



Free my Kubernetes network!

Breaking away from the Kubernetes networking model

FOSDEM 2025

Miguel Duarte

Principal Software Engineer, OpenShift Engineering @ Red Hat, Inc.

Douglas Smith

Principal Software Engineer, OpenShift Engineering @ Red Hat, Inc.



Doug Smith

- Technical lead for OpenShift Network Plumbing Team in OpenShift Engineering
- Multus CNI maintainer
- Network Plumbing Working Group member
- Blog: <https://dougbtv.com>



Miguel Duarte

- OpenShift Virt network engineer
- Network Plumbing Working Group member
- Blog: <https://maiqueb.github.io>



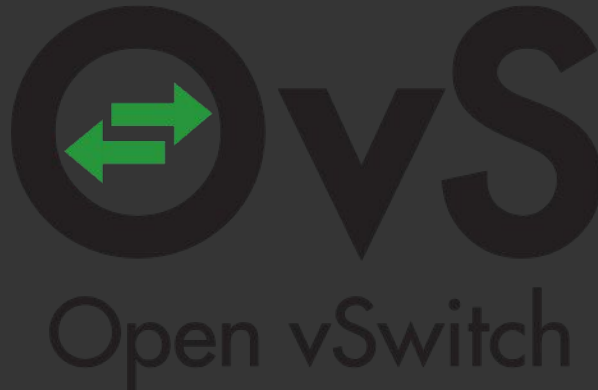
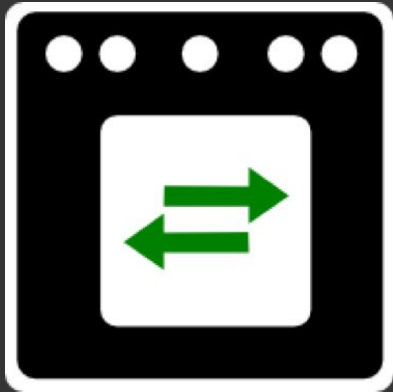
What's on the docket?

Agenda

- ▶ Motivation
- ▶ Problem
- ▶ Use cases
- ▶ Goals
- ▶ Solution
- ▶ Demos
- ▶ Conclusions



MULTUS



Motivation

- Traditional virt user
 - L2 isolation
- Kubernetes savvy user
 - Managed experience
- Stable IP addresses



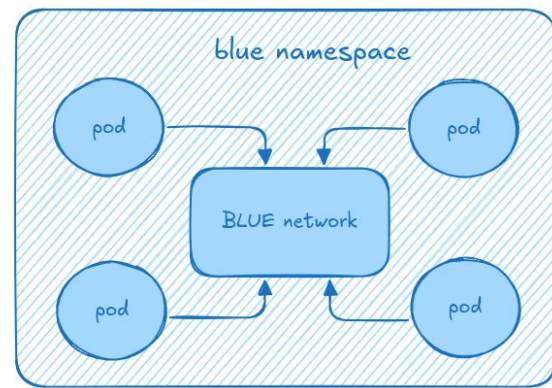
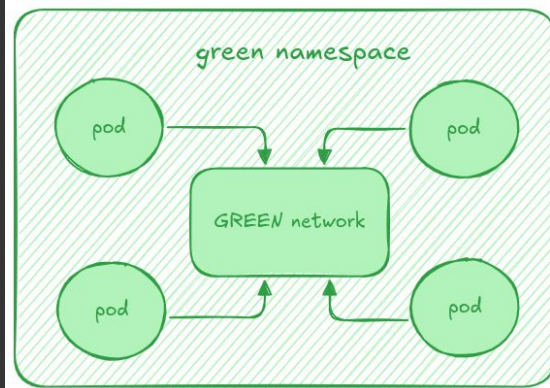
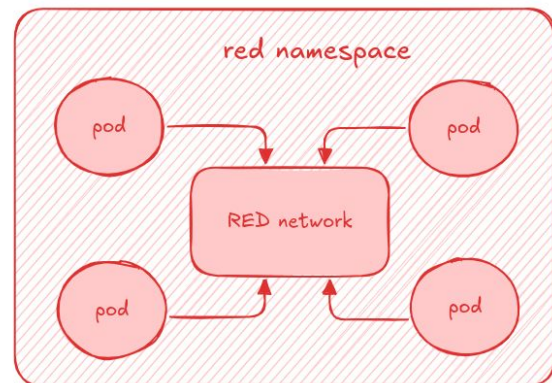
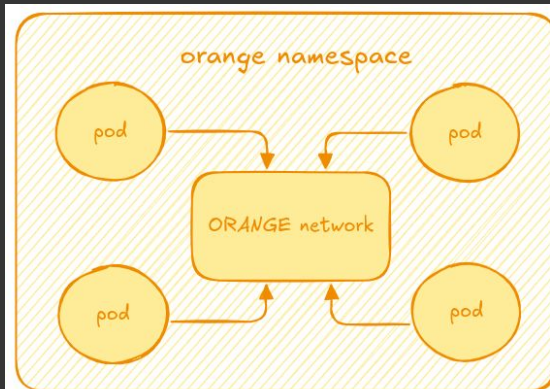
Problem: Kubernetes is opinionated!

