DEVCONF.cz

OVN-Kubernetes -

The new default CNI of OpenShift

Patryk Diak
Senior Software Engineer,
OpenShift Networking Team

Surya Seetharaman
Senior Software Engineer,
OpenShift Networking Team

Agenda

- Key things that will be covered in this session

➔ Kubernetes & OpenShift Networking - At a Glance

➔ What is OVN (**O**pen **V**irtual **N**etworking)-Kubernetes?

➔ Why did we move to using OVN-Kubernetes

- Differences between OVN-Kubernetes and its predecessor OpenShift-SDN (**S**oftware **D**efined **N**etworking)

➔ How are Kubernetes Networking constructs implemented in OVN-Kubernetes?

- OVN-K main components and network topology
- Pods and Services will be the focus
- Demo of OVN-K

Kubernetes Networking

Kubernetes Networking

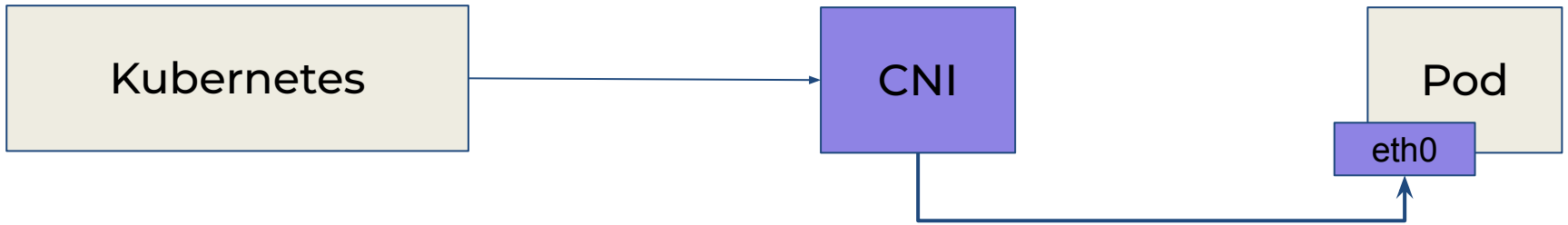
- Where does OVN-Kubernetes belong in the Kubernetes world?

➔ Kubernetes Fundamental Networking requirements

- Every pod must have its own unique IP
- Pod to pod communication must be possible within the cluster without NAT(**N**etwork **A**ddress **T**ranslation)ing
- Node to pod communication must be seamless

➔ Kubernetes Networking Model

- K8s is a bunch of APIs, it is plugin agnostic as far as networking is concerned
- Networking solutions are implemented on top of core K8s using **Container Network Interface (CNI)** Specification
- OVN-Kubernetes is one such CNI (others include flannel, antrea, cilium, calico etc)



OpenShift Networking

OVN-Kubernetes became default CNI of OCP from 4.12 release

OpenShift Networking

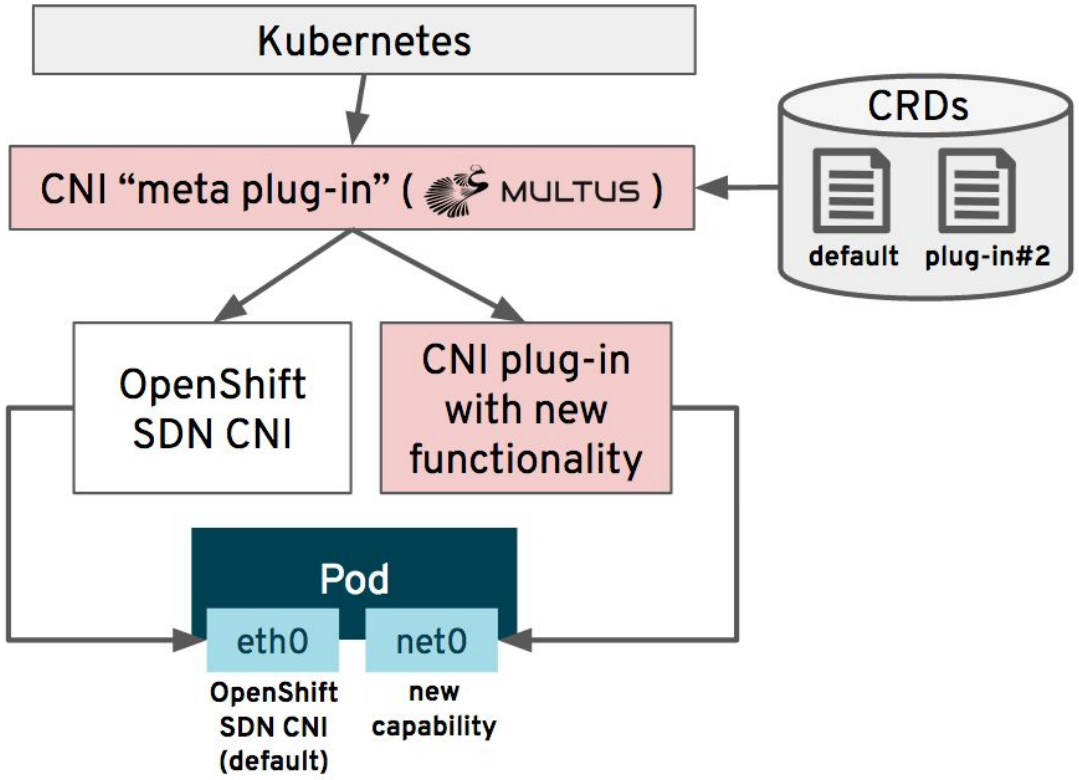
- Where does OVN-Kubernetes belong in the OpenShift world?

