Agenda

Kubernetes and SDN Choices

OpenShift and OpenShift SDN

Typical network security questions on an Enterprise Cluster

- Restricting traffic across tiers
- Handling network zones and isolation
- Securing Egress
- Securing Ingress
- Securing communications between Nodes
- Application Network Security

Kubernetes is a clear winner in the world of Container Orchestration and Management

WHAT DOES IT TAKE TO MAKE K8S ENTERPRISE READY?

How do I support my K8S cluster? version upgrades, fixes/patches etc

You'll need a lot more on your cluster than Kubernetes itself..

Kubernetes

K8S REQUIRES A SECURE ENTERPRISE GRADE LINUX CONTAINER HOST

Kubernetes

Red Hat Enterprise Linux or Red Hat CoreOS

K8S CLUSTER REQUIRES NETWORKING AND STORAGE SOLUTIONS

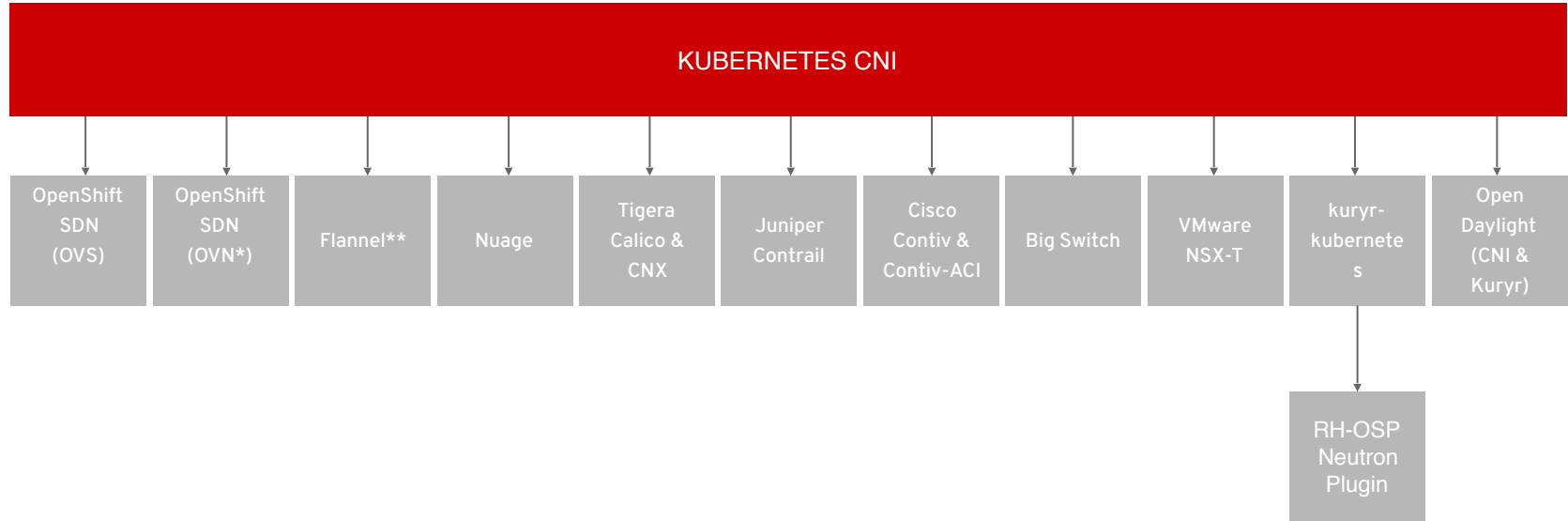| Software Defined Network, Storage Solution |
| :---: |
| Kubernetes (CNI, CSI) |
| Red Hat Enterprise Linux or Red Hat CoreOS |

Software Defined Network Choices

For Network solution, K8S uses CNI

K8S CLUSTER REQUIRES LIFECYCLE MGMT FOR DEVELOPERS AND OPERATORS

Developer Console, Operations Console, Lifecycle Mgmt, Automated Operations
Software Defined Network, Storage
Kubernetes (CNI, CSI)
Red Hat Enterprise Linux or Red Hat CoreOS

YOU'LL NEED CONTAINER REGISTRY, LOGGING, METRICS, CHARGEBACK CAPABILITIES

Cluster Services

Registry
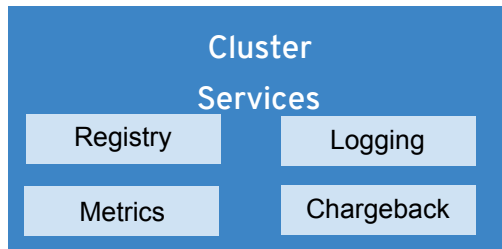
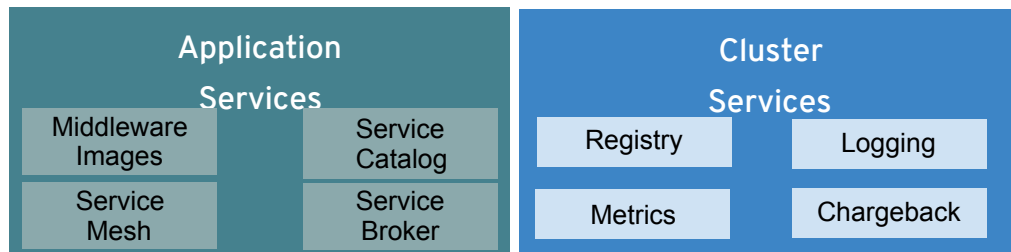Logging

Metrics

Chargeback

Developer Console, Operations Console, Lifecycle Mgmt, Automated Operations

Software Defined Network, Storage

Kubernetes (CNI, CSI)

Red Hat Enterprise Linux or Red Hat CoreOS

YOU'LL NEED TO STANDARDIZE ON MIDDLEWARE, A SERVICE CATALOG, AND MICROSERVICE MGMT

Application Services		Cluster Services	
Middleware Images	Service Catalog	Registry	Logging
Service Mesh	Service Broker	Metrics	Chargeback

Developer Console, Operations Console, Lifecycle Mgmt, Automated Operations
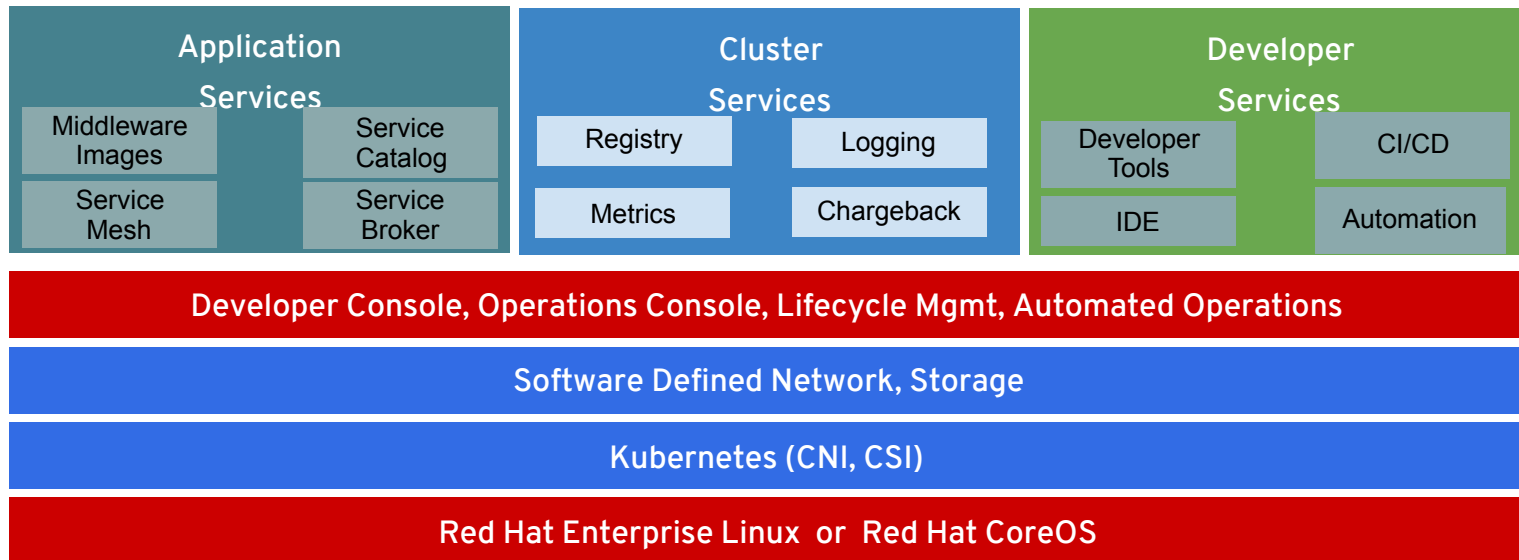
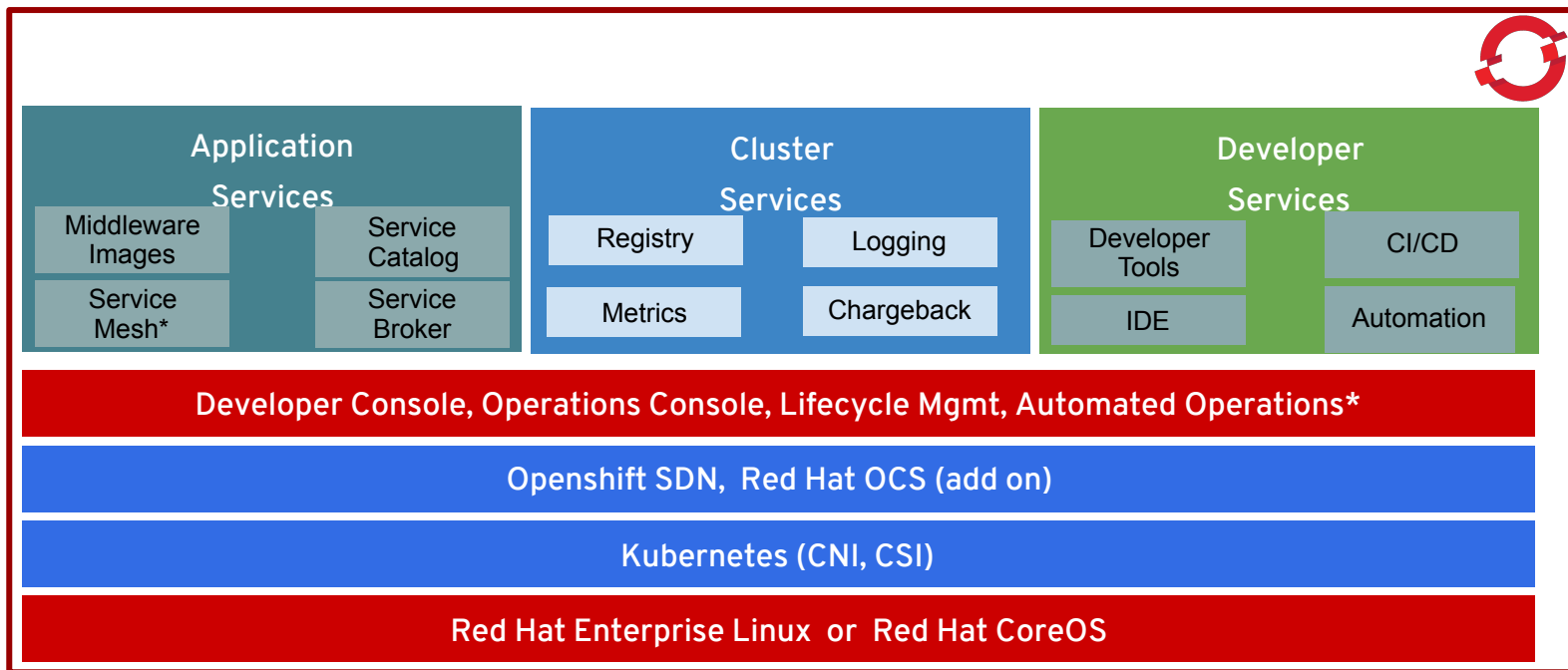**Software Defined Network, Storage**

Kubernetes (CNI, CSI)

Red Hat Enterprise Linux or Red Hat CoreOS

*coming soon

DEVELOPERS NEED IDEs, BUILD MGMT, CICD, DEBUGGING FEATURES AND MORE..

