CalicoCon 2019

November 18, 2019

KubeCon Co-Located Event

Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

PROJECT CALICO

Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

A

Welcome to CalicoCon

About Today!

PROJECT CALICO

CalicoCon: What We Will Accomplish Today

- What You'll Learn
  - Working knowledge of Kubernetes Networking and Network Policy
- How You Can Apply These Learnings
  - Segment Clusters for multiple users & teams
  - Deploy applications to a Zero Trust Architecture
  - Integrate with your enterprise network

Sponsored by Tigera

Inventors and lead maintainers of open source Project Calico

Products and Services

PROJECT CALICO

Networking
& Network Policy

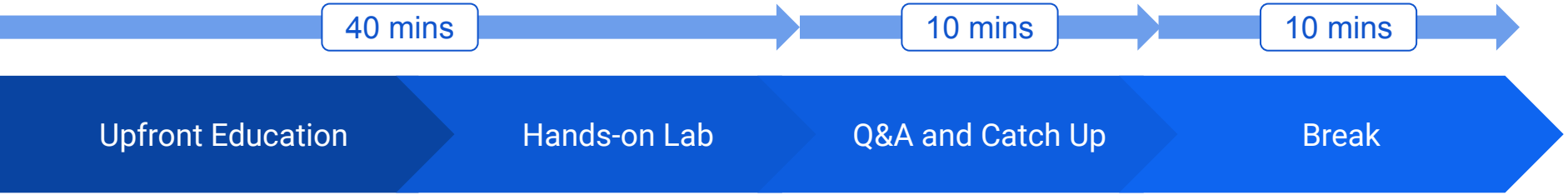
CALICO ESSENTIALS

Kubernetes Journey
Acceleration

CALICO ENTERPRISE

Enterprise Teams,
Visibility,
Network Security,
Compliance

Lab Structure



Training on concepts related to the lab.

What we are going to accomplish in this lab.

Each student walks through lab exercise.

Lab exercise is located in GitHub.

Summary & additional considerations.

Extra time to catch up

Q&A

We will take a 10 minute break between labs.

WIFI

Network Name: `LogDNALovesDevs`

Password: `cloudnative`

How to Get Help

Stuck on a step? Broken lab environment? We're here to help!

slack

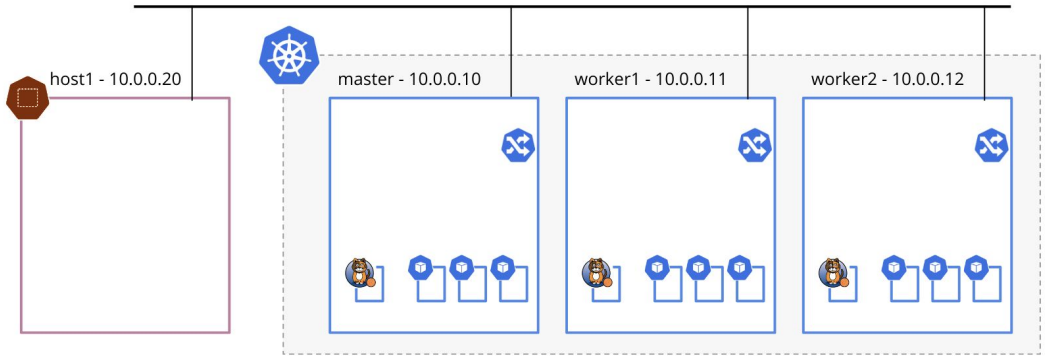
https://calicousers.slack.com/

#calicocon-2019

Lab Environment

4 Hosts

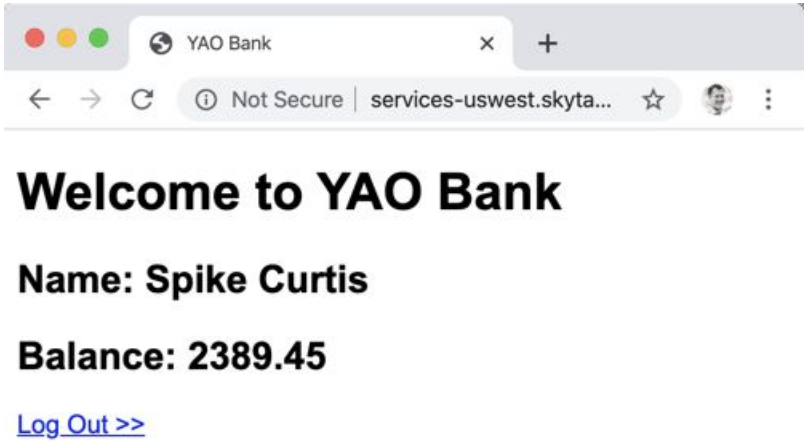
- Standalone Host (and jump server you will work from)
- Kubernetes Master
- 2 Kubernetes Workers