➔ OpenShift implements it's networking fabric using the **C**luster **N**etwork **O**perator (CNO)

➔ CNO Supports 2 network types:

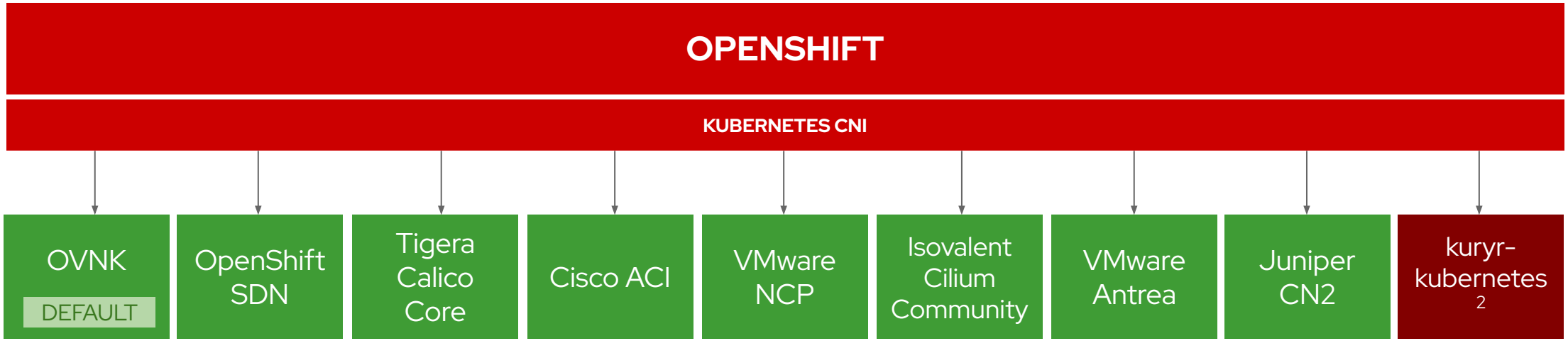
- OpenShift SDN - legacy default plugin
- OVN-Kubernetes - current default plugin since 4.12 OCP

➔ Multus is a meta CNI plugin that allows pods to have multiple interfaces by interfacing with more than one type of CNI plugin



OpenShift Networking

- Where does OVN-Kubernetes belong in the OpenShift world?



Interoperability Support Matrix

Version 2023-05-17

OVN-Kubernetes - What?

OpenShift-SDN: Legacy Network Provider

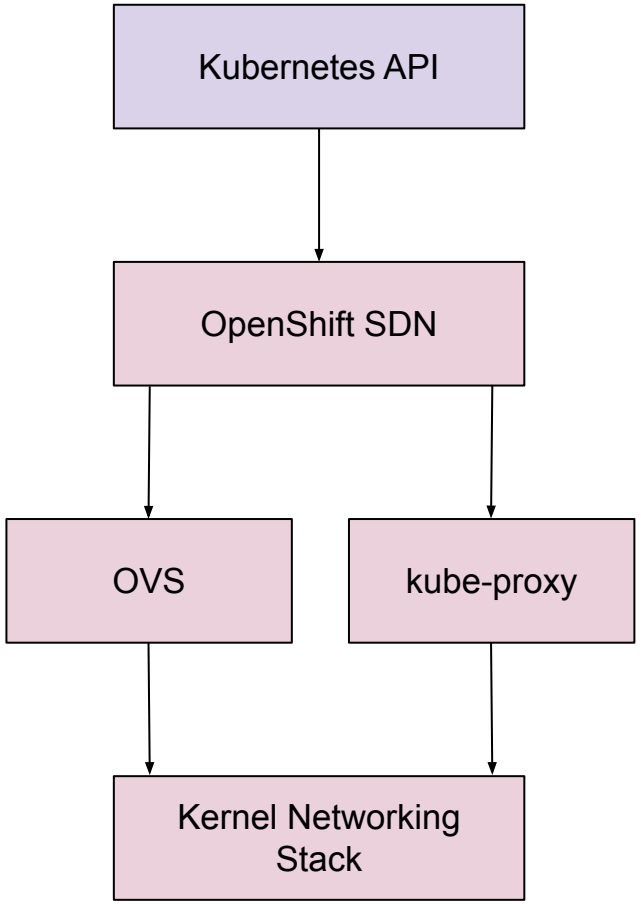
- What is OpenShift - **S**oftware **D**efined **N**etworking?