Application Services

Middleware Images	Service Catalog
Service Mesh	Service Broker

Cluster Services

Registry	Logging
Metrics	Chargeback

Developer Services

Developer Tools	CI/CD
IDE	Automation

Developer Console, Operations Console, Lifecycle Mgmt, Automated Operations

Software Defined Network, Storage

Kubernetes (CNI, CSI)
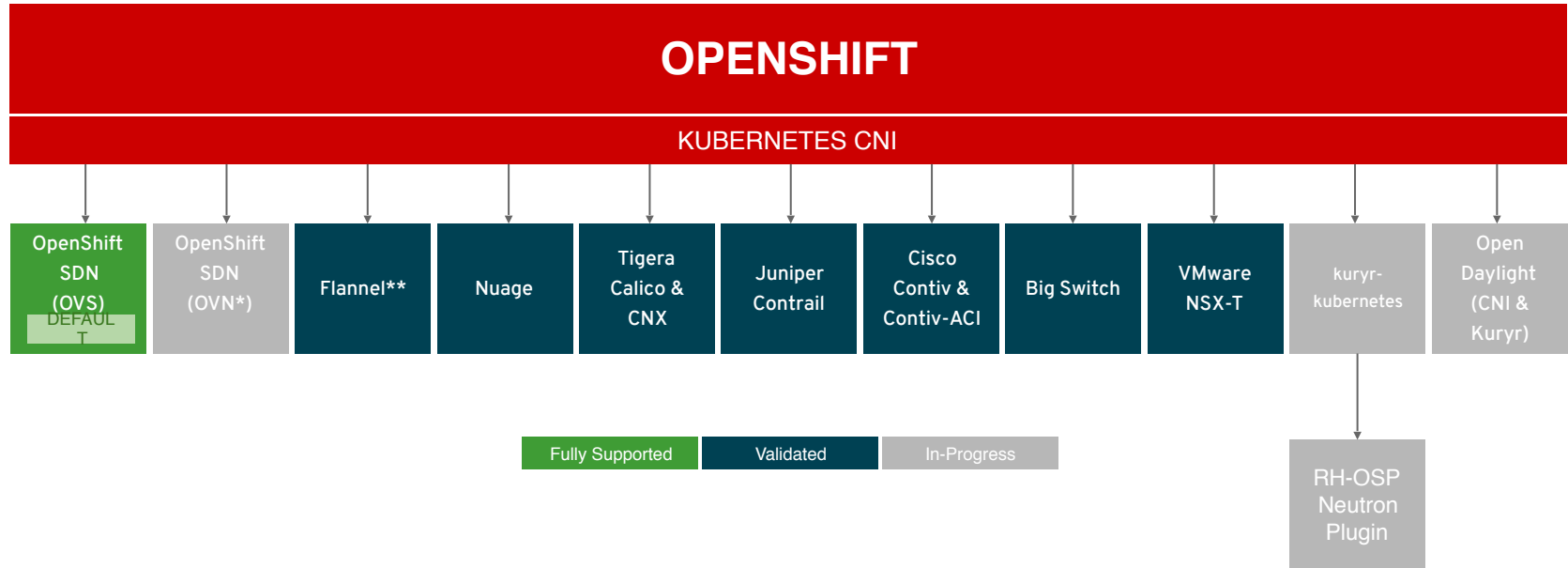
Red Hat Enterprise Linux or Red Hat CoreOS

THIS MAKES A REFERENCE ARCHITECTURE
FOR ENTERPRISE KUBERNETES
aka OPENSHIFT

Application Services		Cluster Services		Developer Services	
Middleware Images	Service Catalog	Registry	Logging	Developer Tools	CI/CD
Service Mesh*	Service Broker	Metrics	Chargeback	IDE	Automation

Developer Console, Operations Console, Lifecycle Mgmt, Automated Operations*

Openshift SDN, Red Hat OCS (add on)

Kubernetes (CNI, CSI)

Red Hat Enterprise Linux or Red Hat CoreOS

OPENSHIFT NETWORK PLUGINS

OPENSHIFT

KUBERNETES CNI

| OpenShift SDN (OVS) DEFAULT | OpenShift SDN (OVN*) | Flannel** | Nuage | Tigera Calico & CNX | Juniper Contrail | Cisco Contiv & Contiv-ACI | Big Switch | VMware NSX-T | kuryr-kubernetes | Open Daylight (CNI & Kuryr) |

Fully Supported — **Validated** — In-Progress

RH-OSP Neutron Plugin

* Coming as default in OCP 4.1

** Flannel is minimally verified and is supported only and exactly as deployed in the OpenShift on OpenStack reference architecture

Typical Network Questions

1.   Restricting traffic across tiers

#SecuritySymposium

Traffic Restrictions Across Application Tiers
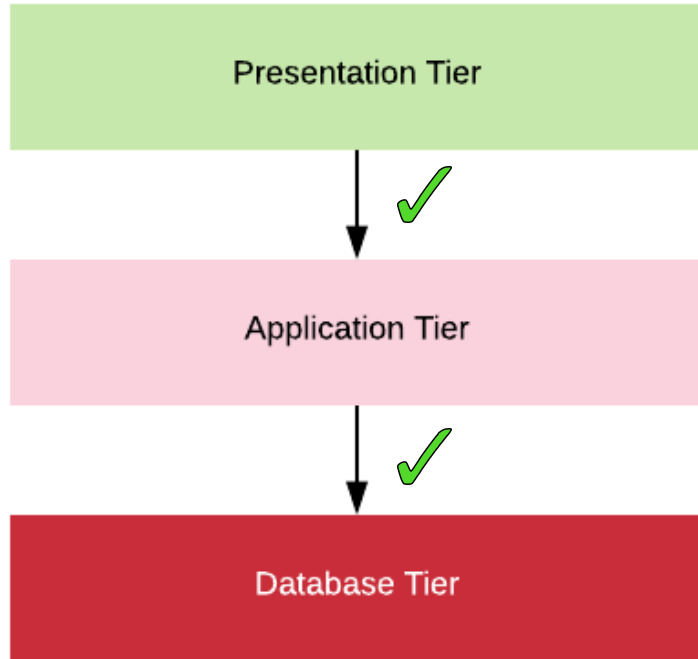
Allowed connections

Disallowed connections

How can we restrict traffic across Application Tiers?

Network Policy Objects

Enables **Microsegmentation**

Allows configuring individual policies at the Pod Level

Apply to ingress traffic for pods and services

Allows restricting traffic between the pods within a project/namespace

Allows traffic to specific pods from other projects/namespaces

Network Policy Objects

PROJECT A

PROJECT B

8080

5432

POD

POD

POD

POD

POD

POD

POD

POD
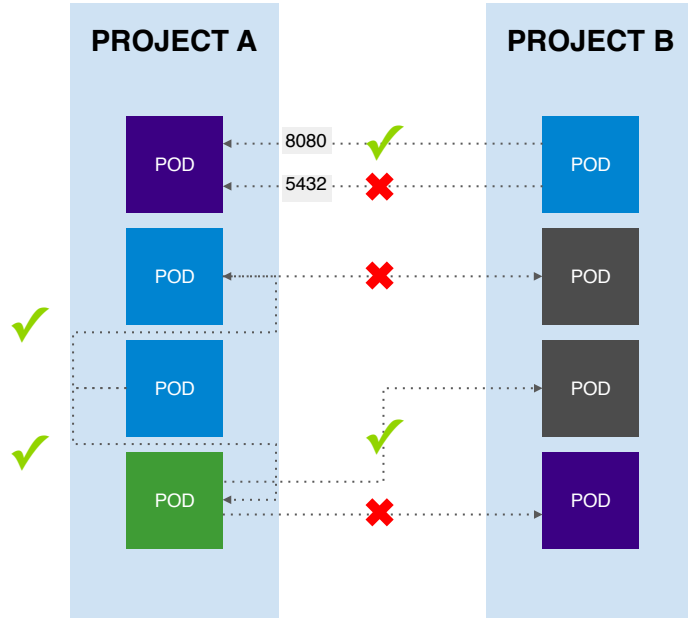
Example Policies
- Allow all traffic inside the project
- Allow traffic from green to gray
- Allow traffic to purple on 8080

