You make **possible**

OpenShift and Cisco ACI Integration

ACI CNI Plugin for OpenShift

Domenico Dastoli – TME INSBU

BRKACI-3330

Barcelona | January 27-31, 2020

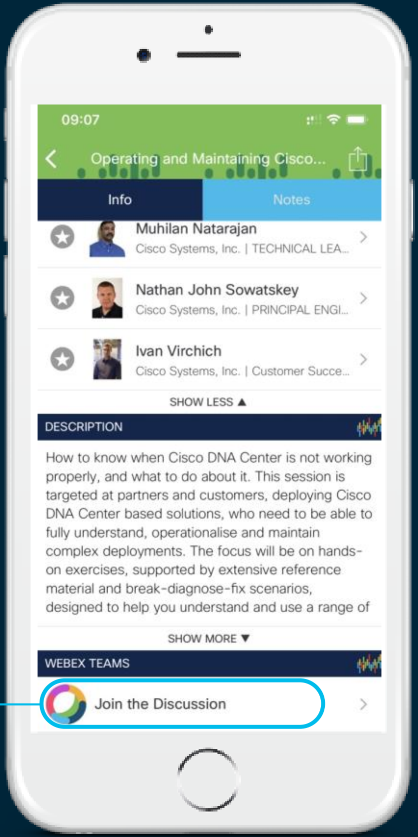
Cisco Webex Teams

Questions?

Use Cisco Webex Teams to chat
with the speaker after the session

How

1. Find this session in the Cisco Events Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space



09:07

Operating and Maintaining Cisco...

Info | Notes

Muhilan Natarajan
Cisco Systems, Inc. | TECHNICAL LEA...

Nathan John Sowatskey
Cisco Systems, Inc. | PRINCIPAL ENGI...

Ivan Virchich
Cisco Systems, Inc. | Customer Succe...

SHOW LESS ▲

DESCRIPTION

How to know when Cisco DNA Center is not working properly, and what to do about it. This session is targeted at partners and customers, deploying Cisco DNA Center based solutions, who need to be able to fully understand, operationalise and maintain complex deployments. The focus will be on hands-on exercises, supported by extensive reference material and break-diagnose-fix scenarios, designed to help you understand and use a range of

SHOW MORE ▼

WEBEX TEAMS

Join the Discussion

Agenda

- OCP Network Fundamentals and challenges
- ACI and OpenShift: Better together
- ACI and OpenShift:
  - Installation
  - Components
  - Use cases and Demos

Why ACI for Application Container Platforms

Turnkey solution for node and container connectivity

Flexible policy: Native platform policy API and ACI policies

Hardware-accelerated: Integrated load balancing

Visibility: Live statistics in APIC per container and health metrics

Enhanced Multitenancy and unified networking for containers, VMs, bare metal

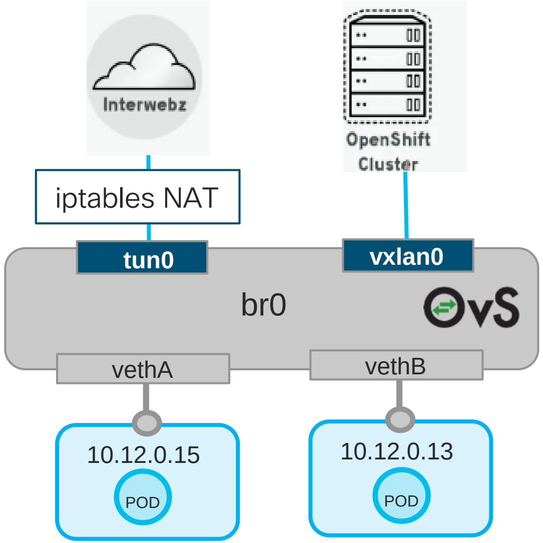
Fast, easy, secure and scalable **networking** for your Application Container Platform

Red Hat OCP Network Fundamentals

OpenShift Native Networking Details*

OPENSHIFT

- OpenShift uses Open vSwitch and can leverage VXLAN for POD to POD communication.
- POD vEth is added to a OVS bridge
- POD to external uses a Tun0 interface and NAT



(*) This is the most common standard design with Red Hat OpenShift

OpenShift Networking

- 3 Native Networking modes:
  - ovs-subnet: "flat" pod network where every pod can communicate with every other pod and service.
  - ovs-multitenant: project-level isolation for pods and services.
  - ovs-networkpolicy: project administrators to configure their own isolation policies using NetworkPolicy objects.
- Or delegates to 3rd party SDN with **CNI Plugin → for example, ACI CNI Plugin**

OpenShift supports Kubernetes Service options

- POD to POD via **ClusterIP** Service
- Exposing services externally:
  - Using Node IP via **NodePortIP**
  - Using external **LoadBalancer**
- **Routes** allow exposing services outside of the cluster, similar to ingress controller on Kubernetes:
  - Services exposed with a “route” or URL which can be easily mapped to wildcard DNS entry: all FQDN matching that wildcard will be treated by the OCP route and LB to the appropriate service.

Red Hat OCP ‘Native’ Network Challenges

Operations and Visibility

- Skills gap between network and OCP admins
- Visibility and governance of network policies
- Simplified Network Operations

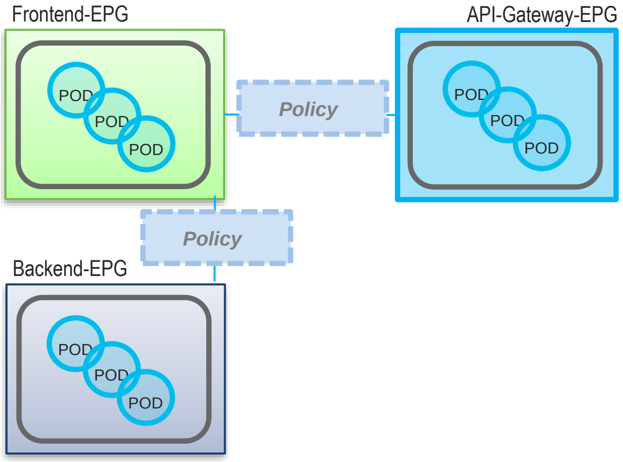
Developer

Infosec

Network Administrator

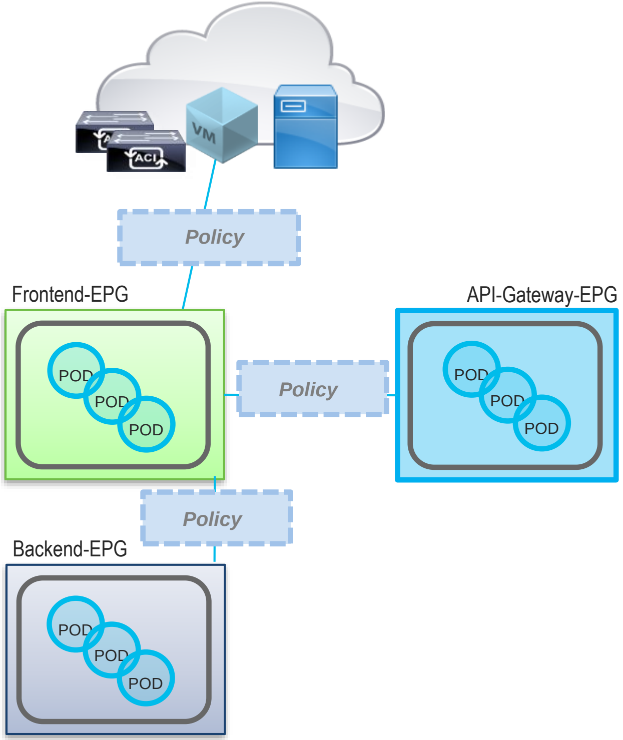
Segmentation

- Secure K8s **infrastructure**:
  - network isolation for kube-system and other infrastructure related objects (i.e. heapster, hawkular, etc.)
- Network isolation between **namespaces**



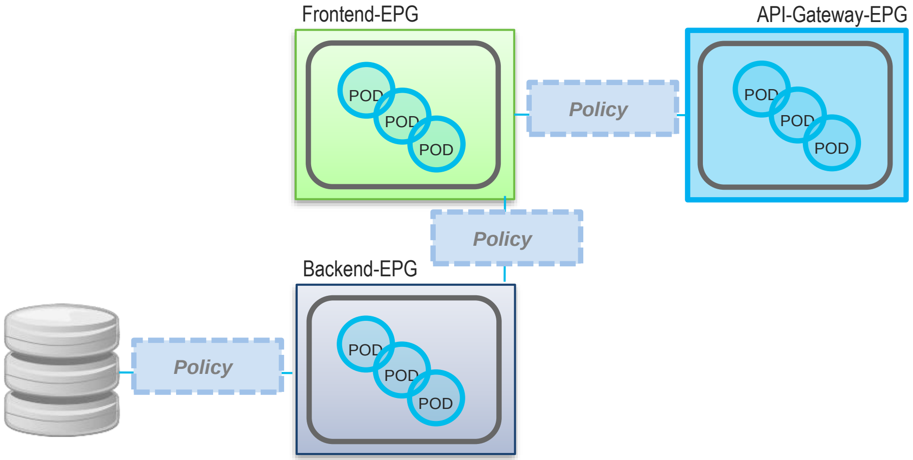
Communications outside of the Cluster

- Non-Cluster endpoints communicating with Cluster:
  - Exposing external services, how? NodePort? LoadBalancer?
  - Scaling-out ingress controllers, how can you scale?
- Cluster endpoints communicating with non-cluster endpoints:
  - POD access to external services and endpoints



Storage Access from Nodes

- Applications running in OpenShift that need high-bandwidth, low-latency traffic to data external to the cluster suffer the bottleneck imposed by the egress router implementation. i.e. centralized storage from node or PODs:
  - iSCSI, NFS, GlusterFS, CEPH, etc.
  - HyperFlex



ACI and OpenShift better Together!

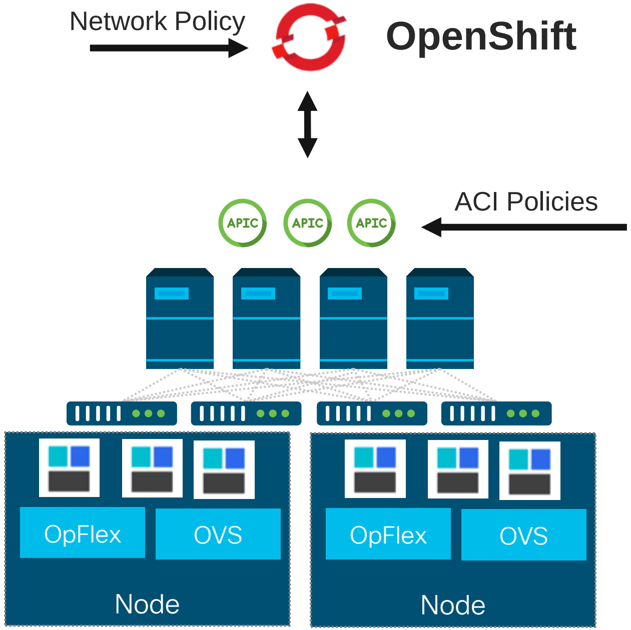
OpenShift supports CNI Plugins. There are multiple CNI plugins available. Including ACI CNI Plugin.

Cisco ACI CNI Plugin Benefits

1. **Simplified Operations and Enhanced visibility**
2. **Granular security**: security can be implemented by using native NetworkPolicy or by using ACI EPGs and contracts, or both models complementing each other.
3. **Unified networking**: Pod and Service endpoints become first class citizens at the same level as Bare Metal or Virtual Machines.
4. **High performance**: low-latency secure connectivity without egress routers
5. **Hardware-assisted load balancing**: ingress connections to LoadBalancer-type services using ACI's Policy Based Redirect technology

Cisco ACI CNI plugin features

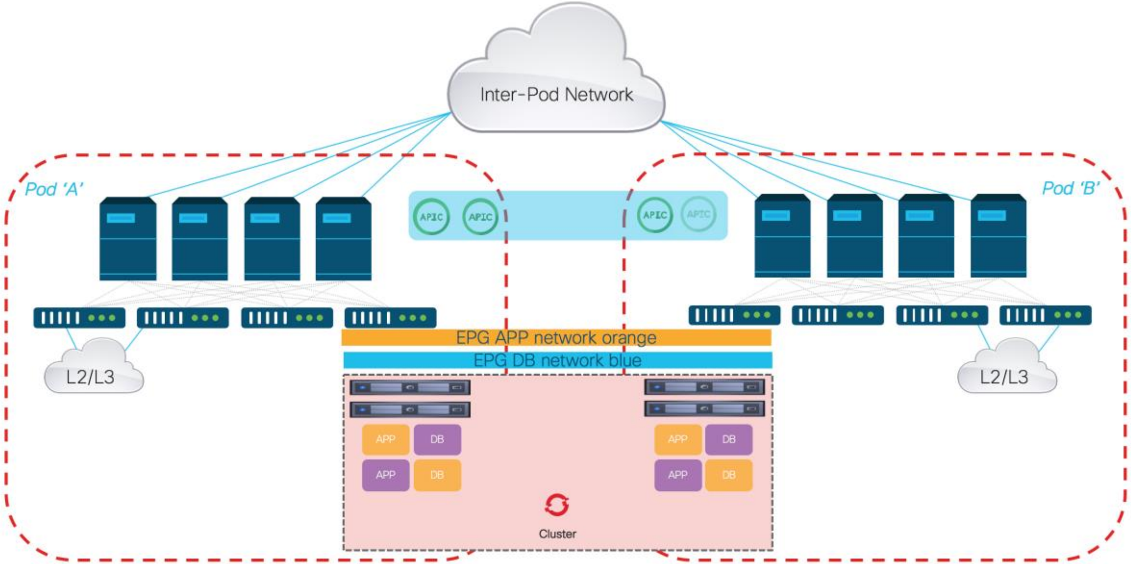
- IP Address Management and SNAT* (optional) for Pods and Services
- Distributed Routing and Switching with integrated VXLAN overlays implemented fabric wide and on Open vSwitch
- Distributed Firewall for Network Policies
- EPG-level segmentation for OCP objects using annotations
- Consolidated visibility of OCP networking via VMM Integration



*supported as from CNI release 4.2(x)

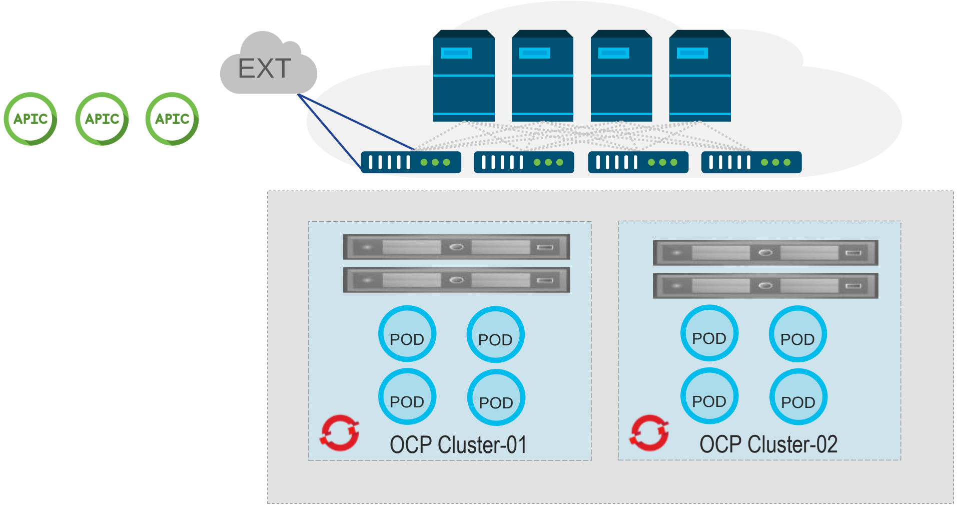
MultiPod and OpenShift

- It is possible to seamlessly extend the OpenShift cluster across different data centers, both increasing redundancy and allowing disaster recovery scenarios.



Multiple OpenShift Cluster on same ACI fabric

- You can have multiple OpenShift Clusters on the same ACI fabric, i.e. Production and Testing etc.