- Single network !!!
- Everything's connected !!!
- Micro-segmentation via NetworkPolicy

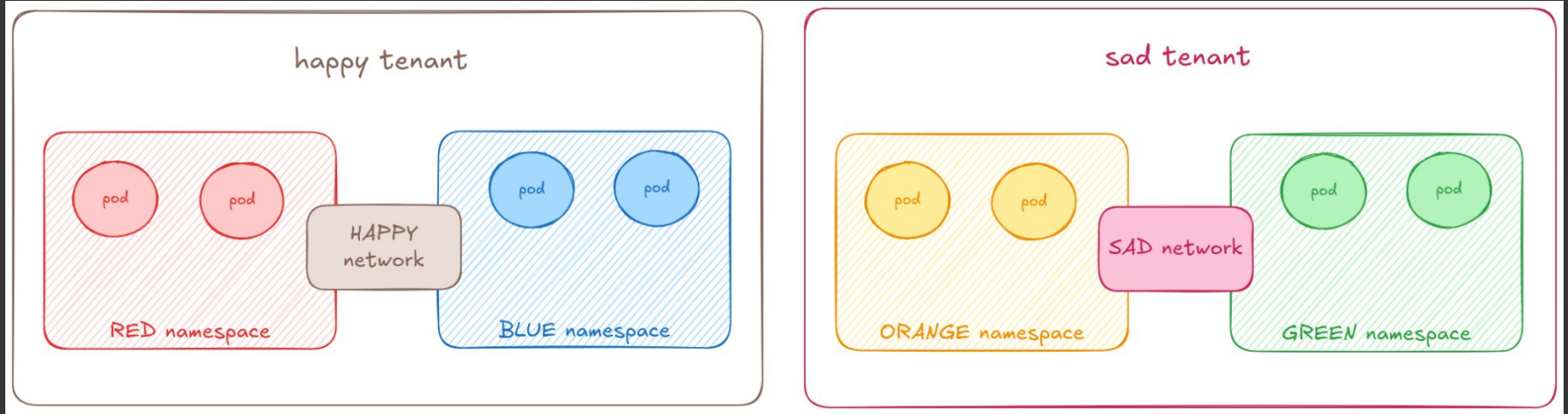


Use Cases

Native Namespace Isolation



Native cluster-wide network isolation



Goals

Goals



Workload/Tenant Isolation

Ability to group different types of applications in different isolated networks that cannot talk to each other



Overlapping podIPs

Ability to create Multiple Networks in your cluster with same pod Subnet range thereby possible to have copies of setups!



Kubernetes APIs supported!

Primary UDNs will have full support for **services**, **network policies**, **admin network policies**

Goals



Stable IPAM configuration
Workloads require their IPs,
GW, and DNS configuration
to be stable during their
lifecycle



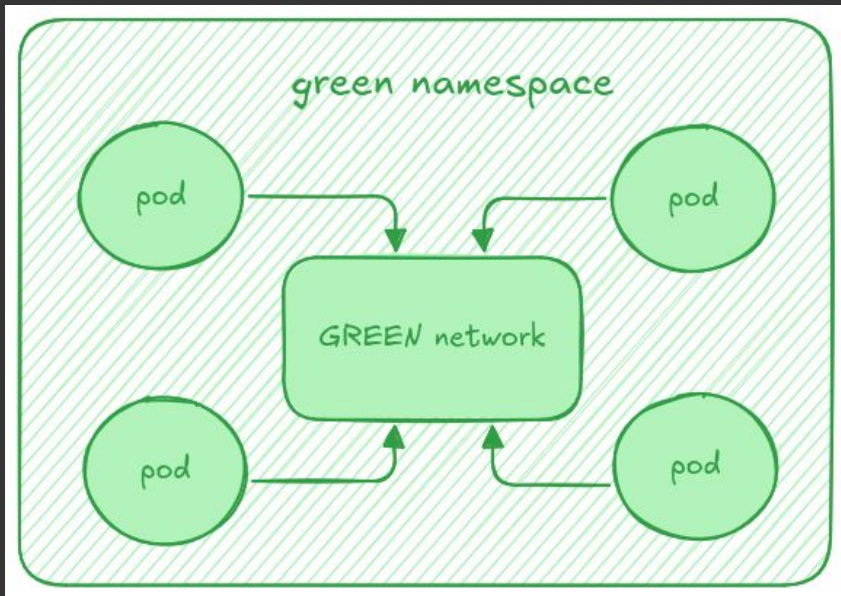
Cloud platform support
Packets must egress the
cluster w/ the IP addresses
of the *node* it runs on, to
appease cloud providers



Access to some kube services
Workloads attached to
primary UDNs will *still* have
access to kube-dns *and*
kube-api