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: allow-to-purple-on-8080
spec:
  podSelector:
    matchLabels:
      color: purple
  ingress:
  - ports:
    - protocol: tcp
      port: 8080
```

2. Isolating zones

Network Zones separated by Firewalls
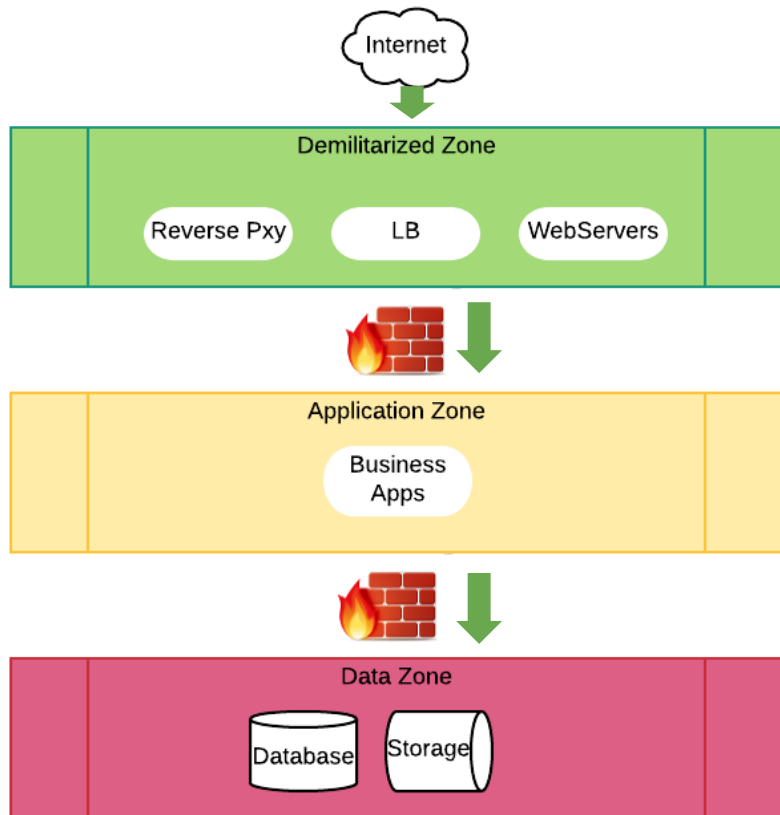
External traffic allowed to touch DMZ

Holes punched in firewalls to allow specific traffic from

DMZ to Application Zone

and from

How do I setup K8S/OpenShift here?

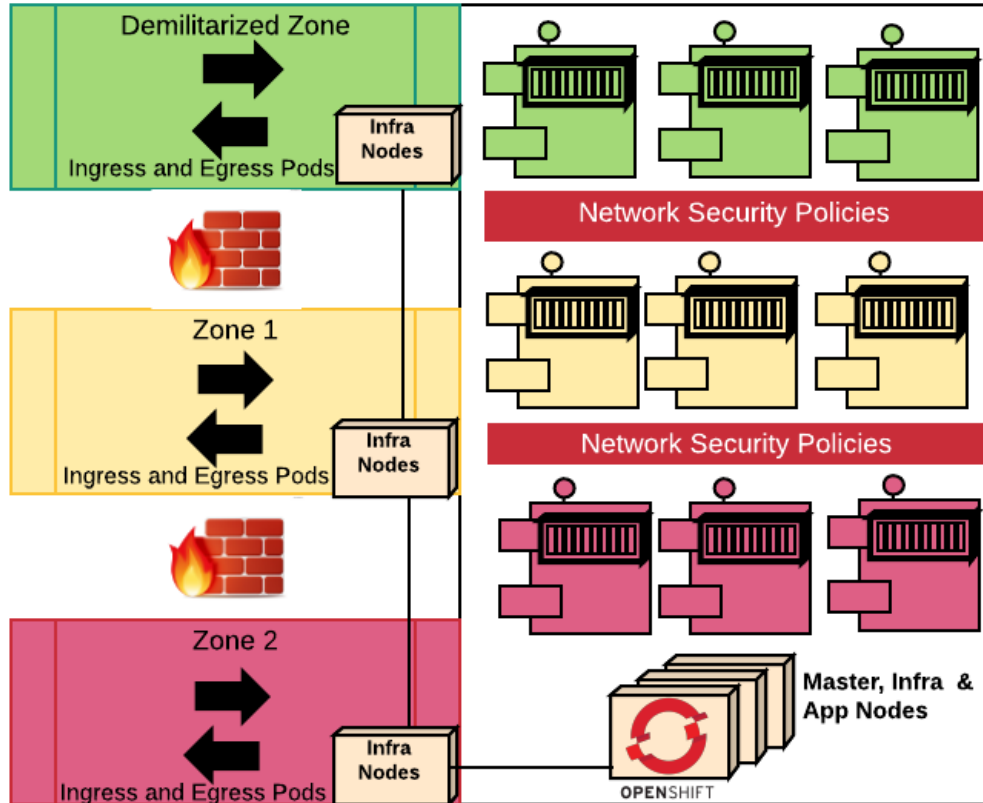# Option 1: OpenShift cluster per Zone
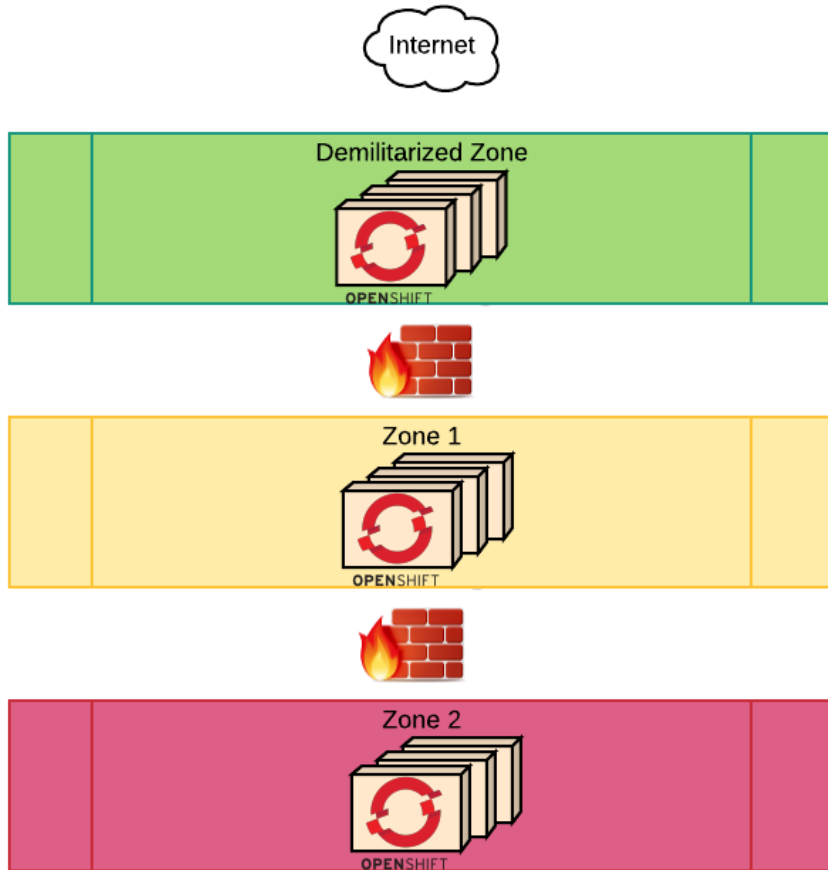
Useful to demonstrate compliance with Security Standards and Regulations

Additional actions needed to protect Master APIs, and other URLs in DMZ that are not supposed to be exposed to Internet

Cost of maintenance is high

Option 2: OpenShift Cluster covering Multiple Zones

Application pods run on one Cluster. Microsegmented with Network Security policies.

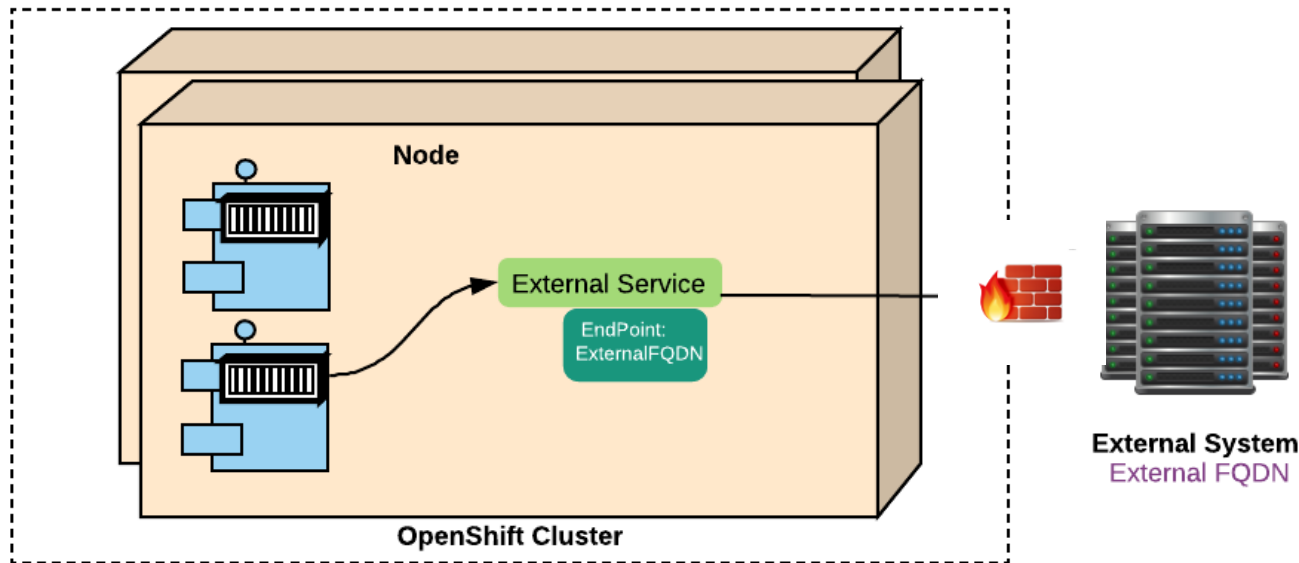Infra Nodes in each zone run Ingress and Egress pods for specific zones

If required, physical isolation of pods to specific nodes is possible with node-selectors. But that defeats the purpose of a shared cluster. Microsegmentation with SDN is the way to go.

3. Securing Egress
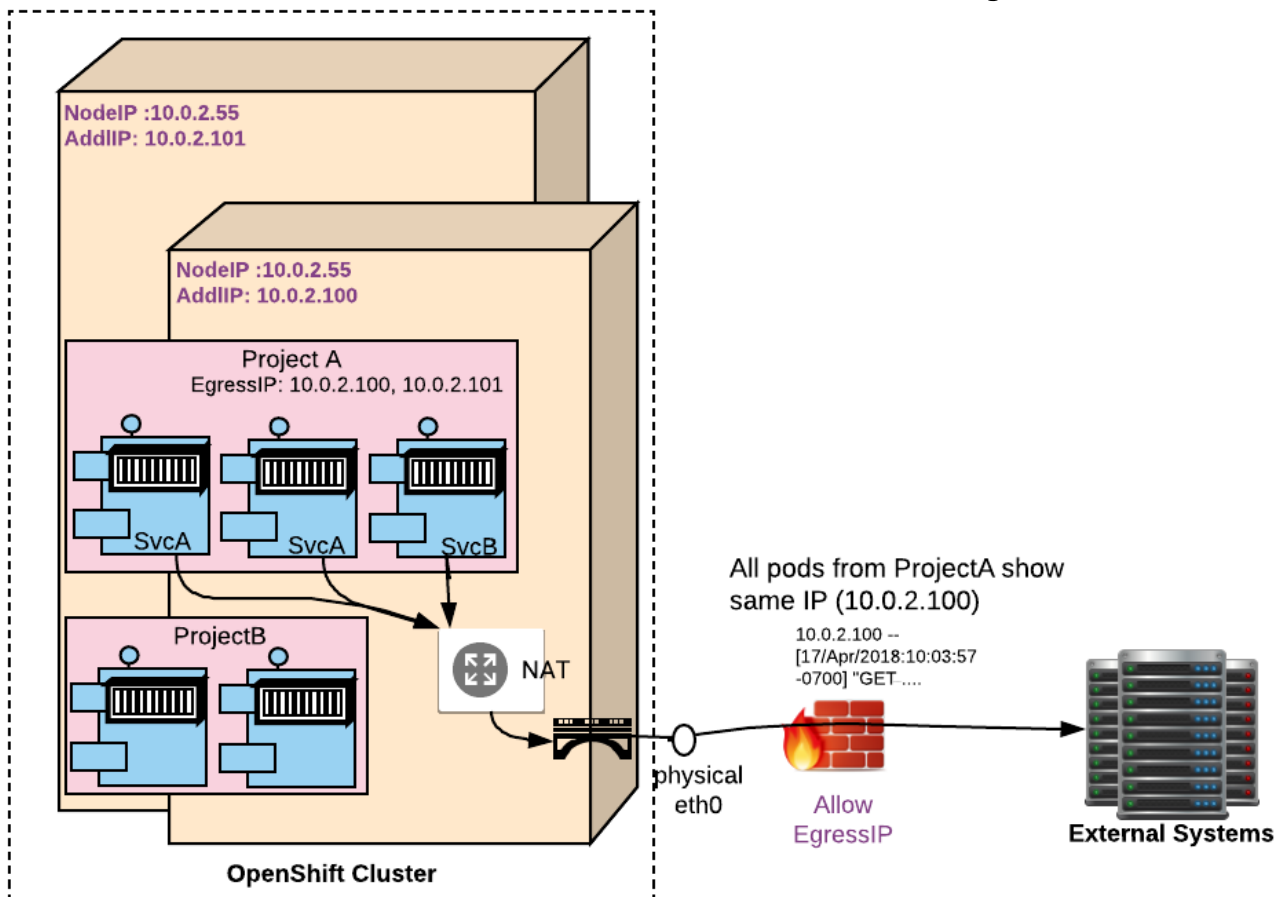
#SecuritySymposium

Connecting via External Service