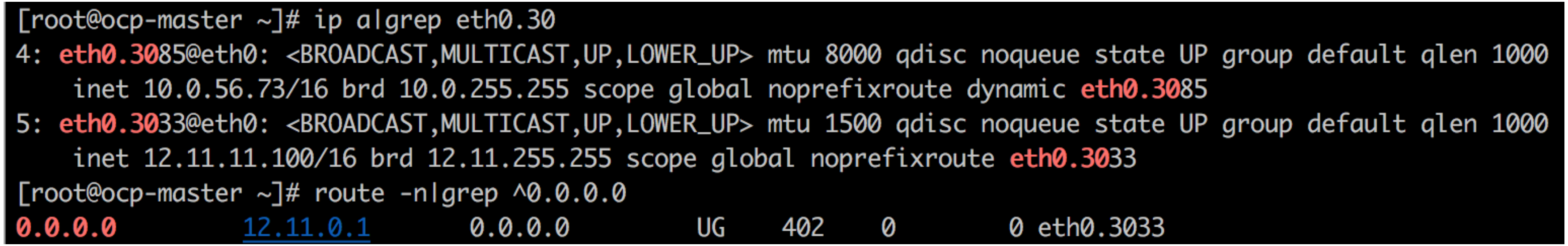


OpenShift nodes to ACI Connectivity Requirements

Pre reqs Kubernetes Nodes sub-interfaces

Manual config

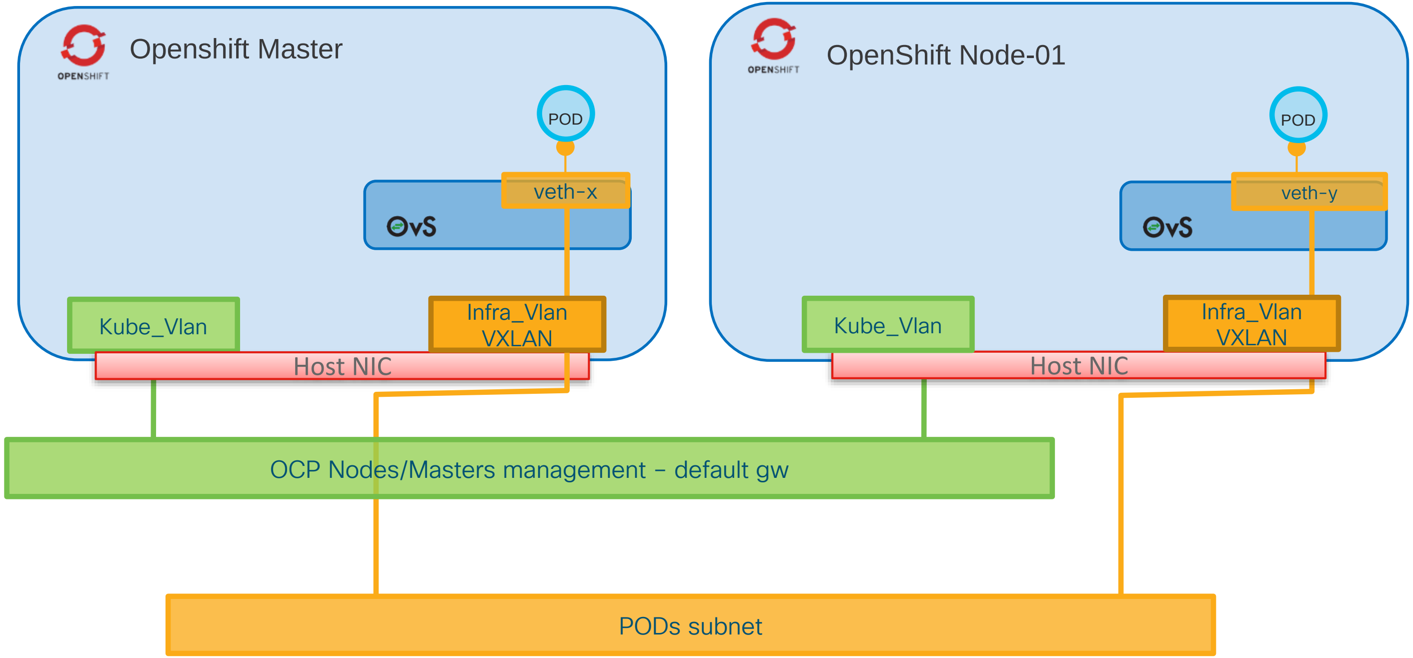
- **Infra VLAN** – OpFlex channel and Container Data Path
- **Kube-API VLAN** – Kubernetes API host IP address



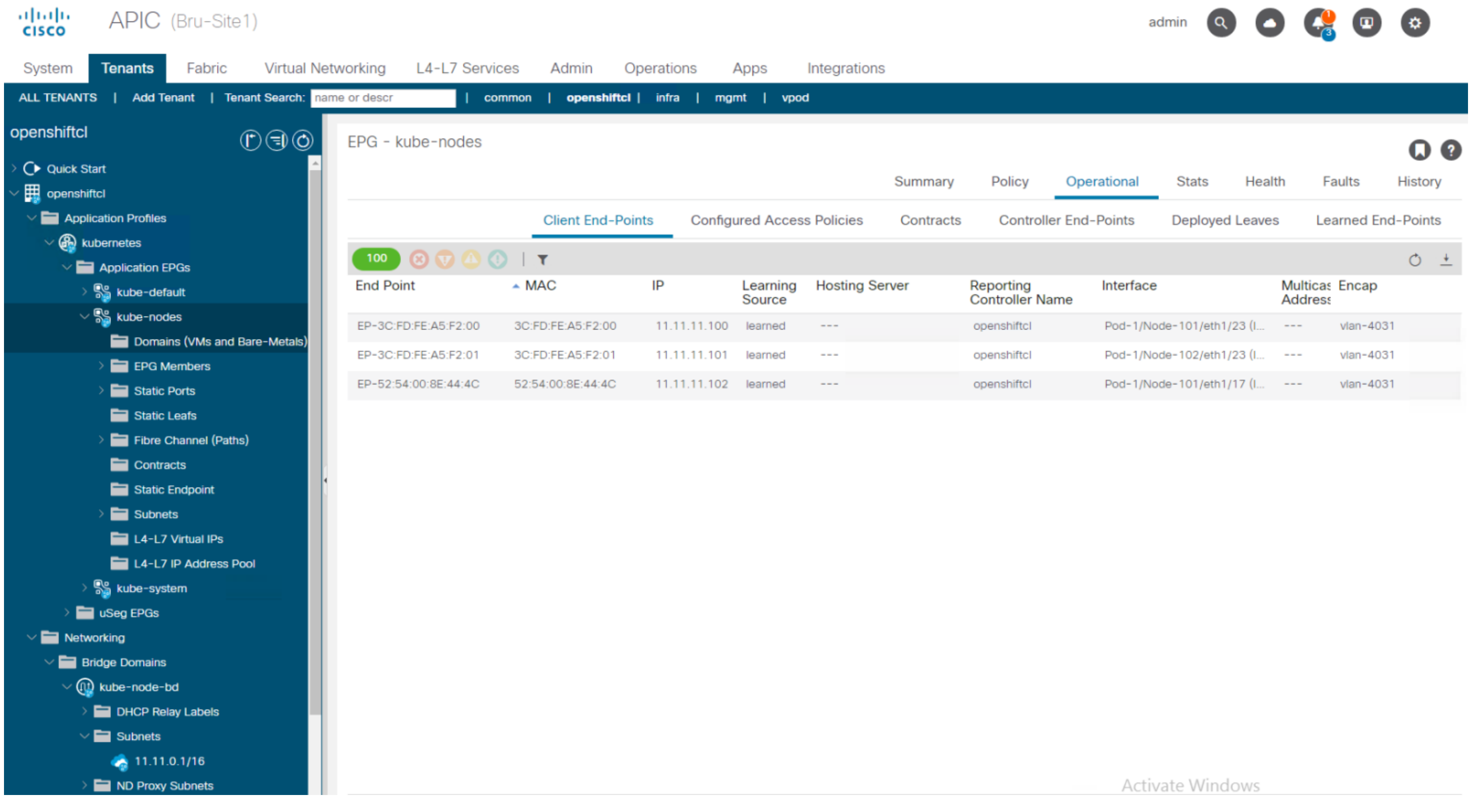
```
[root@ocp-master ~]# ip a|grep eth0.30
4: eth0.3085@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8000 qdisc noqueue state UP group default qlen 1000
    inet 10.0.56.73/16 brd 10.0.255.255 scope global noprefixroute dynamic eth0.3085
5: eth0.3033@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    inet 12.11.11.100/16 brd 12.11.255.255 scope global noprefixroute eth0.3033
[root@ocp-master ~]# route -n|grep ^0.0.0.0
0.0.0.0         12.11.0.1       0.0.0.0         UG    402    0        0 eth0.3033
```

- Default route goes through the Kube-API VLAN
- The Infra VLAN is used to encapsulate VXLAN traffic of the PODs

Kubernetes Nodes connectivity



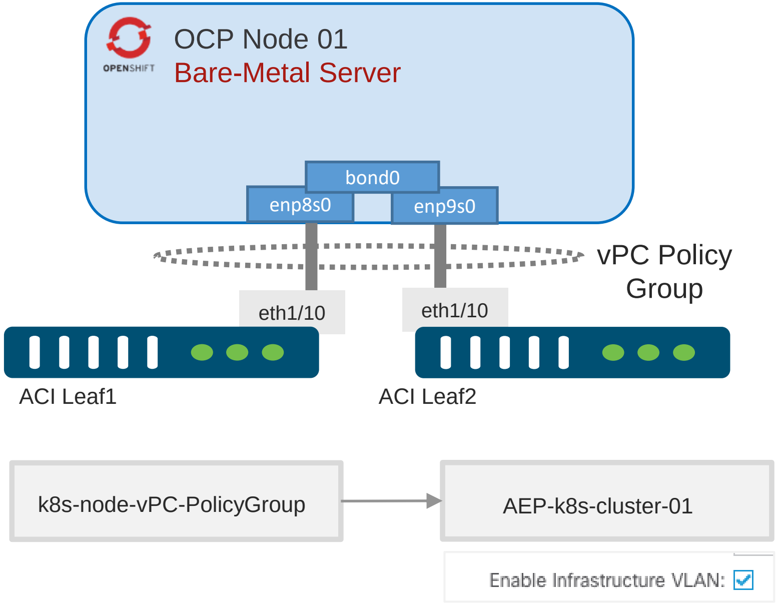
Visibility of OCP nodes in ACI



End Point	MAC	IP	Learning Source	Hosting Server	Reporting Controller Name	Interface	Multicas Address	Encap
EP-3C:FD:FE:A5:F2:00	3C:FD:FE:A5:F2:00	11.11.11.100	learned	---	openshiftcl	Pod-1/Node-101/eth1/23 (I...	---	vlan-4031
EP-3C:FD:FE:A5:F2:01	3C:FD:FE:A5:F2:01	11.11.11.101	learned	---	openshiftcl	Pod-1/Node-102/eth1/23 (I...	---	vlan-4031
EP-52:54:00:8E:44:4C	52:54:00:8E:44:4C	11.11.11.102	learned	---	openshiftcl	Pod-1/Node-101/eth1/17 (I...	---	vlan-4031

Nodes as Bare-Metal Servers

- Virtual Port Channels enable simple and fast link redundancy for Kubernetes bare metal nodes
- Can use standard based LACP between K8s nodes and leaf pair for optimal load balancing and link-failure convergence

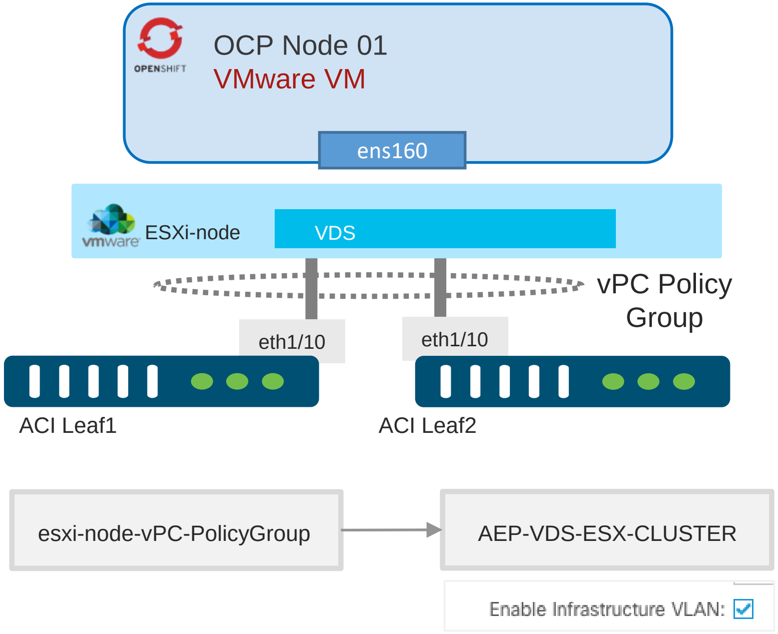


Enable Infrastructure VLAN: ☑

Make sure InfraVLAN is enabled for AEP

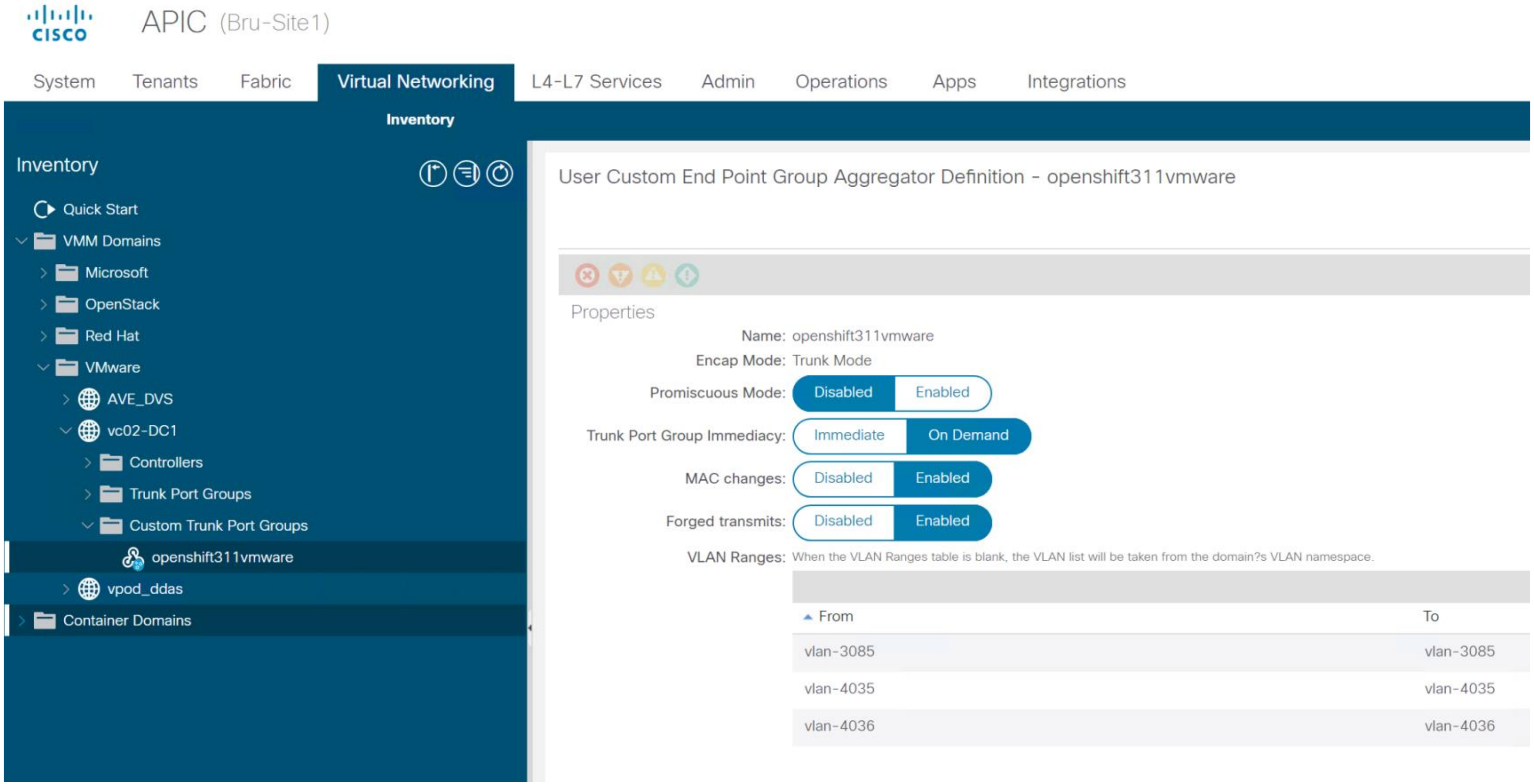
Nodes as VMware Virtual Machines

- The VMs can be running on a VMware VMM Domain.
- The VMs will be connected to a PortGroup that will be created by the ACI CNI installer tool.



Make sure InfraVLAN is enabled for AEP

Node port group shows as Custom Trunk Port*



User Custom End Point Group Aggregator Definition - openshift311vmware

Properties

Name: openshift311vmware

Encap Mode: Trunk Mode

Promiscuous Mode: [✓ Disabled] Enabled

Trunk Port Group Immediacy: Immediate [✓ On Demand]

MAC changes: Disabled [✓ Enabled]

Forged transmits: Disabled [✓ Enabled]

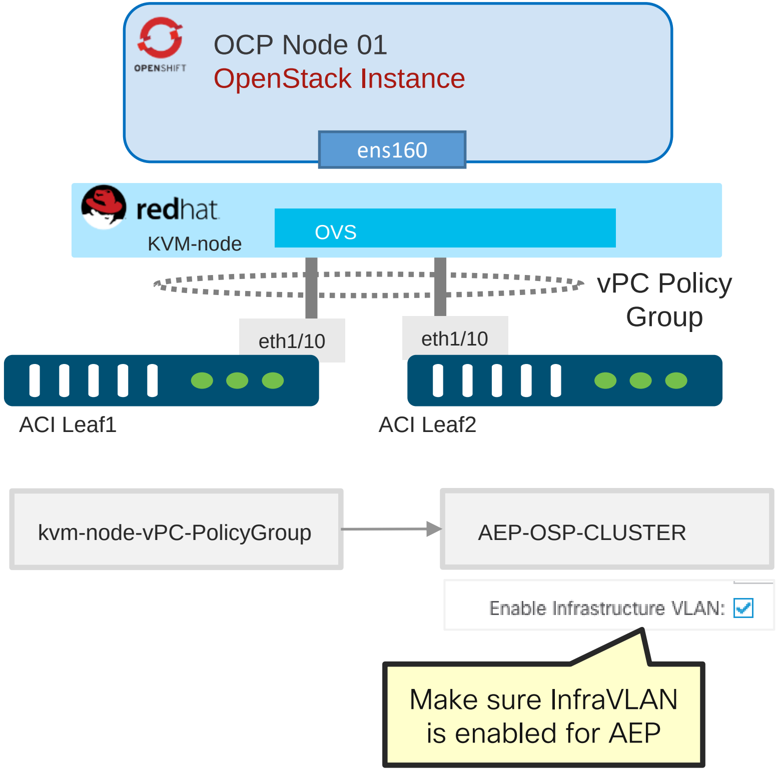
VLAN Ranges: When the VLAN Ranges table is blank, the VLAN list will be taken from the domain?s VLAN namespace.

From	To
vlan-3085	vlan-3085
vlan-4035	vlan-4035
vlan-4036	vlan-4036

*with VCenter VMM Domain

Nodes as RH OSP Virtual Machines

- As of ACI 3.2(x) it is supported to also run the ACI CNI plugin for clusters running on Red Hat OpenStack VMs.
- The VMs should be running on a OpenStack VMM Domain.



