CISCO Live!

Let’s go

The bridge to possible

Kubernetes Infrastructure Connectivity for ACI

Camillo Rossi, Technical Marketing Engineer

Agenda

- Kubernetes Refresh
- Kubernetes Network Challenges
- ACI-CNI
- BGP Based Architecture
  - Calico, Cilium and Kube-Router
  - Automation and Visibility
- Which solution is right for me?
- Q&A

Kubernetes Refresh

Kubernetes - pod

- A pod is the scheduling unit in Kubernetes. It is a logical collection of one or more containers which are always scheduled together.
- The set of containers composed together in a pod share an IP.

```
 [root@k8s-01-p1 ~]# kubectl get pod  --namespace=kube-system
NAME                                      READY     STATUS    RESTARTS   AGE
aci-containers-controller-1201600828-qsw5g   1/1       Running   1          69d
aci-containers-host-lt9kl                 3/3       Running   0          72d
aci-containers-host-xnwkr                 3/3       Running   0          58d
aci-containers-openvswitch-0rjbw          1/1       Running   0          58d
aci-containers-openvswitch-7j1h5          1/1       Running   0          72d
```

Kubernetes – Deployment

- Deployments are a collection of pods providing the same service
- You describe the desired state in a Deployment object, and the Deployment controller will change the actual state to the desired state at a controlled rate for you
- For example you can create a deployment that declare you need to have 2 copies of your front-end pod.

```
[root@k8s-01-p1 ~]# kubectl get deployment --namespace=kube-system
NAME                        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
aci-containers-controller   1         1         1            1           72d
```

Kubernetes – Services

- A service tells the rest of the Kubernetes environment (including other pods and Deployments) what services your application provides.
- While pods come and go, the service IP addresses and ports remain the same.
- Kubernetes automatically load balance the load across the replicas in the deployment that you expose through a Service
- Other applications can find your service through Kubernetes service discovery.
    - Every time a service is create a DNS entry is added to kube-dns

```
[root@k8s-01-p1 ~]# kubectl get svc --namespace=kube-system
NAME        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kube-dns    11.96.0.10    <none>         53/UDP,53/TCP    72d
```

Kubernetes – External Services

- If there are external IPs that route to one or more cluster nodes, Kubernetes services can be exposed on those external IPs.
- Traffic that ingresses into the cluster with the external IP (as destination IP), on the service port, will be routed to one of the service endpoints.
- External IPs are not managed by Kubernetes and are the responsibility of the cluster administrator.

```
[root@k8s-01-p1 ~]# kubectl get svc front-end --namespace=guest-book
NAME        CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
front-end   11.96.0.33   11.3.0.2      80:30002/TCP   3m
```

Kubernetes - Annotations

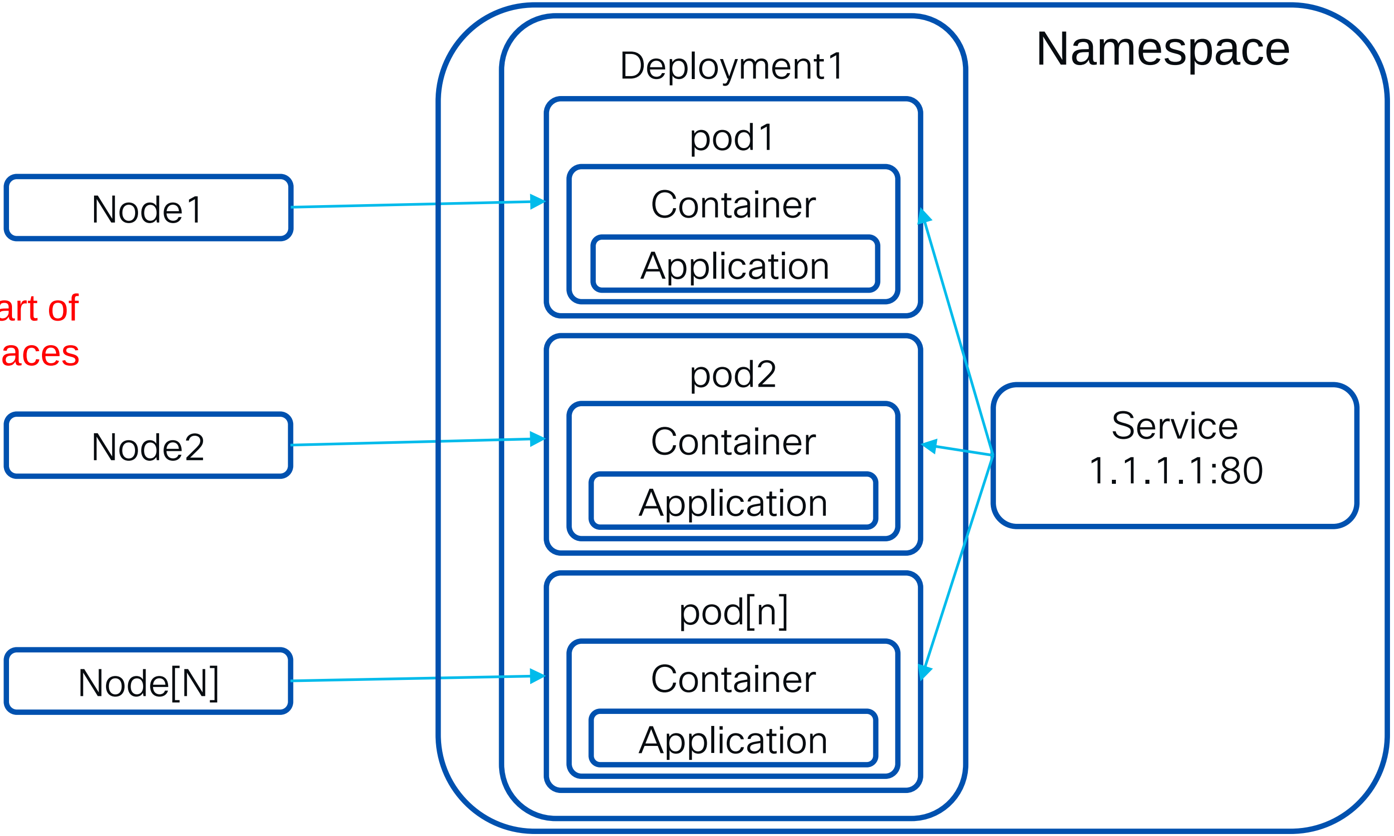
- Similar to labels but are NOT used to identify and select object

```
[root@k8s-01-p1 ~]# kubectl describe node k8s-01-p1 | more
Name:                 k8s-01-p1
Role:
Labels:                     beta.kubernetes.io/arch=amd64
                      beta.kubernetes.io/os=linux
                      kubernetes.io/hostname=k8s-01-p1
                      node-role.kubernetes.io/master=
Annotations:                node.alpha.kubernetes.io/ttl=0
                      opflex.cisco.com/pod-network-ranges={"V4":[{"start":"11.2.0.130","end":"11.2.1.1"}]}
                      opflex.cisco.com/service-endpoint={"mac":"66:85:9a:e9:ef:2f","ipv4":"11.5.0.3"}
                      volumes.kubernetes.io/controller-managed-attach-detach=true
```

