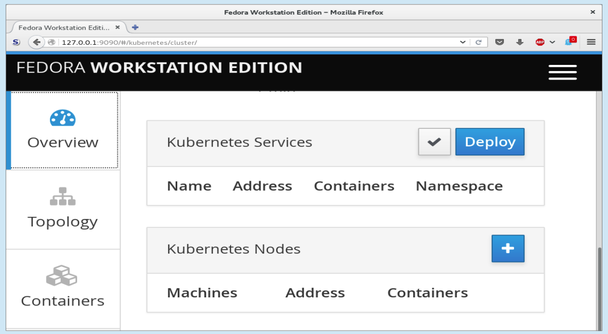
Networking Containers with Kubernetes and OpenShift

Dan Williams
Networking Services, Red Hat

Kubernetes Components

Web UI



Command-line interface

```
$ kubectl
kubectl controls the Kubernetes cluster
manager.

Find more information at
https://github.com/GoogleCloudPlatform/ku
bernetes.

Usage:
  kubectl [flags]
  kubectl [command]
```

Kubernetes Master

apiserver

etcd

scheduler

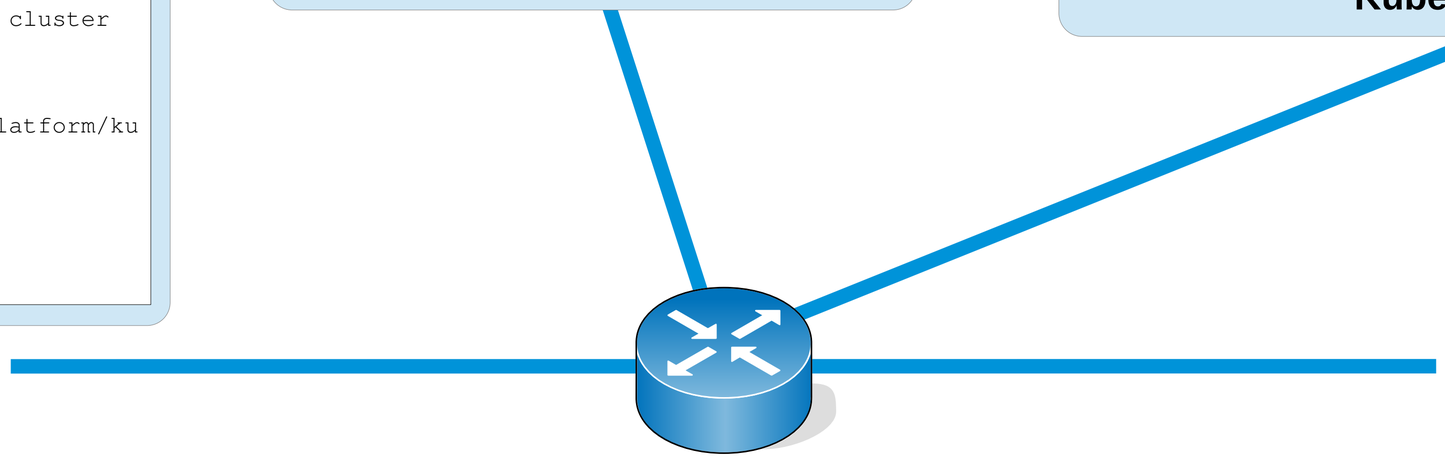
controller

Kubernetes Node

Pod

Container 1

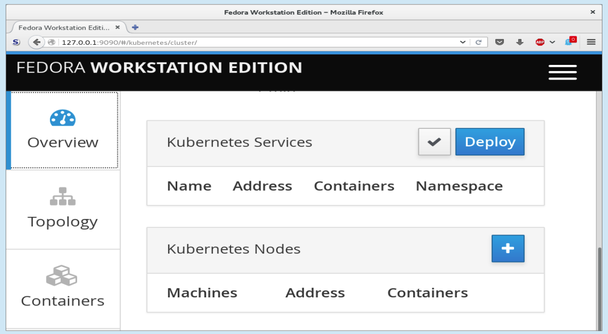
Container 2



redhat.