➔ OpenShift specific legacy CNI

➔ Uses iptables and OVS - **O**pen **V**irtual **S**witch to configure the Kubernetes networking

➔ Nodes in the cluster are connected using **V**irtual **Ex**tensible **L**ocal **A**rea **N**etwork (VXLAN) tunnels to provide overlay networking (east-west traffic)

➔ Runs kube-proxy on each node to provide support for Kubernetes services



OVN-Kubernetes

- Open Virtual Networking for Kubernetes a.k.a OVN-K8s

➔ Open Source project that provides a robust networking solution for Kubernetes clusters

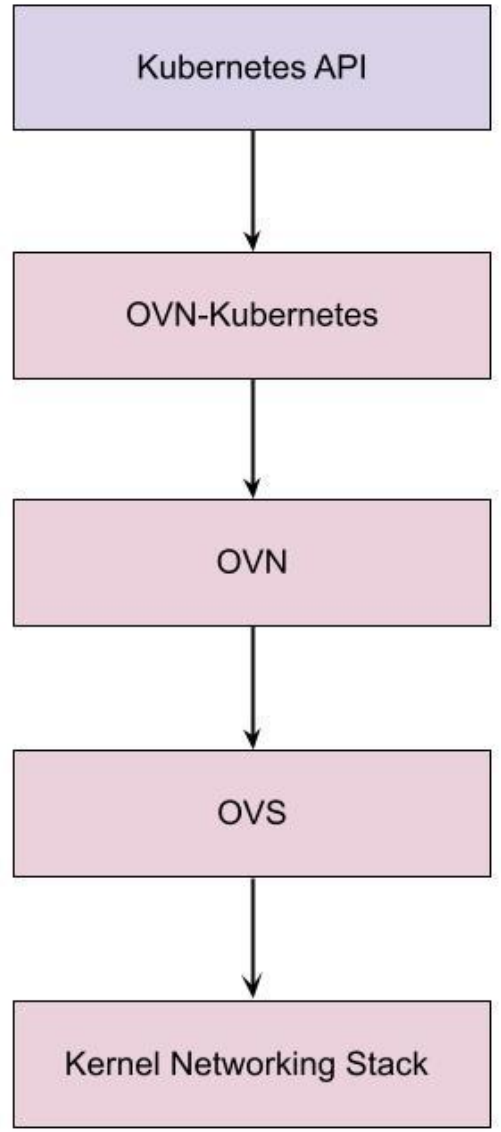
➔ Has OVN - **O**pen **V**irtual **N**etworking and OVS - **O**pen **V**irtual **S**witch at its core

- Uses them as the abstraction layers to manage cluster networking traffic flows on the node

➔ Uses GENEVE (**G**eneric **N**etwork **V**irtualization **E**ncapsulation) protocol between nodes to provide overlay networking (east-west traffic)

➔ Creates logical network topology per node and connects them together

- logical routers, switches, pod ports



Technology Highlights Comparison

Wait a Minute...! Does that look too familiar? If so, what changed and why did we move?

Legacy OpenShift SDN Plugin	OVN-Kubernetes
Creates veth pairs	Creates veth pairs
Creates an OVS switch	Creates an OVS switch
Central IPAM controller	Central IPAM controller
OVS flows for Network Policies	OVS flows for Network Policies
VXLAN tunnels between nodes	GENEVE tunnels between nodes
IPTable Rules for Services	OVN constructs for Services
IPTable Rules for NATing	OVN constructs for NATing

- veth - Virtual Ethernet
- IPAM - Internet Protocol Address Manager (here in context of allocating podIPs)

OVN-Kubernetes - Why?

Why did we move to OVN-K?

Easier*** development

- ➔ OVN as the new abstraction layer!
  - SDN: Directly touching the openflows which was complicated
  - OVN-K: Add human readable logical constructs closer to real world representations

Why did we move to OVN-K?

Flexible Architecture close to real world representations

➔ OVN as the new abstraction layer!

- SDN: Directly touching the openflows which was complicated
- OVN-K: Add human readable logical constructs closer to real world representations

➔ OVNK has a flexible/expandable architecture that enables faster feature development

- SDN: IPTables (kube-proxy for services) is restrictive
- OVNK: We have an entire (OVN team) engine

Why did we move to OVN-K?