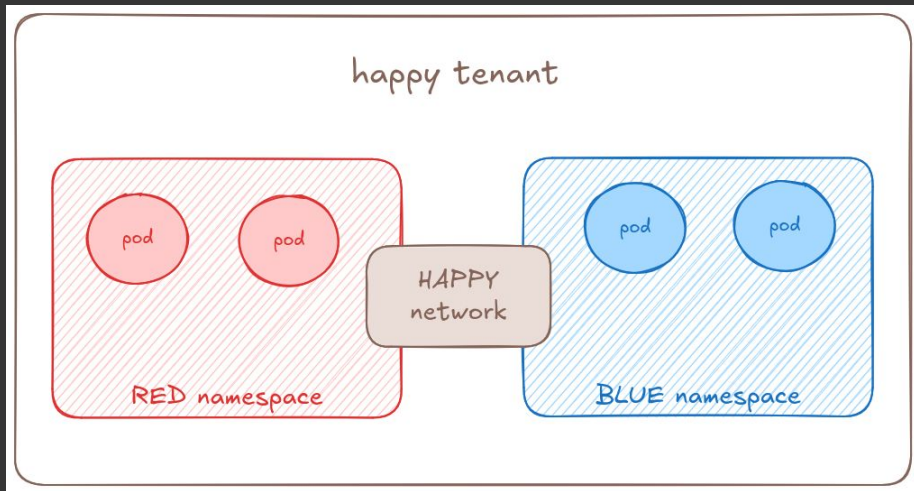
The API

UserDefinedNetwork



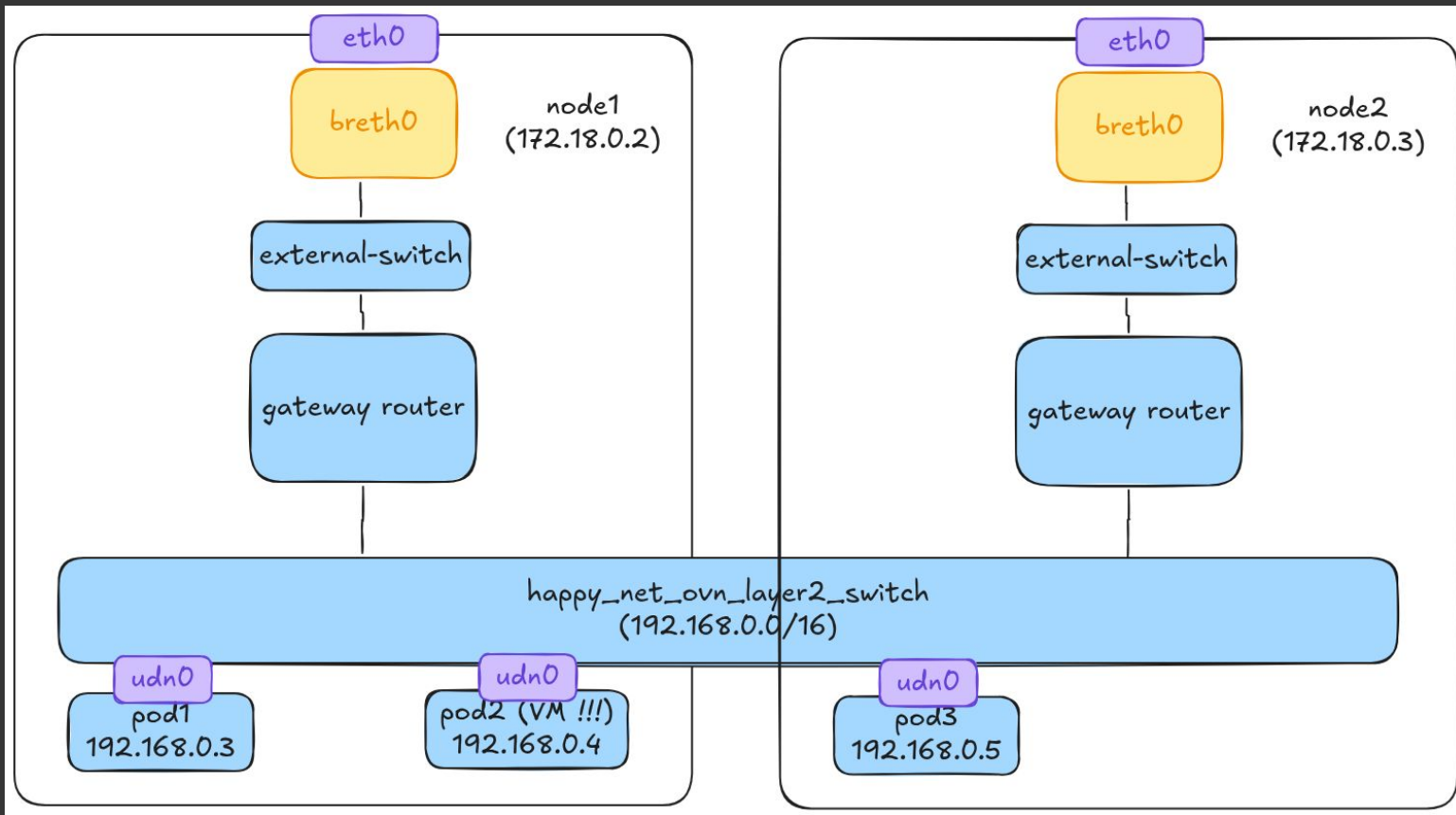
```
apiVersion: k8s.ovn.org/v1
kind: UserDefinedNetwork
metadata:
  name: namespace-scoped
  namespace: green
spec:
  topology: Layer2
  layer2:
    role: Primary
    subnets:
      - 203.203.0.0/16
  ipam:
    lifecycle: Persistent
```