Application connecting to External System talks to an External Service whose Endpoint is set as Destination IP & Port
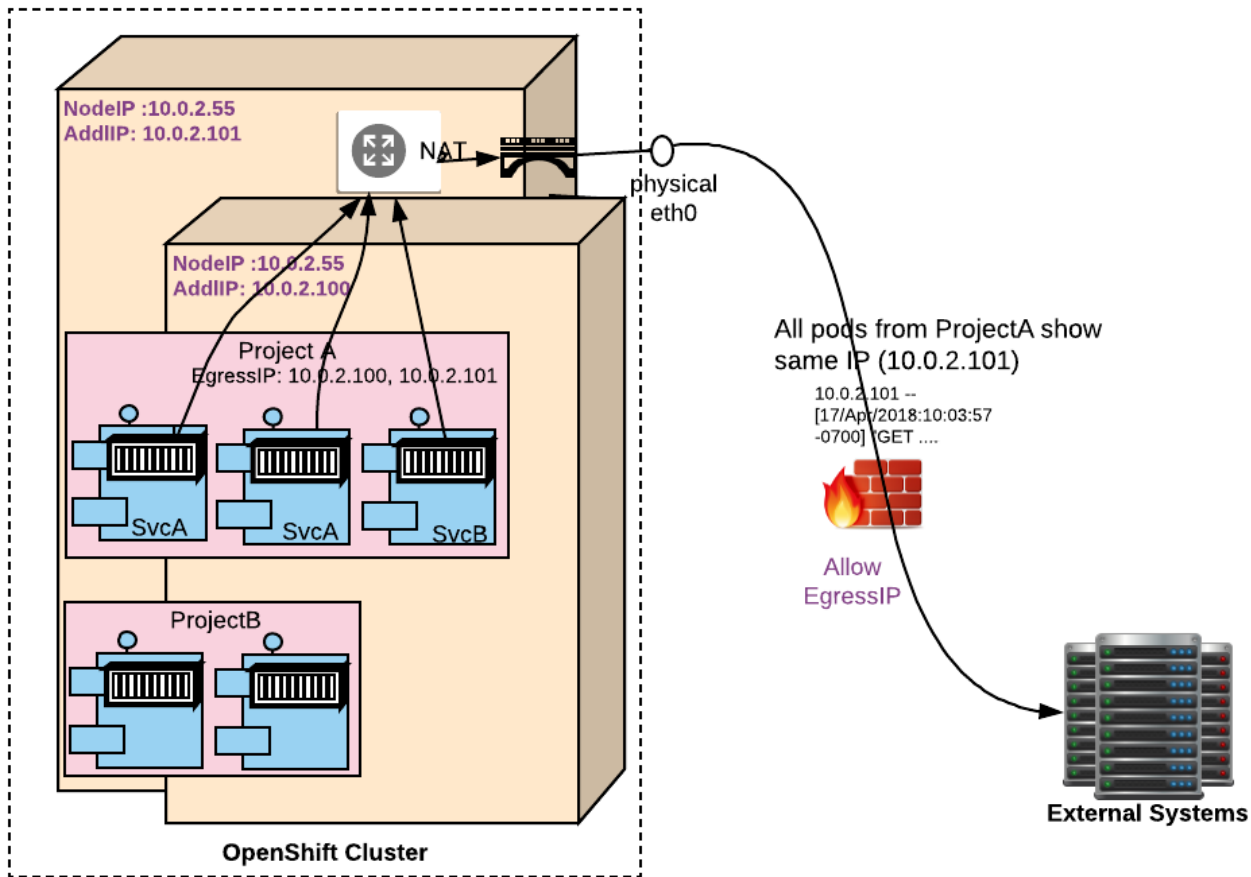Or a Fully qualified domain name (FQDN) of the external system and port

But, what if we have a firewall in front of the External System that allows only Specific IPs?

Static IP for all traffic from a Project

Static IP for all traffic from a Project

NodeIP :10.0.2.55
AddlIP: 10.0.2.101

NAT

physical
eth0

NodeIP :10.0.2.55
AddlIP: 10.0.2.100

Project A
EgressIP: 10.0.2.100, 10.0.2.101

SvcA   SvcA   SvcB

ProjectB

OpenShift Cluster

High availability
scenario

All pods from ProjectA show
same IP (10.0.2.101)

10.0.2.101 --
[17/Apr/2018:10:03:57
-0700] "GET ....

Allow
EgressIP

External Systems

Egress Firewall to Limit Access

Cluster admin can limit the external addresses accessed by some or all pods
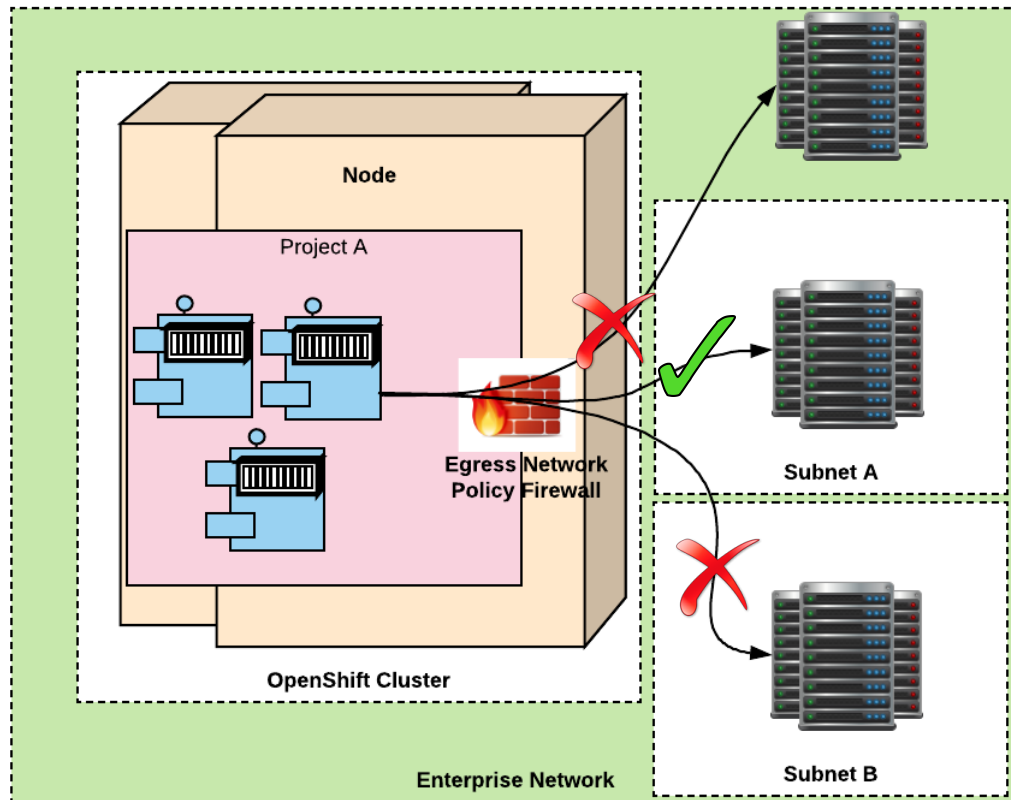
Examples:

A pod can talk to hosts (outside OpenShift cluster) but cannot connect to public internet

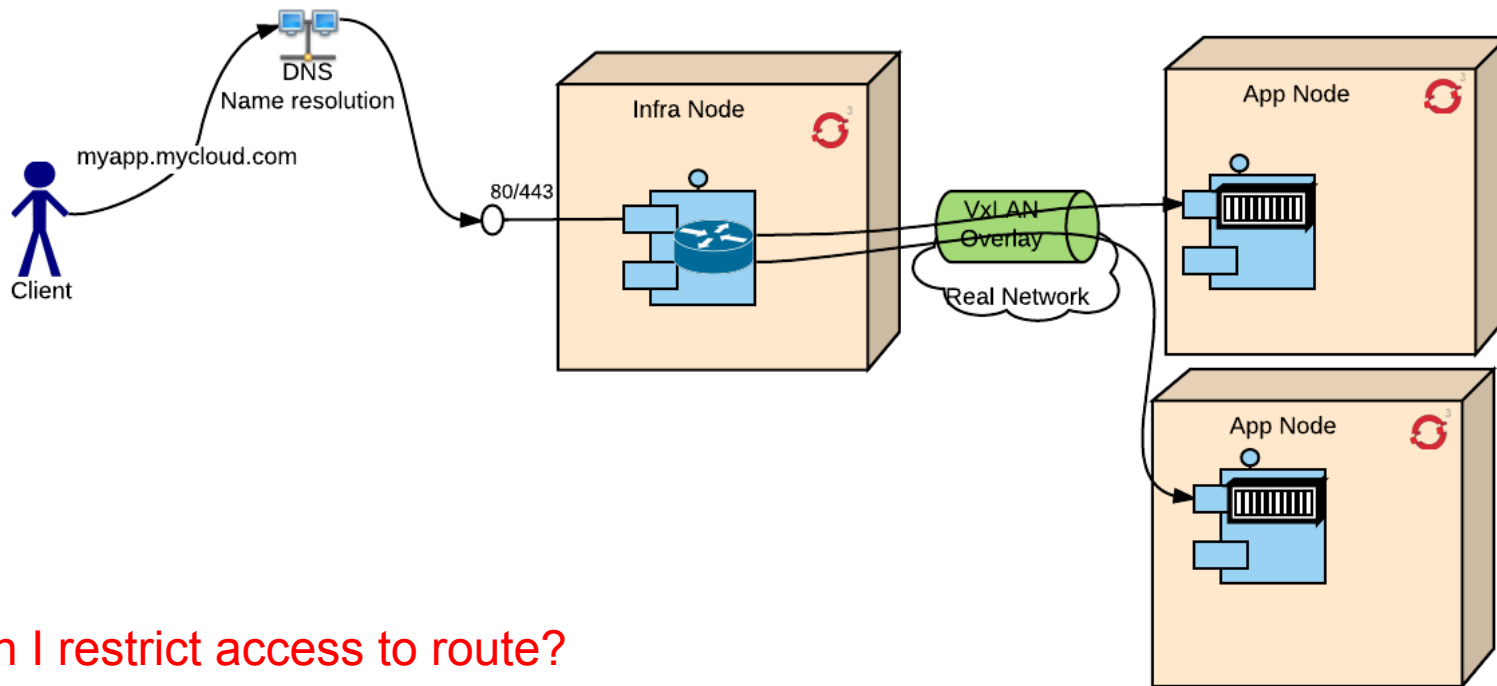A pod can talk to public internet, but cannot connect to hosts (outside OpenShift cluster)

A pod cannot reach specific subnets/hosts

4. Securing Ingress

OpenShift Router as Ingress

Can I restrict access to route?

Route Specific IP Whitelists

- Restrict access to a route to a select IP address(es)
- Annotate the route with the whitelisted/allowed IP addresses
- Connections from any other IPs are blocked