Win-win at the community level

- ➔ OVN as the new abstraction layer!
  - SDN: Directly touching the openflows which was complicated
  - OVN-K: Add human readable logical constructs closer to real world representations

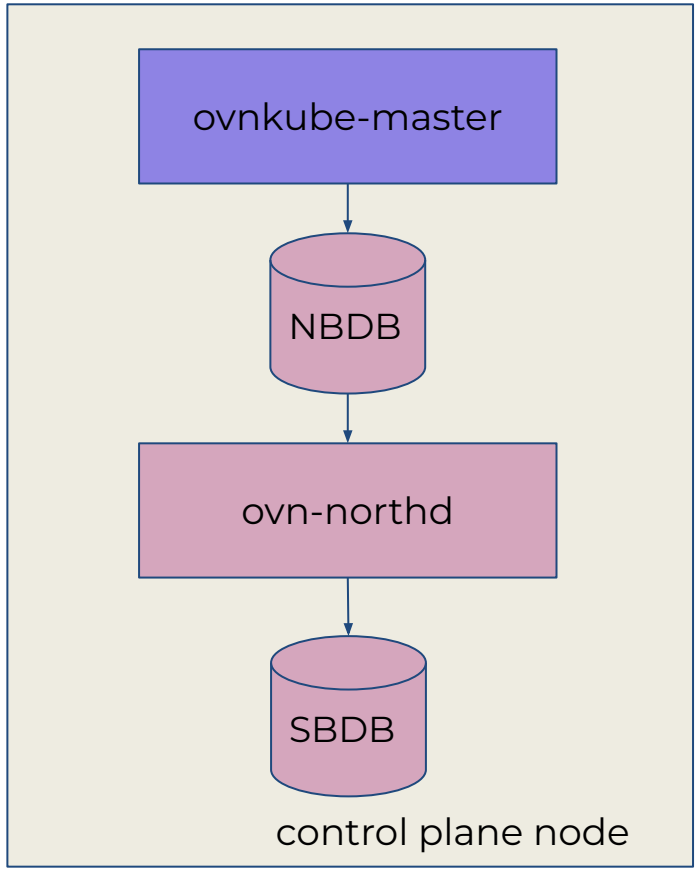
- ➔ OVNK has a flexible/expandable architecture that enables faster feature development
  - SDN: IPTables (kube-proxy for services) is restrictive
  - OVNK: We have an entire (OVN team) engine

- ➔ OVN-K is an open source project
  - SDN: Was used only in OpenShift
  - OVNK: Vibrant upstream community & easier alignment downstream

OVN-Kubernetes - How?

OVN-Kubernetes Architecture

- Components running in the Control Plane



➔ **OVN-Kubernetes Master**

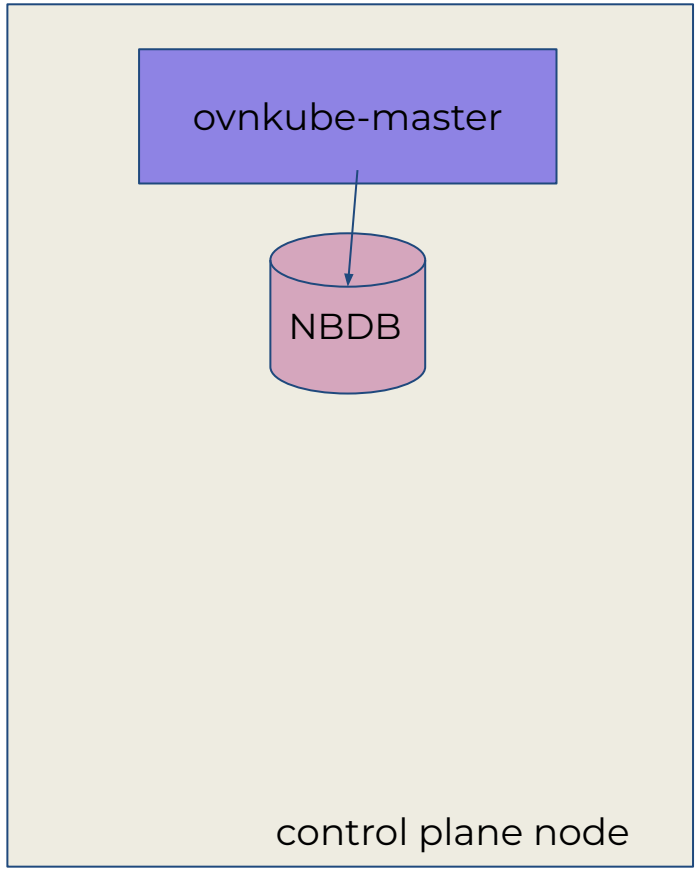
- OVN-Kubernetes component
- Watches K8s API for objects - namespaces, pods, services, endpoints, network policies
- Translates K8s objects into OVN logical entities
- Stores OVN entities in **N**orth**B**ound**D**ata**B**ase (NBDB)
- Manages pod subnet allocation to nodes (pod IPAM)

➔ **OVN-Native components**

- NBDB, Northd, SBDB, that translate these entities created by OVNK-Master into OVN logical (traffic) flows

OVN-Kubernetes Architecture

- Components running in the Control Plane



➔ **OVN-Kubernetes Master**

- OVN-Kubernetes component
- Watches K8s API for objects - namespaces, pods, services, endpoints, network policies
- Translates K8s objects into OVN logical entities
- Stores OVN entities in NorthBound Database (NBDB)
- Manages pod subnet allocation to nodes (pod IPAM)