Enable Infrastructure VLAN: ☑

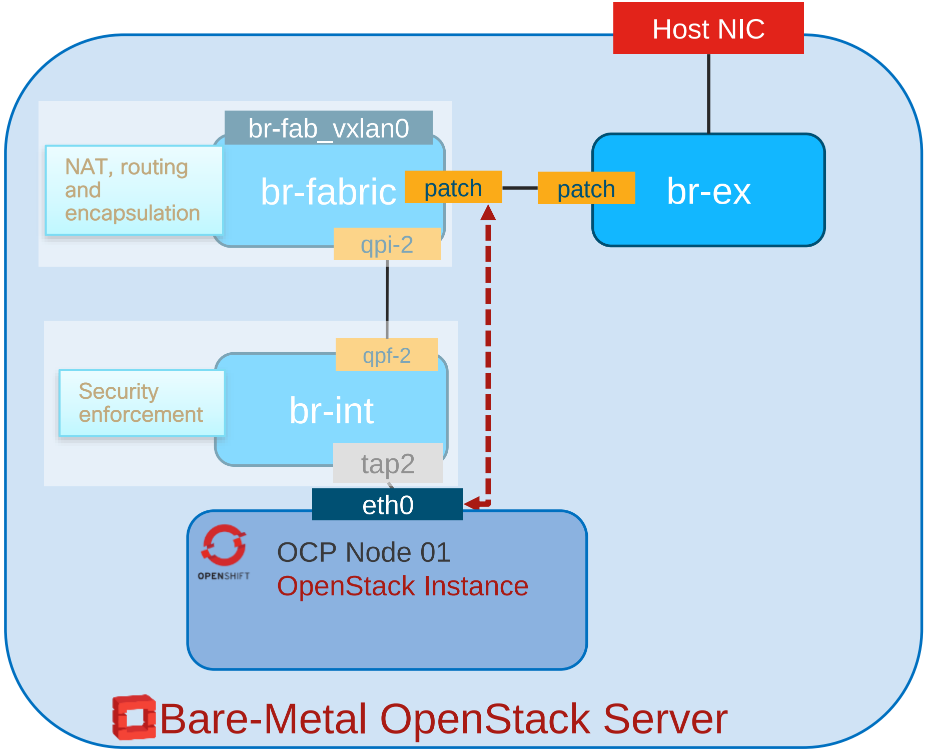
Make sure InfraVLAN is enabled for AEP

OCP on OSP: Key Differentiators

- OpenStack and OpenShift network policy configurations are orthogonal
  - changes to one do not affect the other
  - e.g. no OpenStack Security Groups are required to allow OpenShift traffic
  - APIC serves as the normalization point for common policy and provides one place for administering it
- Unlike other nested solutions which do double encapsulation (like Flannel over Neutron), the data path here preserves single encapsulation
- Container networking is a first class citizen (seen as ACI EP) even though the containers are nested inside the VM

OVS bridge patching for OpenShift on OpenStack

- Traffic destined to the OCP node bypasses the OV bridges for the 'standard' OpenStack traffic.
- Bypassing any OSP rule allows avoiding double encapsulation



How do you Install OpenShift nodes with ACI CNI Plugin?

3 Simple Steps!

- Define an L3out to be used by OCP
- Run acc-provision tool
- Use Standard means to install OCP

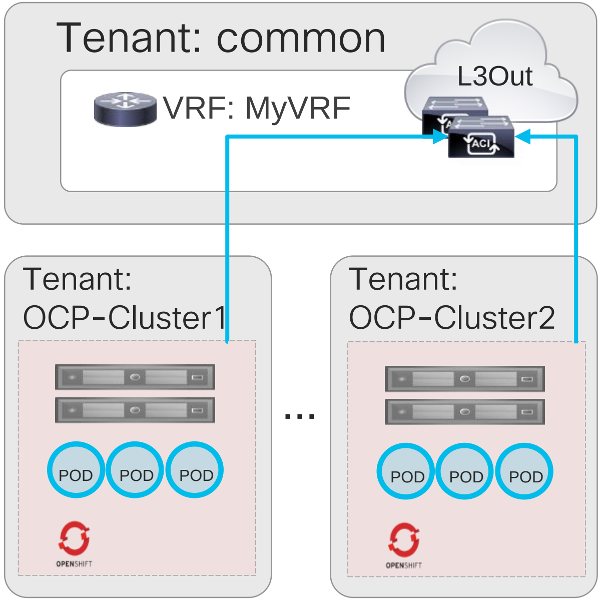
Step 1

ACI Tenant External Connectivity for OCP Cluster

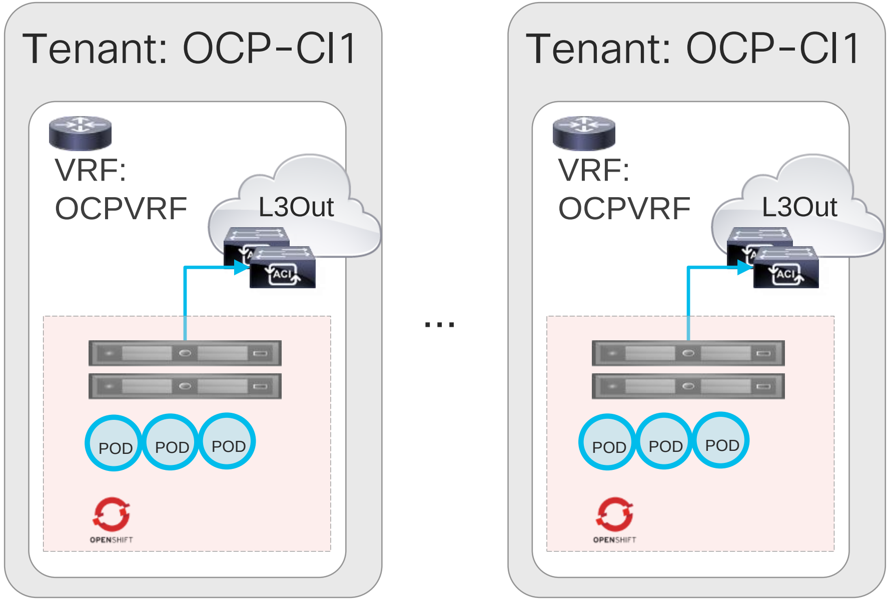
- Fabric Administrator must define the L3OUT that will be used to:
  - Connect OCP Nodes and Pods to Internet
  - Expose external services
- The L3OUT name will be relevant as they will be used by the ACI Container Controller provision tool (acc-provision we will show later).
- An existing L3out could be used for this purpose.

ACI Tenant deployment model: 2 Options

Option1:
Cluster on user tenant, Single Shared VRF and L3Out



Option2:
Cluster on user tenant, Dedicated VRF and L3Out



Step 2

Pre provision ACI Tenant to host OCP cluster: OpenShift ACI Container Controller provision tool

- Available on Cisco.com: executable can run on any Linux host
- ACI Container Controller Provision Tool:
  - Takes a input YAML file containing the parameters of your configuration
  - Generates and pushes most of the ACI config to allow nodes and pods connectivity
  - Generates certificate to allow OCP master node to provision ACI config
  - Output a Kubernetes ACI CNI containers configuration YAML file

```
acc-provision –f openshift-3.11 –c config.yml –o cni_conf.yml
```

Used to select if we are deploying kubernetes 1.6 – 1.13 or OpenShift 3.6 - 3.11

Configuration file

Output file for ACI CNI config

Configuration File: config.yaml

```
aci_config:
  system_id: ocp311
  apic_hosts:
  - 10.48.170.201

  vmm_domain:
    encap_type: vxlan
    mcast_range:
      start: 226.31.1.1
      end: 226.31.255.255

  aep: OCP_AAEP
  vrf:
    name: default
    tenant: common
  l3out:
    name: openshiftL3
    external_networks:
    - extEpg

net_config:
  node_subnet: 12.11.0.1/16
  pod_subnet: 12.12.0.1/16
  node_svc_subnet: 12.15.0.1/24
  kubeapi_vlan: 3033
  service_vlan: 3034
  infra_vlan: 3085
```

# Every opflex cluster must have a distict ID

APIC controller(s) IP address(es)

Encapsulation

Multicast group

Attachable Entity Profile used for OCP nodes

L3out for nodes and pods external connectivity

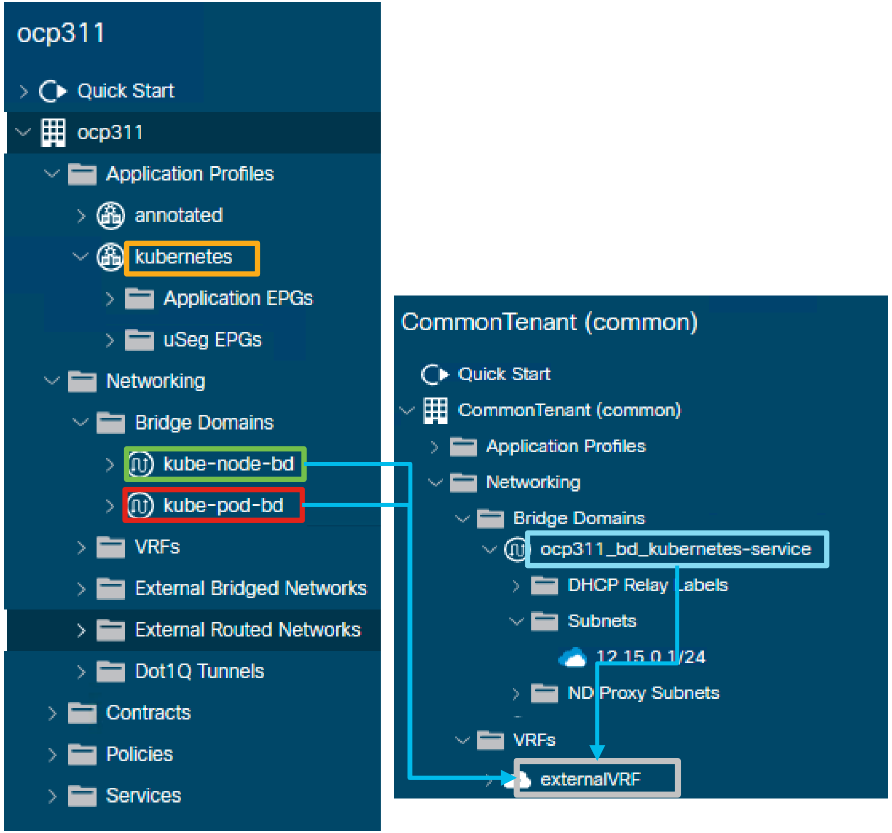
Node subnet
Pod subnet
LB type service subnet

Node VLAN
LB type service VLAN
ACI infra VLAN

- Note that the system_id will be used to create an ACI tenant for the specific OCP Cluster.
- To date it is required to dedicate one ACI tenant to each OCP cluster → next ACI release this requirement will be released

Acc-provision creates ACI tenant with Bridge Domains for OCP nodes, Pods and services

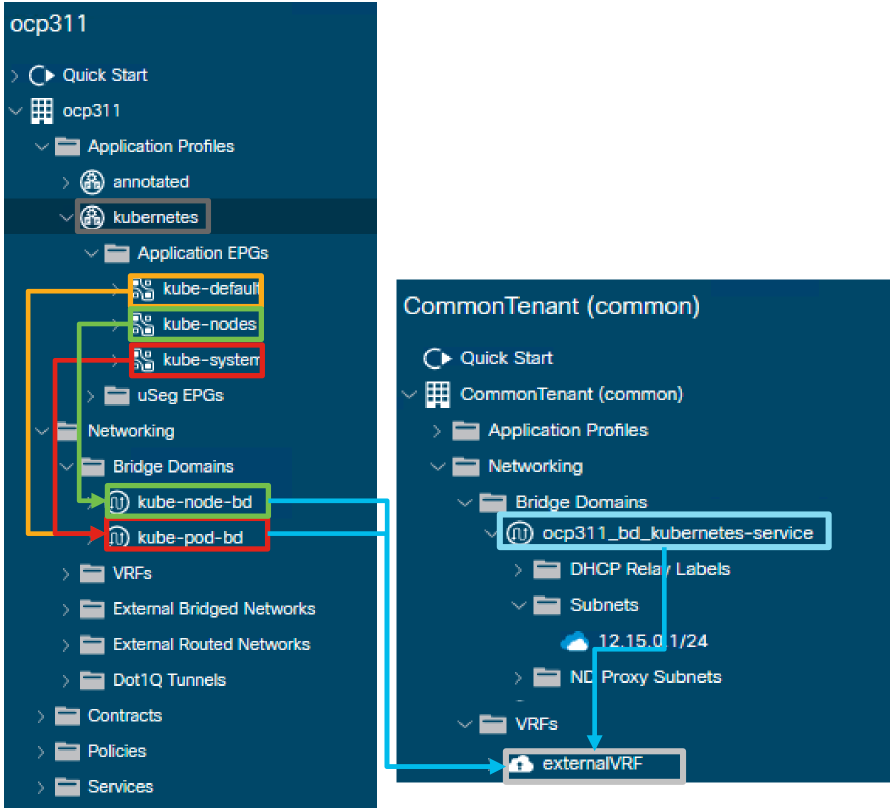
- kube-nodes-bd:
  - Only used for kube-node EPG
  - Maps to node_subnet
- kube-pod-bd:
  - Any pod will be assigned an IP from this BD Subnet
  - Used for kube-default, kube-system and any other user defined POD EPGs.
  - Maps to pod_subnet
- Cluster service BD:
  - BD for PBR/SG services
  - Created when ACI CNI plugin is deployed



cisco Live!

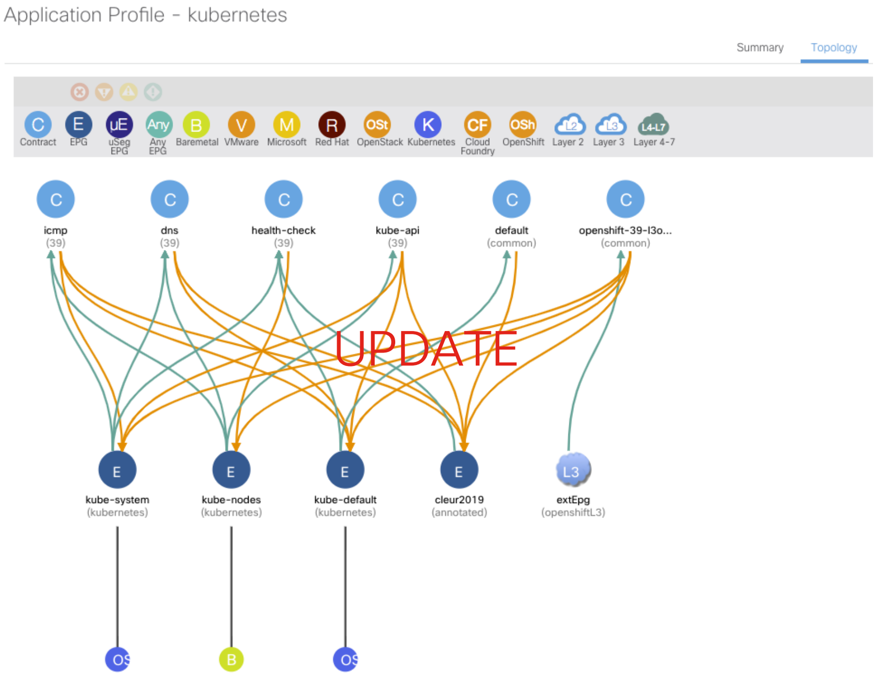
Acc-provision creates the ACI tenant with EPGs for OCP nodes and PODs

- Within the tenant selected the provisioning tool creates a ‘Kubernetes’ Application Profile with three EPGs:
- ‘kube-nodes’: for node interfaces, mapped to PhysDom
- ‘kube-system’: for system PODs, mapped to VMMDom
- ‘kube-default’: base EPG for all containers on any project by default, mapped to VMMDom



Acc-provision creates the contracts to allow connectivity between OCP nodes and PODs

- The required minimum set of contracts are automatically configured to ensure basic cluster functionality
  - DNS
  - Health-check
  - ICMP
  - Kube-API
- Administrator can define additional contracts if/when required



Step 3

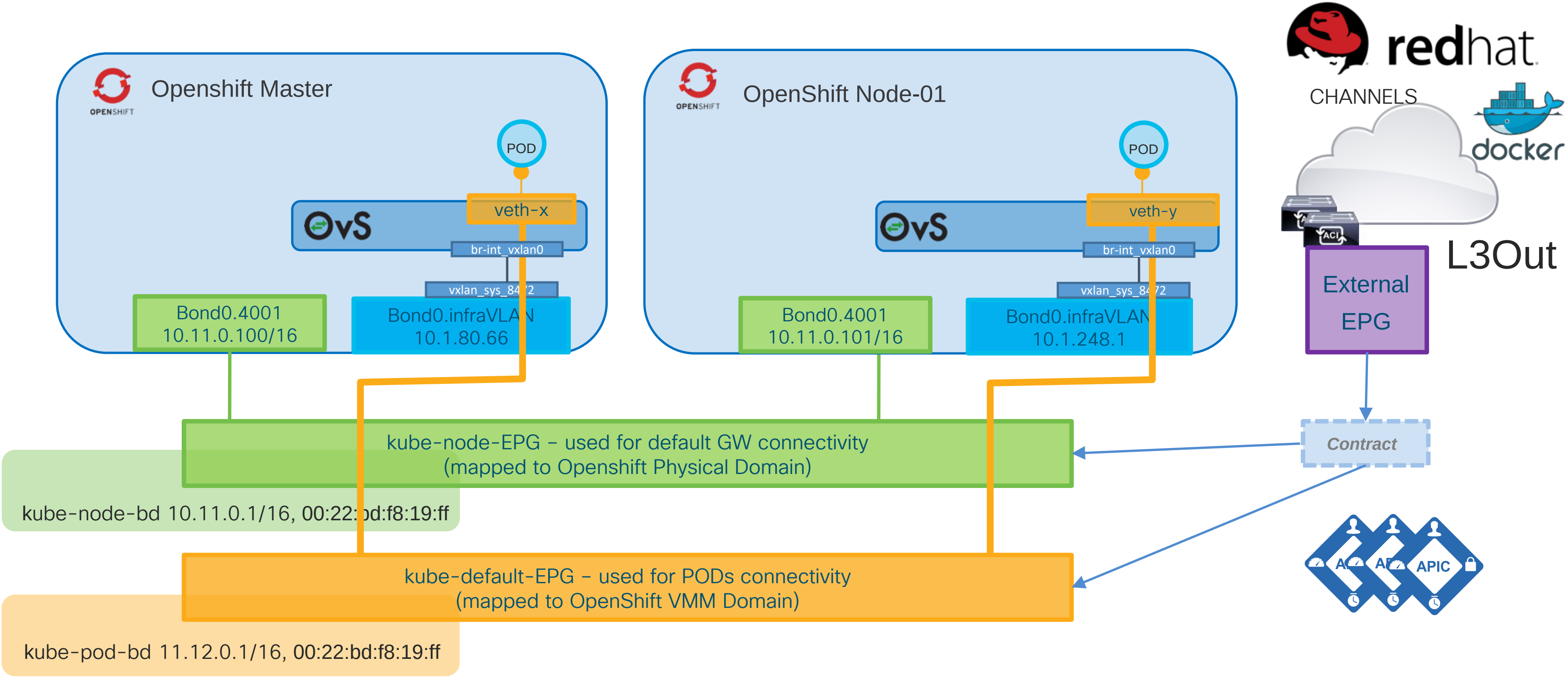
Ansible configuration file to provision OCP

```
[root@ocp-master ~]# cat /etc/ansible/hosts
[…]
[OSEv3:vars]
openshift_use_aci=True
openshift_use_openshift_sdn=False
os_sdn_network_plugin_name='cni'
aci_deployment_yaml_file='/root/cniConfig311.yaml’
[…]
```