Sample Application

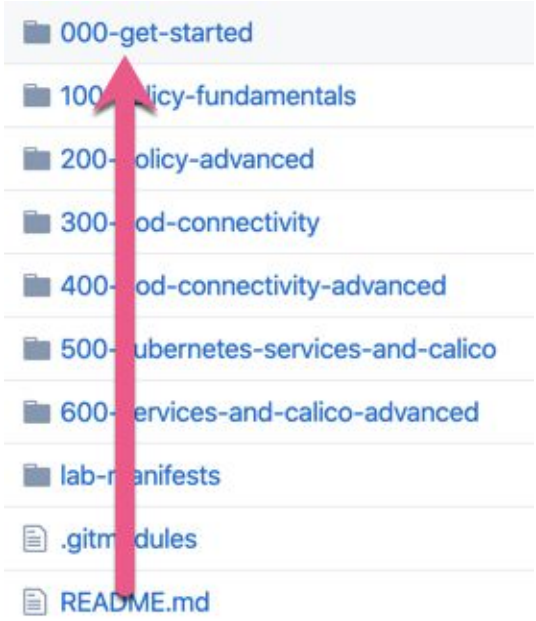
- Yet Another Online Bank (YAO Bank)

YAO Bank Sample Application – Pods



Lab Notebook

https://github.com/projectcalico/calicocon



Accessing Your Lab

Check Your Email for 2 Links (from andy.wright@tigera.io)

1. Link to HTML5 terminal on your jump server
2. Link to YaoBank web frontend you will install

Don't see the link? Ask for help in Slack.

The Platform Operators Journey in Kubernetes

Education | **Lab / Sandbox** | **Pilot / Pre - Production** | **Prod - Initial Apps** | **Prod - Mass Migration**

Focus for CalicoCon Workshops Today

Education	Lab / Sandbox	Pilot / Pre - Production	Prod - Initial Apps	Prod - Mass Migration
Get basic cluster working with networking	Network policy Policy best practices Managing trust across teams	Understand networking at scale Integrate with enterprise network	Understand services at scale Select best options for service types Integrate services with rest of infrastructure	
000 Lab	**100 & 200 Lab**	**300 & 400 Lab**	**500 & 600 Lab**	

The Platform Operators Networking Journey

Get a working network	Figure out security	Optimize network
Nothing works until you have this! Get something working Overlay is fine - it just needs to work **000 Lab**	Network policy Policy best practices Managing trust across teams **100 & 200 Lab**	Understand the nuts and bolts on network Get rid of overlays Integrate with enterprise network **300 & 400 Lab** Understand the nuts and bolts on services Select best options for service types **500 & 600 Lab**

Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

PROJECT CALICO

B

State of the K8s

The Big Picture

PROJECT CALICO

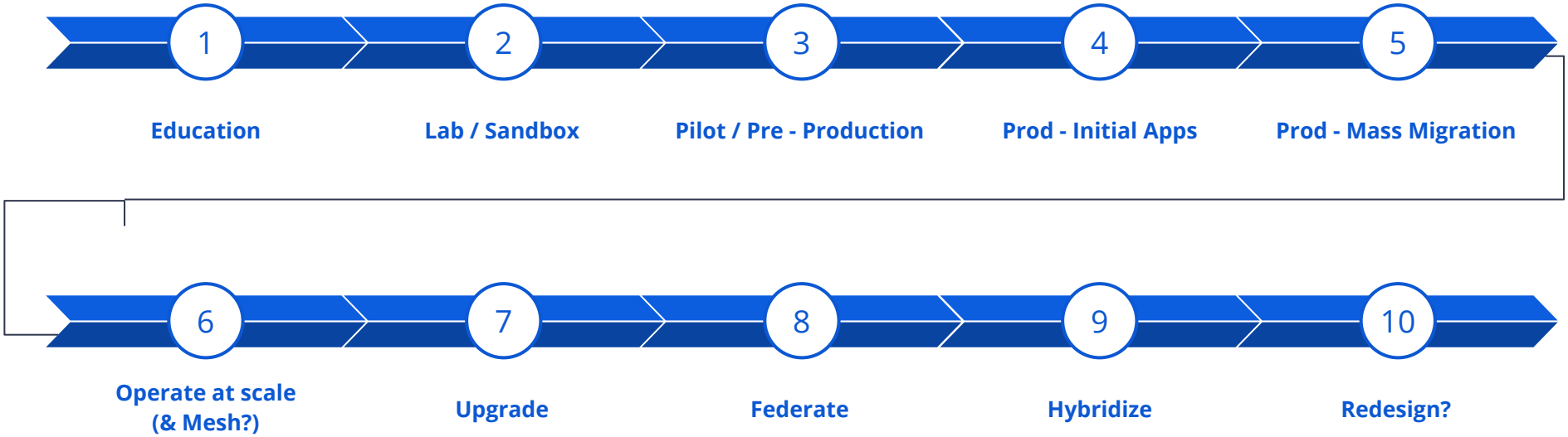

DNS & Service discovery
Network Policy
Services & kube-proxy
Ingress & gateways
CNI & Pod Networks
Service Mesh

The Platform Operators Journey in Kubernetes

Education | **Lab / Sandbox** | **Pilot / Pre - Production** | **Prod - Initial Apps** | **Prod - Mass Migration**

The Platform Operators Journey in Kubernetes

(Wait, there's more! a.k.a: Why didn't we think of this earlier?!)