➔ **OVN-NorthBoundDataBase (NBDB)**

- Native OVN component
- 3 replicas across control plane nodes

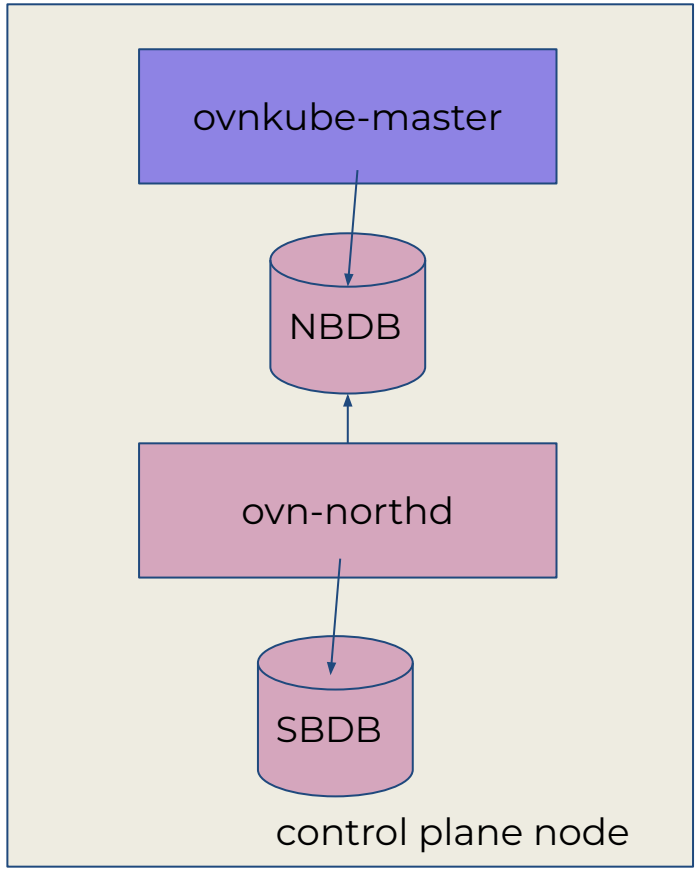
Mapping of Kubernetes objects to OVN entities

- NorthBoundDataBase contains the OVN logical Entities

Kubernetes Objects	**OVN Logical Entities**
Node	Switch, Router, Routes
Namespace	Address-Sets (a collection of podIPs in OVN world)
Pod	Logical Switch Ports, Address-Sets, NATs, Router Policies, Routes
Service, Endpoint	Load Balancer
Network Policy, EgressFirewall	Port Groups, **A**ccess **C**ontrol **L**ist (ACL), Address-Sets
EgressIP, EgressService	NAT, Router Policies
EgressQoS	Logical Switch **Q**uality **O**f **S**ervice (QoS) Rule

OVN-Kubernetes Architecture

- Components running in the Control Plane



➔ **OVN-Northd**

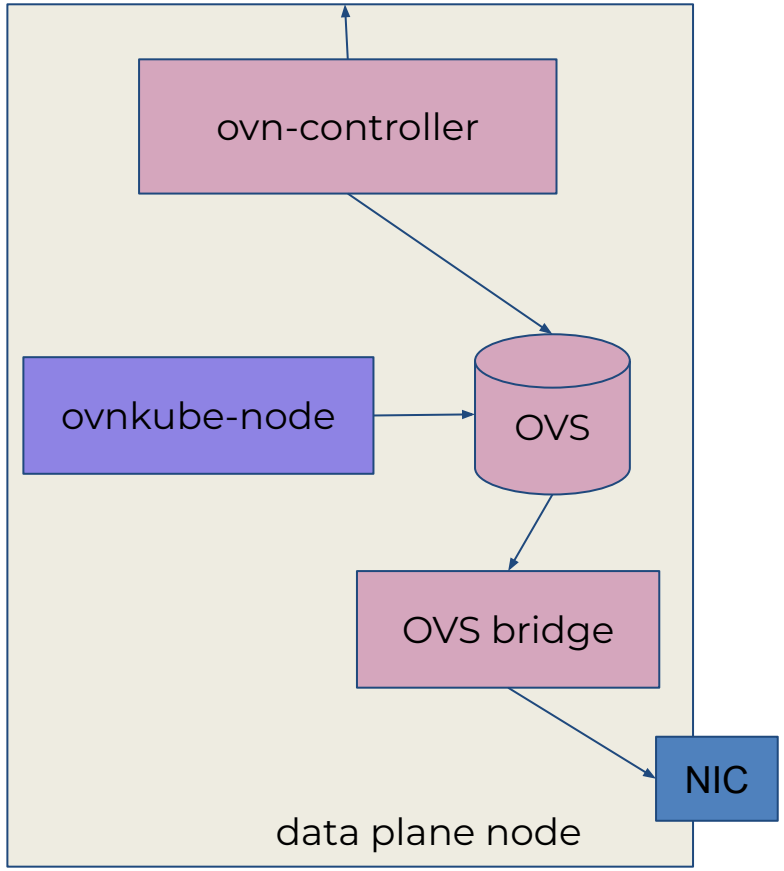
- Native OVN component
- Reads changes in NBDB
- Translates the OVN logical entities into OVN logical flows
- Saves these flows into SBDB