```
metadata:
  annotations:
    haproxy.router.openshift.io/ip_whitelist: 192.168.1.10 192.168.1.11
```

What about ingress traffic on ports that are not 80 or 443?

Using NodePort as Ingress to Service
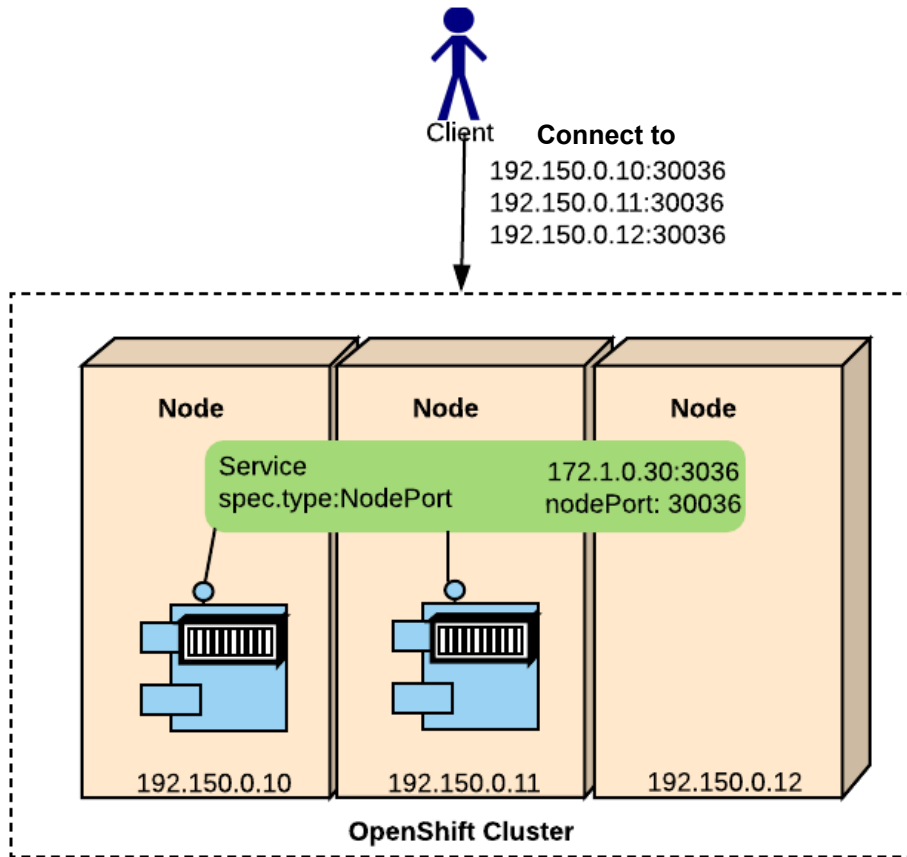
Binds service to a unique port on every node in the cluster

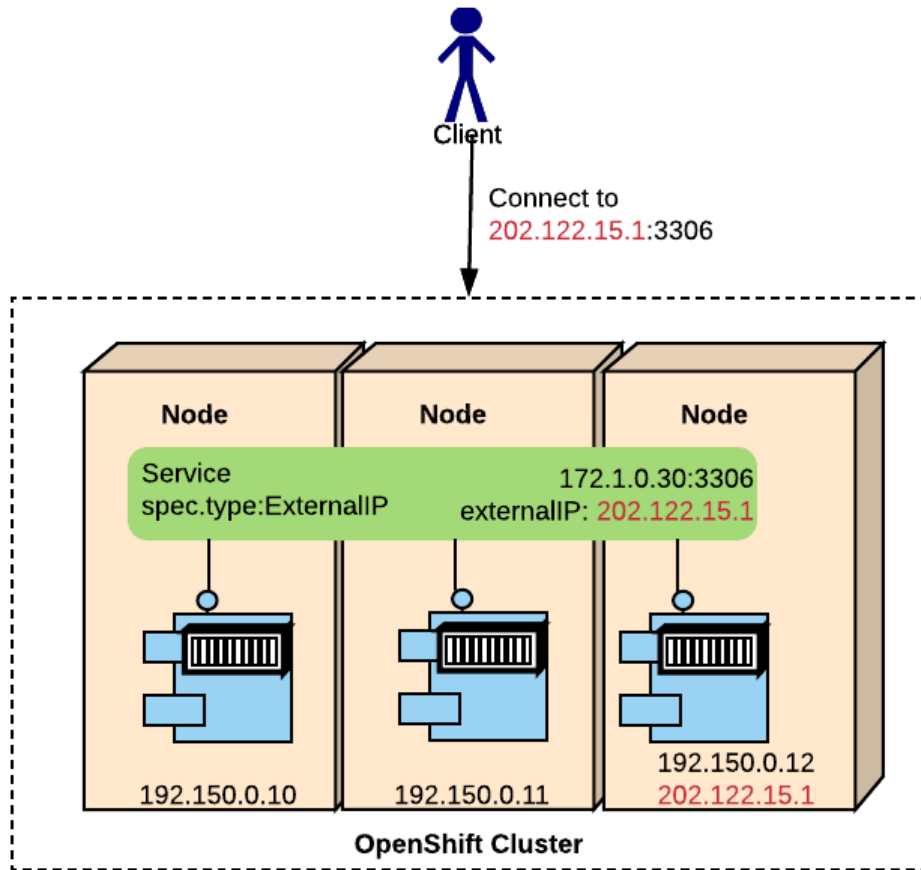Port randomly assigned or optionally picked from port range 30000-32767

All nodes act as ingress point at the port assigned

Every node in the cluster redirects traffic to service service endpoints even if a corresponding pod is not running on that node

Firewall rules should not prevent nodes listening on these ports

Every exposed service uses up a port on all the nodes in a cluster. Are there alternatives?

Assigning External IP to a Service with Ingress

Client

Connect to
202.122.15.1:3306

Node

Service
spec.type:ExternalIP

172.1.0.30:3306
externalIP: 202.122.15.1

Node

Node

192.150.0.10

192.150.0.11

192.150.0.12
202.122.15.1

OpenShift Cluster

Admin defines ExternalIP address range.
Assigns these extra IPs to nodes.

OpenShift assigns both internal IP and external
IP to a service. Or a specific External IP can be
chosen.

Node to which ExternalIP is assigned acts as
the ingress point to the service.

ExternalIP can be a VIP. You can set up
ipfailover to reassign VIP to other nodes.
Ipfailover runs as a privileged pod and handles
VIP assignment.

5. Securing communications between nodes

Secured Communications between Hosts
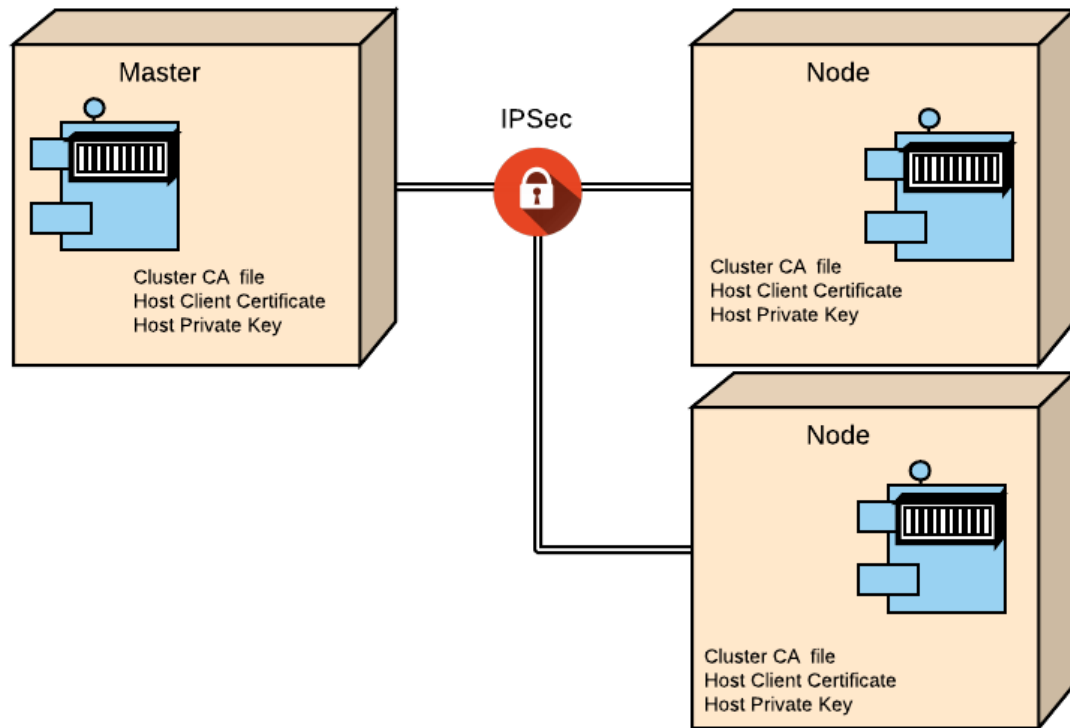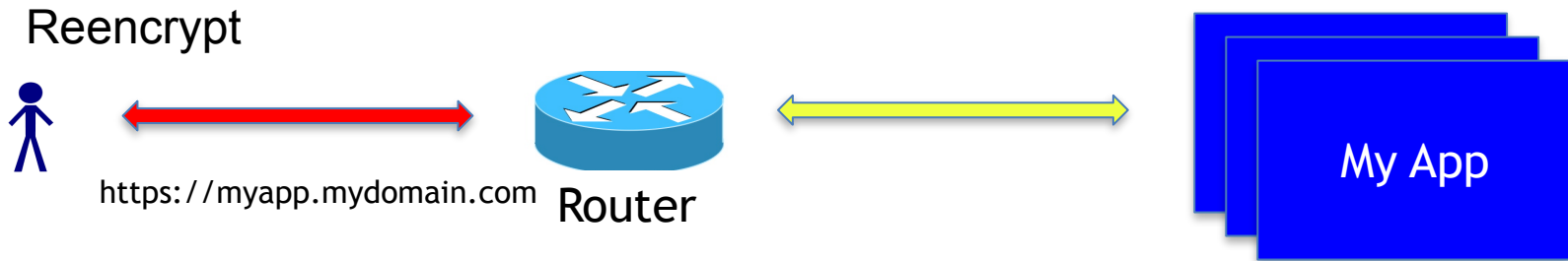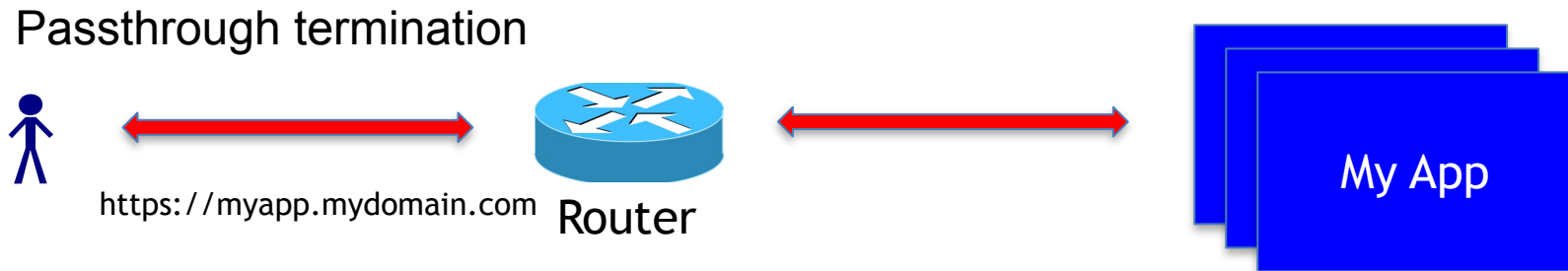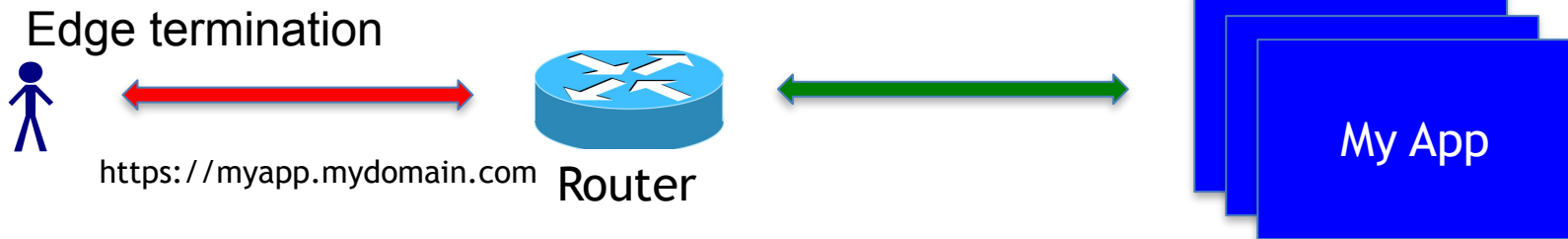

Secures cluster communications with IPsec

- Encryption between all Master and Node hosts (L3)