Kubernetes – Namespace

- Groups everything together:
  - Pod
  - Deployment
  - Volumes
  - Services
  - Etc…

All Together: A K8S Cluster



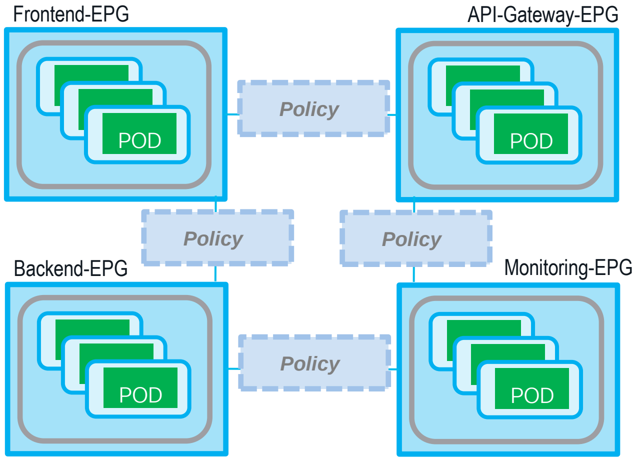
A node can be part of
Several Namespaces

Agenda

- Kubernetes Refresh
- **Kubernetes Network Challenges**
- ACI-CNI
- BGP Based Architecture
  - Calico, Cilium and Kube-Router
  - Automation and Visibility
- Which solution is right for me?
- Q&A

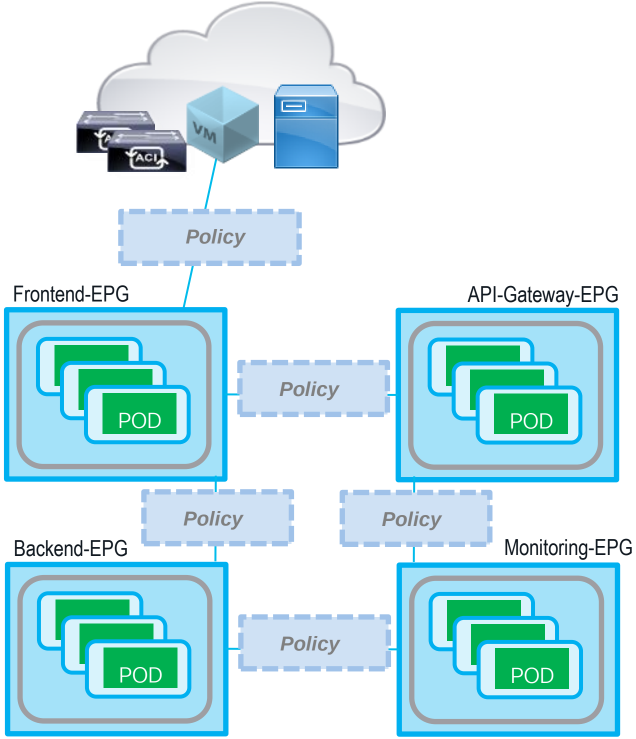
Segmentation

- 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?
- Cluster endpoints communicating with non-cluster endpoints:
  - POD access to external services and endpoints
- Cluster accessing shared resources like Storage



Agenda

- Kubernetes Refresh
- Kubernetes Network Challenges
- ACI-CNI
- BGP Based Architecture
  - Calico, Cilium and Kube-Router
  - Automation and Visibility
- Which solution is right for me?
- Q&A

ACI-CNI
Solution Overview

Why ACI-CNI 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 and Source NAT

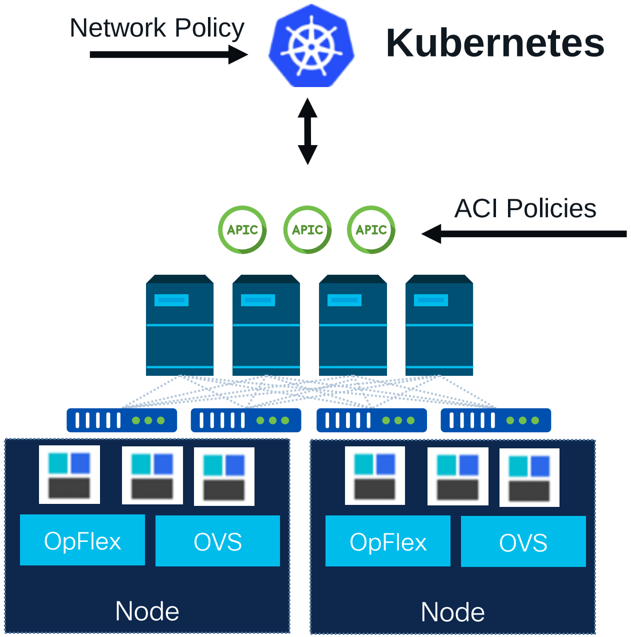
Visibility: Live statistics in APIC per container and health metrics

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

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

Cisco ACI CNI plugin features

- IP Address Management for Pods and Services
- Distributed Routing and Switching with integrated VXLAN overlays implemented fabric wide and on Open vSwitch
- Distributed Firewall for implementing Network Policies
- EPG-level segmentation for K8s objects using annotations
- Consolidated visibility of K8s networking via VMM Integration



ACI-CNI Configuration

Kubernetes Nodes will require the following interfaces

- InfraVLAN – sub-interface over which we build the opflex channel
- Node IP – sub-interface used for the Kubernetes API host IP address
- (Optional) OOB Management – sub-interface or physical interface used optionally for OOB access.

acc-provision – configuration file (1)

```
aci_config:
  system_id: Kubernetes     # Tenant Name and Controller Domain Name
  apic_hosts:               # List of APIC hosts to connect for APIC API
  - 10.67.185.102
  vmm_domain:               # Kubernetes VMM domain configuration
    encap_type: vxlan       # Encap mode: vxlan or vlan
    mcast_range:             # mcast range for BUM replication
      start: 225.22.1.1
      end: 225.22.255.255
    mcast_fabric: 225.1.2.4
    nested_inside:          # (OPTIONAL) If running k8s node as VMs specify the VMM Type and Name.
      type: vmware
      name: ACI
    # The following resources must already exist on the APIC,
      # they are used, but not created by the provisioning tool.
      aep: ACI_AttEntityP       # The AEP for ports/VPCs used by this cluster
      vrf:                  # The VRF can be placed in the same Tenant or in Common.
        name: vrf1
        tenant: KubeSpray       # This can be the system-id or common
      l3out:
        name: l3out               # Used to provision external IPs
        external_networks:
        - default_extepg        # Default Ext EPG, used for PBR redirection
```

