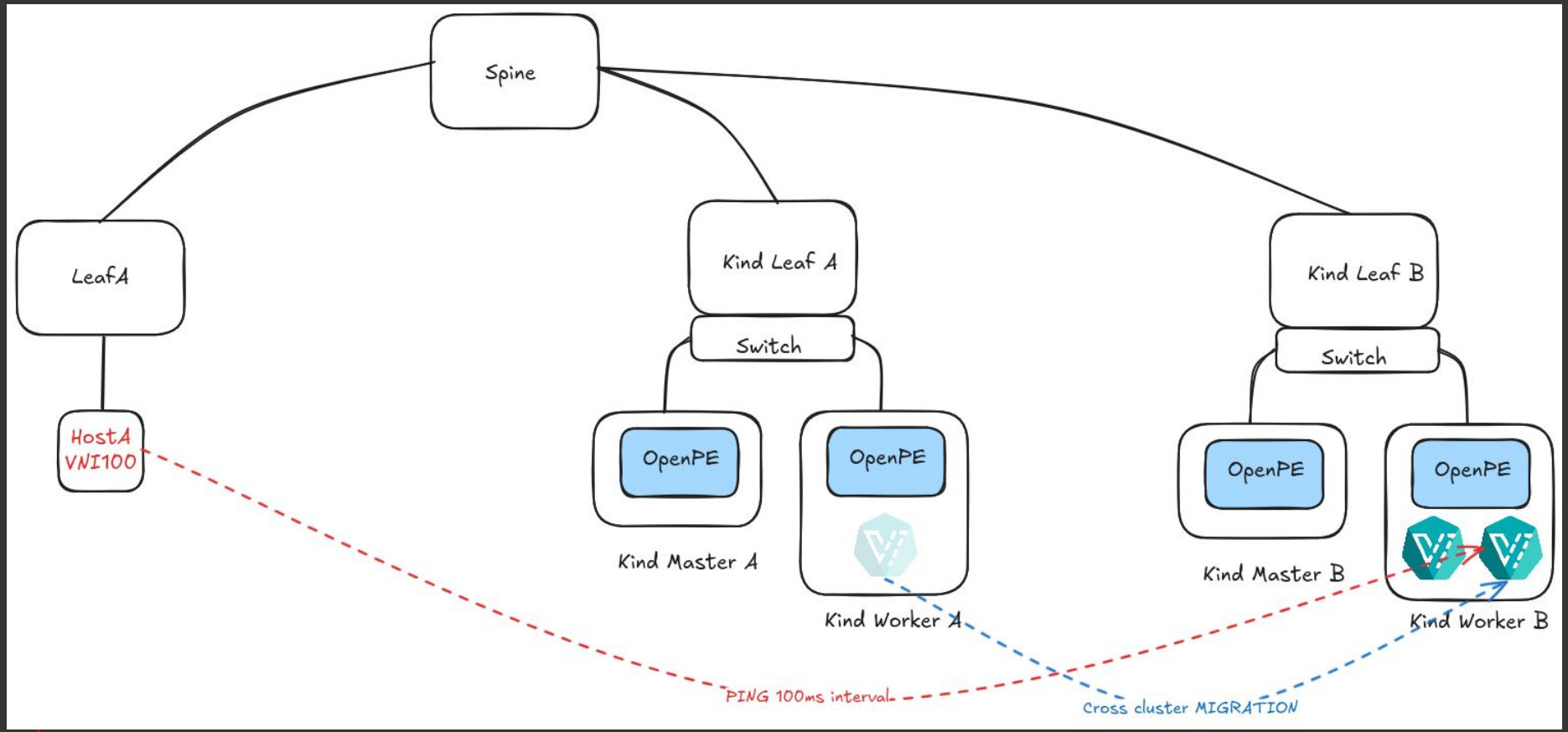
The demo scenario



The demo scenario



The demo scenario



Repo with demo script:
https://github.com/maiqueb/fosdem2026-openperouter
Asciinema link: https://tinyurl.com/superduperrouter

Conclusions

- L2VNI can be used to stretch a Layer 2 overlay across multiple Kubernetes clusters
  - or sites
- L3VNI can be used to create Layer 3 overlays
  - allows VMs running in Kubernetes to access services in the exposed provider networks
  - ... or the other way around
  - Provides direct ingress into the KubeVirt VMs
- L2VNIs can provide cross-cluster live migration

the end ...

- OpenPERouter docs
- KubeVirt docs
- KubeVirt related blog posts
  - Stretching an L2 between clusters
  - Dedicated migration networks using an L2VNI
- OpenPERouter examples
  - KubeVirt single cluster integration
  - KubeVirt multi-cluster integration
- FRR documentation
  - EVPN