ClusterUserDefinedNetwork



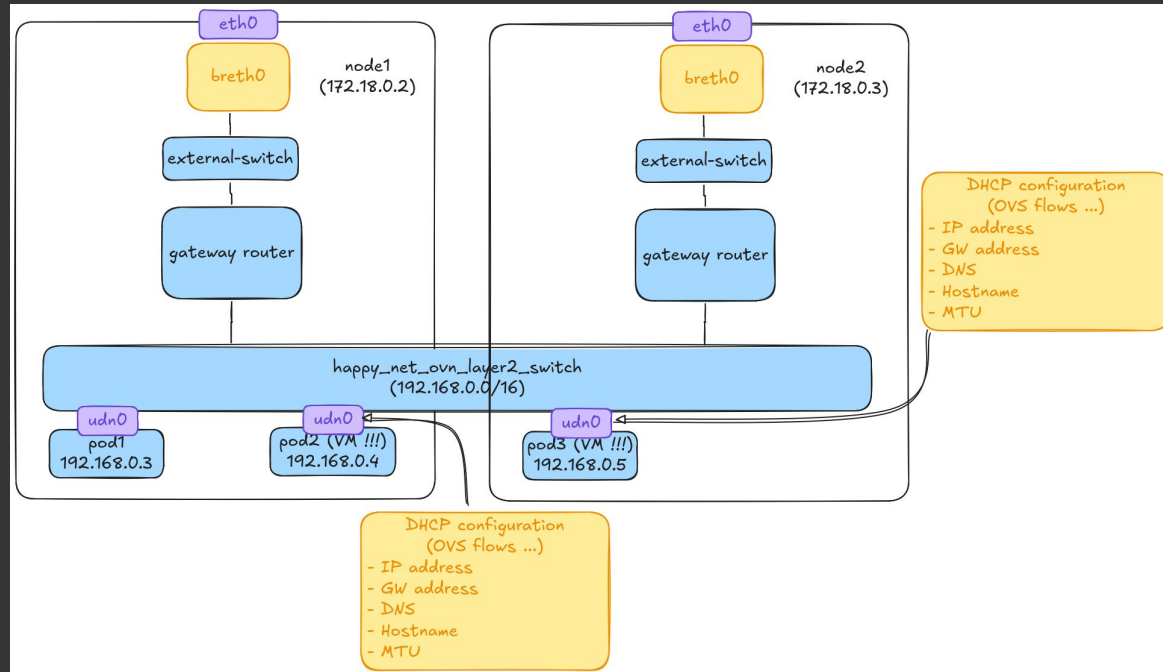
```
apiVersion: k8s.ovn.org/v1
kind: ClusterUserDefinedNetwork
metadata:
  name: happy-tenant
spec:
  namespaceSelector:
    matchExpressions:
      - key: kubernetes.io/metadata.name
        operator: In
        values:
          - red-namespace
          - blue-namespace
  network:
    topology: Layer2
    layer2:
      role: Primary
    ipam:
      lifecycle: Persistent
    subnets:
      - 192.168.0.0/16
```