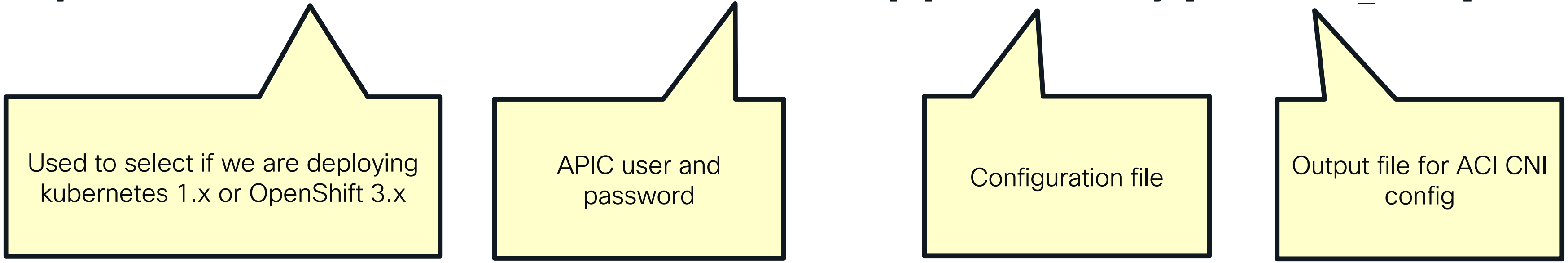
acc-provision – configuration file (2)

```
#
# Networks used by Kubernetes
#
net_config:
  node_subnet: 10.32.0.1/16      # Subnet to use for nodes
  pod_subnet: 10.33.0.1/16       # Subnet to use for Kubernetes Pods
  extern_dynamic: 10.34.0.1/24   # Subnet to use for dynamic external IPs
  extern_static: 10.35.0.1/24    # Subnet to use for static external IPs
  node_svc_subnet: 10.36.0.1/24  # Subnet to use for service graph
  kubeapi_vlan: 4011             # The VLAN used by for nodes to node API communications
  service_vlan: 4013             # The VLAN used by LoadBalancer services
  infra_vlan: 3456               # The ACI infra VLAN used to establish the OpFlex tunnel with the leaf
```

acc-provision

- ACI Container Controller Provision:
  - Takes a YAML file containing the parameters of your configuration
  - Generates and pushes most of the ACI config
  - Generates Kubernetes ACI CNI containers configuration

```
acc-provision --flavor=kubernetes-1.25 -a -u admin -p pass -c config.yml -o cni_conf.yml
```



Used to select if we are deploying kubernetes 1.x or OpenShift 3.x

APIC user and password

Configuration file

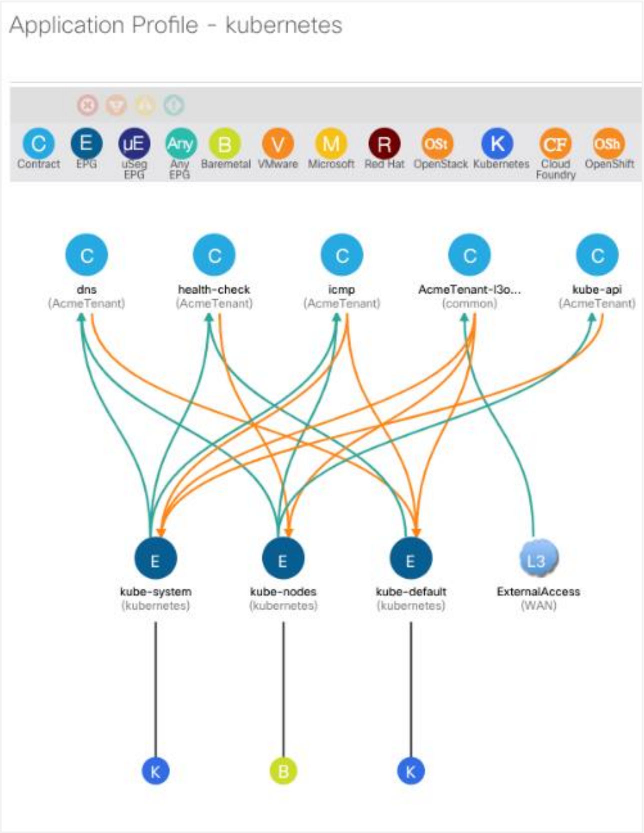
Output file for ACI CNI config

acc-provision will now create

EPGs for nodes and Pods

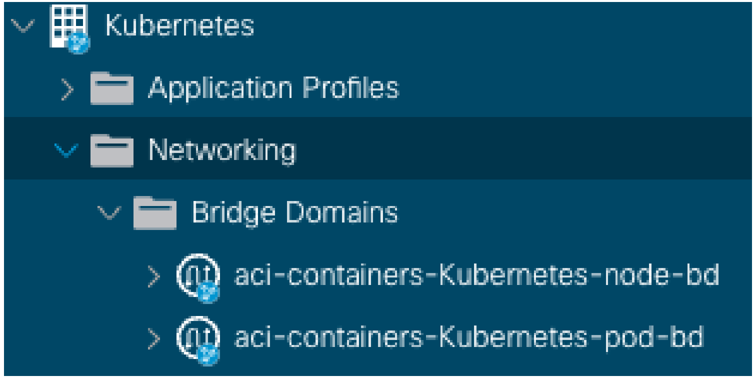
Within the tenant selected the provisioning tool creates a ‘Kubernetes’ Application Profile with three EPGs:

- for the node interfaces
- for the system PODs
- Default EPG for all containers on any namespace