This is the output of the acc-provision tool previously used

```
[root@ocp-master ~]# ansible-playbook /usr/share/ansible/openshift-
ansible/playbooks/deploy_cluster.yml -i /etc/ansible/hosts
[…]
PLAY RECAP
***************************************************************************************
*****************************************************************************
localhost                  : ok=12   changed=0    unreachable=0    failed=0
ocp-master.domlab.cisco.com : ok=1032 changed=413  unreachable=0    failed=0
ocp-node1.domlab.cisco.com : ok=139  changed=71   unreachable=0    failed=0
ocp-node2.domlab.cisco.com : ok=139  changed=70   unreachable=0    failed=0
```

OpenShift and ACI Architecture



Demo
OCP and ACI: Visibility of OCP resources

ACI CNI Plugin components

ACI CNI Plugin Components – OCP Side

- ACI Containers Controller (ACC, aci-containers-controller)
  - It is a Kubernetes **Deployment** running one POD instance.
  - Handles IPAM
  - Management of endpoint state
  - Policy Mapping (annotations)
  - Controls Load Balancing
  - Synchronises configurations into the APIC
  - Apply SNAT to PODs

```
[root@dom-master1 ~]# oc get deployments --namespace=aci-containers-system
NAME                       DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
aci-containers-controller  1         1         1            1           223d
```

ACI CNI Plugin Components - OCP Side

- Aci Containers Host (ACH, aci-container-host):
  - is a DaemonSet composed of 3 containers running on every node
  - mcast-daemon:
    - Handles Broadcast, unknown unicast and multicast replication
  - aci-containers-host:
    - Endpoint metadata, Pod IPAM, Container Interface Configuration
  - opflex-agent:
    - Support for Stateful Security Groups, Manage configuration of OVS, Render policy to openflow rules to program OVS, handles loadbalancer services

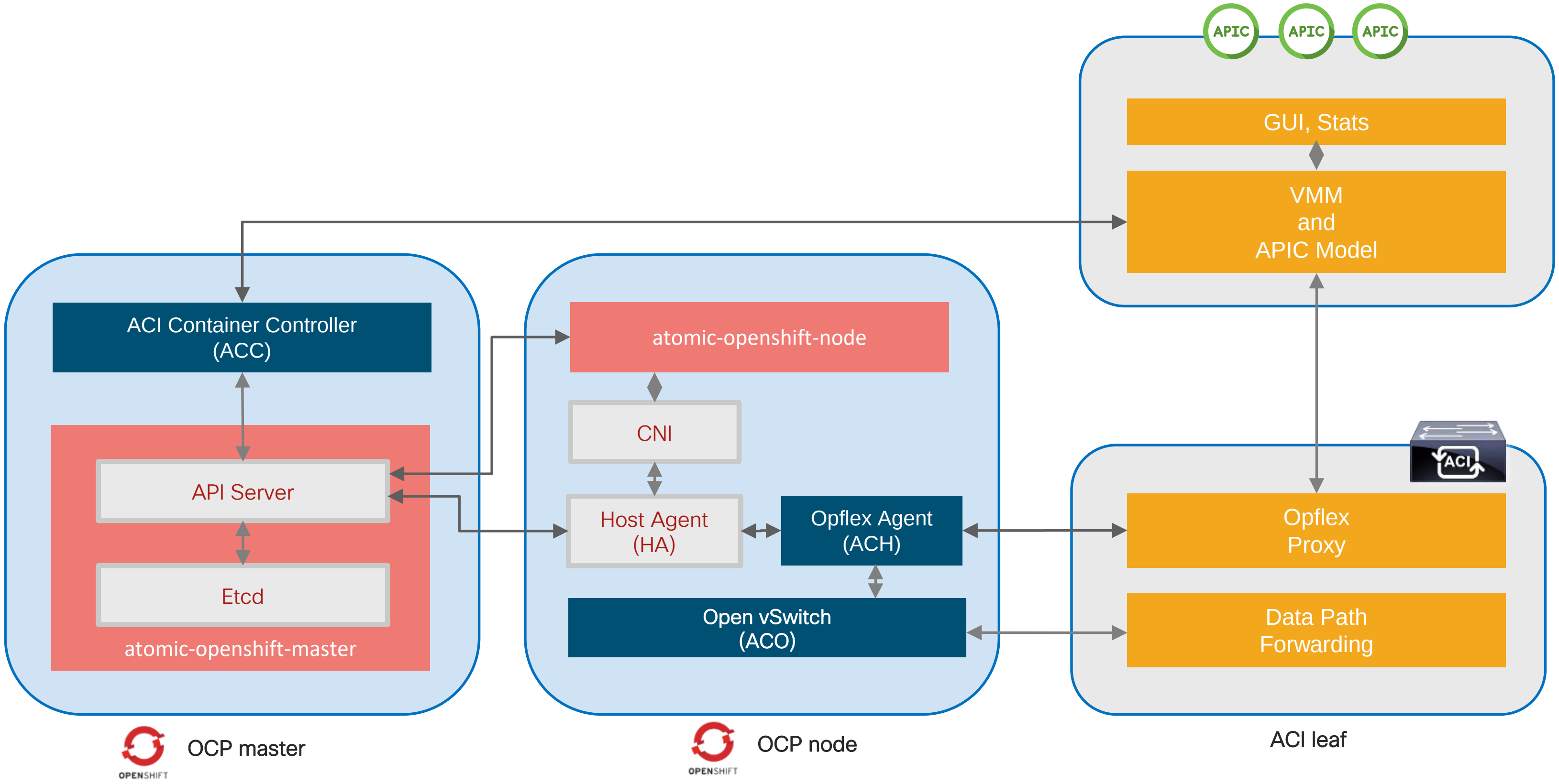
```
[root@dom-master1 ~]# oc get daemonsets --namespace=aci-containers-system |grep host
NAME                   DESIRED   CURRENT   READY    UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
aci-containers-host     3         3         3        3             3          <none>          223d
```

ACI CNI Plugin Components - OCP Side

- Aci Containers Openvswitch (ACO, aci-container-openvswitch)
  - DaemonSet composed of 3 containers running on every node
  - It is the Open vSwitch enforcing the required networking and security policies provisioned through the OpFlex agent

```
[root@dom-master1 ~]# oc get daemonsets --namespace=aci-containers-system |grep vswitch
NAME                         DESIRED   CURRENT   READY    UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
aci-containers-openvswitch   3         3         3        3            3           <none>          223d
```

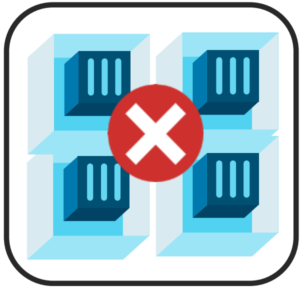
ACI CNI Plugin Components - Overview



ACI CNI Plugin: Segmentation

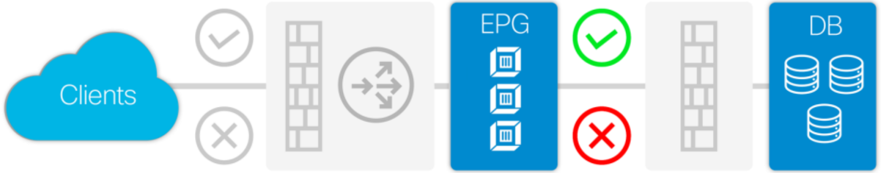
Dual level Policy Enforcement by ACI

Both Kubernetes Network Policy and ACI Contracts are enforced in the Linux kernel of every server node that containers run on.



Native API Default deny all traffic

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec: podSelector: {}
policyTypes:
- Ingress
- Egress
```



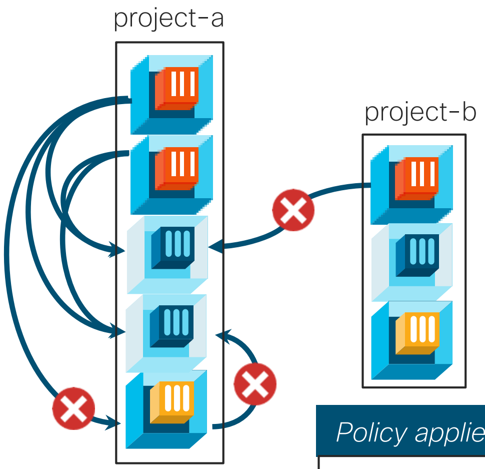
Containers are mapped to EPGs and contracts between EPGs are also enforced on all switches in the fabric where applicable.

Both policy mechanisms can be used in conjunction.

Also ISTIO is supported!

Segmentation through K8s Network Policy

- Specification of how selections of pods are allowed to communicate with each other and other network endpoints.
- Network namespace isolation using defined labels
  - directional: allowed ingress pod-to-pod traffic
  - filters traffic from pods in other projects
  - can specify protocol and ports (e.g. tcp/80)

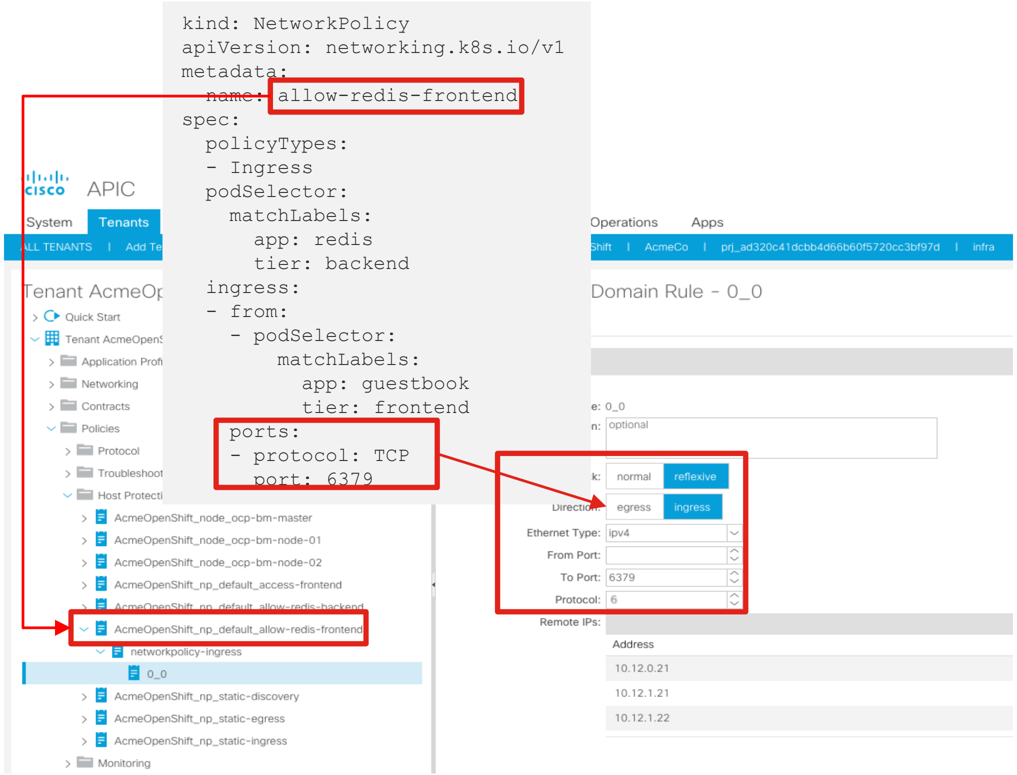


Policy applied to namespace: namespace-a

```
kind: NetworkPolicy
apiVersion: extensions/v1beta1
metadata:
  name: allow-red-to-blue-same-ns
spec:
  podSelector:
    matchLabels:
      type: blue
  ingress:
  - from:
    - podSelector:
        matchLabels:
          type: red
```

ACI Host Protection Profiles

- K8s Network Policies are implemented in ACI as HPP rules.
- HPP are enforced in OVS rules through OpFlex protocol.
- OVSs enforce distributed stateful firewall rules.



```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-redis-frontend
spec:
  policyTypes:
  - Ingress
  podSelector:
    matchLabels:
      app: redis
      tier: backend
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: guestbook
          tier: frontend
    ports:
    - protocol: TCP
      port: 6379
```

APIC
System
Tenants
Operations
Apps
ALL TENANTS | Add Te
Shift | AcmeCo | prj_ad320c41dcbb4d66b60f5720cc3bf97d | infra
Tenant AcmeOp
Quick Start
Tenant AcmeOpenS
Application Profi
Networking
Contracts
Policies
Protocol
Troubleshoot
Host Protecti
AcmeOpenShift_node_ocp-bm-master
AcmeOpenShift_node_ocp-bm-node-01
AcmeOpenShift_node_ocp-bm-node-02
AcmeOpenShift_np_default_access-frontend
AcmeOpenShift_np_default_allow-redis-backend
AcmeOpenShift_np_default_allow-redis-frontend
networkpolicy-ingress
0_0
AcmeOpenShift_np_static-discovery
AcmeOpenShift_np_static-egress
AcmeOpenShift_np_static-ingress
Monitoring
Domain Rule - 0_0

e: 0_0

n: optional

k: normal [✓ reflexive]

Direction: egress [✓ ingress]

Ethernet Type: ipv4

From Port:

To Port: 6379

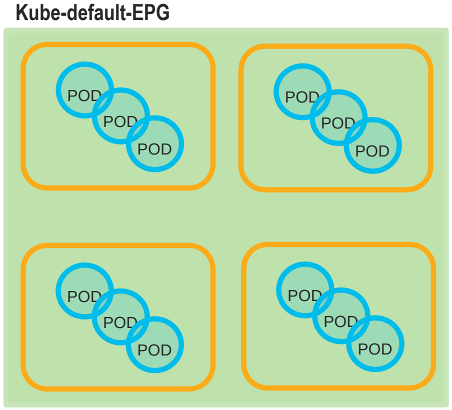
Protocol: 6

Remote IPs:

Address
10.12.0.21
10.12.1.21
10.12.1.22

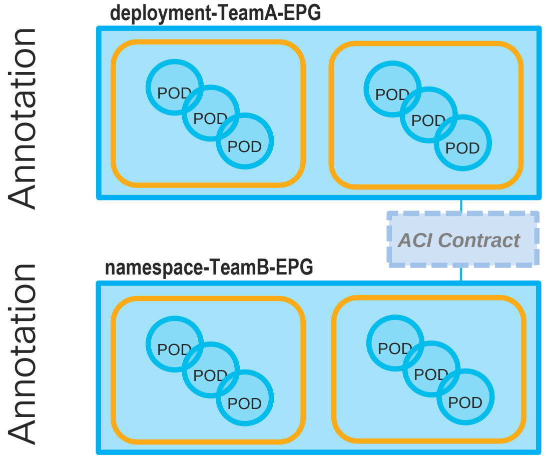
Mapping Network Policy and EPGs

Cluster Isolation



- Default behavior: single EPG for all user PODs in the cluster
- No need for ACI contracts between Pods
- K8s Network Policy to isolate namespaces

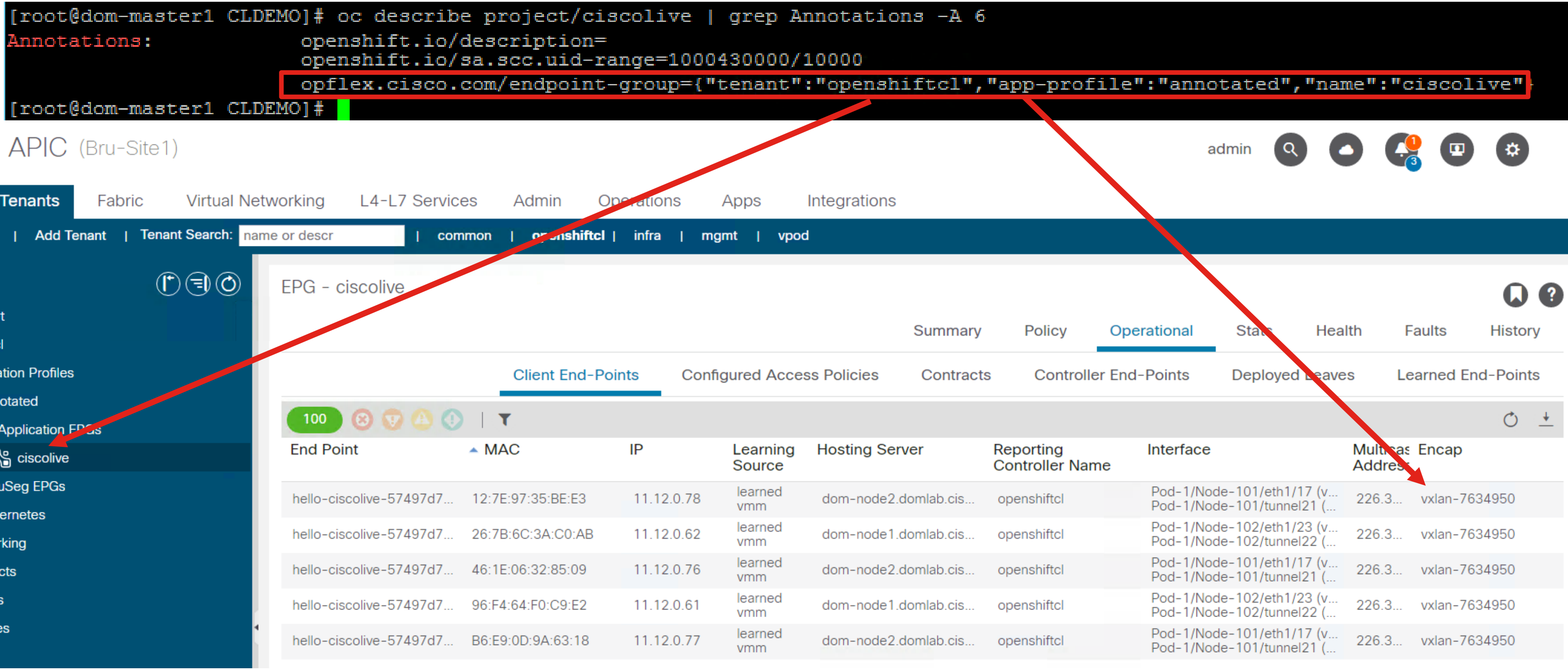
Namespace or Deployment Isolation



- Namespaces or Deployments mapped to EPGs
- Contracts for inter-namespace traffic required to enable connectivity
- K8s Network Policy to isolate tiers inside namespaces

□ K8s Network Policy/ISTIO

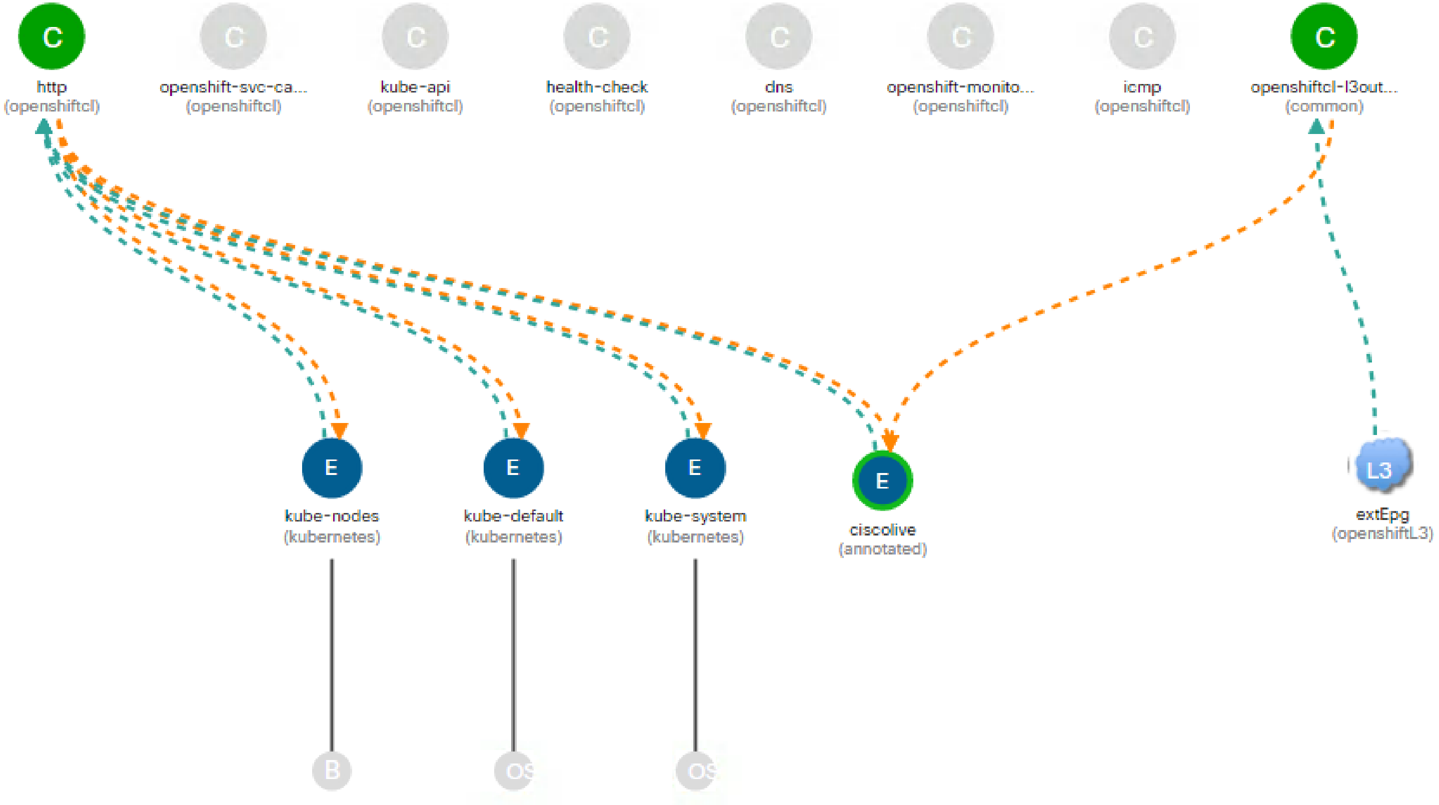
Annotation of Project/Deployment