Kubernetes Networking Out-of-the-Box

Web UI

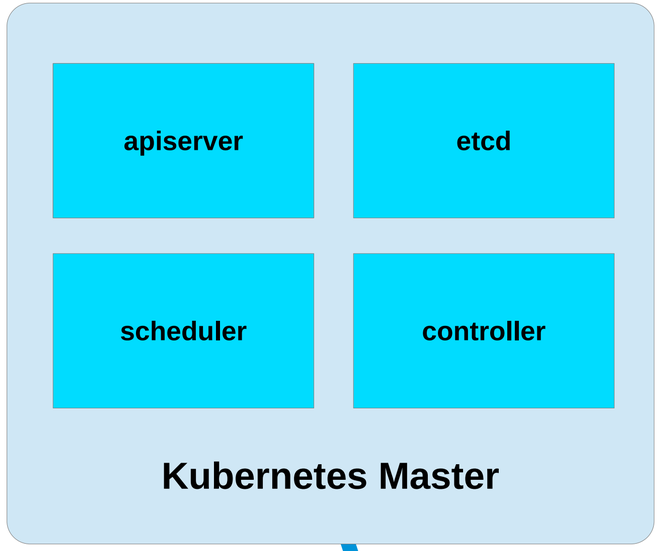


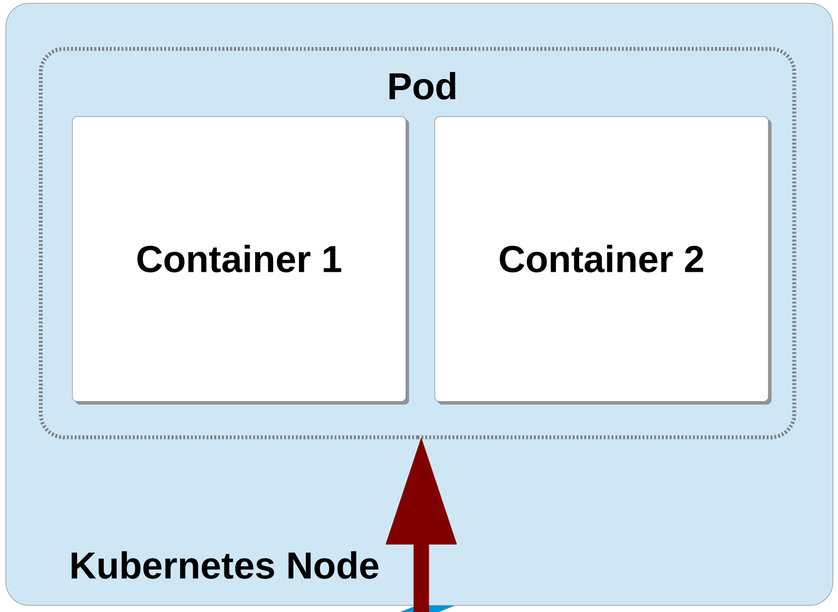
Command-line interface

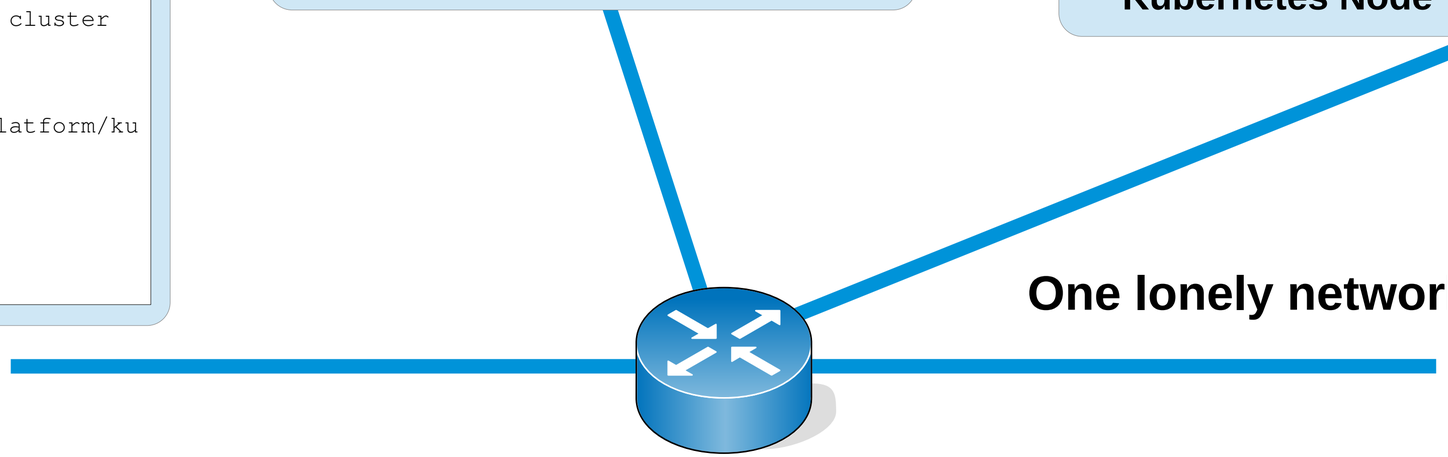
```
$ kubectl
kubectl controls the Kubernetes cluster
manager.

Find more information at
https://github.com/GoogleCloudPlatform/ku
bernetes.

Usage:
  kubectl [flags]
  kubectl [command]
```