➔ **OVN-SouthBoundDataBase (SBDB)**

- Native OVN component
- 3 replicas across control plane nodes

OVN-Kubernetes Architecture

- Components running in the Data Plane



➔ **OVN-Kubernetes Node**

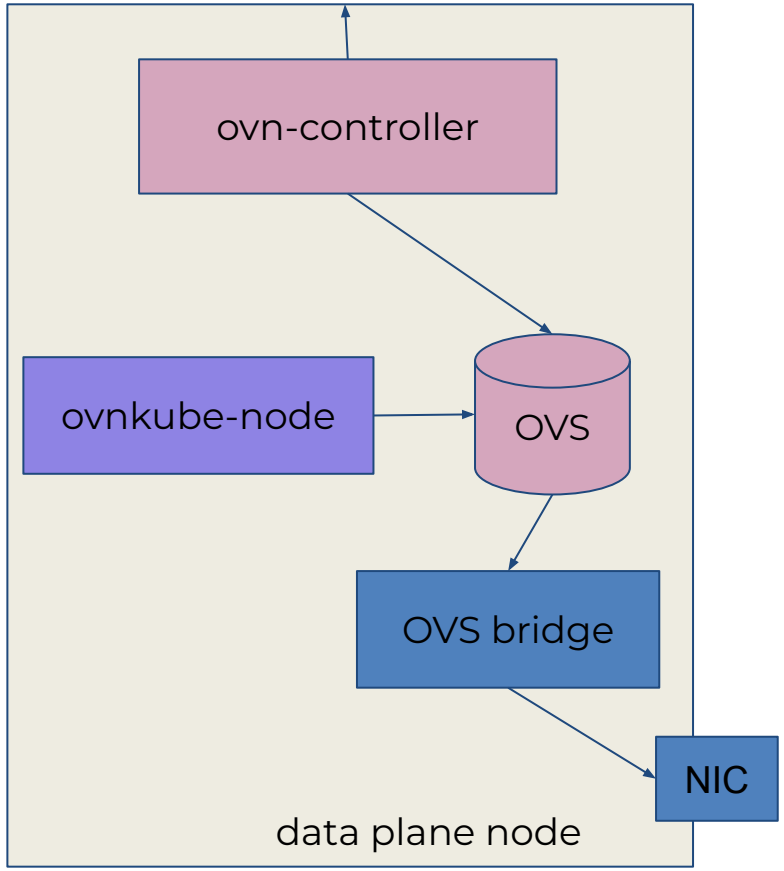
- OVN Kubernetes component
- Runs the CNI executable (CNI ADD/DEL)
- Digests the IPAM annotation set on pod by master
- Creates the veth pair for the pod
- Creates the ovs port on bridge

➔ **OVN and OVS native components**

- OVN-Controller, OVS, that translate these entities created by control plane into OVS (traffic) flows

OVN-Kubernetes Architecture

- Components running in the Data Plane



➔ **OVN-Kubernetes Node**

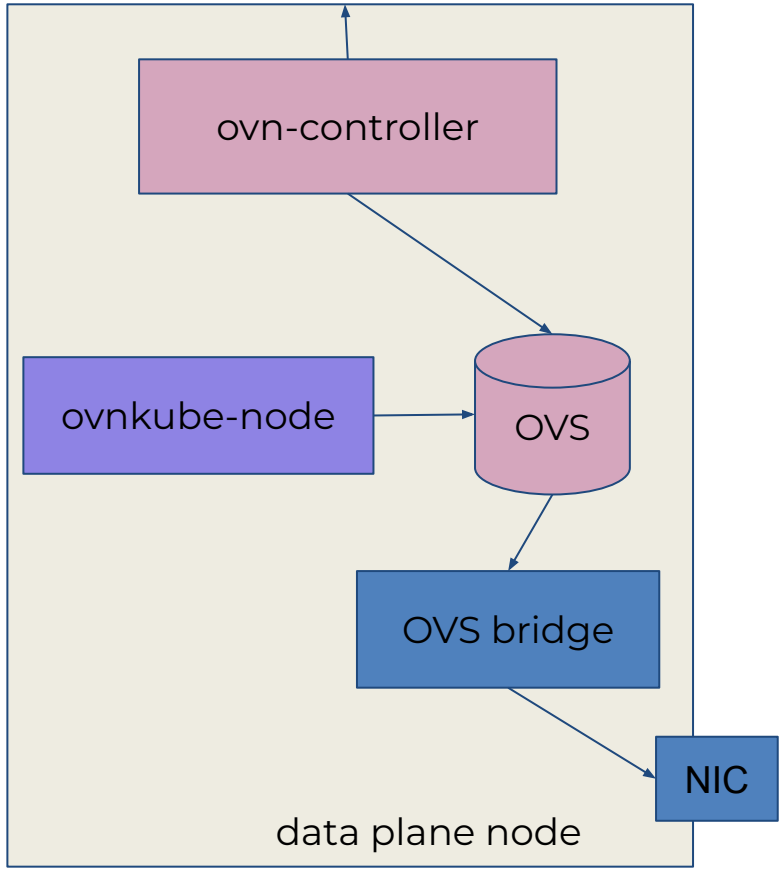
- OVN Kubernetes component
- Runs the CNI executable (CNI ADD/DEL)
- Digests the IPAM annotation set on pod by master
- Creates the veth pair for the pod
- Creates the ovs port on bridge