- Uses OpenShift CA and existing certificates
- Simple setup via policy defn
  - Groups (e.g. subnets)
  - Individual hosts

6. Security at Application Level

SSL at Ingress (with OpenShift Routes)

Edge termination

👤 ⟷ https://myapp.mydomain.com Router ⟷ My App

Passthrough termination

👤 ⟷ https://myapp.mydomain.com Router ⟷ My App

Reencrypt

👤 ⟷ https://myapp.mydomain.com Router ⟷ My App

Layer 7 Application Security

Application specific monitoring East-West
container traffic

Web Application Firewalls

Granular traffic control, Packet Inspections

Denial of Service, Ransomware, Viruses
Detection and Mitigation

Runtime Security, Forensics, Incident
capture, Audits, Alerts

Container runtime monitoring

Partner Solutions
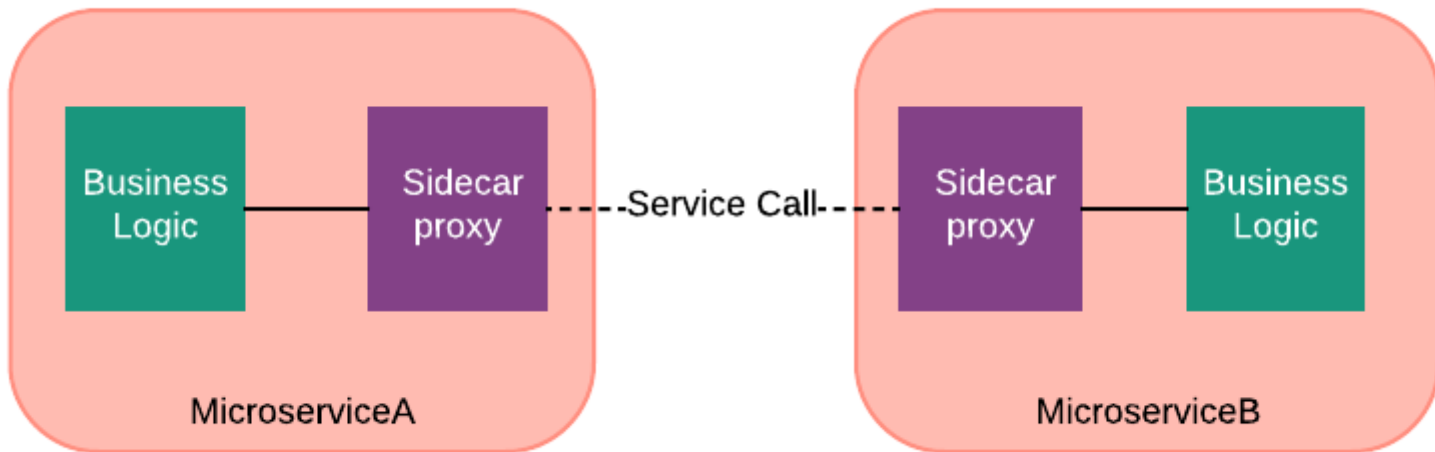
7. (Tech Preview) Application network security with Istio

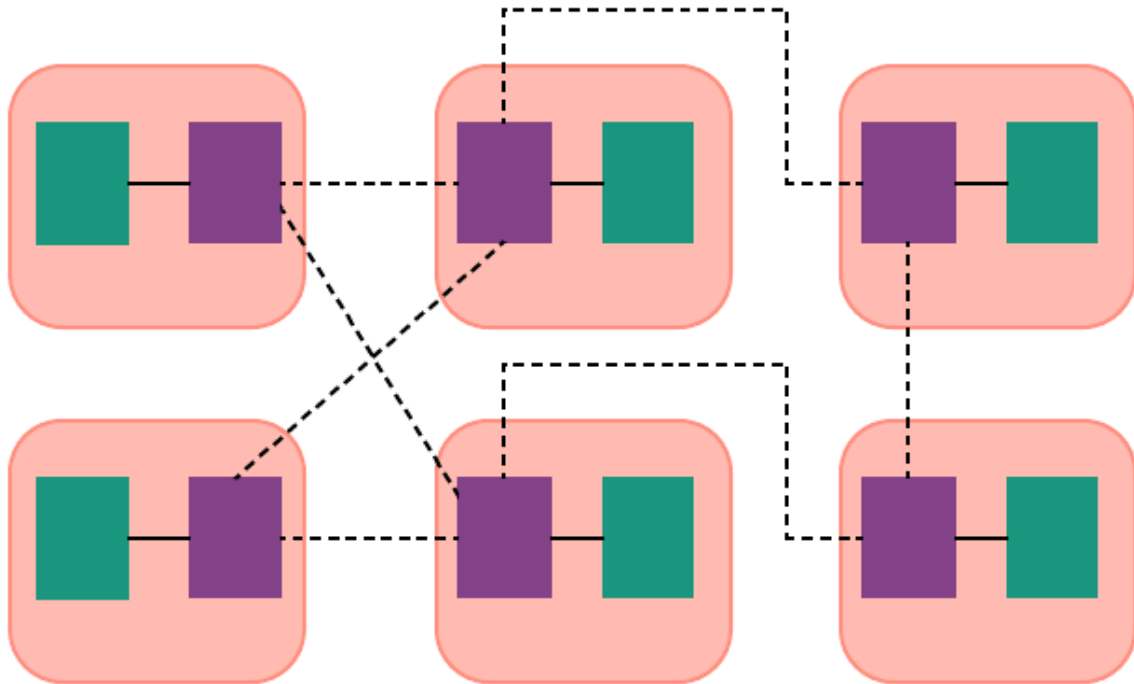# Istio Concepts - Sidecar Proxy

SideCar Proxy
- Intercepts all network communication between microservices
- Encapsulates Service Infrastructure code
- Application code (business logic) unaware of Sidecar proxy
- Examples - Linkerd, Envoy

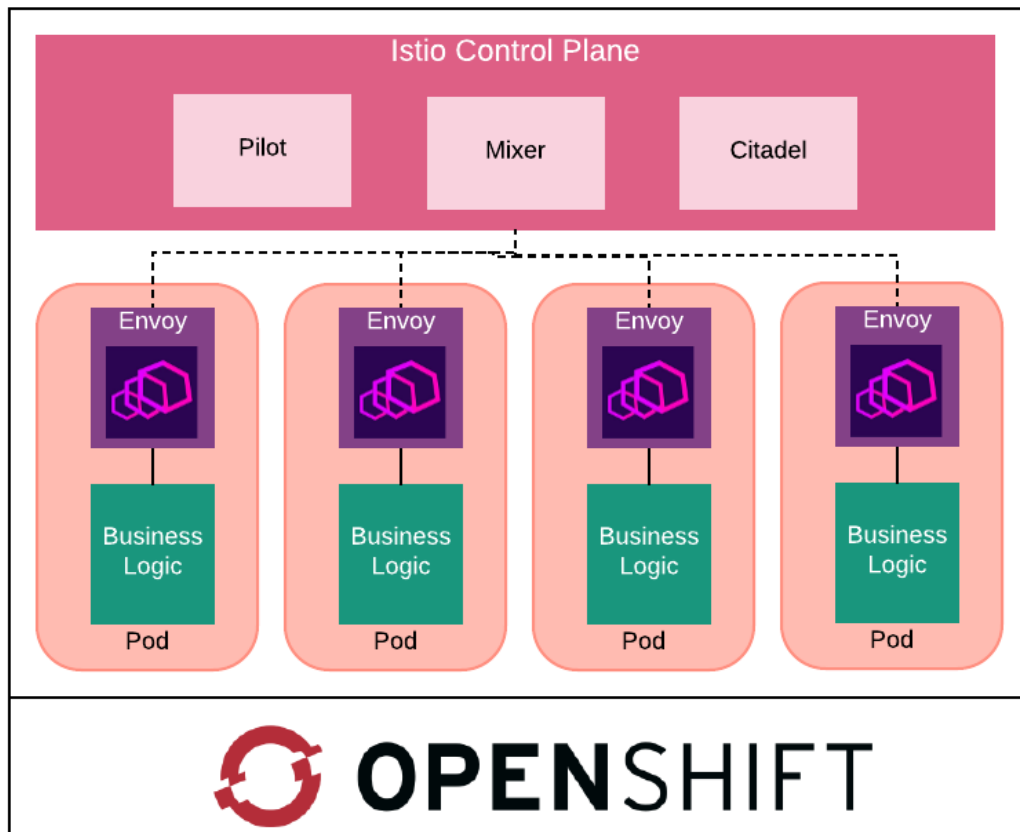# Istio Concepts - Service Mesh

Network of Microservices

Service Mesh is a dedicated infrastructure layer to handle service-service communications

Typically implemented as an array of lightweight network proxies deployed alongside application code

Interconnected Proxies form a mesh network

Istio Service Mesh on OpenShift

Connect, Manage, and Secure Microservices, transparently

- Intelligent Routing
- Load Balancing
- Service Resilience
- Telemetry and Reporting
- Policy Enforcement
- Content based Filtering (Layer 7)
- mTLS between services
- East-West traffic control