Topology



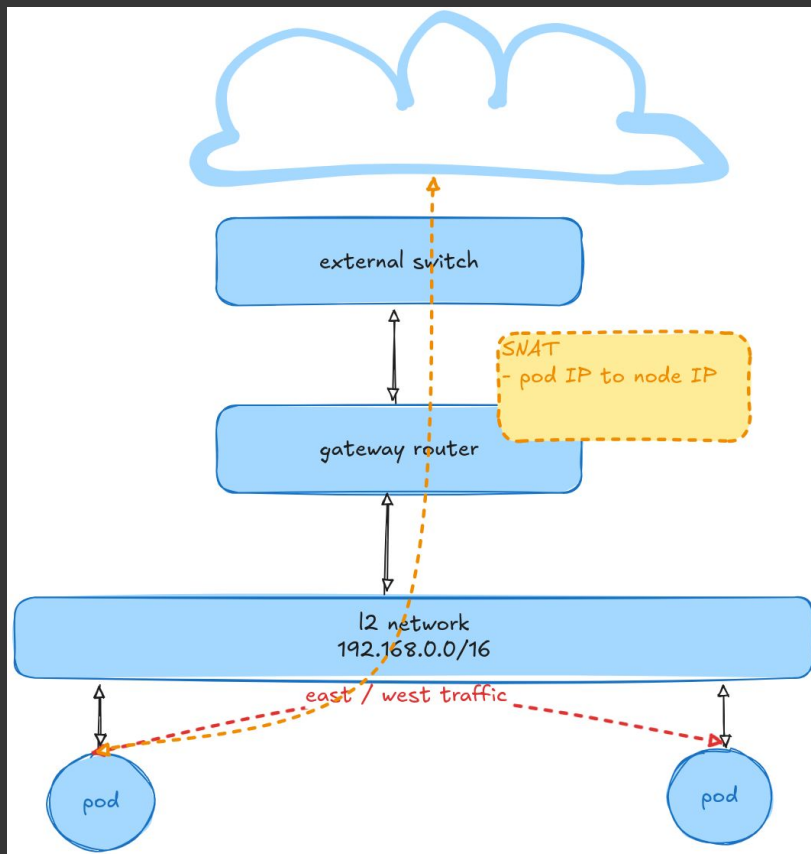
VM Guest IPAM config

Per port DHCP configuration in OVN

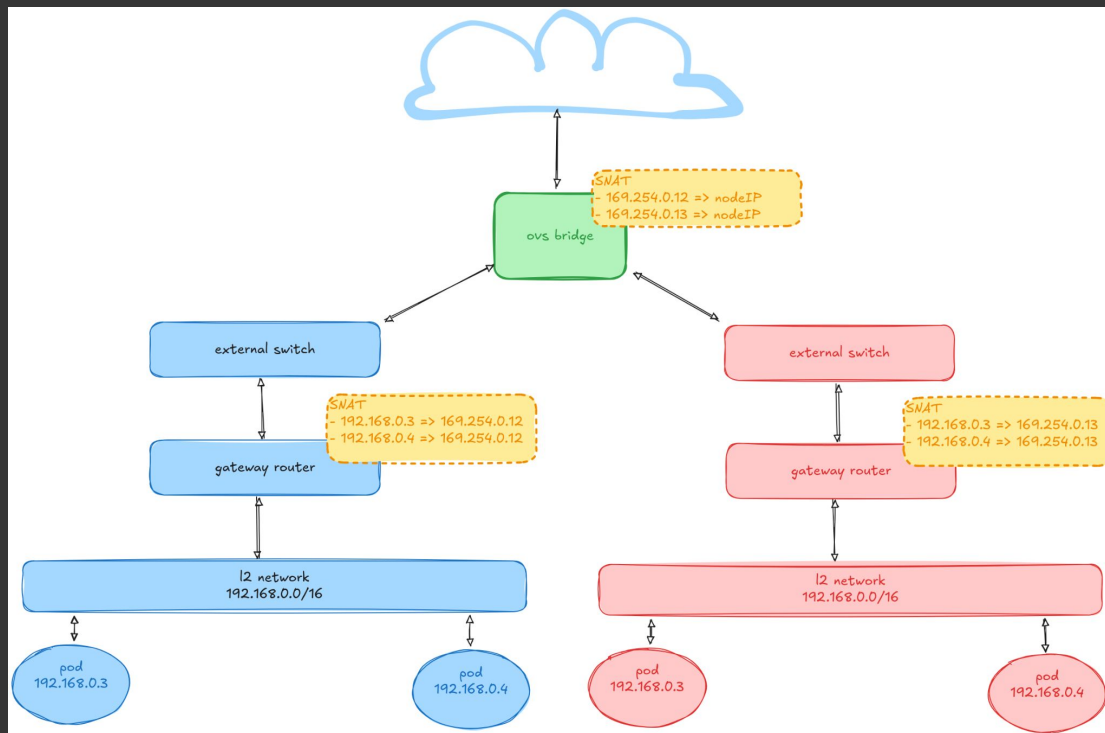


Enabling UDN for cloud platforms

NAT'ed egress

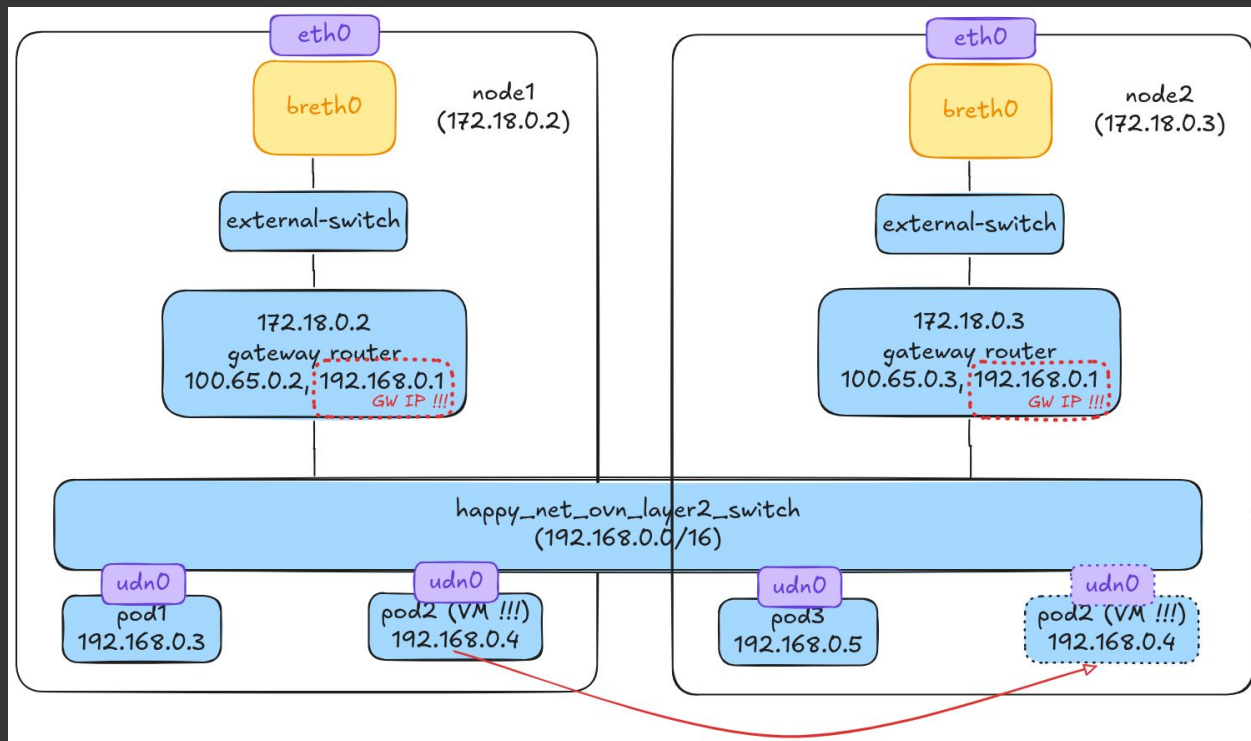


Overlapping subnets

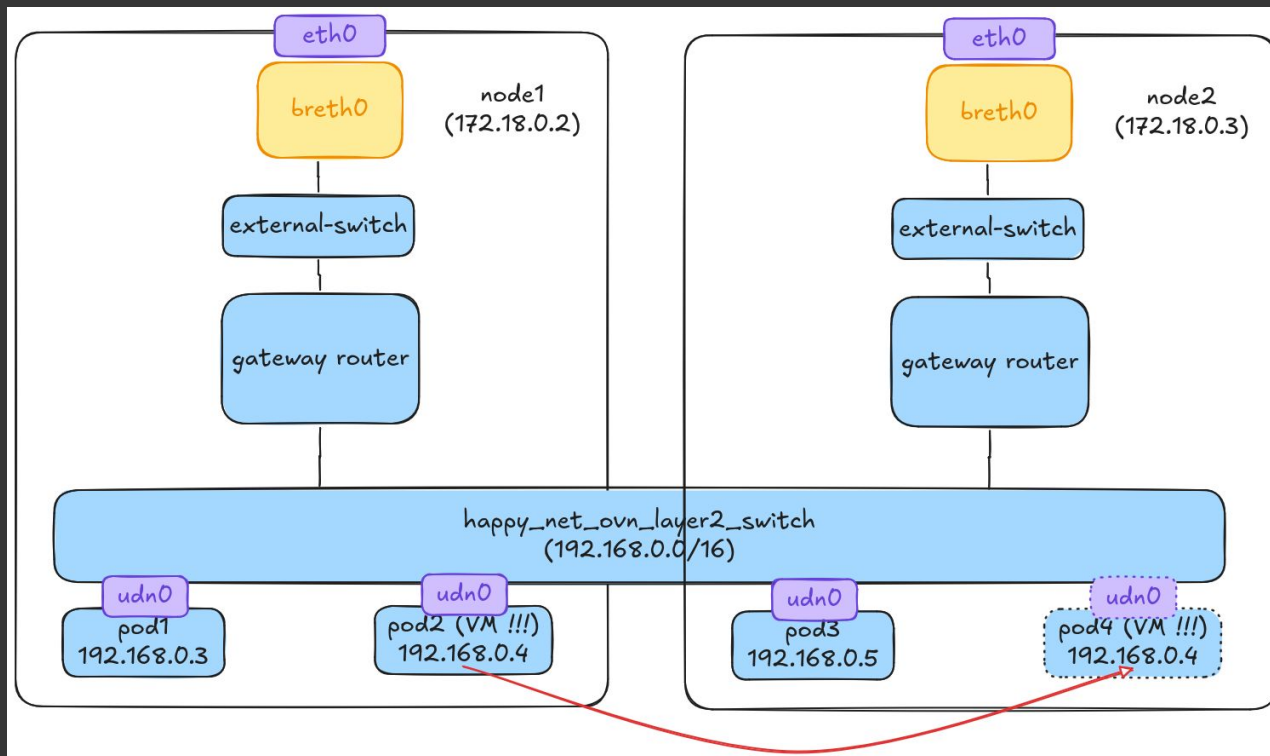


Stable IPAM