```
[root@dom-master1 CLDEMO]# oc describe project/ciscolive | grep Annotations -A 6
Annotations:            openshift.io/description=
                        openshift.io/sa.scc.uid-range=1000430000/10000
                        opflex.cisco.com/endpoint-group={"tenant":"openshiftcl","app-profile":"annotated","name":"ciscolive"
[root@dom-master1 CLDEMO]#
```

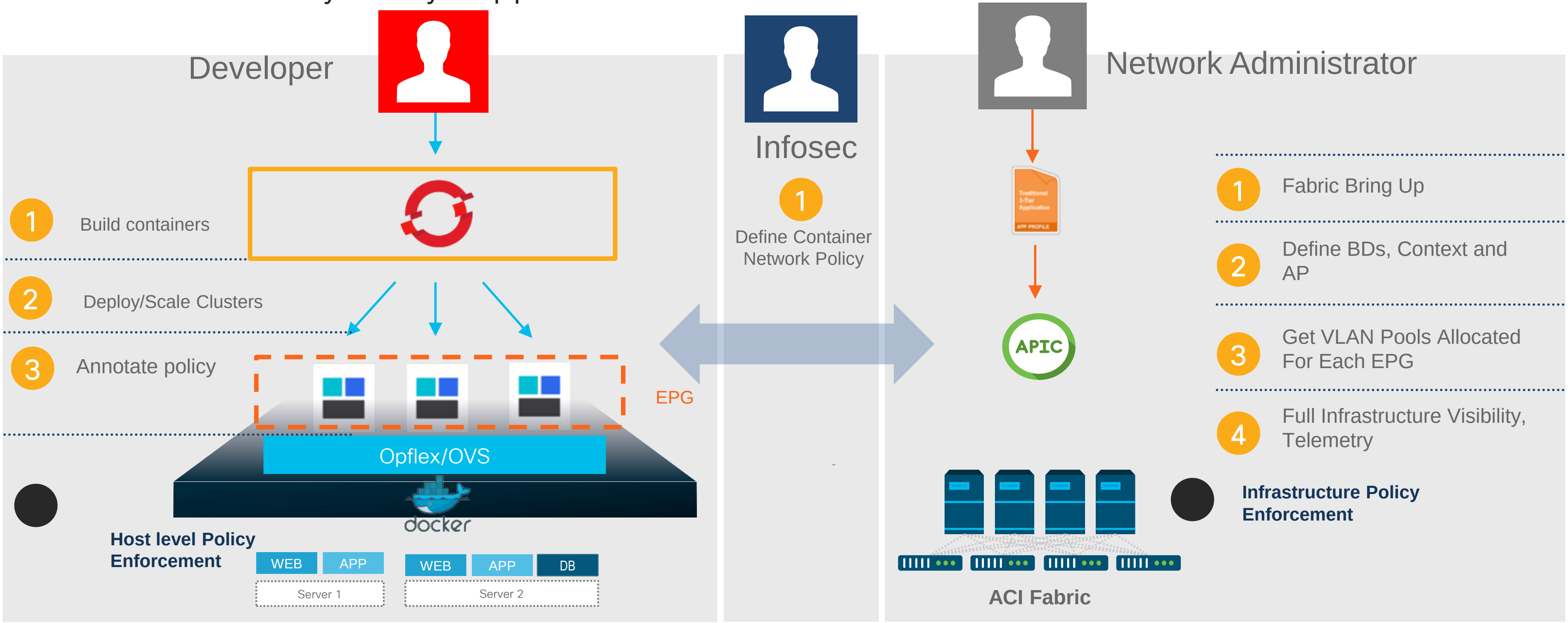
End Point	MAC	IP	Learning Source	Hosting Server	Reporting Controller Name	Interface	Multicast Address	Encap
hello-ciscolive-57497d7...	12:7E:97:35:BE:E3	11.12.0.78	learned vmm	dom-node2.domlab.cis...	openshiftcl	Pod-1/Node-101/eth1/17 (v... Pod-1/Node-101/tunnel21 (...	226.3...	vxlan-7634950
hello-ciscolive-57497d7...	26:7B:6C:3A:C0:AB	11.12.0.62	learned vmm	dom-node1.domlab.cis...	openshiftcl	Pod-1/Node-102/eth1/23 (v... Pod-1/Node-102/tunnel22 (...	226.3...	vxlan-7634950
hello-ciscolive-57497d7...	46:1E:06:32:85:09	11.12.0.76	learned vmm	dom-node2.domlab.cis...	openshiftcl	Pod-1/Node-101/eth1/17 (v... Pod-1/Node-101/tunnel21 (...	226.3...	vxlan-7634950
hello-ciscolive-57497d7...	96:F4:64:F0:C9:E2	11.12.0.61	learned vmm	dom-node1.domlab.cis...	openshiftcl	Pod-1/Node-102/eth1/23 (v... Pod-1/Node-102/tunnel22 (...	226.3...	vxlan-7634950
hello-ciscolive-57497d7...	B6:E9:0D:9A:63:18	11.12.0.77	learned vmm	dom-node2.domlab.cis...	openshiftcl	Pod-1/Node-101/eth1/17 (v... Pod-1/Node-101/tunnel21 (...	226.3...	vxlan-7634950

Segmentation: EPG to connect other resources



ACI Network Plugin for OpenShift

- Native Security Policy Support



Developer

1 Build containers

2 Deploy/Scale Clusters

3 Annotate policy

Host level Policy Enforcement

Infosec

1 Define Container Network Policy

Network Administrator

1 Fabric Bring Up

2 Define BDs, Context and AP

3 Get VLAN Pools Allocated For Each EPG

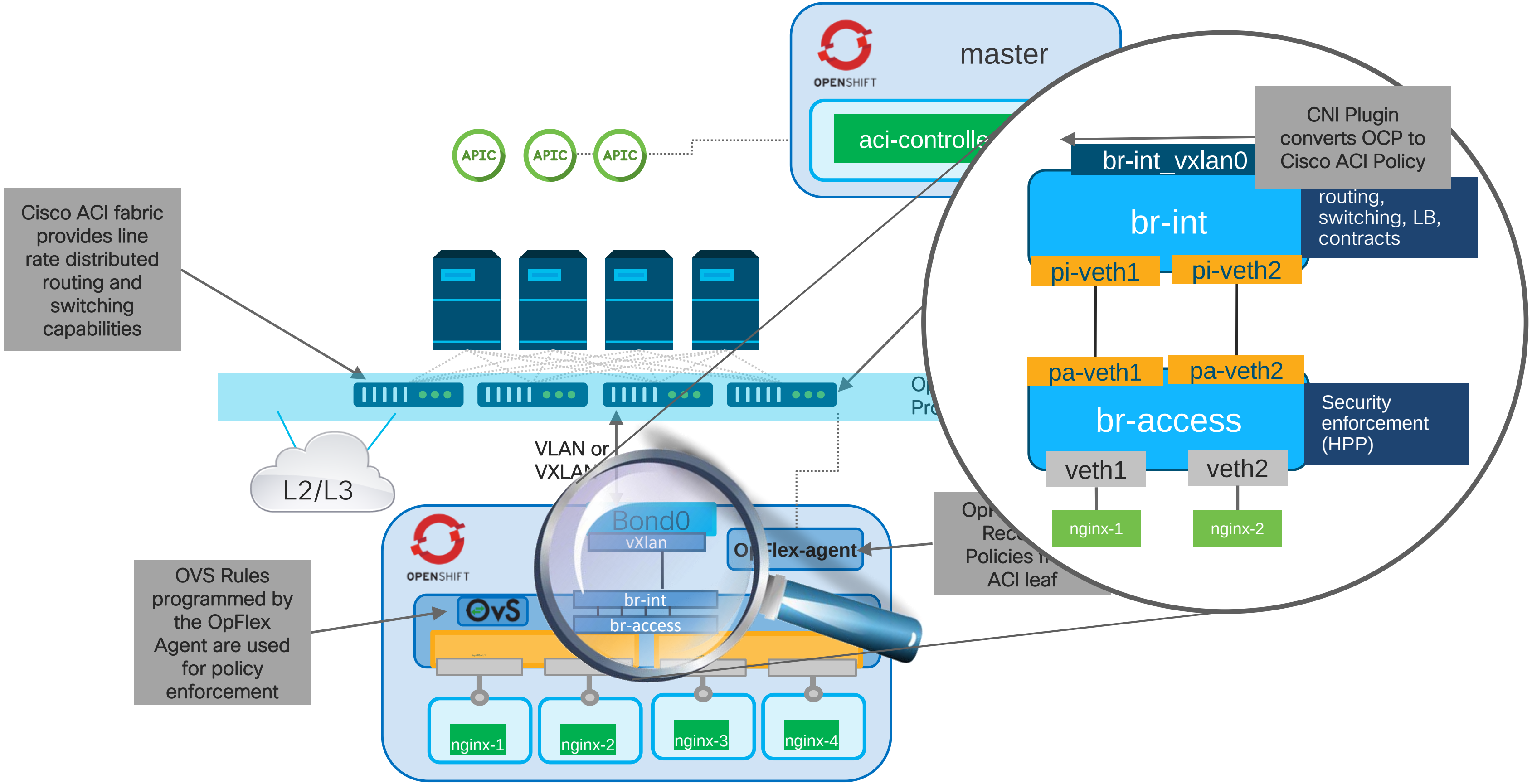
4 Full Infrastructure Visibility, Telemetry

Infrastructure Policy Enforcement

ACI Fabric

Demo:
Creation of annotated Project and Visibility

OVS rules provisioning



Cisco ACI fabric provides line rate distributed routing and switching capabilities

OVS Rules programmed by the OpFlex Agent are used for policy enforcement

CNI Plugin converts OCP to Cisco ACI Policy

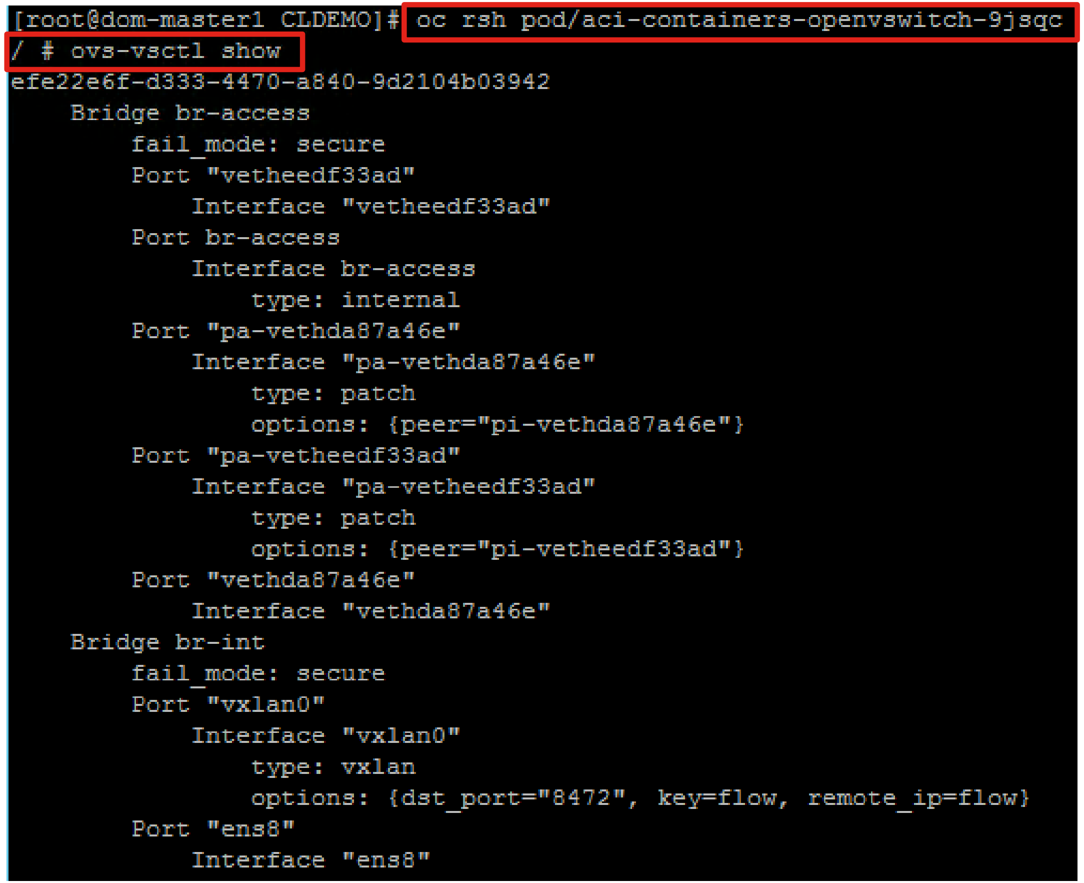
routing, switching, LB, contracts

Security enforcement (HPP)

Op
Rec
Policies f
ACI leaf

cisco Live!

Debugging OVS



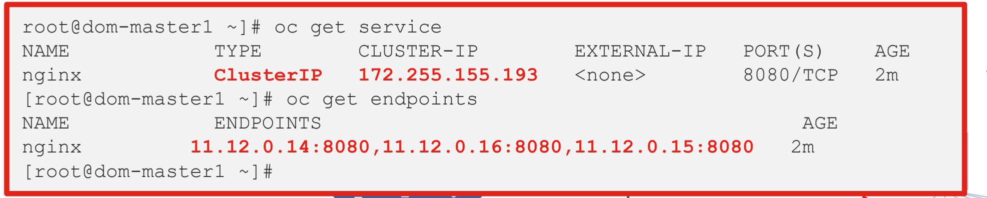
```
[root@dom-master1 CLDEMO]# oc rsh pod/aci-containers-openvswitch-9jsqc
/ # ovs-vsctl show
efe22e6f-d333-4470-a840-9d2104b03942
    Bridge br-access
        fail_mode: secure
        Port "vetheedf33ad"
            Interface "vetheedf33ad"
        Port br-access
            Interface br-access
                type: internal
        Port "pa-vethda87a46e"
            Interface "pa-vethda87a46e"
                type: patch
                options: {peer="pi-vethda87a46e"}
        Port "pa-vetheedf33ad"
            Interface "pa-vetheedf33ad"
                type: patch
                options: {peer="pi-vetheedf33ad"}
        Port "vethda87a46e"
            Interface "vethda87a46e"
    Bridge br-int
        fail_mode: secure
        Port "vxlan0"
            Interface "vxlan0"
                type: vxlan
                options: {dst_port="8472", key=flow, remote_ip=flow}
        Port "ens8"
            Interface "ens8"
```

ACI CNI Plugin and OpenShift Services and routes

OpenShift Service - ClusterIP

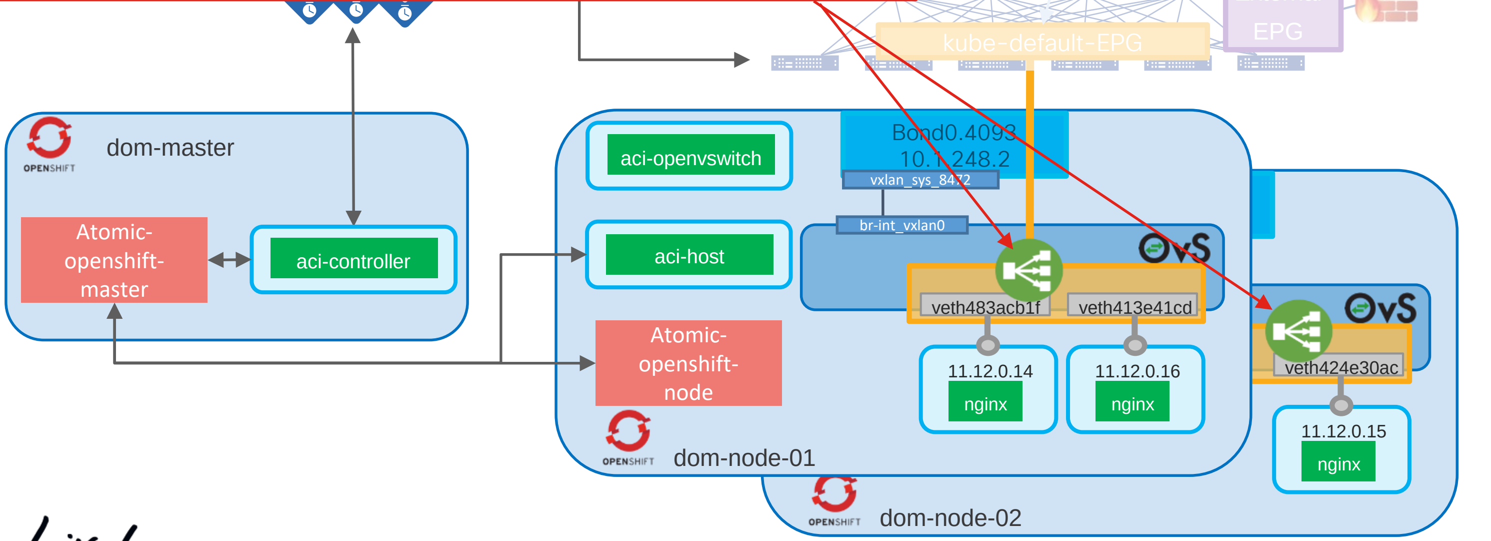
- Exposes a service internally, using a virtual IP address that is permanent but only visible in the cluster (for POD to POD conversations)
- Essentially, this is for East-West traffic within the OpenShift Cluster.
- Cisco ACI CNI plugin implements everything required for service objects of type ClusterIP:
  - When a new service object is created, the ACI CNI plugin assigns it an IP address from the OpenShift_portal_net CIDR.
  - The plugin listens to the endpoint API and learns the list of Pods that are backed by the service.

ClusterIP – Distributed Load Balancing

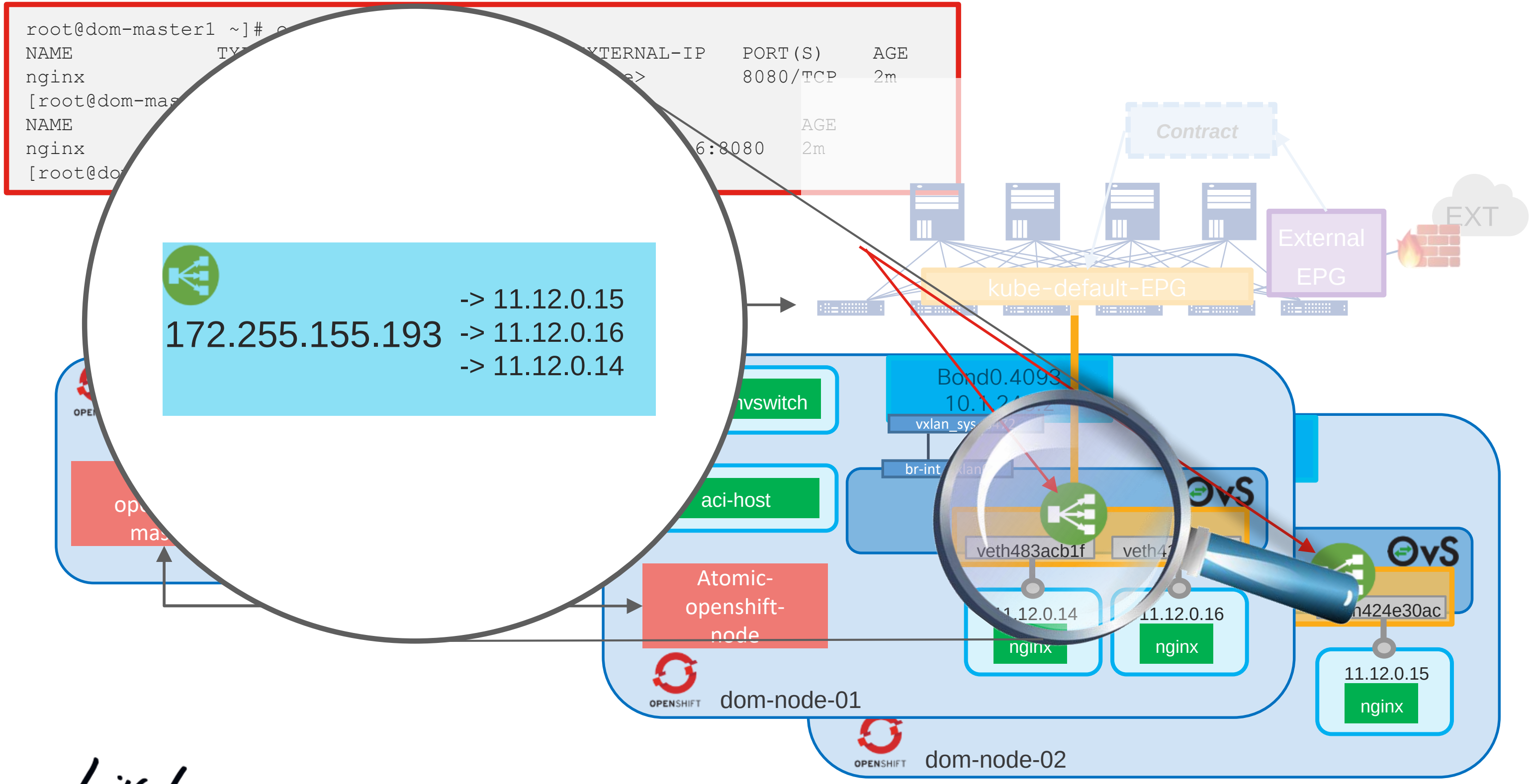