Focus for CalicoCon WorkshopsToday

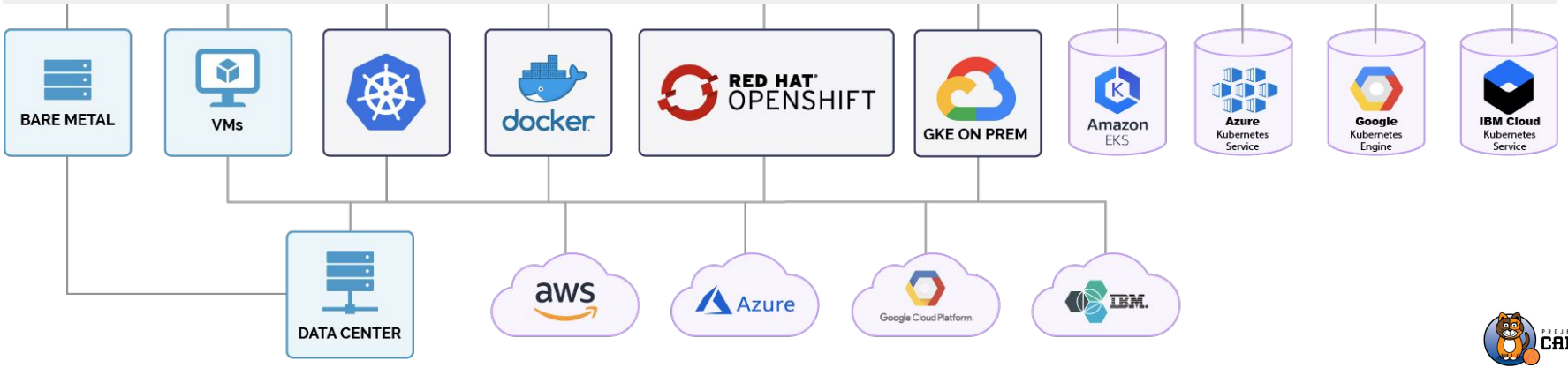
Education	Lab / Sandbox	Pilot / Pre - Production	Prod - Initial Apps	Prod - Mass Migration
Get basic cluster working with networking	Network policy Policy best practices Managing trust across teams	Understand the nuts and bolts of networking Integrate with enterprise network	Understand the nuts and bolts on services Select best options for service types Integrate services with rest of infrastructure
000 Lab	**100 & 200 Lab**	**300 & 400 Lab**	**500 & 600 Lab**

Calico Overview

- **Most popular network policy implementation for Kubernetes**
- **Provided as the default by major Kubernetes hosted services**
  - AWS EKS, Azure AKS, Google GKE, IBM Cloud IKS, etc.
- **Included in many popular Kubernetes distributions**
  - Docker Enterprise, Rancher, Canonical, etc.
- **Reference implementation of Kubernetes network policy**
  - Complete implementation of Kubernetes Network Policy
  - Provides a superset of network policy features allowing for real-world operator controls
  - Implementation tuned for robustness and Kubernetes scale
- **CNI plugin providing elegant, scalable pod network connectivity leveraging standard network designs**
  - Provides common encapsulation options but also flat, non-encapsulated IP connectivity if desired
  - Enables advanced pod and service connectivity configurations