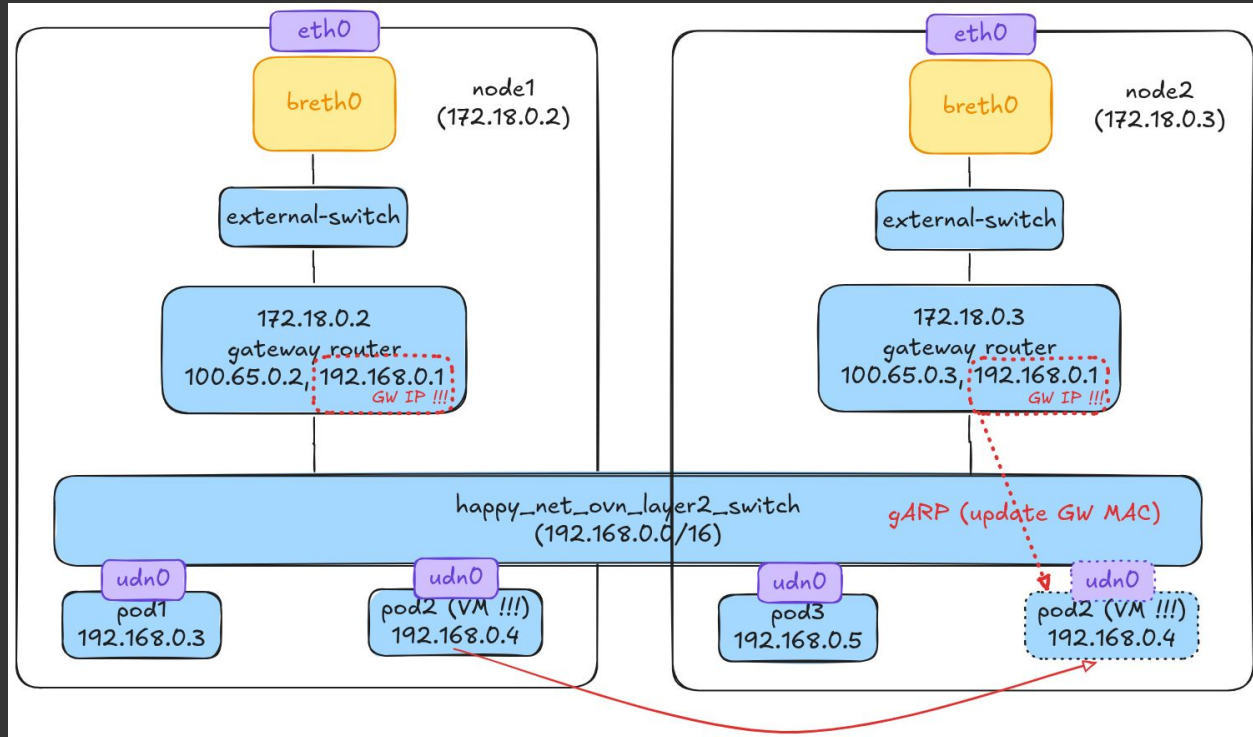
VM Live Migration



VM Live Migration



VM Live Migration



Demos

<https://github.com/maiqueb/fosdem2025-p-udn>

Namespace isolation



<https://asciinema.org/a/699323>

Cluster-wide network / cluster ingress



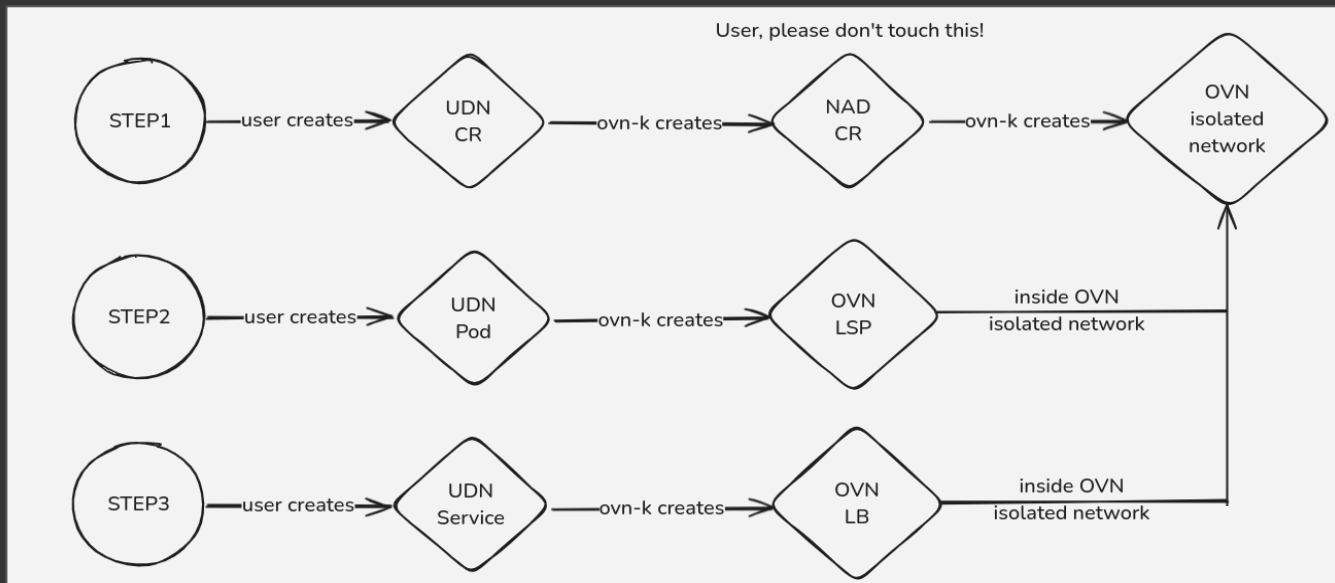
<https://asciinema.org/a/699643>

Conclusions