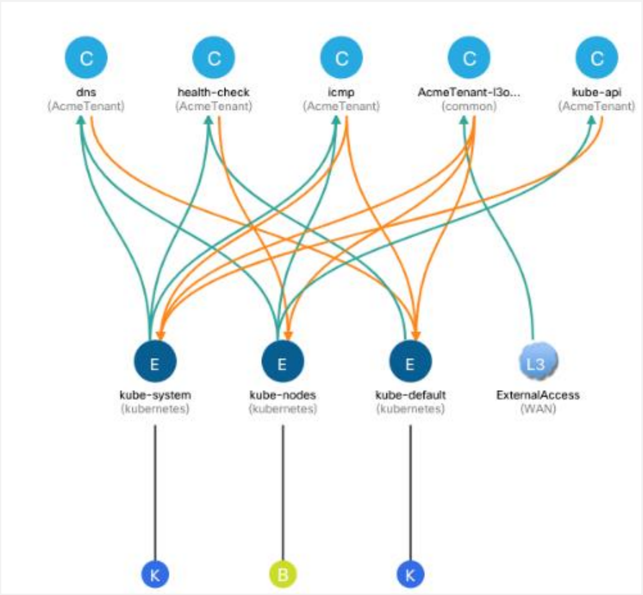


BDs and Contracts

- minimum set of contracts to ensure basic cluster functionality and security

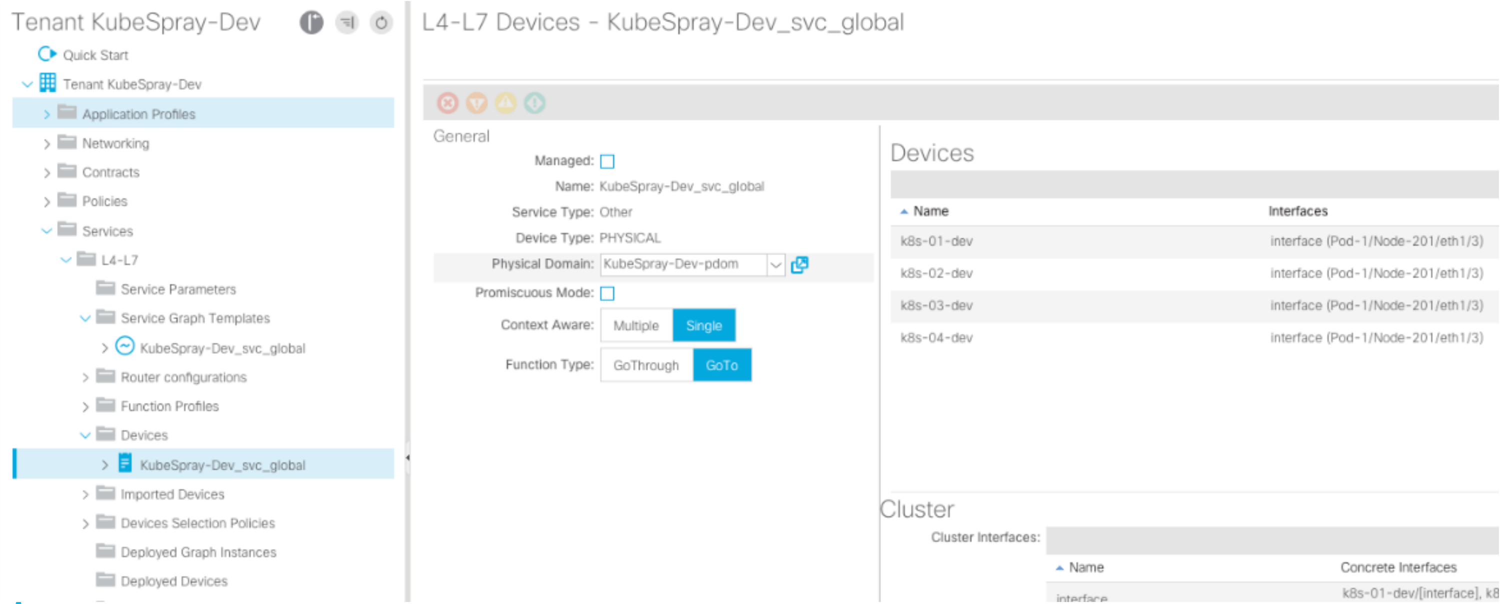


- Kubernetes
  - Application Profiles
  - Networking
    - Bridge Domains
      - aci-containers-Kubernetes-node-bd
      - aci-containers-Kubernetes-pod-bd



L4-L7 Devices

- Dynamically updated if nodes are added or removed from the k8s cluster
- Service Graph Template



L4-L7 Devices - KubeSpray-Dev_svc_global

General

Managed: □

Name: KubeSpray-Dev_svc_global

Service Type: Other

Device Type: PHYSICAL

Physical Domain: KubeSpray-Dev-pdom

Promiscuous Mode: □

Context Aware: Multiple [✓ Single]

Function Type: GoThrough [✓ GoTo]

Devices

Name	Interfaces
k8s-01-dev	interface (Pod-1/Node-201/eth1/3)
k8s-02-dev	interface (Pod-1/Node-201/eth1/3)
k8s-03-dev	interface (Pod-1/Node-201/eth1/3)
k8s-04-dev	interface (Pod-1/Node-201/eth1/3)

Cluster

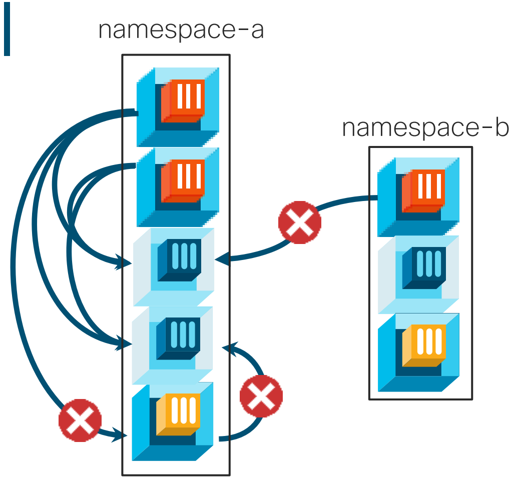
Cluster Interfaces:

Name	Concrete Interfaces
interface	k8s-01-dev/[interface], k8

ACI-CNI:
K8s Security Model

Support for Network Policy in ACI

- 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)

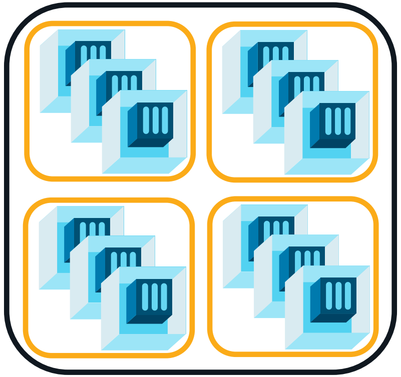


Policy applied to namespace: namespace-a

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

Mapping Network Policy and EPGs

Cluster Isolation

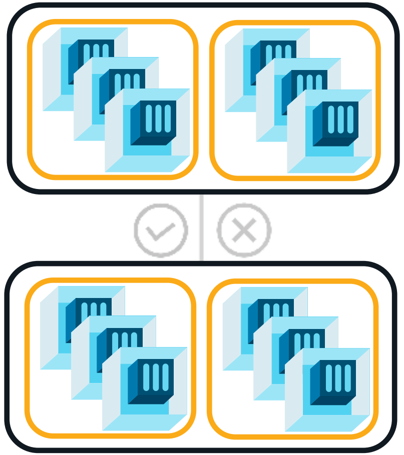


Single EPG for entire cluster.

(Default behavior)

No need for any internal contracts.

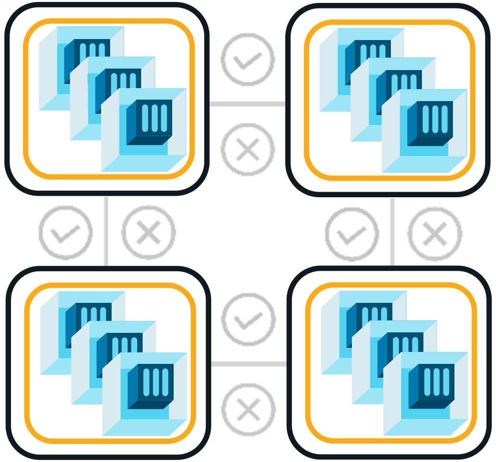
Namespace Isolation



Each namespace is mapped to its own EPG.

Contracts for inter-namespace traffic.