PROJECT CALICO

Widely Deployed at Scale in Advanced, Multi-Cloud Environments



PROJECT CALICO

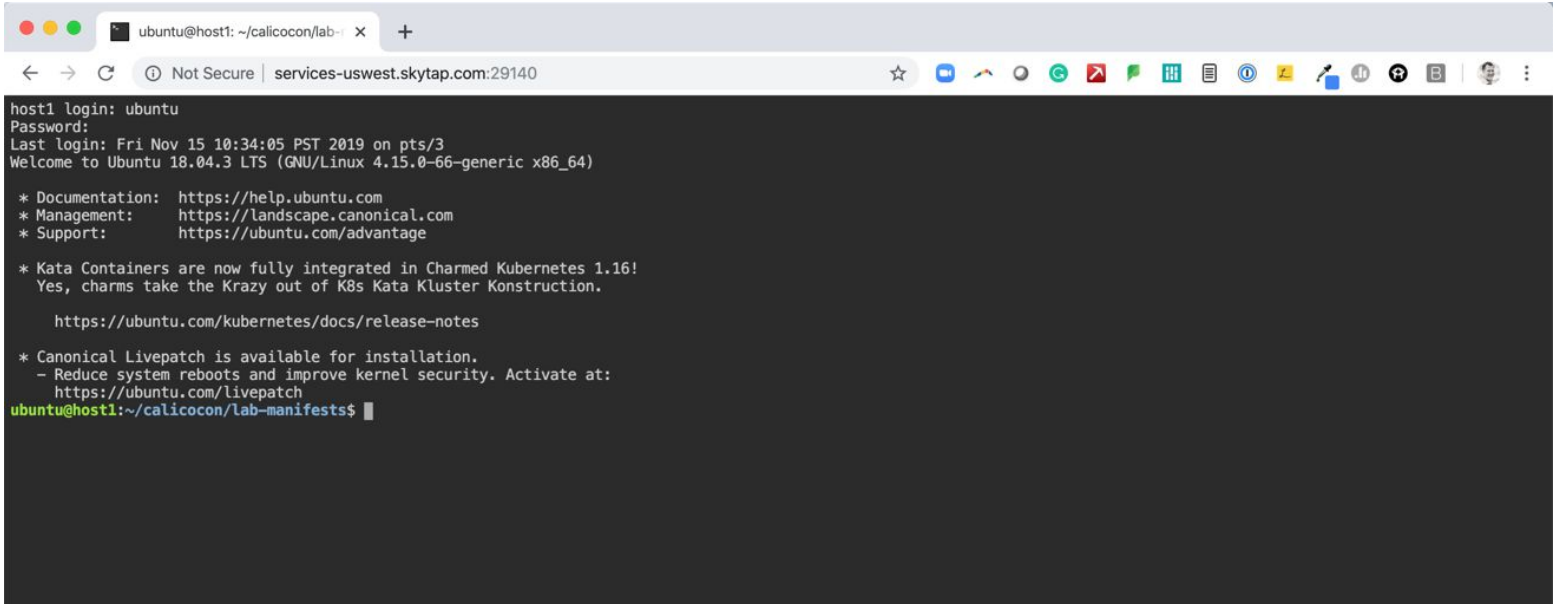
Let's Get Started

000-get-started

Let's Get Started: Login to Your Jump Server

Username: ubuntu

Password: %TigeraCalicoCon19%



Prepare Your Jump Server for Remote Support

tmux (terminal multiplexer) enables others to collaborate in the same terminal

```
tmux new -t calico
```

Download the Lab Notebooks and Manifests

The manifest files you will be using are on GitHub

```
git pull
```

Install Calico

Calico will be the networking (CNI) and network policy implementation used today

```
kubectl apply -f 010-calico.yaml
```

Launch Sample Application (YaoBank)

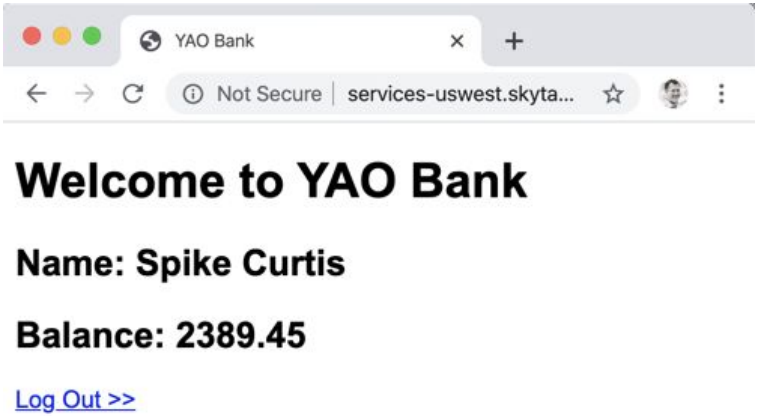

```
kubectl apply -f 110-yaobank.yaml
```

Access YaoBank

Make sure your pods are running

```
kubectl get pods -n yaobank
```

Browse to the "Web Application" URL sent with your lab login details.



Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

PROJECT CALICO

C

Policy Fundamentals

PROJECT CALICO

Why is Network Policy important?

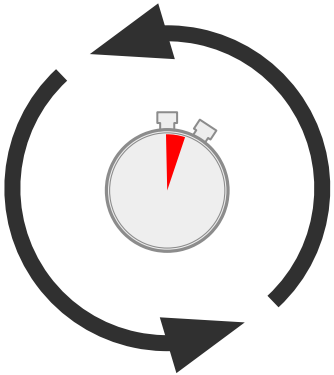
- The Kubernetes network model is a “flat” network
  - Pods on a node can communicate with all pods on all nodes without NAT
- Dynamic scheduling of pods
  - IP addresses are ephemeral
  - Pod locations are non-deterministic (*normally)
- Securing the perimeter is not enough