Application Traffic Encryption with Istio Auth (Future)
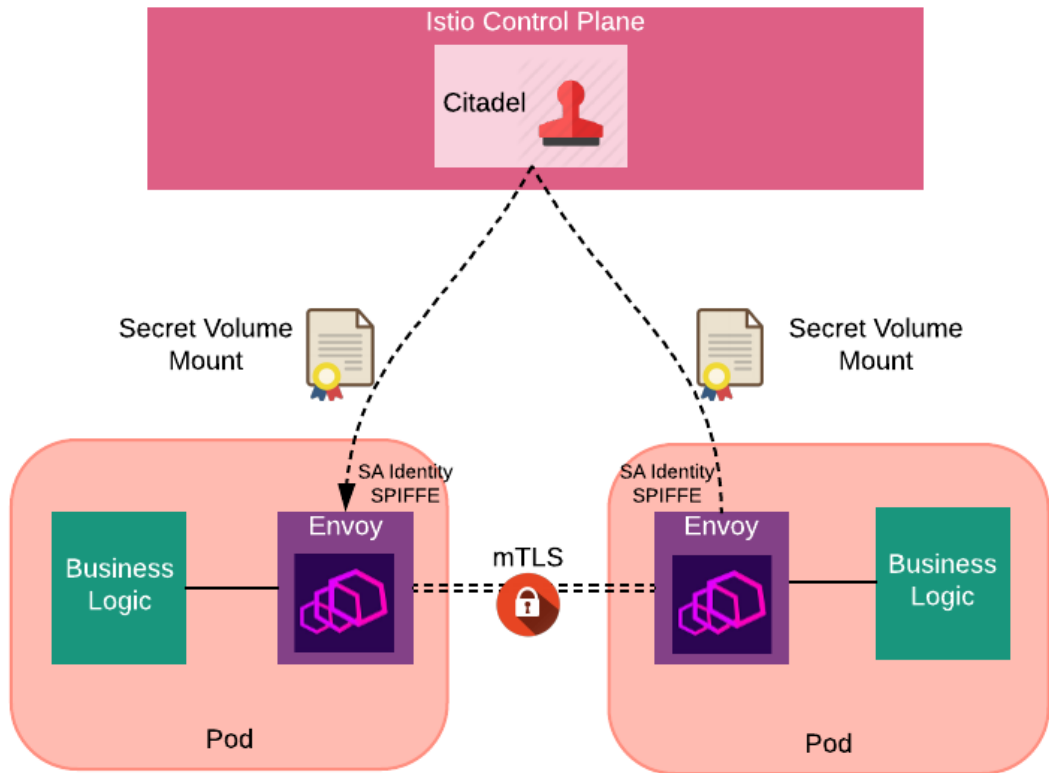
Uses Service Account as Identity.
SPIFFE Id format

spiffe://*<domain>*/ns/*<namespace>*/sa/*<serviceaccount>*

Mutual TLS between sidecars

Istio CA

- Generate cert pair and SPIFFE key for each SA
- Distribute key and cert pairs
- Rotate keys and certs periodically
- Revoke key and cert when need

Questions?
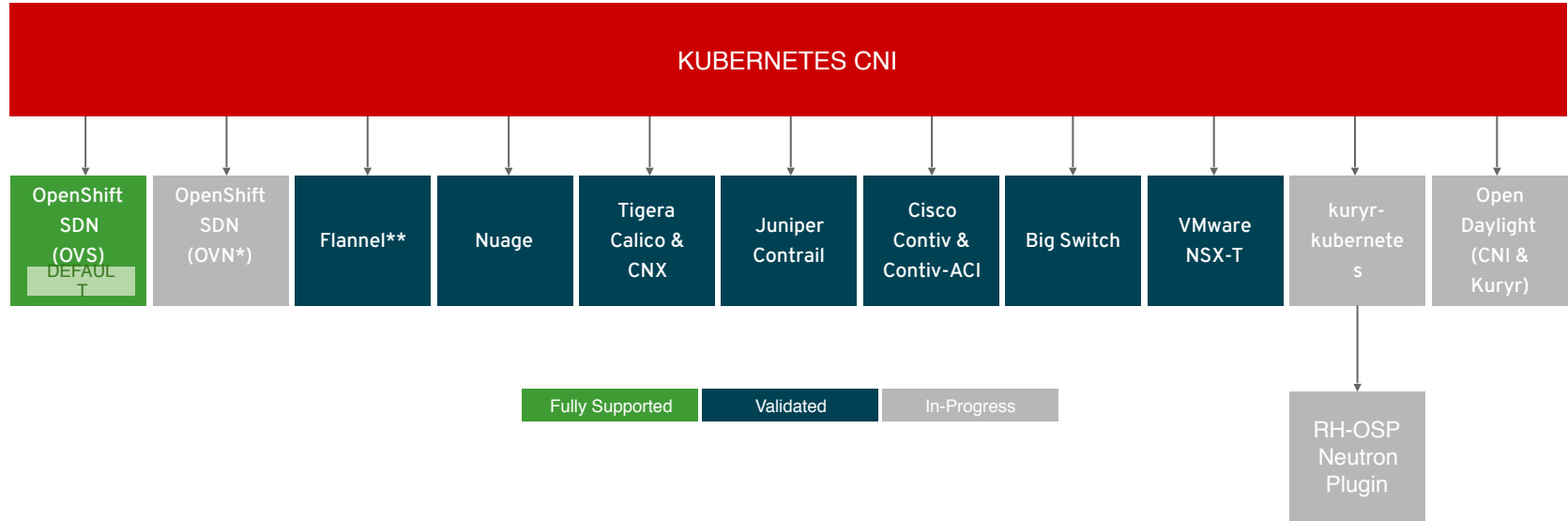
THANK YOU

#SecuritySymposium

OpenShift SDN Overview

#SecuritySymposium

Kubernetes uses CNI

KUBERNETES CNI

| OpenShift SDN (OVS) DEFAULT | OpenShift SDN (OVN*) | Flannel** | Nuage | Tigera Calico & CNX | Juniper Contrail | Cisco Contiv & Contiv-ACI | Big Switch | VMware NSX-T | kuryr-kubernetes | Open Daylight (CNI & Kuryr) |

Fully Supported | Validated | In-Progress

RH-OSP Neutron Plugin

* Coming as default in OCP 4.1

** Flannel is minimally verified and is supported only and exactly as deployed in the OpenShift on OpenStack reference architecture

OpenShift Networking

Software Defined Networking (SDN) for pod-pod communication

- Configures overlay network using Open vSwitch (OVS)
- Three types of plugins
    - **ovs-subnet** : flat network every pod can talk to every other pod
    - **ovs-multitenant**: project level isolation for pod-pod communication. Unique VNID per project

        You can join projects to get them the same VNID

        'default' project (VNID 0) privileged to communicate with other pods

    - **ovs-networkpolicy**: fine-grained isolation using network policy objects

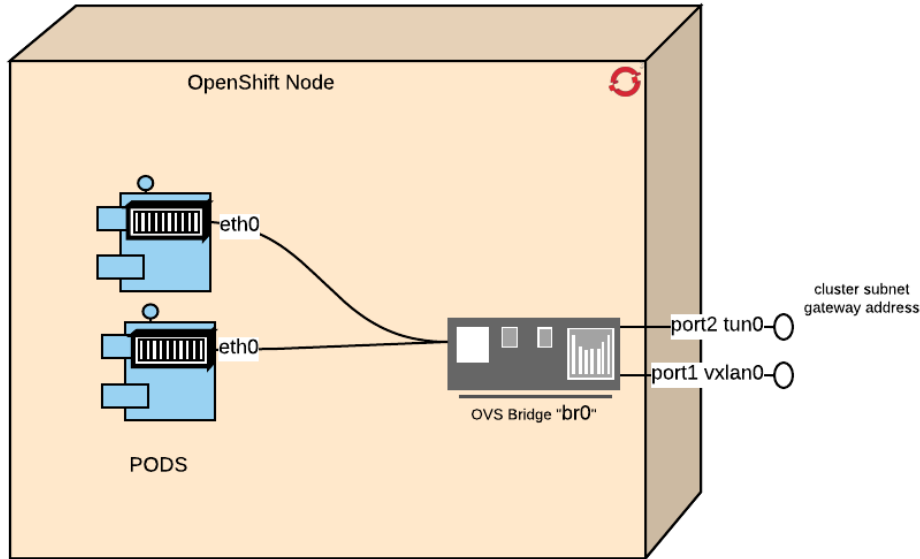# OpenShift SDN manages Node Registry
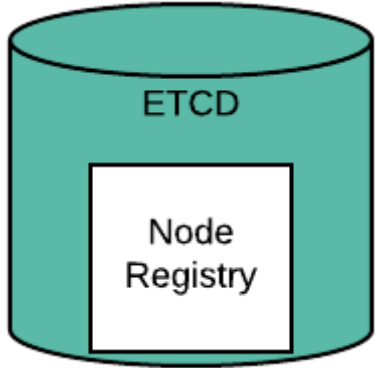
Master allocates a subnet to the node.