Deployment Isolation



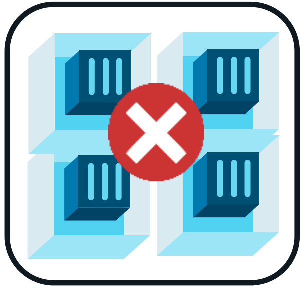
Each deployment mapped to an EPG

Contracts tightly control service traffic

Key Map EPG NetworkPolicy ✓ | ✗ Contract

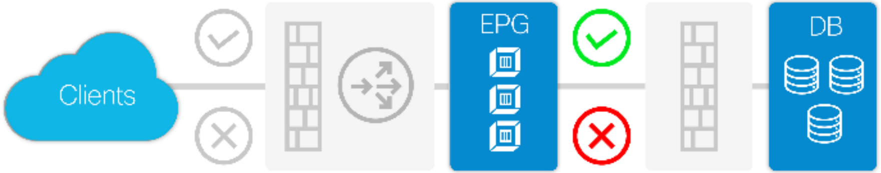
Dual level Policy Enforcement by ACI

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.

Exposing Services

Automated LoadBalancing

- Create a service of type “LoadBalancer” (as per K8s standard)
- ACI CNI will:
  - Allocate an external IP from a user-defined subnet
  - Deploy a Service Graph with PBR redirection to LoadBalance the traffic between any K8s Nodes that have PODs for the exposed service

```
cisco@k8s-01:~/demo/guestbook1$ kubectl --namespace=guestbook get svc frontend
NAME        CLUSTER-IP     EXTERNAL-IP    PORT(S)         AGE
frontend    10.37.0.124    10.34.0.5      80:32677/TCP    5h
```

extern_ip

POD SNAT

POD Networking - Recap

- During the ACI CNI installation a Bridge Domain with a dedicated subnet is created for your POD Networking.
- Every POD that is created in the Kubernetes cluster will be assigned an IP address from the POD Subnet.
- In most cases this is an advantage to other CNI implementation:
  - the POD IP/Subnet is equivalent to any other IP/Subnet in ACI
  - The POD/IP Subnet can communicate directly with any other IP/Subnet directly connect to ACI
  - The POD/IP Subnet can communicate directly with any other IP/Subnet outside of ACI via a standard L3OUT
  - Your PODs are first class citizen in ACI!

Some challenges

Challenge 1: External Firewall Configuration

- The POD IP is ephemeral:
  - It is not possible to predict what IP address a POD will be assigned.
  - It is not possible to manually assign an IP to a POD
- This standard Kubernetes behavior but it does not work well with firewalls:
  - Is not possible to configure the firewalls ACLs based on the POD IPs as the POD IP can change at any time.
- The same Kubernetes cluster can host multiple applications:
  - It is not possible to use the POD subnet as security boundary

Challenge 2: POD Subnet Routing

- The POD Subnet is, most likely, a private subnet
- In certain scenarios a POD might need communicated with an external environment (i.e. internet) and the POD IP address needs to be natted

ACI CNI SNAT to the rescue!

- POD Initiated traffic can be natted to an IP address selected by the user
  - SNAT IP: Single IP or Range
  - Ability to apply SNAT Policy at different levels:
    - Cluster Level: connection initiated by any POD in any Namespaces is natted to the selected SNAT IP
    - Namespace: connection initiated by any POD in the selected Namespaces is natted to the selected SNAT IP
    - Deployment: connection initiated by any POD in the selected Deployment is natted to the selected SNAT IP
    - LoadBalanced Service: connection initiated by any POD mapped to an external Service of Type LoadBalance are natted to the external Service IP.

Agenda

- Kubernetes Refresh
- Kubernetes Network Challenges
- ACI-CNI
- **BGP Based Architecture**
  - Calico, Cilium and Kube-Router
  - Automation and Visibility
- Which solution is right for me?
- Q&A

BGP Based Architecture

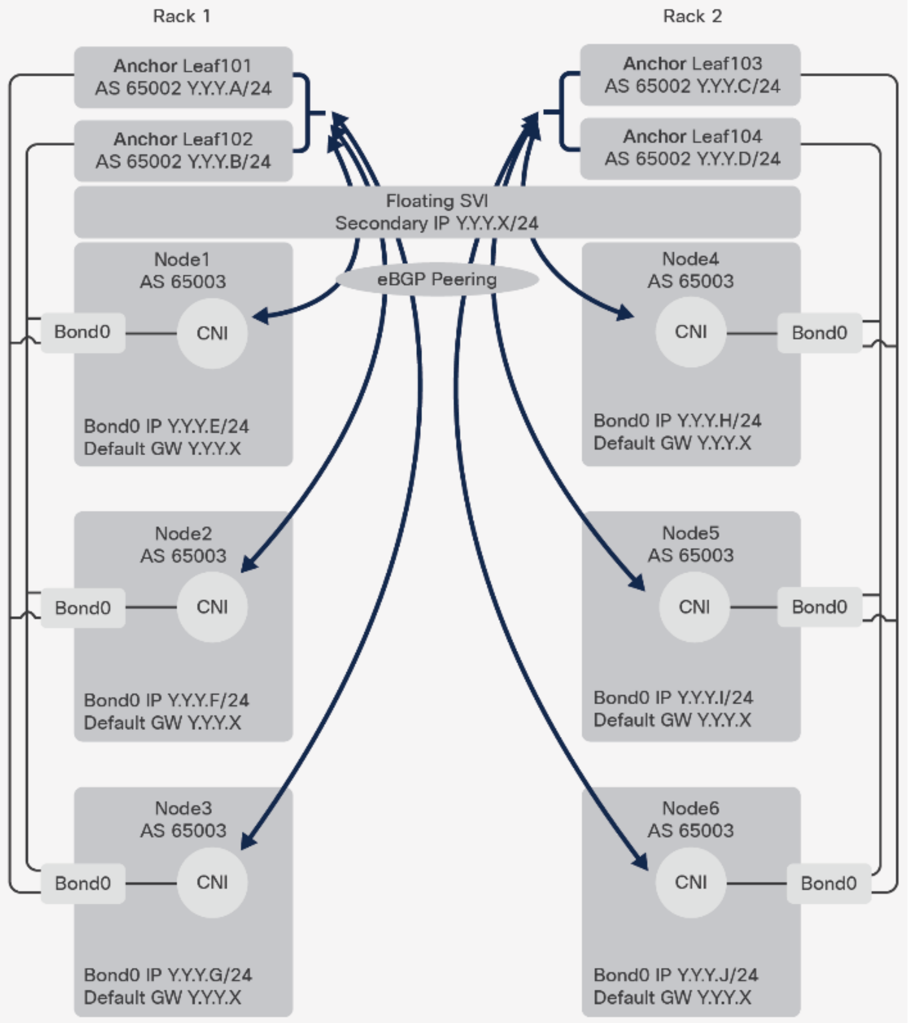
BGP Based Integration Benefits – why?