```
root@dom-master1 ~]# oc get service
NAME                TYPE        CLUSTER-IP        EXTERNAL-IP   PORT(S)     AGE
nginx               ClusterIP   172.255.155.193   <none>        8080/TCP    2m
[root@dom-master1 ~]# oc get endpoints
NAME                ENDPOINTS                                          AGE
nginx          11.12.0.14:8080,11.12.0.16:8080,11.12.0.15:8080    2m
[root@dom-master1 ~]#
```

The service configuration triggers the opflex agent to create of a load balancer on OVS

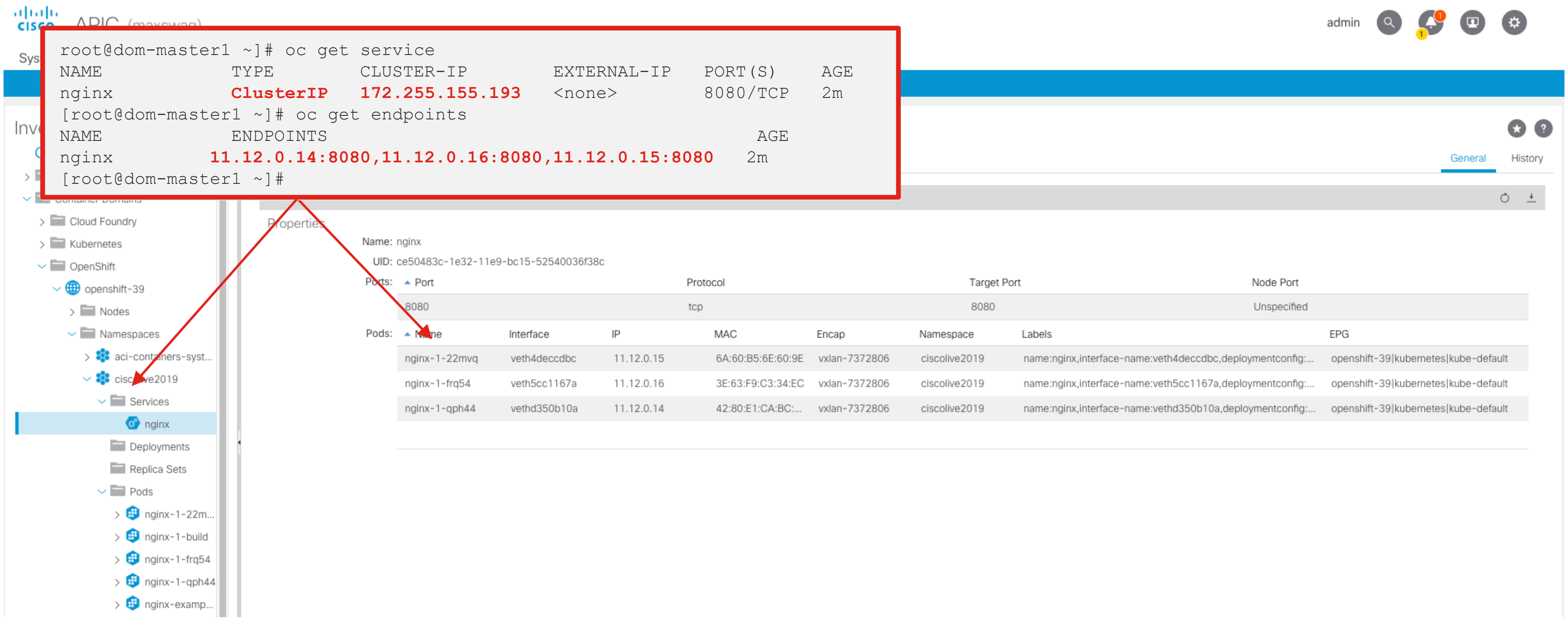


ClusterIP – Distributed Load Balancing



```
root@dom-master1 ~]# c
NAME              TY                   XTERNAL-IP    PORT(S)     AGE
nginx                                    e>           8080/TCP    2m
[root@dom-mas
NAME                                                              AGE
nginx                                              6:8080         2m
[root@do
```

ClusterIP – ACI VMM Visibility



```
root@dom-master1 ~]# oc get service
NAME                TYPE         CLUSTER-IP        EXTERNAL-IP    PORT(S)    AGE
nginx               ClusterIP    172.255.155.193   <none>         8080/TCP   2m
[root@dom-master1 ~]# oc get endpoints
NAME                ENDPOINTS                                              AGE
nginx          11.12.0.14:8080,11.12.0.16:8080,11.12.0.15:8080        2m
[root@dom-master1 ~]#
```

Name: nginx

UID: ce50483c-1e32-11e9-bc15-52540036f38c

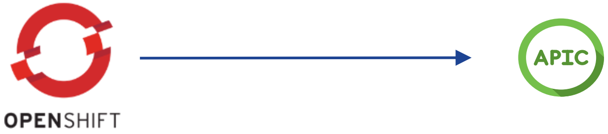
Ports:

Port	Protocol	Target Port	Node Port
8080	tcp	8080	Unspecified

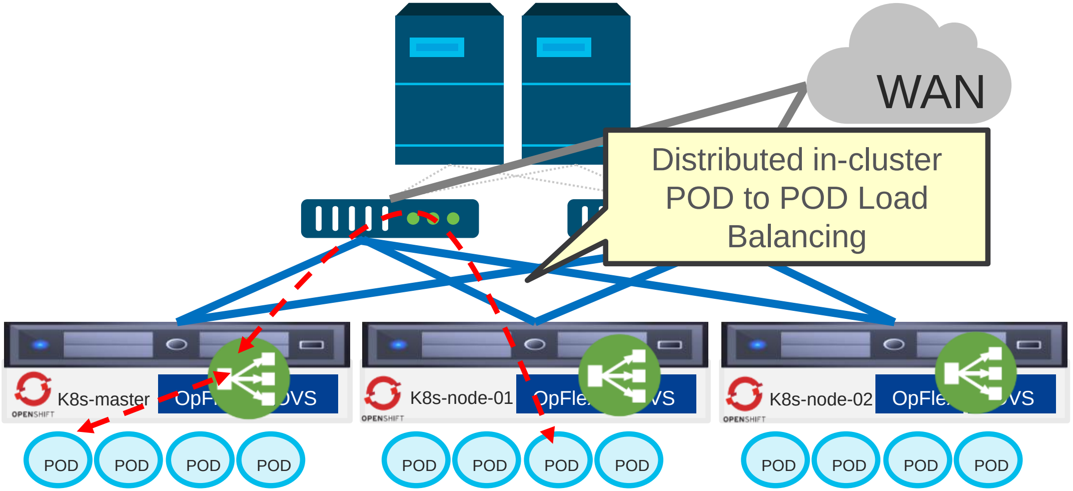
Pods:

Name	Interface	IP	MAC	Encap	Namespace	Labels	EPG
nginx-1-22mvq	veth4deccdbc	11.12.0.15	6A:60:B5:6E:60:9E	vxlan-7372806	ciscolive2019	name:nginx,interface-name:veth4deccdbc,deploymentconfig:...	openshift-39|kubernetes|kube-default
nginx-1-frq54	veth5cc1167a	11.12.0.16	3E:63:F9:C3:34:EC	vxlan-7372806	ciscolive2019	name:nginx,interface-name:veth5cc1167a,deploymentconfig:...	openshift-39|kubernetes|kube-default
nginx-1-qph44	vethd350b10a	11.12.0.14	42:80:E1:CA:BC:...	vxlan-7372806	ciscolive2019	name:nginx,interface-name:vethd350b10a,deploymentconfig:...	openshift-39|kubernetes|kube-default

Openshift Load Balancing with ACI - ClusterIP



```
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx
  namespace: default
spec:
  ports:
  - name: 80-tcp
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: ClusterIP
```

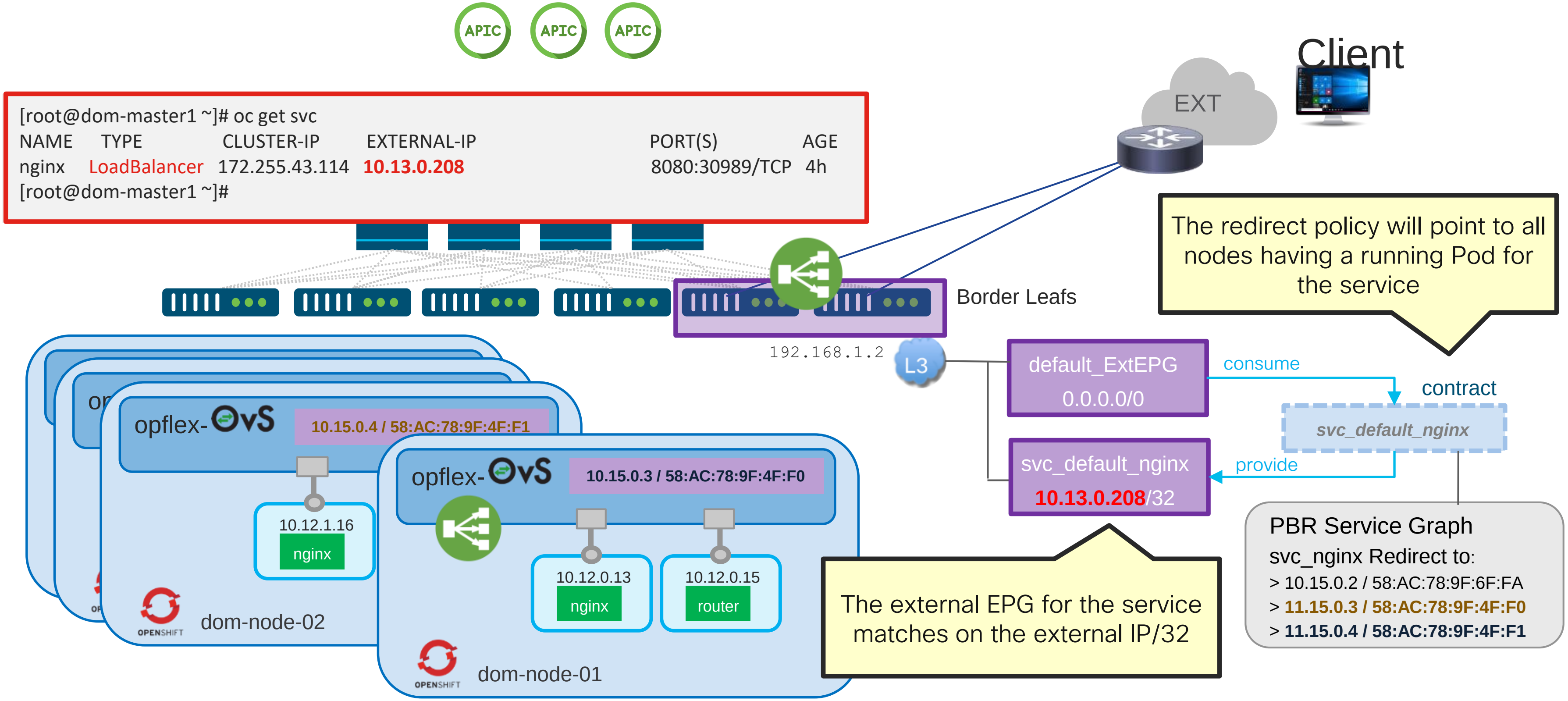


Distributed in-cluster POD to POD Load Balancing

OpenShift Service - LoadBalancer

- Works with a cloud provider to expose an external service IP address (statically or dynamically assigned)
- Essentially, this is for North-South traffic.
- Cisco ACI CNI plugin implements everything required for service objects of type LoadBalancer:
    - ACI CNI plugin assigns a L3out external EPG matching the service IP.
    - Creates and applies PBR matching the external EPG to enforce HW load balancing to the OpenShift nodes with active endpoints.
    - Listens to the endpoint API and constantly updates list of endpoints.

Service LoadBalancer: Life of a packet