➔ **OVN-Controller**

- OVN Native component
- Connects to SBDB running in control plane using TLS
- Converts SBDB logical flows into openflows
- Write them to OVS

OVN-Kubernetes Architecture

- Components running in the Data Plane

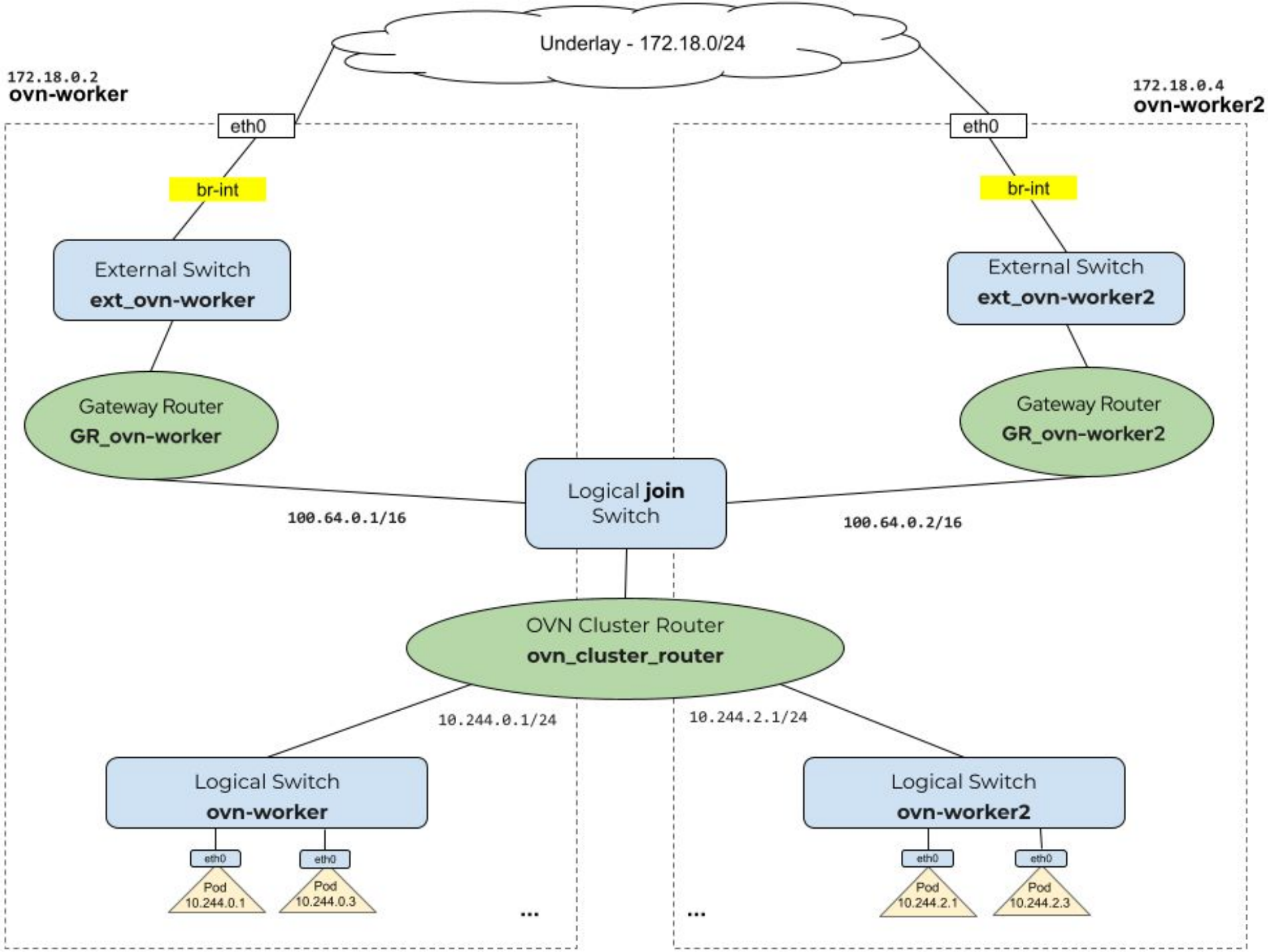


➔ **Open vSwitch**

- OVS Native component
- OVS process run via systemd on the host
- virtual switch that pushes the network plumbing to the edge on the node