1. **Relies on a well-established protocol (BGP)**
2. **Unified networking**: Node, Pod and Service endpoints are accessible from an L3OUT providing easy connectivity across and outside the fabric
3. **(Limited) ACI Security:** ability to use external EPG classification to secure communications to Node/Pod/Service Subnets (no /32 granularity)
4. **High performance**: low-latency connectivity without egress routers if no Overlay are used
5. **Hardware-assisted load balancing**: ECMP up to 64 paths/Nodes
6. **Any Hypervisor/Bare Metal:** allows to mix form factors together

Architecture and Configuration

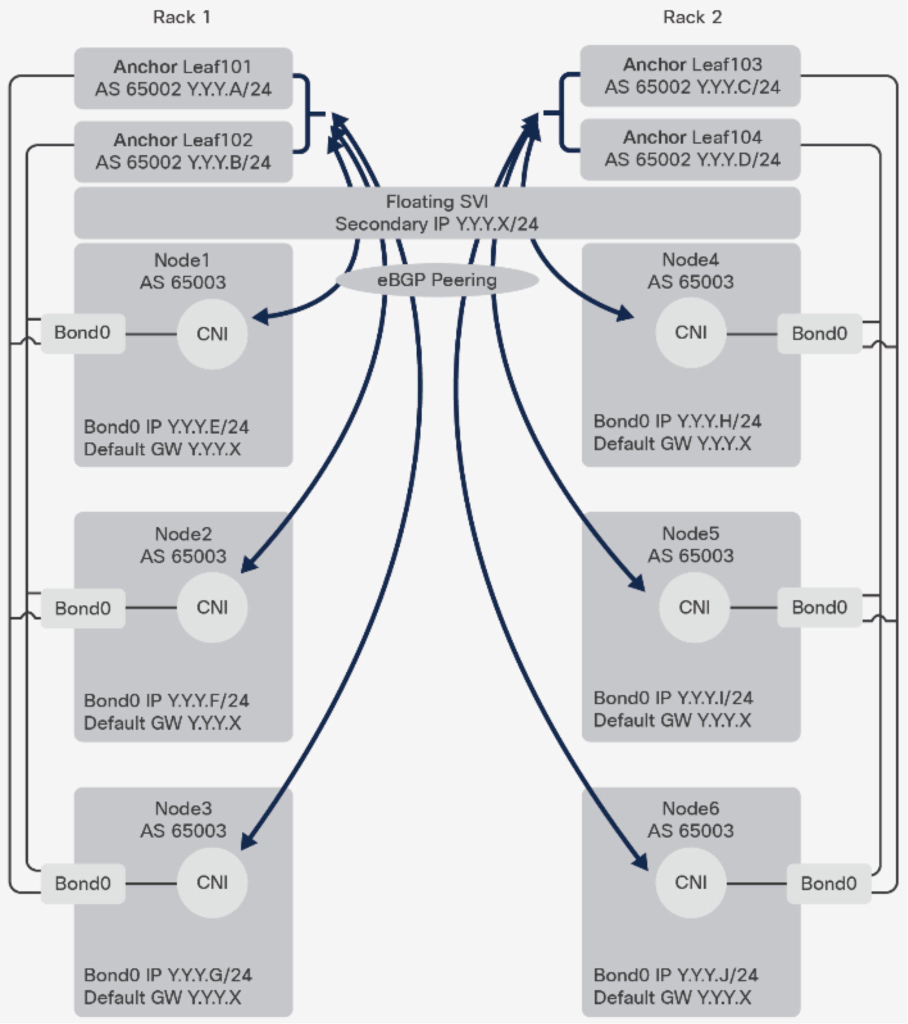
Architecture

- Each K8s Node will peer with a pair of border leaves
- Single AS for the whole cluster
    - Simpler ACI config (can use a subnet for passive peering)
- CNI Advertise all the K8s subnets to ACI as well as host-routes for exposed services leveraging ECMP for LoadBalancing



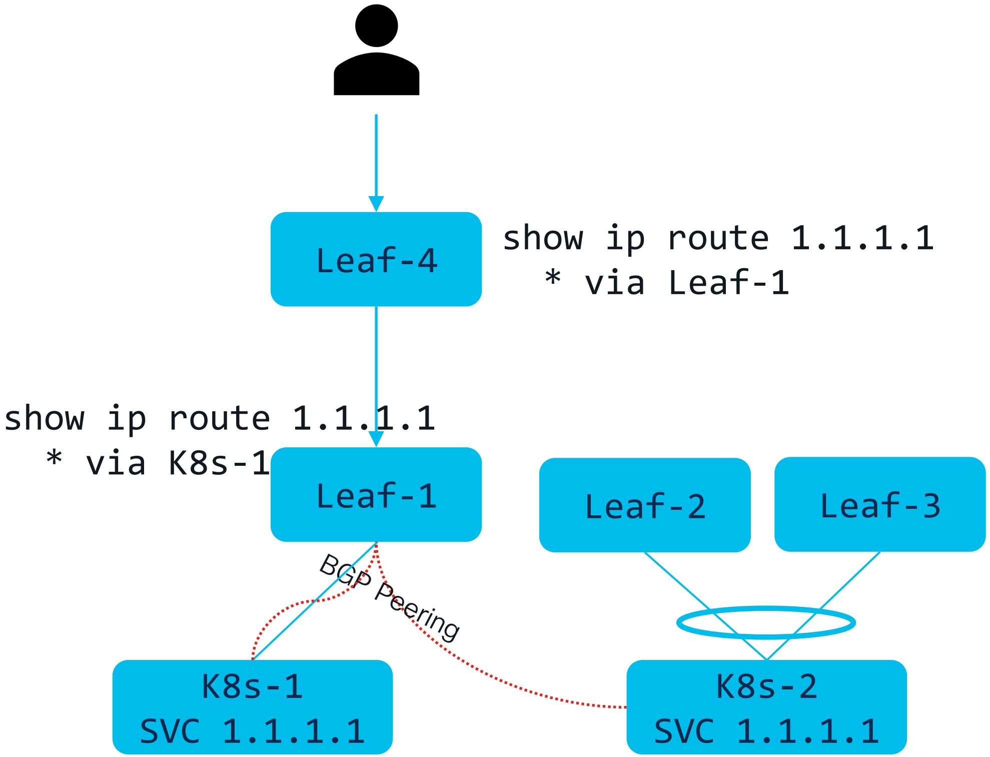
L3OUT Design

- K8s Nodes are connected to an L3OUT via vPC
  - External EPGs can be used to classify the traffic coming from the cluster
- Floating L3OUT
  - VM Mobility
  - Ability to mix BareMetal and VMs running on any hypervisor



ACI Best Practice: Peer to local ToR

- If some K8s nodes are connected to **Anchor** and some to **Non-**Anchor Leaves and are advertising the same Service IP only the one connected to the **Anchor** Leaves are selected as valid next hop.
- This happens because the Next-Hop cost is higher for to Non-Anchor Leaves connected K8s Nodes.
- We are working to address this in an upcoming ACI release



ACI BGP Tuning

- AS override and Disable Peer AS Check: To support having a single AS per cluster without the presence of Route Reflectors or Full Mesh inside the cluster
- BGP Graceful Restart
- BGP timers tuned to 1s/3s for quick eBGP node down detection
- Relax AS path policy to allow installing more than one ECMP path for the same route
- Increase Max BGP ECMP path to 64 for better load balancing

ACI BGP Hardening (Optional)

- Enabled BGP password authentication
- Set the maximum AS limit to one
- Configure BGP import route control to accept only the expected subnets from the Kubernetes cluster:
  - Pod subnet(s)
  - Node subnet(s)
  - Service subnet(s)
- Set a limit on the number of received prefixes from the nodes.