=> Requires a new approach to network security

Network Policy

Label-based

Declarative

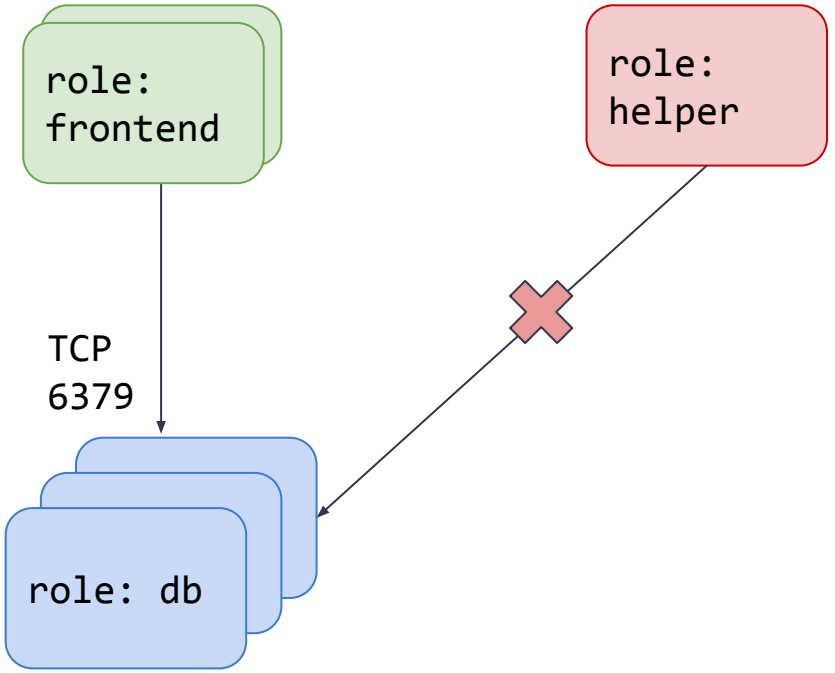


Dynamic

Calico

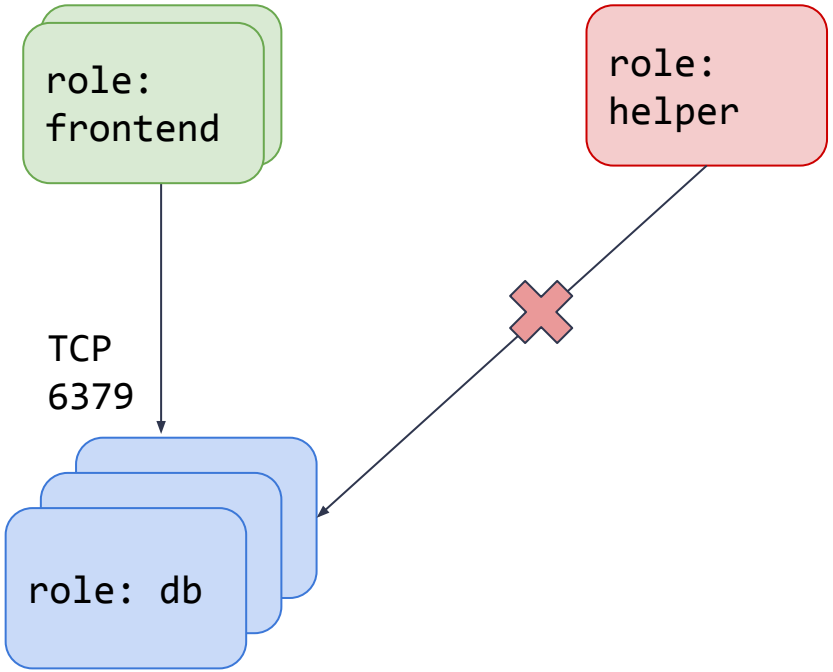
Kubernetes Network Policy Example

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: my-network-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: db
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
```



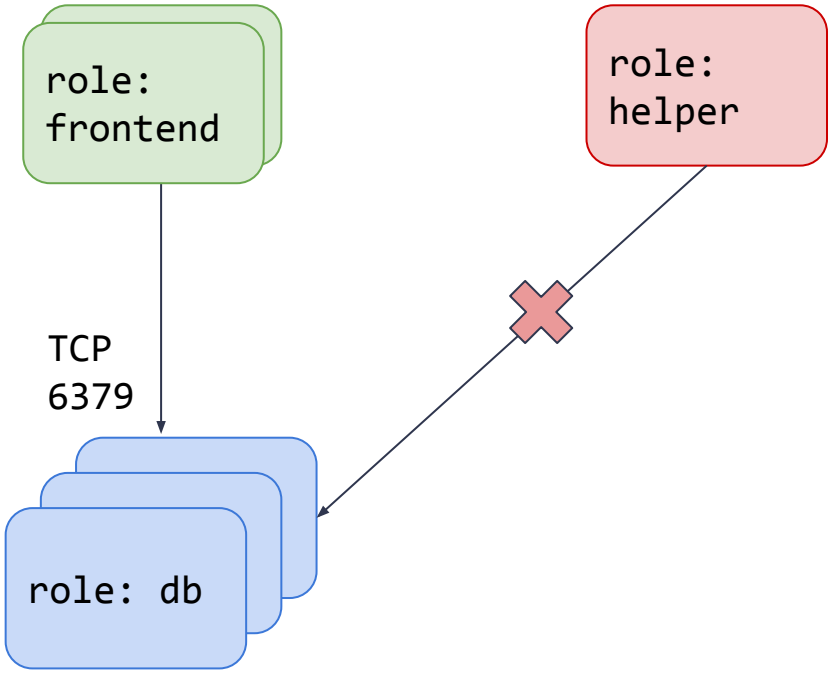
Calico Policy Model