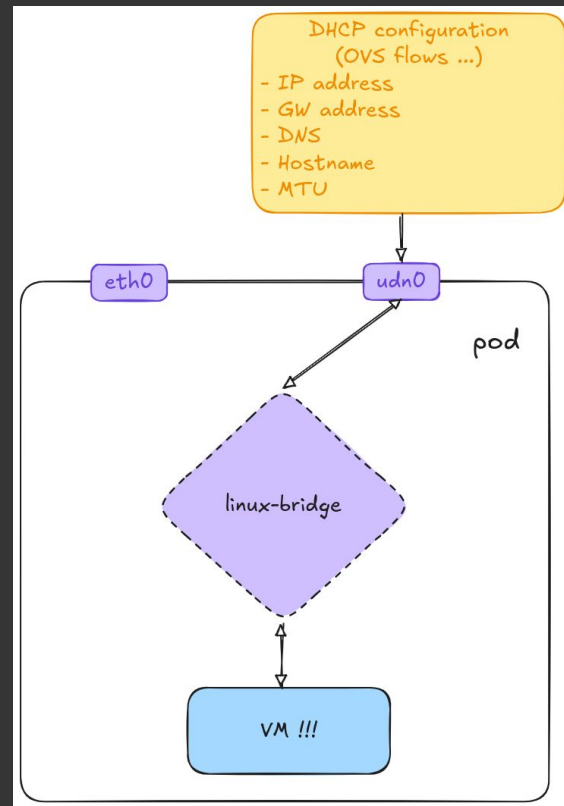
- Primary UDN provides
 - OpenStack neutron like(ish) type of networks in Kubernetes
 - Managed experience
- VM network requirements >> pod network requirements
- Integrated w/ Kubernetes API - net pol / services / ...
- Overlapping IPs in primary UDNs
- Cloud platforms are picky ! (as they should ...)

the end ...