Expected Routing Behaviour

- Nodes, pods and service IPs Subnets will be advertised to the ACI fabric
- Every K8s nodes is allocated one or more subnets from the POD Supernet. Each subnets is advertised to ACI as well
- Exposed Services will be advertised to ACI as host routes from every nodes that has a running POD associated to the service.

Agenda

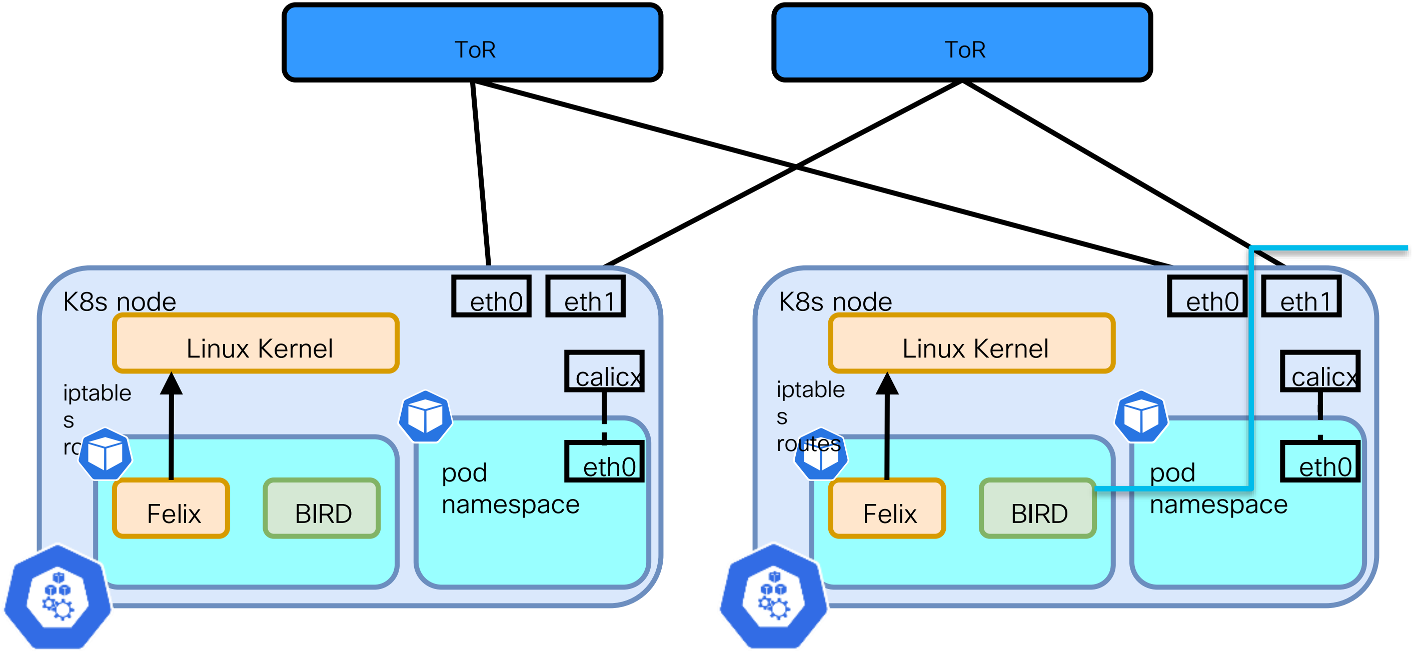
- Kubernetes Refresh
- Kubernetes Network Challenges
- ACI-CNI
- BGP Based Architecture
  - **Calico, Cilium and Kube-Router**
  - Automation and Visibility
- Which solution is right for me?
- Q&A

What is Calico

A Kubernetes CNI plugin

Calico

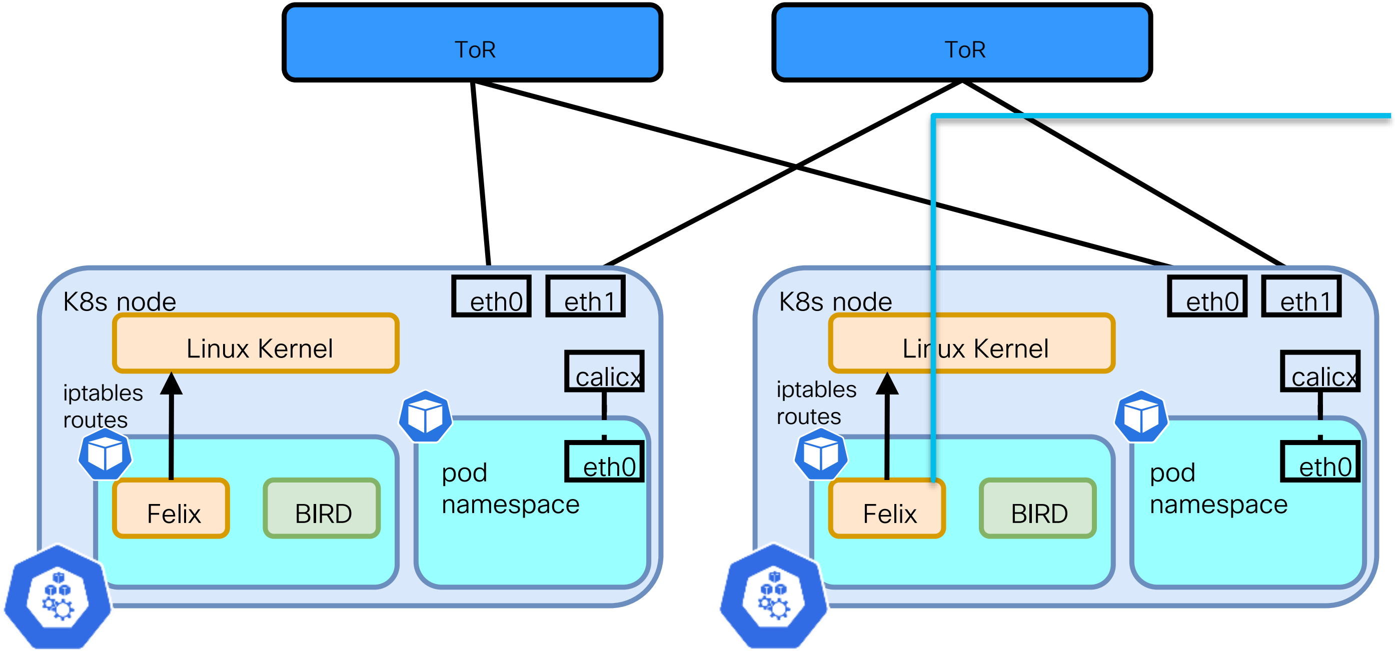
A CNI plugin of Kubernetes



BIRD: It is a routing daemon responsible for peering with other K8s nodes and exchanging routes of pod network and service network for inter-node communication.