One lonely networking integration point

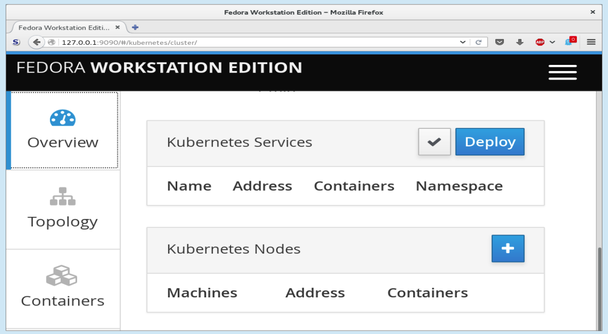
redhat.

How can we improve Kubernetes networking?

- Enhance the existing network plugin architecture
- Add multi-tenancy support through network objects
- Implement a flexible, fine-grained network security policy
- Make sure UI understands these concepts
- Make sure they are easy for administrators and developers to use

Kubernetes Networking Out-of-the-Box

Web UI

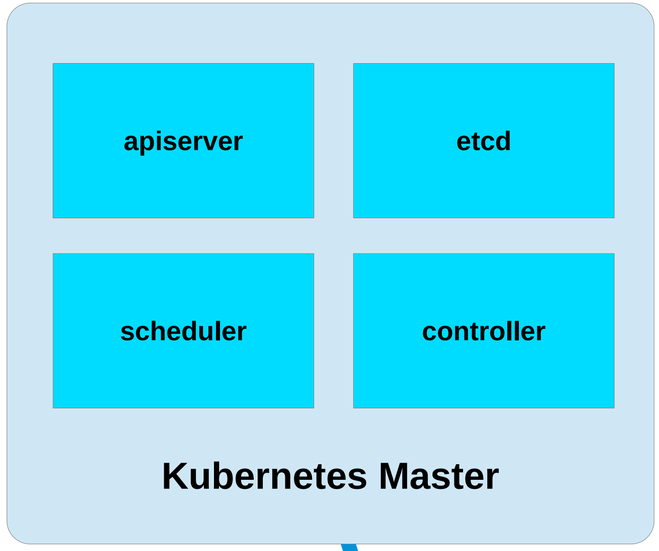


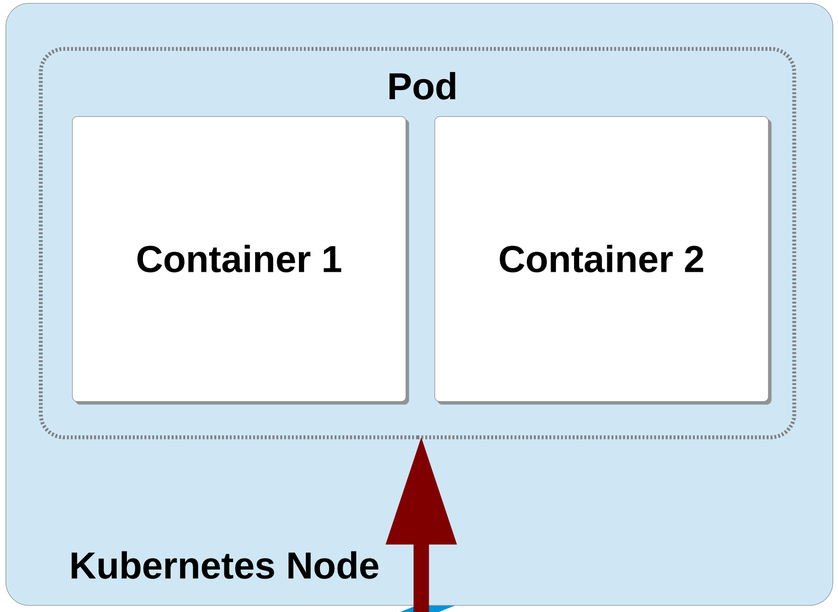
Command-line interface

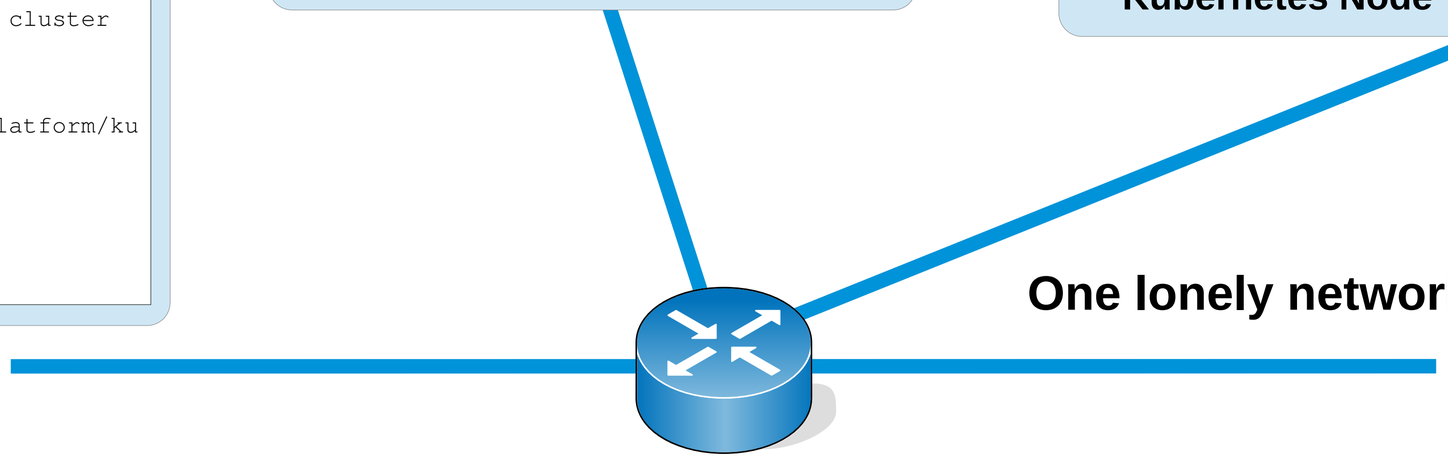
```
$ kubectl
kubectl controls the Kubernetes cluster