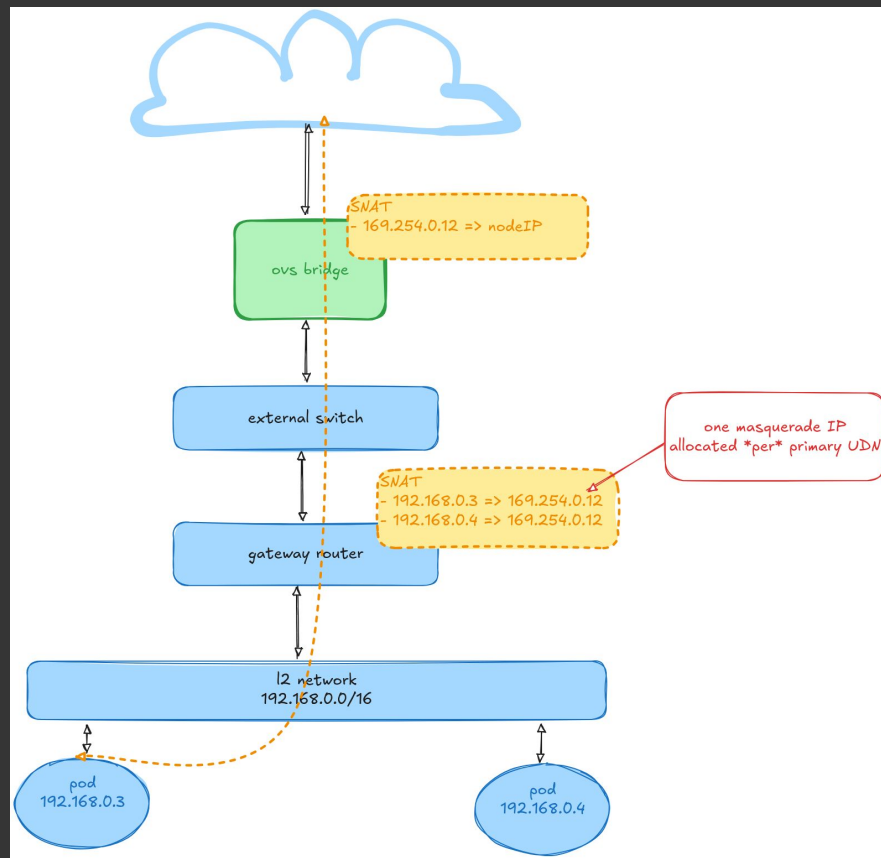
UDN configuration steps



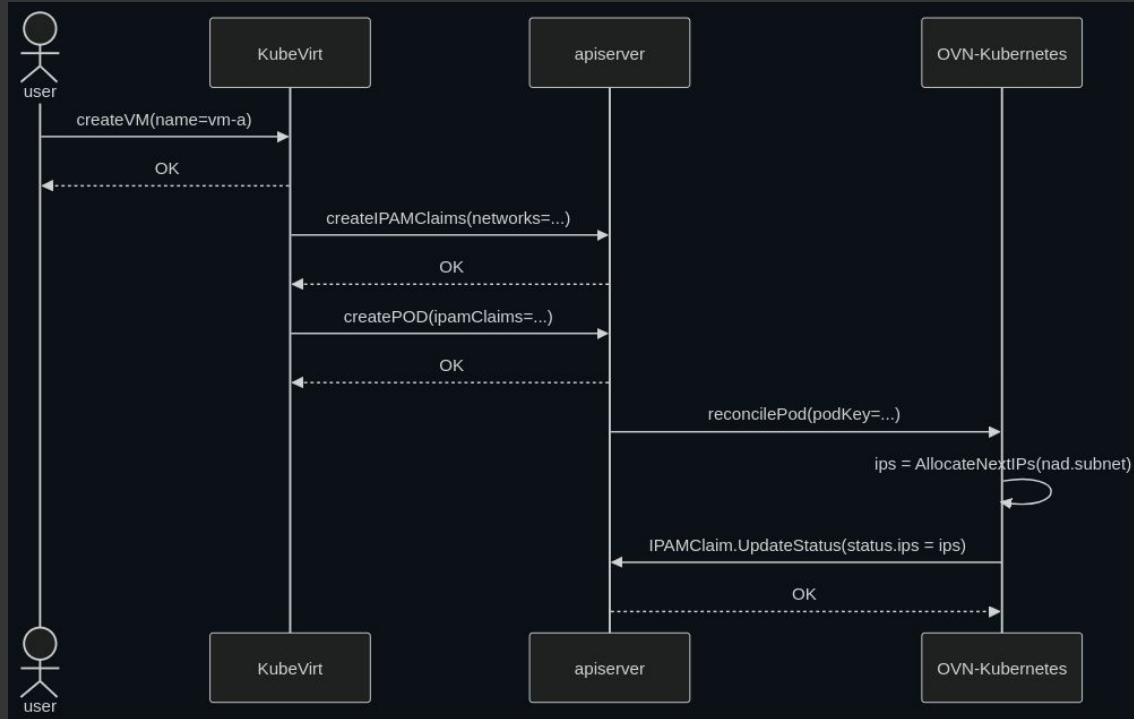
KubeVirt network binding



Overlapping subnets



Persisting IP addresses: IP allocation



Persisting IP addresses: IP recovery