```
apiVersion: projectcalico.org/v3
kind: NetworkPolicy
metadata:
  name: my-network-policy
  namespace: my-namespace
spec:
  order: 900
  selector: role == "helper"
  ingress:
    - action: Deny
      protocol: TCP
      source:
        serviceAccounts:
          names:
            - sre-account
          selector: (role == "db")||(stage == "dev")
    - action: LOG
      protocol: ICMP
      source:
        namespaceSelector: color == "green"
```



Calico Policy Model with Application Layer Rules

```
apiVersion: projectcalico.org/v3
kind: NetworkPolicy
metadata:
  name: my-network-policy
  namespace: my-namespace
spec:
  order: 900
  selector: role == "helper"
  ingress:
    - action: Deny
      protocol: TCP
      source:
        namespaceSelector: color == "green"
        serviceAccounts:
          names:
            - sre-account
          selector: (role == "db")||(stage == "dev")
      http:
        methods:
          - POST
        paths:
          - exact: /table/*/customers
```



Compare this approach with virtual network security appliances

Calico cloud-native approach

- Enforced close to workload (container network interface and/or service mesh)
- Enforcement inline with existing datapath
- Kubernetes pod (daemonset) agent
- Rules dynamically calculated based on declarative policy and workload labels
- Changes applied in milliseconds
- Integrates with Kubernetes to understand workload topology, labeling, namespaces, etc

Traditional network security approach

- Enforced in separate firewall appliance
- Enforcement requires hairpin / additional hops
- Virtual machine
- Static policy model dependent on IP ranges
- Changes require manual process (weeks)
- No knowledge of container orchestration metadata

Policy Considerations in Production

- IPSets w/ IPTables (& eBPF)
- XDP (& eBPF)
- Network Policy
- Application Layer Policy (& Service Mesh)
- Dynamic Workload Identity
- Distributed Policy Compute from Centralized Intent
- Scale

Takeaways!

- Kubernetes Network Policy enables applications to declare segmentation controls to restrict access to authorized workloads
- Calico Network Policy is a sophisticated superset of Kubernetes network policy that includes a number of advanced policy features facilitating real world use cases
- Offered as the *out-of-the-box* option by the major cloud providers for their hosted Kubernetes services (AWS EKS, Azure AKS, Google GKE, IBM Cloud IKS, etc.)
- Calico provides a scalable implementation of network policy and proven in very large-scale production deployments yet simple to use and operate

Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

D

Advanced Policy

PROJECT CALICO

Calico Policy: Recap

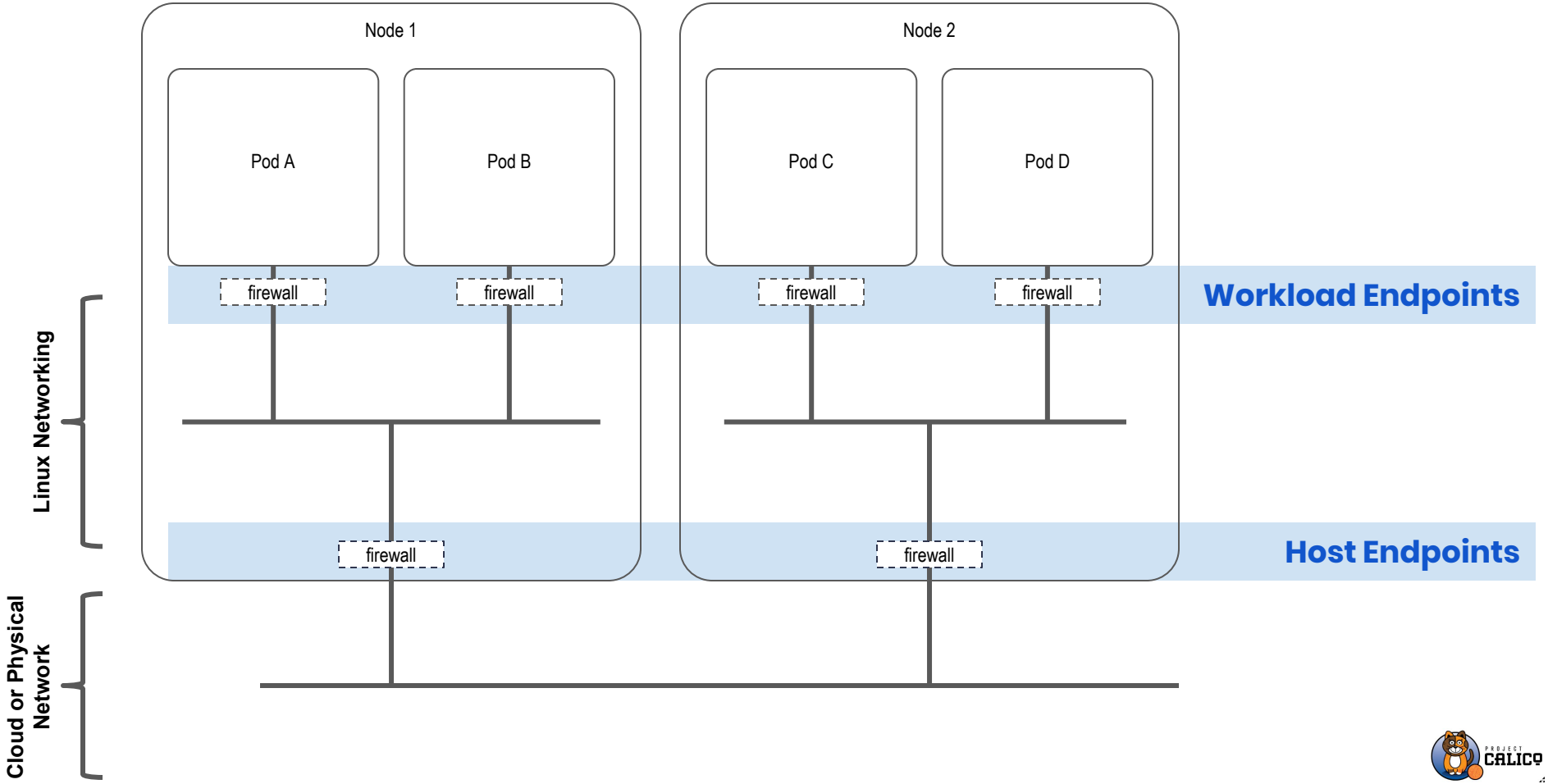
- Kubernetes Network Policy enables applications to declare segmentation controls to restrict access to authorized workloads

- **Calico Network Policy is a sophisticated superset of Kubernetes network policy** that includes a number of advanced policy features facilitating real world use cases

- **Offered as the *out-of-the-box* option by the major cloud providers** for their hosted Kubernetes services (AWS EKS, Azure AKS, Google GKE, IBM Cloud IKS, etc.)

- Calico provides a scalable implementation of network policy and **proven in very large-scale production deployments** yet simple to use and operate

Calico Network Policy Enforcement



Shift left and managing security across teams

Goals:

- Enable dev teams to define policy for microservices they own
- Enable secops teams to set higher-level guardrails

Tools:

- Calico policies can include namespace and service account label selectors
- Kubernetes RBAC allows independent control of pod, namespace, and service account permissions

Delegate trust with namespace granularity

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy

selector: app == "backend"
ingress:
  - action: Allow
    source:
      selector: app == "frontend"
egress:
  - action: Allow
```