manager.

Find more information at
https://github.com/GoogleCloudPlatform/ku
bernetes.

Usage:
  kubectl [flags]
  kubectl [command]
```





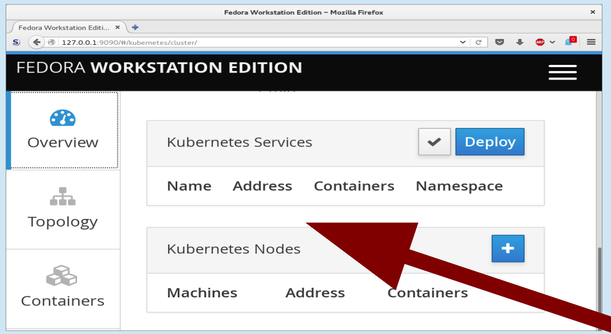


One lonely networking integration point

redhat.

Improve Kubernetes: Plugin Architecture

Web UI



Command-line interface

```
$ kubectl
kubectl controls the Kubernetes cluster
manager.

Find more information at
https://github.com/Go      oudPlatform/ku
bernetes.

Usage:
  kubectl [flags]
  kubectl [command]
```

apiserver

etcd

scheduler

controller

Kubernetes Master

Pod

Container 1

Container 2

Kubernetes Node

Everyone gets an integration point!!!!

redhat.