```
[root@dom-master1 ~]# oc get svc
NAME     TYPE           CLUSTER-IP       EXTERNAL-IP                      PORT(S)          AGE
nginx    LoadBalancer   172.255.43.114   10.13.0.208                      8080:30989/TCP   4h
[root@dom-master1 ~]#
```

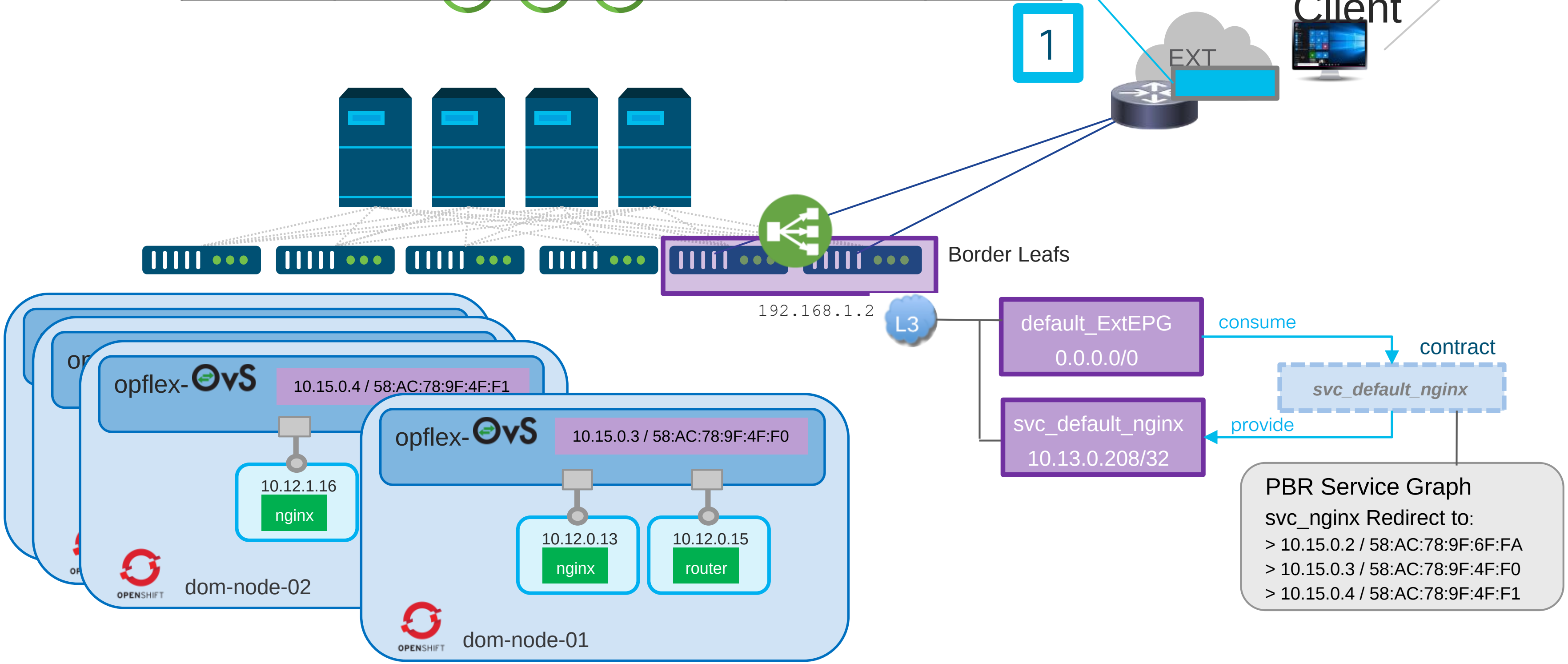
The redirect policy will point to all nodes having a running Pod for the service

The external EPG for the service matches on the external IP/32

PBR Service Graph
svc_nginx Redirect to:
> 10.15.0.2 / 58:AC:78:9F:6F:FA
> 11.15.0.3 / 58:AC:78:9F:4F:F0
> 11.15.0.4 / 58:AC:78:9F:4F:F1

DST_MAC	SRC_MAC	DST_IP	SRC_IP	DST_PORT	SRC_PORT
Router-MAC	Client-MAC	10.13.0.208	192.168.2.2	80	40183

```
#curl http://10.13.0.208
```



PBR Service Graph

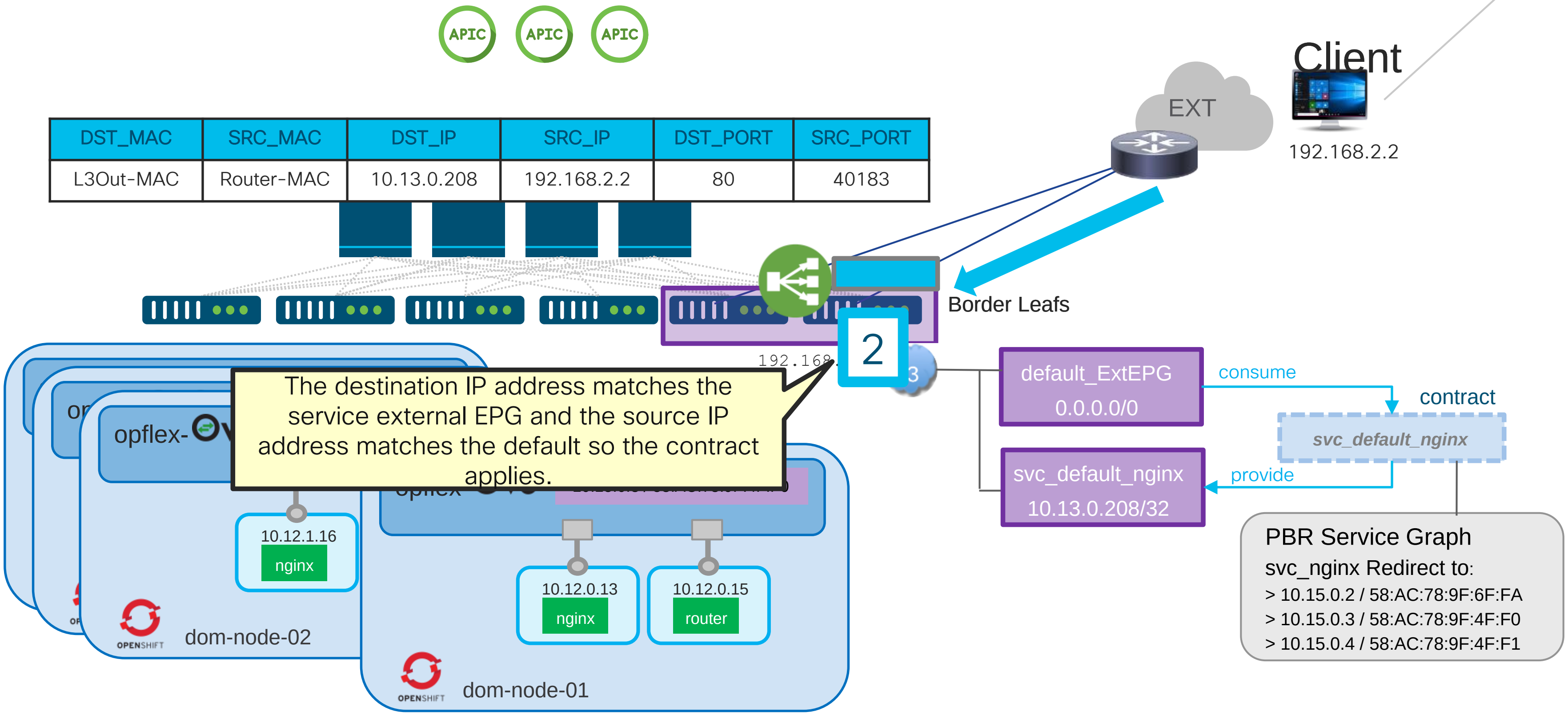
svc_nginx Redirect to:

> 10.15.0.2 / 58:AC:78:9F:6F:FA

> 10.15.0.3 / 58:AC:78:9F:4F:F0

> 10.15.0.4 / 58:AC:78:9F:4F:F1

```
#curl http://10.13.0.208
```



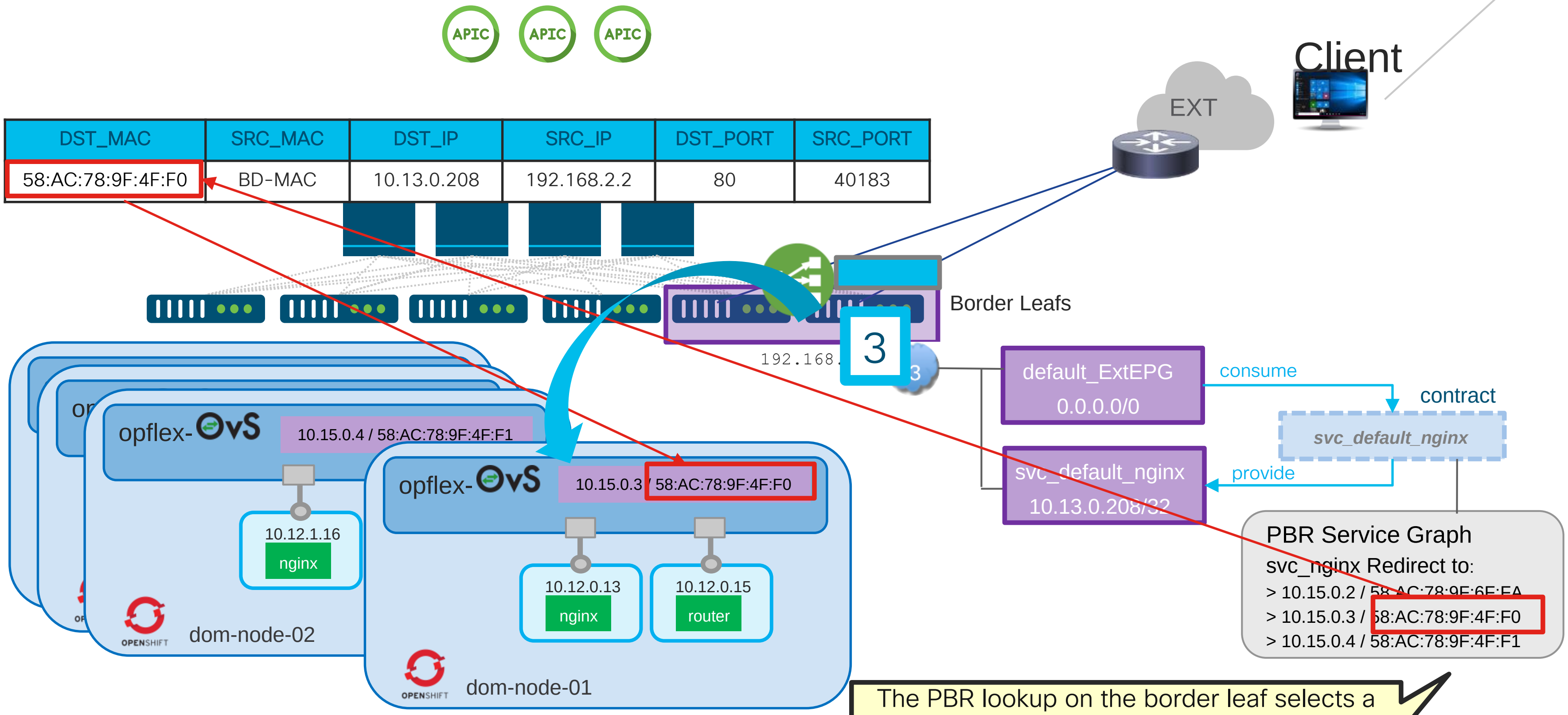
DST_MAC	SRC_MAC	DST_IP	SRC_IP	DST_PORT	SRC_PORT
L3Out-MAC	Router-MAC	10.13.0.208	192.168.2.2	80	40183

The destination IP address matches the service external EPG and the source IP address matches the default so the contract applies.

PBR Service Graph
svc_nginx Redirect to:
> 10.15.0.2 / 58:AC:78:9F:6F:FA
> 10.15.0.3 / 58:AC:78:9F:4F:F0
> 10.15.0.4 / 58:AC:78:9F:4F:F1

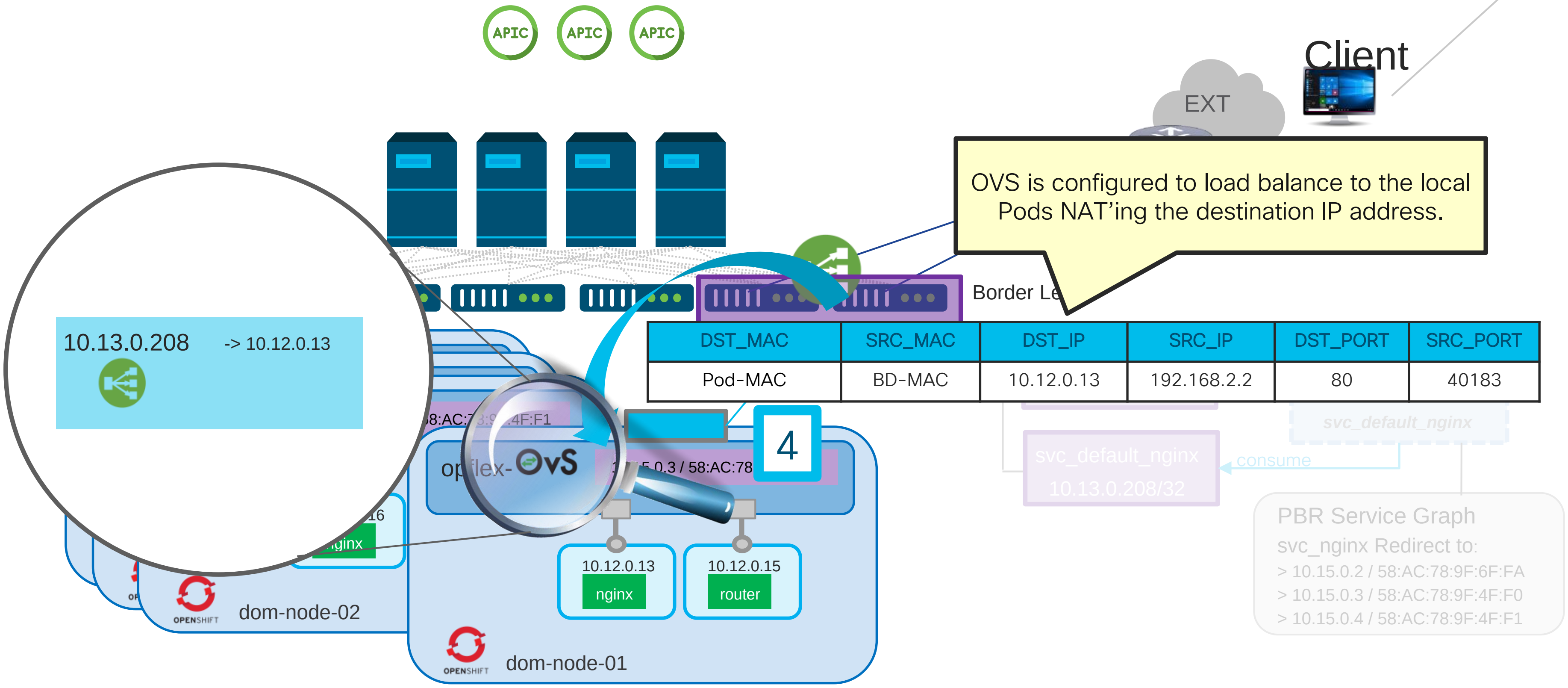
```
#curl http://10.13.0.208
```

DST_MAC	SRC_MAC	DST_IP	SRC_IP	DST_PORT	SRC_PORT
58:AC:78:9F:4F:F0	BD-MAC	10.13.0.208	192.168.2.2	80	40183



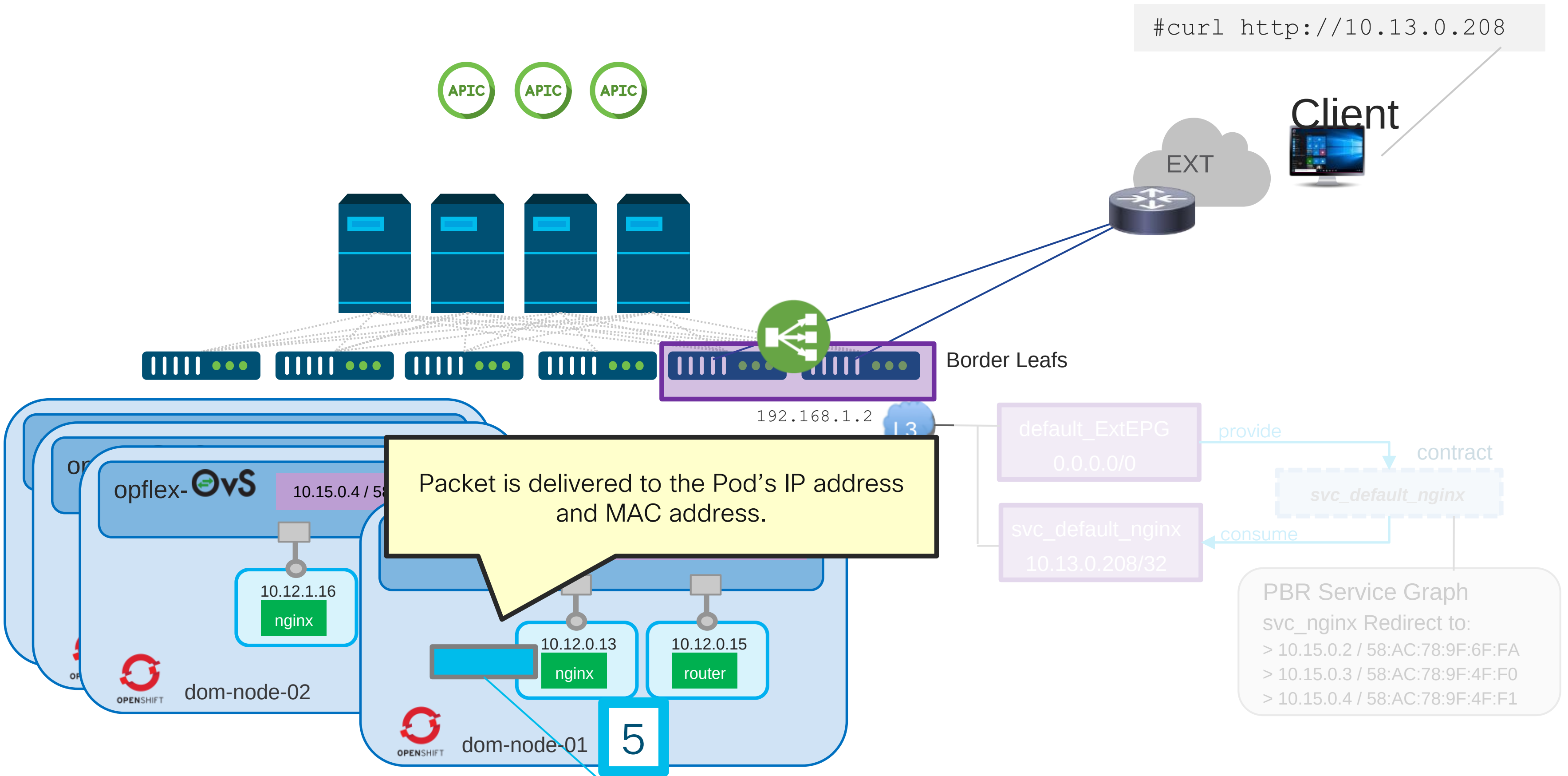
The PBR lookup on the border leaf selects a next-hop node, 10.15.0.3. The packet is routed (VXLAN encapsulated) to the node MAC.