Dev team A:

- RBAC permissions
  - ✔️k8s network policies in namespace “team-a”
  - ❌k8s namespaces
  - ❌Calico network policies
- Has inadvertently given broad access for traffic leaving the “backend” app

```
apiVersion: projectcalico.org/v3
kind: GlobalNetworkPolicy

namespaceSelector: allow-public != "true"
egress:
  - order: 1000
  - action: Deny
    destination:
      selector: netset == "public-ips"
```

Security team:

- RBAC permissions
  - ✔️k8s namespaces
  - ✔️Calico network policies
- Uses labels on namespaces to determine which namespaces are allowed to access the internet

Note: policy examples are pseudo-code (not fully correct syntax)

Delegate trust with service accounts

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy

selector: app == "backend"
ingress:
  - action: Allow
    source:
      selector: app == "frontend"
egress:
  - action: Allow
```

```
apiVersion: projectcalico.org/v3
kind: GlobalNetworkPolicy

serviceAccountSelector: allow-public != "true"
egress:
  - order: 1000
  - action: Deny
    destination:
      selector: netset == "public-ips"
```

Dev team A:

- RBAC permissions
  - ✔️k8s network policies in namespace “team-a”
  - ❌k8s namespaces
  - ❌k8s service accounts
  - ❌Calico network policies
- Has legitimate need for some pods to send traffic to the internet

Security team:

- RBAC permissions
  - ✔️k8s namespaces
  - ✔️k8s service accounts
  - ✔️Calico network policies
- Creates service accounts (with permission labels) in namespace depending on level of trust with dev team

Note: policy examples are pseudo-code (not fully correct syntax)

Label taxonomies and shift-left team interactions

Identity-based

```
selector: app == "backend"
ingress:
  - action: Allow
    source:
      selector: app == "frontend"
egress:
  - action: Allow
    destination:
      selector: app == "database"
```

Pod	Labels
frontend	app:frontend
backend	app:backend
database	app:database

Team owning the microservice must:

- understand which microservices they consume
- understand which microservices consume their microservice
- change the policy every time new microservices wants to consume

Permission-based