Improve Kubernetes: Plugin Architecture

- Two existing network plugin APIs
  - exec
  - Container Network Interface (CNI)
  - Only deals with pod setup/teardown
- Consolidate around one plugin API
- Add hooks at multiple points in the stack
  - master
  - nodes
  - pod setup/teardown
  - user interfaces
- Ensure the needs of multiple networking providers are met

Improve Kubernetes Networking: Multi-tenancy and networks

- Kubernetes is currently 100% network-oblivious
- We must add networks as first-class objects
- Allow external entities to provide network data and events to Kubernetes
- Allow to use multiple networks (distinct from multiple subnets)
- Flexible addressing methods
  - overlapping IPs between networks
  - subnet-per-node
- Service handling and proxies

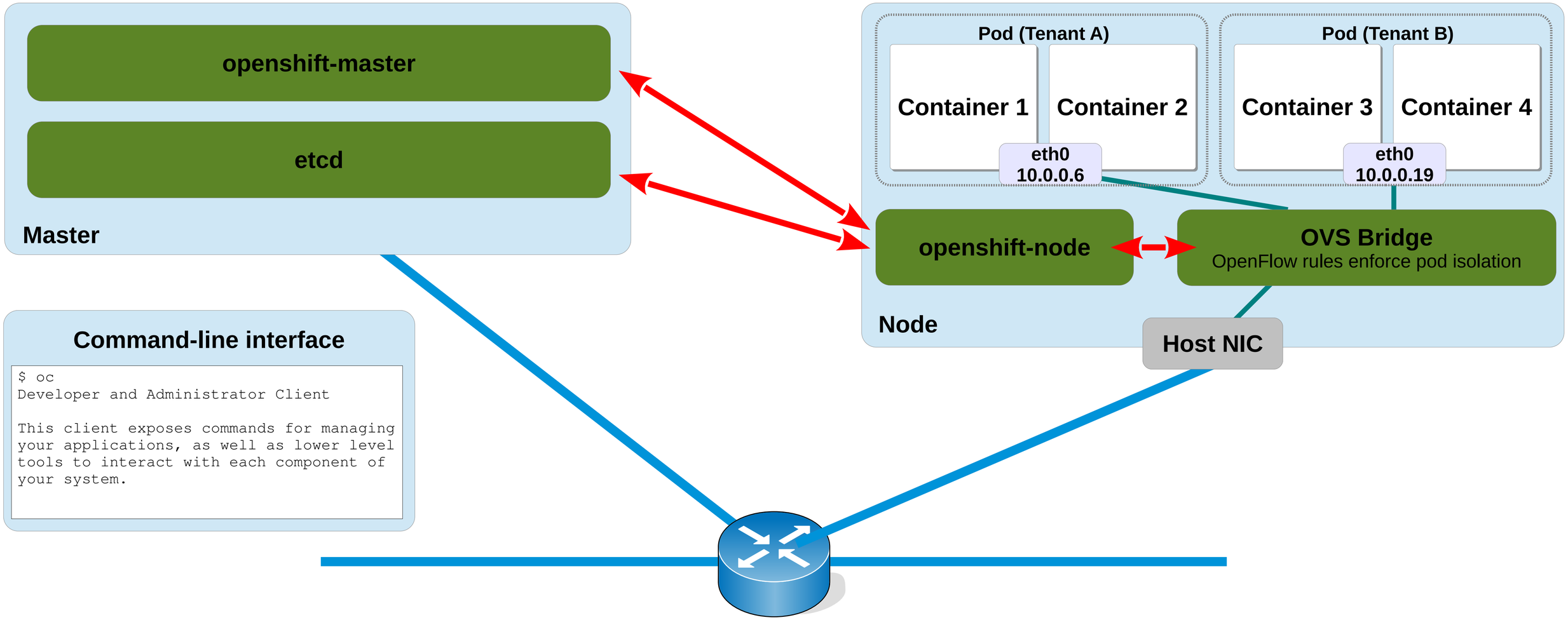
Improve Kubernetes Networking: Network Security Policy

- Security policy enforces which pods can connect to which networks
- Pod definitions cannot control network associations
- Cluster administrator must have control over policy and pod ↔ network mappings
- Must allow pods to map to multiple networks
- Must allow cross-talk between networks if configured