```
#curl http://10.13.0.208
```



OVS is configured to load balance to the local Pods NAT'ing the destination IP address.

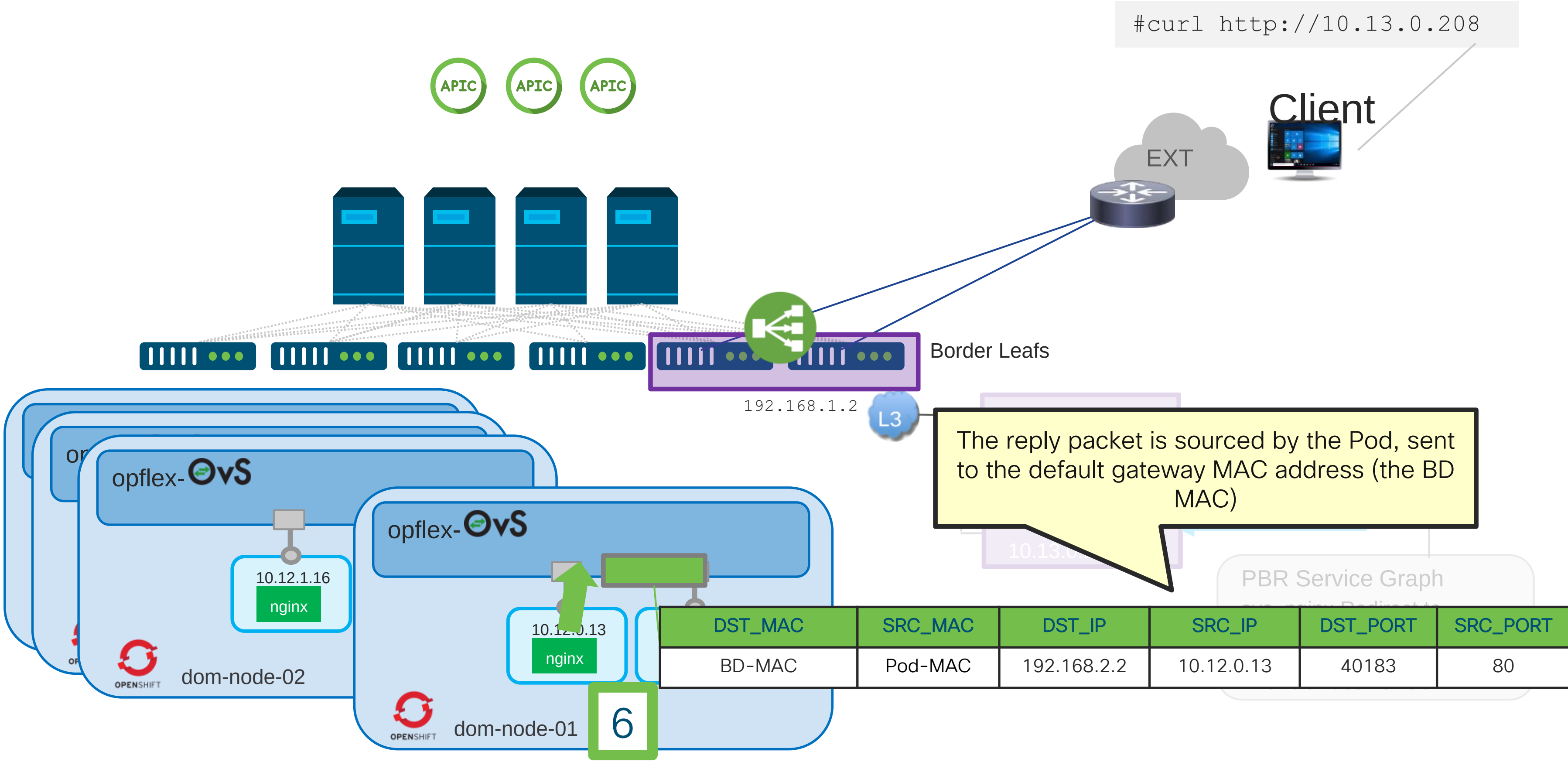
DST_MAC	SRC_MAC	DST_IP	SRC_IP	DST_PORT	SRC_PORT
Pod-MAC	BD-MAC	10.12.0.13	192.168.2.2	80	40183



```
#curl http://10.13.0.208
```

Packet is delivered to the Pod’s IP address and MAC address.

DST_MAC	SRC_MAC	DST_IP	SRC_IP	DST_PORT	SRC_PORT
Pod-MAC	BD-MAC	10.12.0.13	192.168.2.2	80	40183



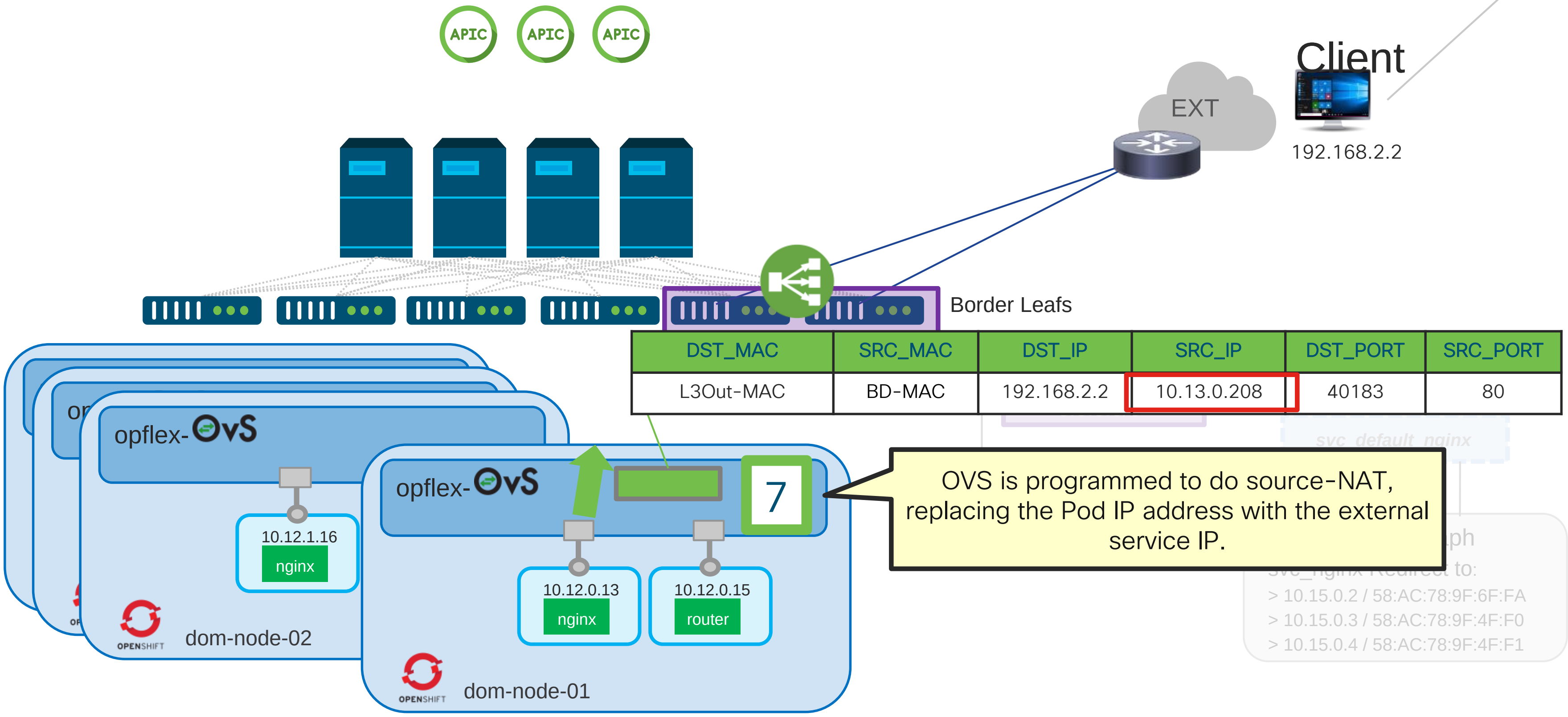
```
#curl http://10.13.0.208
```

The reply packet is sourced by the Pod, sent to the default gateway MAC address (the BD MAC)

DST_MAC	SRC_MAC	DST_IP	SRC_IP	DST_PORT	SRC_PORT
BD-MAC	Pod-MAC	192.168.2.2	10.12.0.13	40183	80

cisco Live!

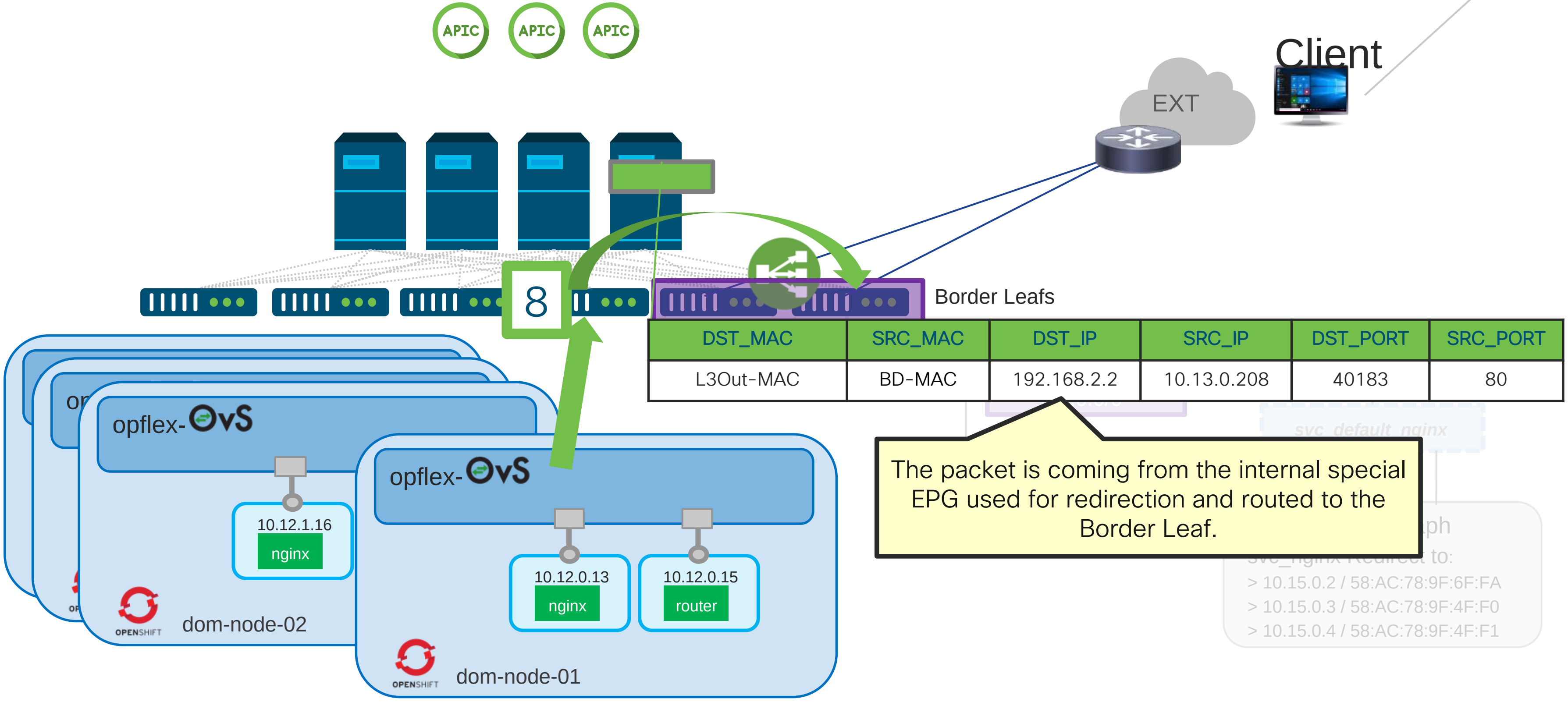
```
#curl http://10.13.0.208
```



DST_MAC	SRC_MAC	DST_IP	SRC_IP	DST_PORT	SRC_PORT
L3Out-MAC	BD-MAC	192.168.2.2	10.13.0.208	40183	80

OVS is programmed to do source-NAT, replacing the Pod IP address with the external service IP.

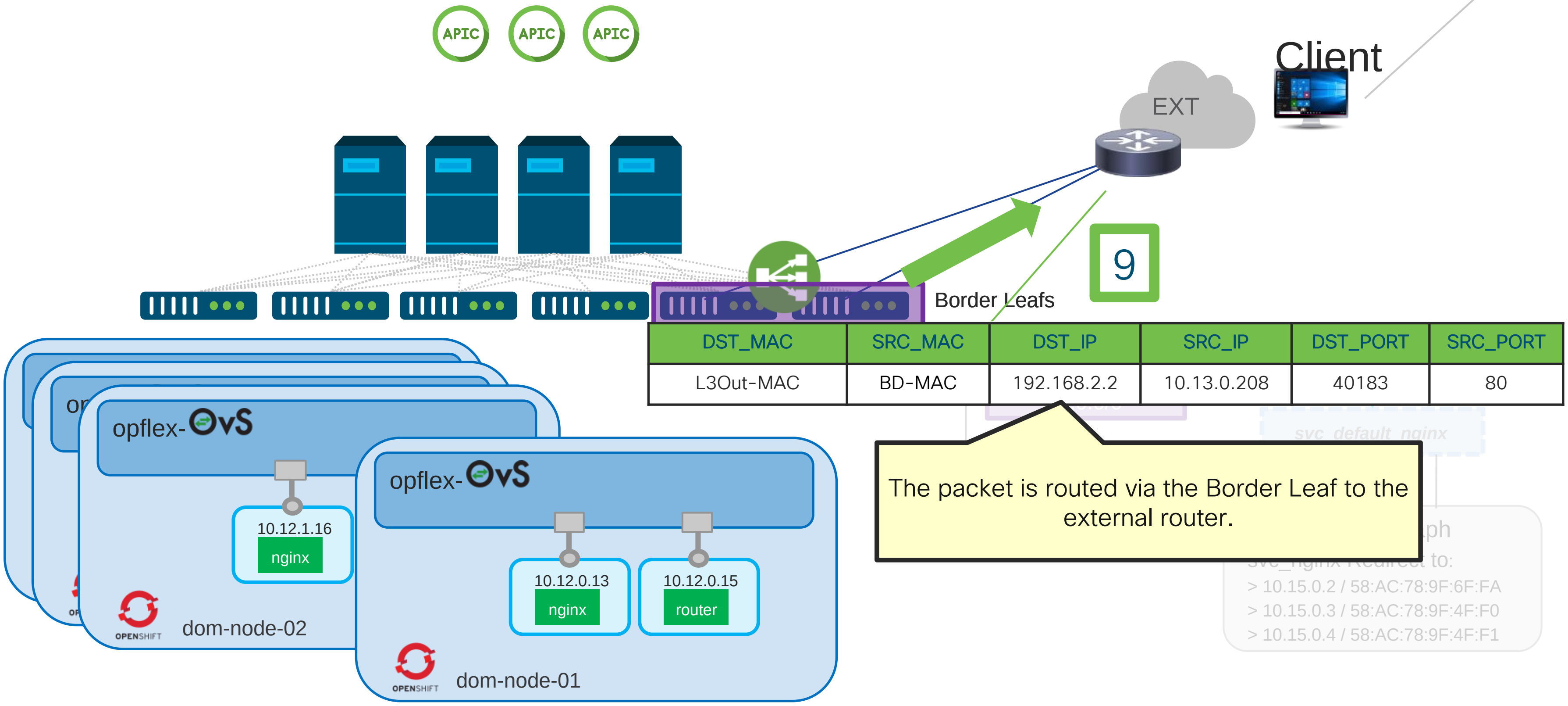
```
#curl http://10.13.0.208
```



DST_MAC	SRC_MAC	DST_IP	SRC_IP	DST_PORT	SRC_PORT
L3Out-MAC	BD-MAC	192.168.2.2	10.13.0.208	40183	80

The packet is coming from the internal special EPG used for redirection and routed to the Border Leaf.

```
#curl http://10.13.0.208
```



DST_MAC	SRC_MAC	DST_IP	SRC_IP	DST_PORT	SRC_PORT
L3Out-MAC	BD-MAC	192.168.2.2	10.13.0.208	40183	80

The packet is routed via the Border Leaf to the external router.

Roadmap

ACI CNI Upcoming Features

- Open Shift 4.2, 4.3 support
- Mixed form factor (VMs and Bare Metals in the same cluster)
- POD and Node BD in common tenant support
  - Support multiple cluster in the same tenant
- ACI CNI in Public Cloud
  - AWS with OpenShift 4.2

A quick Recap?

Why ACI for Application Container Platforms

Turnkey solution for node and container connectivity

Flexible policy: Native platform policy API and ACI policies

Hardware-accelerated: Integrated load balancing

Visibility: Live statistics in APIC per container and health metrics

Enhanced Multitenancy and unified networking for containers, VMs, bare metal

Fast, easy, secure and scalable **networking** for your Application Container Platform

References

Reference Material to Follow up

- Compatibility Matrix
- Archiecture Guide of OpenShift integration with ACI
- ACI and OpenShift CNI Plugin integration guidelines

Complete your online session survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on ciscolive.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.com.

Continue your education

Demos in the Cisco campus

Walk-in labs

Meet the engineer 1:1 meetings

Related sessions

Thank you

You make **possible**