Node creation - Allocated subnet added to Node Registry

Node deletion - subnet removed from the Node Registry

On node creation,SDN registers the host with the SDN master

OpenShift SDN configures network devices on Node

br0 Pod containers attached to this ovs bridge device. Non subnet specific flow rules on br0
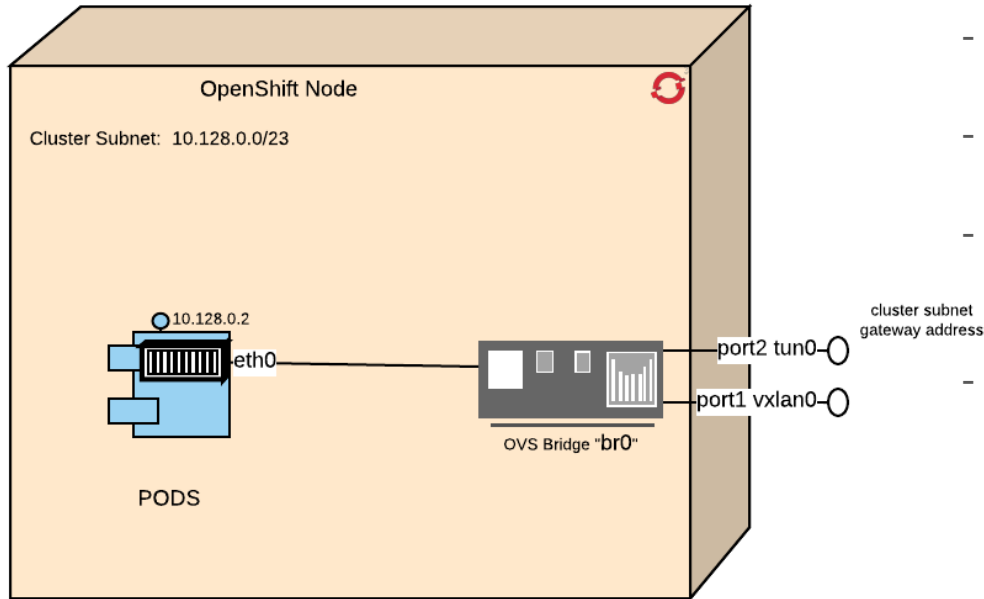
tun0 For external network access via NAT. Cluster subnet gateway address assigned. Configures netfilter and routing rules.

vxlan0 Access to other nodes. OVS VxLAN device

additional node added:

- Watch subnet updates from master
- Add OpenFlow rules on br0 to push traffic to the newly added subnet go to vxlan0

OpenShift SDN Pod Creation

- Assigns an available ip address from the node's cluster subnet to the pod
- Attaches host side of pod's veth interface pair to br0
- Adds OpenFlow rules to OVS DB to route traffic addressed to the new pod to correct OVS port
- For ovs-multitenant, adds OpenFlow rules
  - to attach pod's VNID to outgoing traffic
  - allow traffic to pod when VNID matches
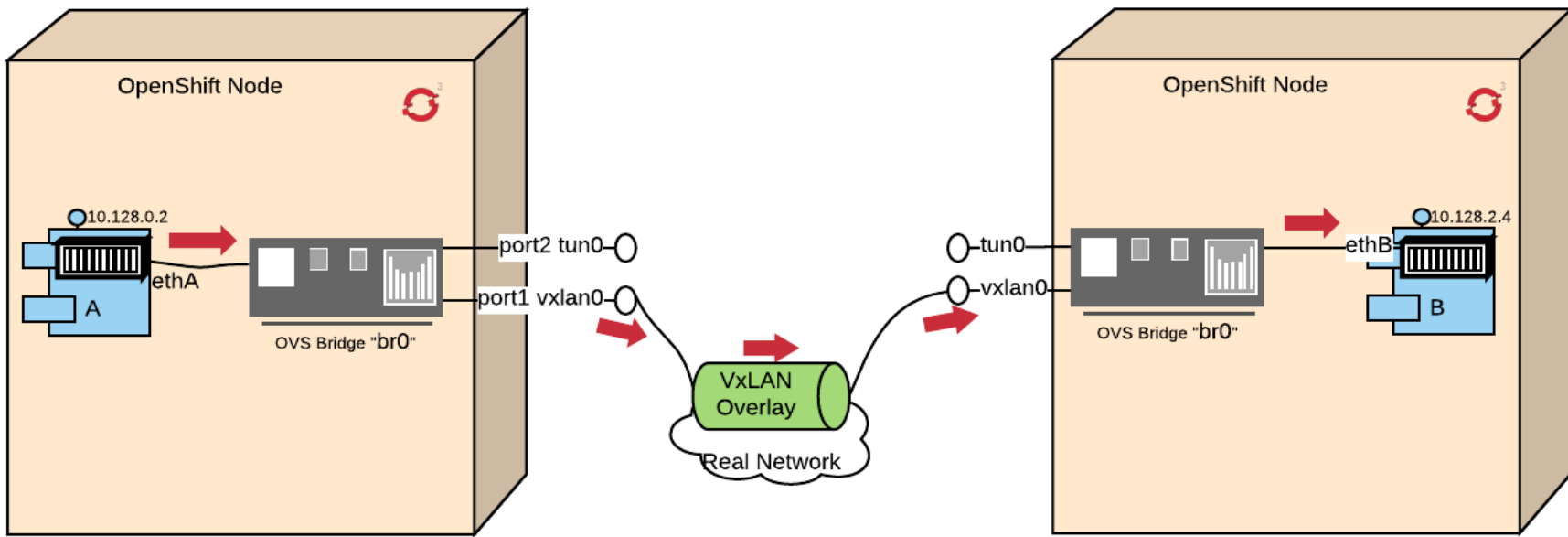
Pod to Pod Traffic - Both pods on the same Node

Flow of traffic

eth0(in A's netns) - vethA - br0 - vethB - eth0(in B's netns)
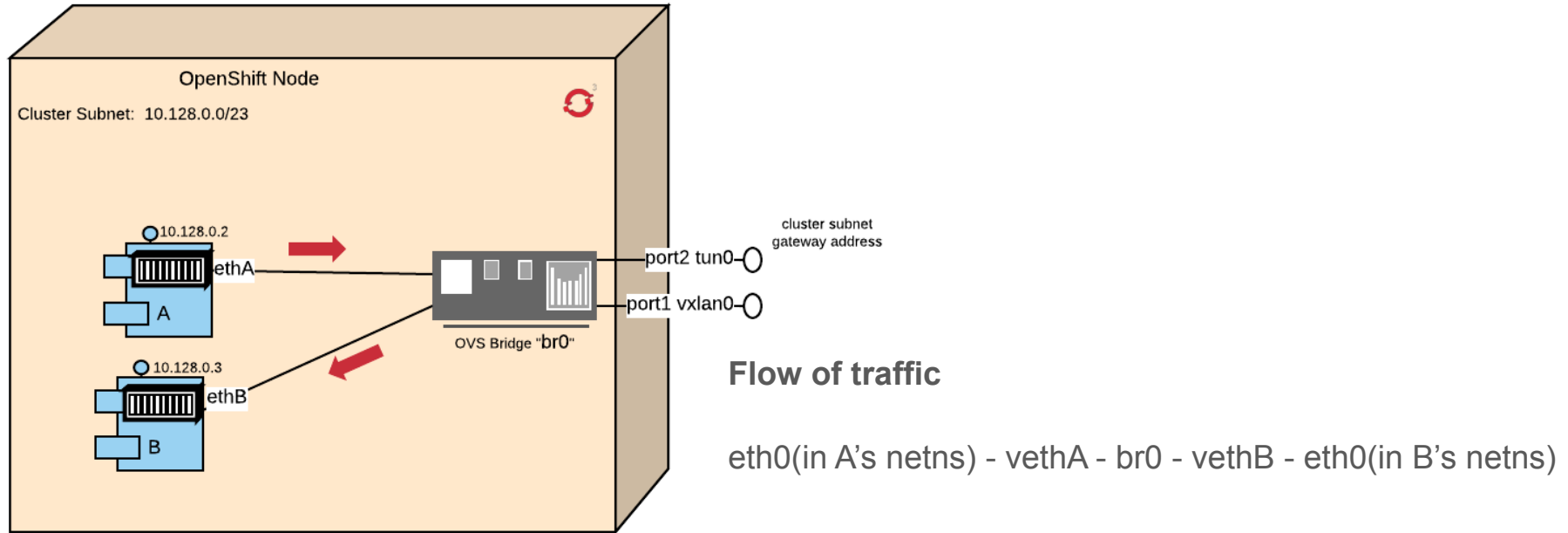
* Peer vEthernet device for container A is named ethA and for container B is named ethB

Pod to Pod Traffic - Pods on two different Nodes

Flow of traffic

eth0(in A's netns) - vethA - br0 - vxlan0 - network - vxlan0 - br0- vethB - eth0(in B's netns)

* Peer vEthernet device for container A is named ethA and for container B is named ethB

Pod to External Systems outside OpenShift

Flow of traffic

eth0(in A's netns) - vethA - br0 - tun0 - (NAT) - eth0(physical device) - Internet