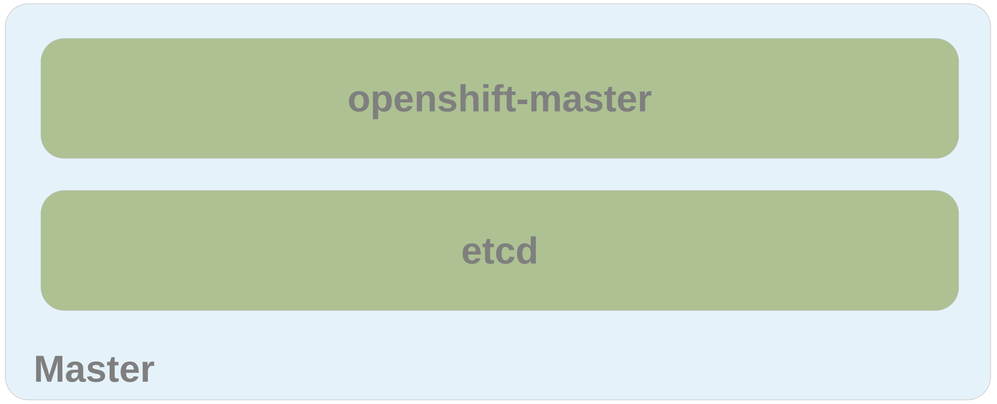
Kubernetes + PaaS = OpenShift

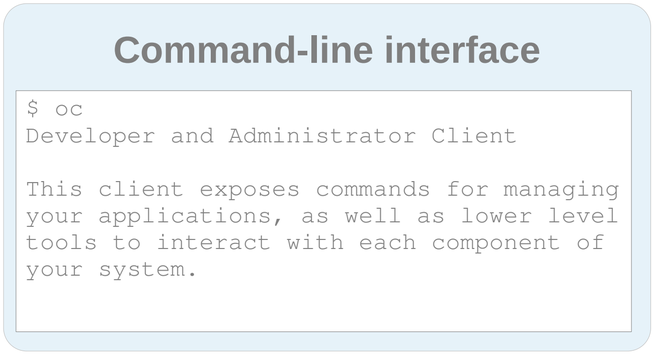
- OpenShift is an open-source project that provides Platform-as-a-Service on top of Kubernetes
- OpenShift wraps Kubernetes and adds:
  - The concept of a complete application
  - Building and deploying docker images from source code (STI)
  - Application lifecycle management (CI, staging, production, ...)
  - Focus on user or administrator experience
  - Out-of-the-box Open vSwitch-based multi-tenant networking
  - Enhanced, flexible access control
  - Secure cluster communication by default