Calico

A CNI plugin of Kubernetes



Felix: Running in same pod as BIRD, programs routes and ACLs (iptables) and anything required on Calico node to provide connectivity for the pods scheduled on that node

Calico BGP Config

- One or more IPPool with all overlays disabled
- BGPConfiguration with:
  - nodeToNodeMeshEnabled set to “false”
  - List of serviceClusterIPs and serviceExternalIPs subnets to enabled host routes advertisement for those subnets
- BGPPeer to define the BGP Peer the K8s nodes connects to
- A Secret, Role and RoleBinding to pass the BGP Password to the Calico BGP Process

Calico IPPool Config – Cont.

```
apiVersion: crd.projectcalico.org/v1
kind: IPPool
metadata:
name: default-ipv4-ippool
spec:
  blockSize: 26 ──────────────→ How to split the POD subnet between nodes
  cidr: 192.168.3.0/24 ───────→ POD Subnet
  ipipMode: Never ────────────→ Disable IP in IP
  nodeSelector: all() ────────→ Allocate this Subnet to all the nodes
  vxlanMode: Never ───────────→ Disable VXLAN Overlay
```

Calico BGPConfiguration – Cont.

```
apiVersion:
crd.projectcalico.org/v1
kind: BGPConfiguration
metadata:
  name: default
spec:
  asNumber: 65003
  listenPort: 179
  logSeverityScreen: Info
  nodeToNodeMeshEnabled: false
  serviceClusterIPs:
  - cidr: 192.168.4.0/24
  serviceExternalIPs:
  - cidr: 192.168.5.0/24
```

K8s Cluster BGP AS

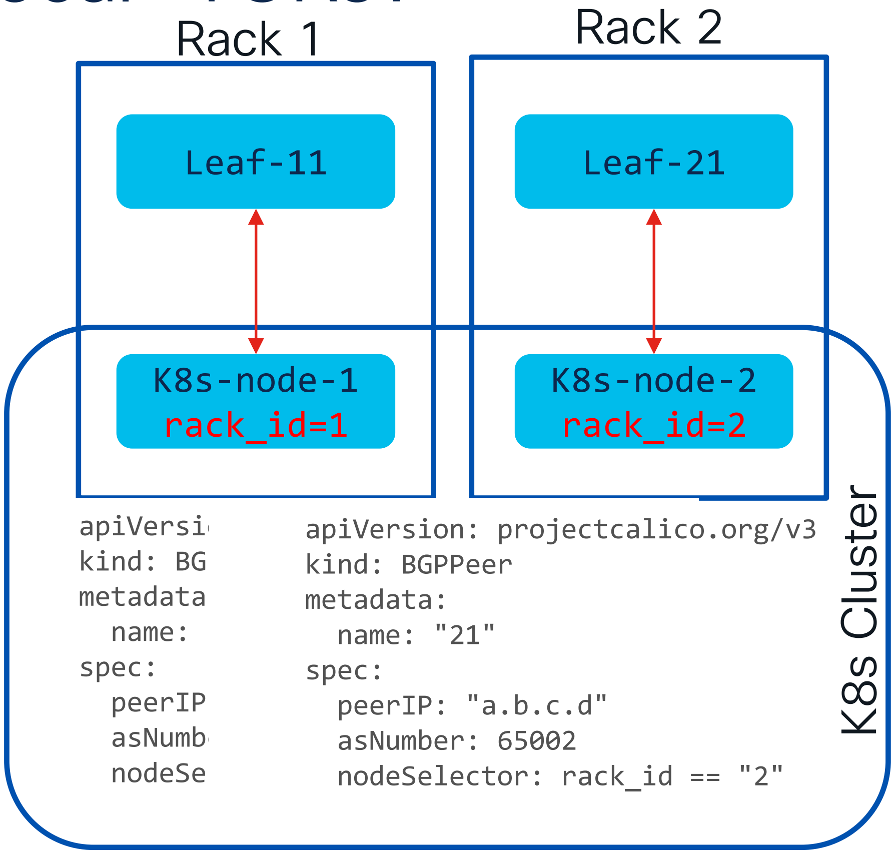
BGP Port

Disable iBGP Full Mesh Peering

Allow Calico to Advertise the Cluster and External service Subnets

How do I peer with the “local” TORs?

- Use a Node label to identify the location of the K8s Node, for example the rack id
- Configure the BGPPeer resource with a *nodeSelector* matching the label of the K8s Nodes
- The result will be that the peering is happening only between K8s Nodes and leaves with a matching rack id



```
apiVersi
kind: BG
metadata
  name:
spec:
  peerIP
  asNumb
  nodeSe
```

```
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: "21"
spec:
  peerIP: "a.b.c.d"
  asNumber: 65002
  nodeSelector: rack_id == "2"
```

BGP CNI Demo

Agenda

- Kubernetes Refresh
- Kubernetes Network Challenges
- ACI-CNI
- BGP Based Architecture
  - Calico, Cilium and Kube-Router
  - **Automation and Visibility**
- Which solution is right for me?
- Q&A

Automation and Visibility Demo

Agenda

- Kubernetes Refresh
- Kubernetes Network Challenges
- ACI-CNI
- BGP Based Architecture
  - Calico, Cilium and Kube-Router
  - Automation and Visibility
- **Which solution is right for me?**
- Q&A

Which solution is right for me?

Which solution is right for me?

- This is an hard question, all the supported CNI plugins are enterprise grade and provide a rich feature set
- Some question you might ask yourself:
  - Is running the same CNI plugin on ANY network infrastructure important?
  - Is the K8s team already using a specific CNI ?
  - Are the advanced ACI CNI feature important?
    - Ability to place PODs into EPG, SNAT, PBR based load balancing
  - Is having a single vendor for the networking stack support important ?

CNI Comparison

	ACI CNI	Calico	Kuber-Router	Cilium
Support	TAC	OpenSource/Pay	OpenSource	OpenSource/Pay
Network Infra	ACI Only	Any	Any	Any
Network Config	Automated	Automated	Automated	Automated
Linux OS Support	Ubuntu/CoreOS	Any	Any	Any
Service LoadBalancing	ACI PBR	BGP ECMP	BGP ECMP	BGP ECMP
POD SNAT	NAT Pools	NAT To Node IP	NAT To Node IP	NAT To Node IP
Security Model	ACI Policy Model + K8s Network Policies	Calico Network Policies	K8s Network Policies	Extended K8s Network Policies
End To End Visibility	Yes	Via Opensource Tools	Via Opensource Tools	Hubble
Data Plane	OVS	Linux or eBPF	Linux + IPVS for LoadBalancing	eBPF

The bridge to possible

Thank you

cisco Live!

Let’s go