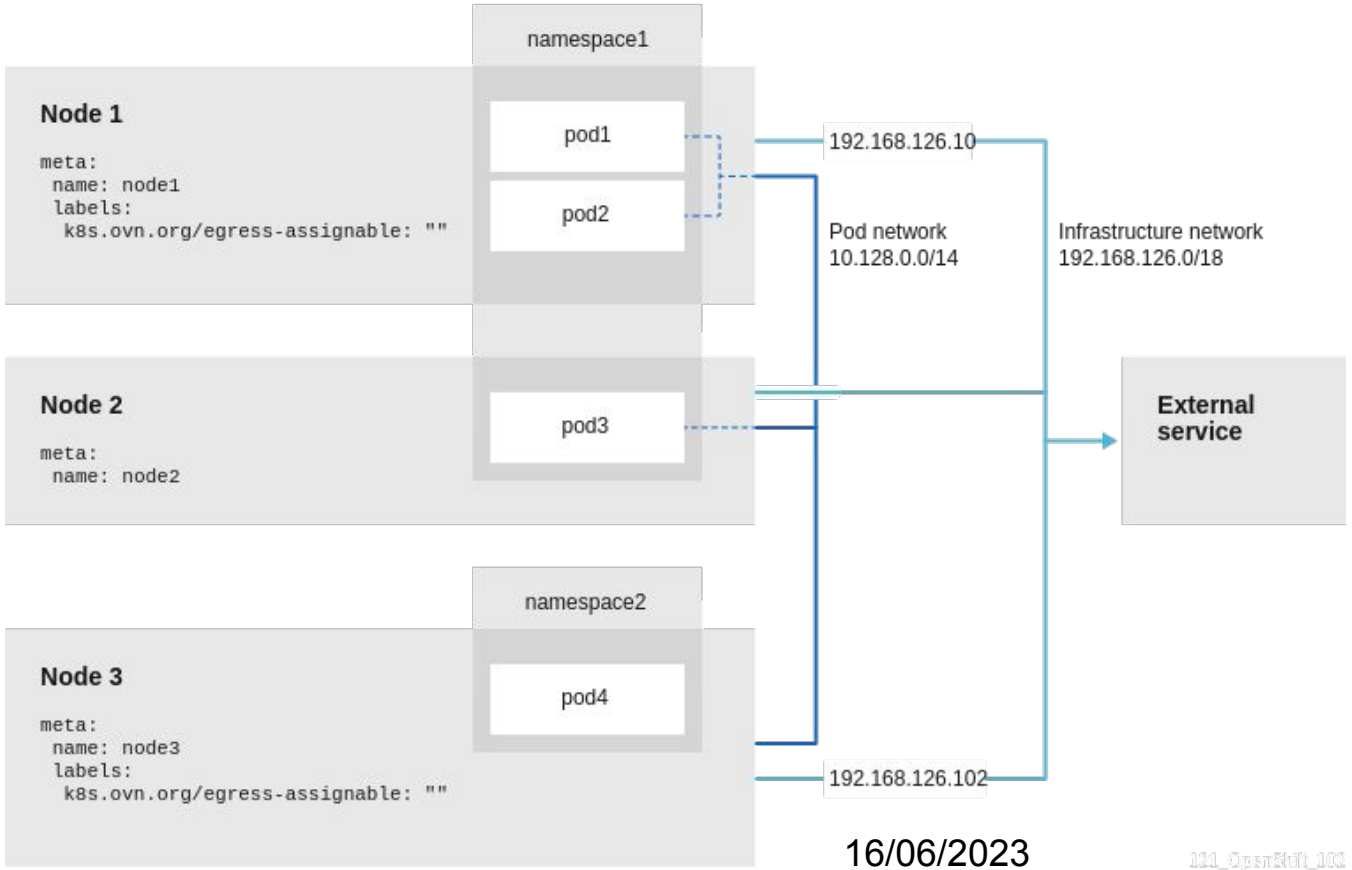
OVN-Kubernetes Demo

OVN-Kubernetes Networking Topology


Underlay - 172.18.0/24
172.18.0.2
ovn-worker
172.18.0.4
ovn-worker2
eth0
eth0
br-int
br-int
External Switch
ext_ovn-worker
External Switch
ext_ovn-worker2
Gateway Router
GR_ovn-worker
Gateway Router
GR_ovn-worker2
Logical join
Switch
100.64.0.1/16
100.64.0.2/16
OVN Cluster Router
ovn_cluster_router
10.244.0.1/24
10.244.2.1/24
Logical Switch
ovn-worker
Logical Switch
ovn-worker2
eth0
eth0
Pod
10.244.0.1
Pod
10.244.0.3
eth0
eth0
Pod
10.244.2.1
Pod
10.244.2.3
...
...

Additional Features in OVN-Kubernetes

- Egress IP
  - ensure that the traffic from selected pods has a consistent source IP address for services outside the cluster network



Additional Features in OVN-Kubernetes

- Egress Firewall
  - Restricts egress traffic leaving OpenShift cluster
  - Allows users to specify DNS names that resolve to an IP
- Hybrid overlay
  - Allow networking in a cluster consisting of both Windows and Linux nodes
- And more...

Thank You
Q&A?