OpenShift Networking with Open vSwitch



OpenShift Networking: The Master

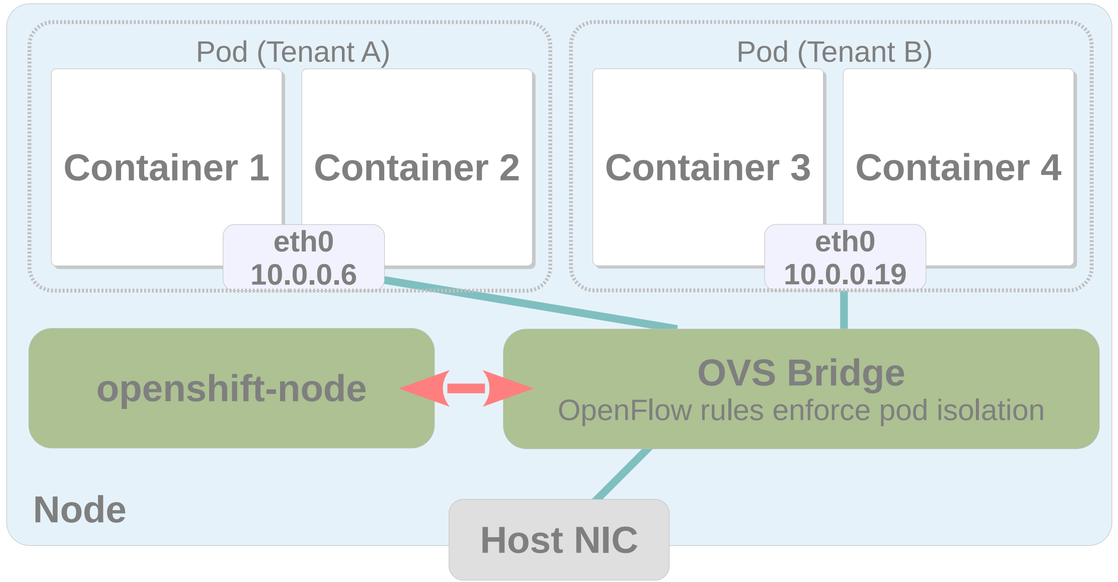




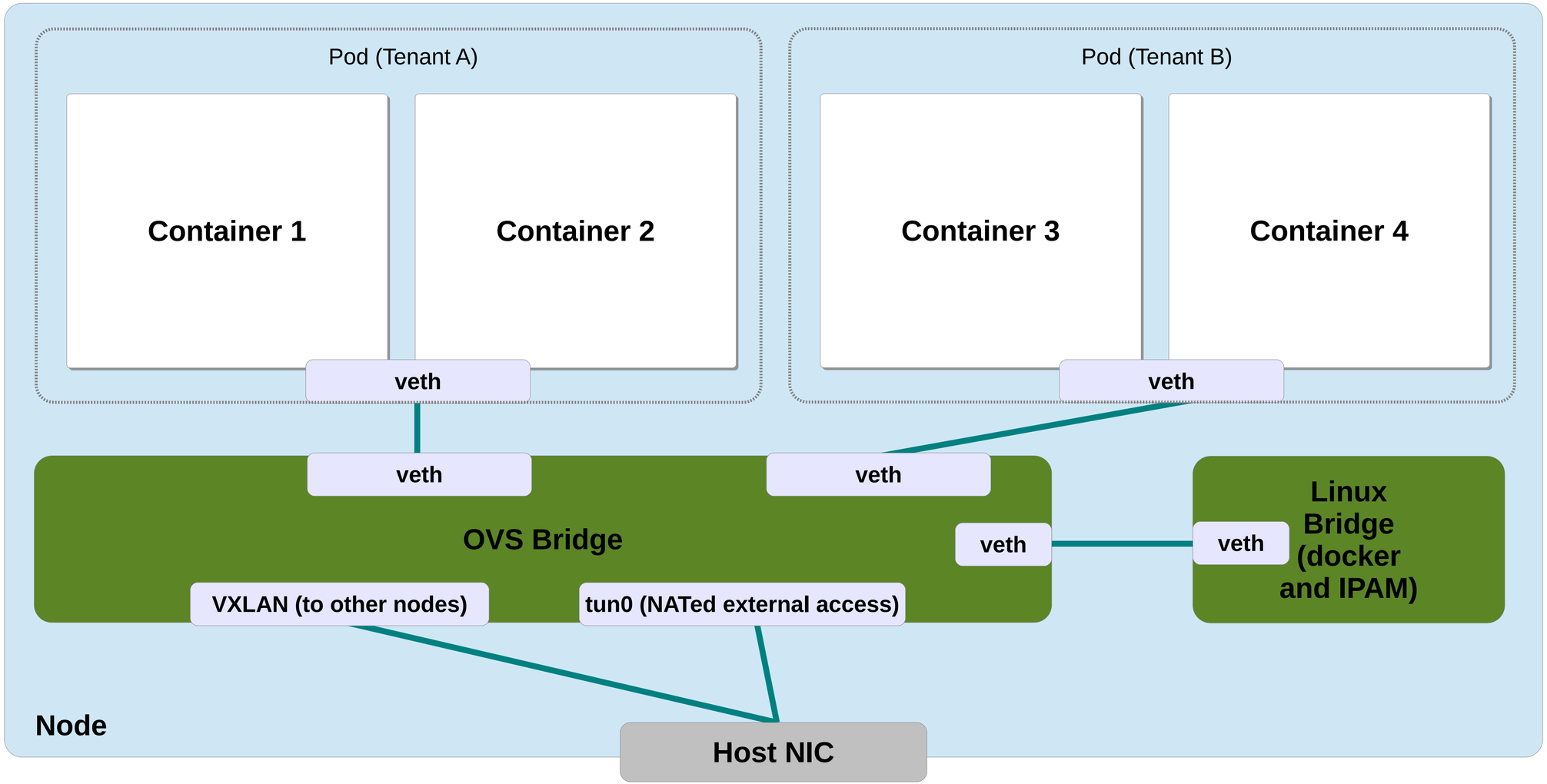
- OpenShift projects are mapped to tenant networks
  - oc new-project TenantA
  - oc create -f <pod template>
  - oc new-project TenantB
  - oc create -f <pod template>
- Tenant networks can be isolated from each other, joined, or “admin”
- Master keeps mapping of projects/networks to Virtual Network ID
- Administration of networks via the openshift-client ('oc') command

OpenShift Networking: The Node

- openshift-node updates OVS bridge flows for:
  - node changes
  - service changes
  - network namespace changes
- Provides a Kubernetes CNI plugin for:
  - pod setup and teardown
  - network namespace changes, joins, and splits
- Each node allocated a node subnet from the cluster subnet
- IPAM provided by Docker using node subnet
- All pods on a node share common subnet, with isolation enforced by OVS flow rules
- Isolation between nodes is enforced through Virtual Network IDs (derived from VXLAN tunnel ID) which are checked on each node
- External network access through NAT-ed tun interface



OpenShift Networking: Node Architecture



OpenShift Networking: OVS Flows

All traffic enters OVS bridge here:

cookie=0x0, table=1, in_port=1 actions=goto_table:2 **[vxlan0]**

cookie=0x0, table=1, in_port=2 actions=goto_table:5 **[tun0]**

cookie=0x0, table=1, actions=goto_table:3

VXLAN ingress from other nodes:

cookie=0x0, table=2, priority=100,ip,nw_dst=10.1.0.0/24 actions=move:NXM_NX_TUN_ID[0..31]->NXM_NX_REG0[],goto_table:6

cookie=0x0, table=2, tun_id=0 actions=goto_table:5 **['admin' networks]**

ingress from pods:

cookie=0x3, table=3, priority=100,ip,in_port=3,nw_src=10.1.0.2 actions=load:0xd->NXM_NX_REG0[],goto_table:4 ***[VNI tagging]***

cookie=0x4, table=3, priority=100,ip,in_port=4,nw_src=10.1.0.3 actions=load:0xe->NXM_NX_REG0[],goto_table:4 ***[VNI tagging]***

services handling rules:

cookie=0x0, table=4, priority=200,tcp,reg0=0xa,nw_dst=172.30.0.1,tp_dst=443 actions=output:2 **[service rule]**

cookie=0x0, table=4, priority=100,ip,nw_dst=172.30.0.0/16 actions=drop

cookie=0x0, table=4, priority=0 actions=goto_table:5

general routing:

cookie=0x0, table=5, priority=200,ip,nw_dst=10.1.0.1 actions=output:2 **[traffic to external networks]**

cookie=0x0, table=5, priority=150,ip,nw_dst=10.1.0.0/24 actions=goto_table:6 **[traffic to pods on the node]**

cookie=0x0, table=5, priority=100,ip,nw_dst=10.1.0.0/16 actions=goto_table:7 **[cluster network egress]**

cookie=0x0, table=5, priority=0,ip actions=output:2

egress to pods:

cookie=0x0, table=6, priority=200,ip,reg0=0 actions=goto_table:8 **['admin' networks]**

cookie=0x4, table=6, priority=100,ip,reg0=0xe,nw_dst=10.1.0.3 actions=output:4 ***[pod filter rule]***

cookie=0x3, table=6, priority=100,ip,reg0=0xd,nw_dst=10.1.0.2 actions=output:3 ***[pod filter rule]***

egress to nodes via VXLAN:

cookie=0xaf50204,table=7 priority=100,ip,nw_dst=10.1.1.0/24 actions=move:NXM_NX_REG0[]->NXM_NX_TUN_ID[0..31],set_field:10.245.2.4->tun_dst,output:1

How to Make OpenShift Networking Better

- Move OpenShift networking to external projects
  - Drive improvements to Kubernetes network plugin API and multi-network support
  - Contribute multi-network and other improvements to flannel
  - Develop more community around simple OVS-based container networking
- Continue improving tunnel performance
  - VXLAN and Geneve offloading and optimization
- Use OVS internal ports across kernel network namespaces
  - Simplifies container network interface management
- Use OVS conntrack and NAT instead of kernel iptables
- Move IPAM from docker to CNI plugin
  - Better control and flexibility over addressing

Questions?