```
selector: app == "backend"
ingress:
  - action: Allow
    source:
      selector: be-client == "true"
egress:
  - action: Allow
    destination:
      selector: app == "database"
```

Pod	Labels
frontend	app:frontend be-client:true
backend	app:backend db-client:true
database	app:database

Team owning the microservice must:

- understand which microservices they consume
- trust teams wanting to consume their microservice

Takeaways!

- Kubernetes Network Policy is developer-centric: rules to allow TCP/UDP ports and cidrs after enabling default-deny

- Calico Network Policy provides **advanced policy features** (including controls that can be leveraged by platform cluster operators and security teams), **namespace and ServiceAccount labels with RBAC** and **actions besides "Allow"** often required by cluster operators and platform/security teams

- Calico is designed for **operational simplicity and Kubernetes scale**, and **widely proven in production**
  - Leverages the best underlying Linux kernel facilities, including **IPSets with IPTables**, **XDP** and on **Windows**, Host Networking with VFP

Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

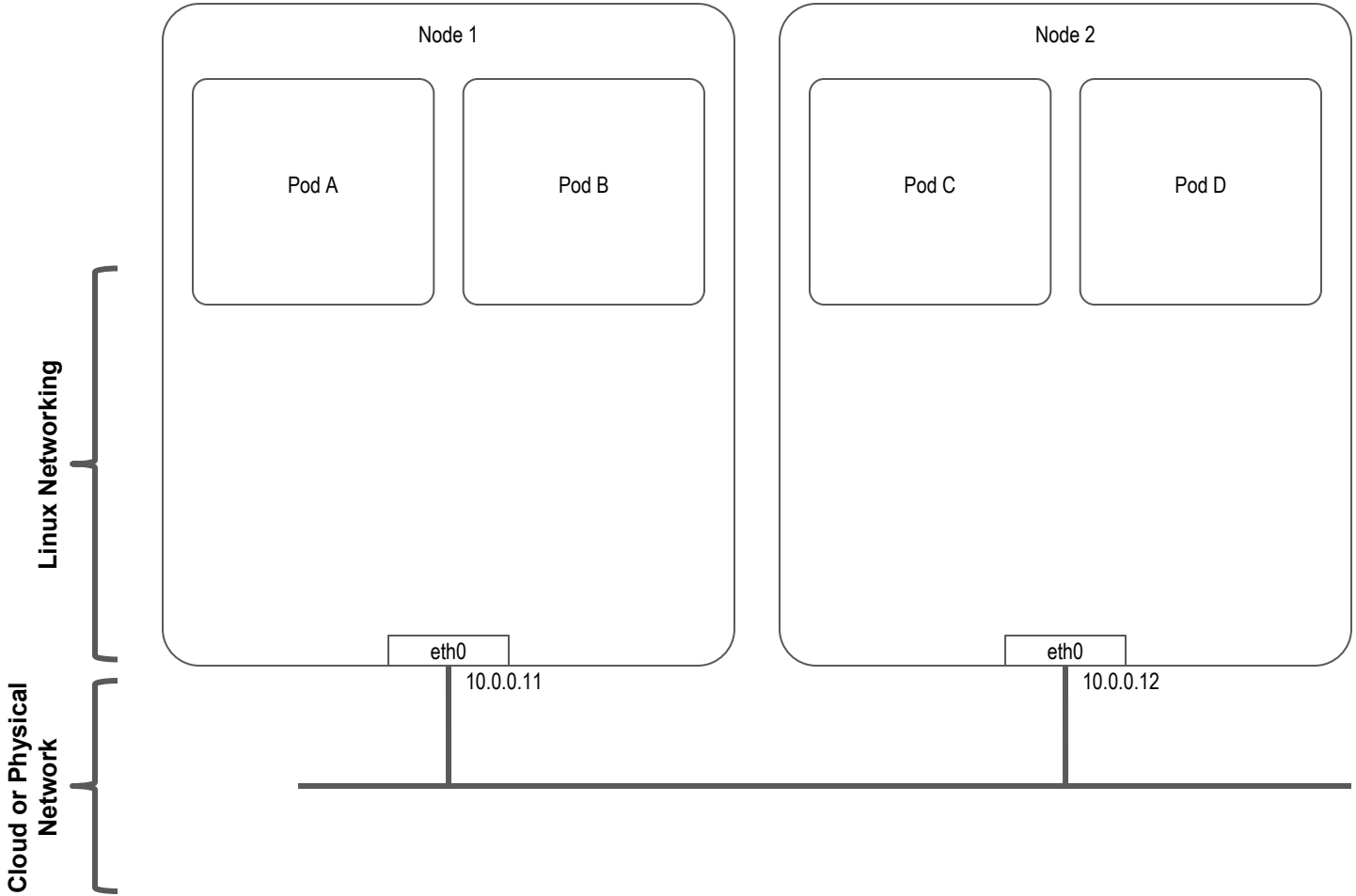
E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

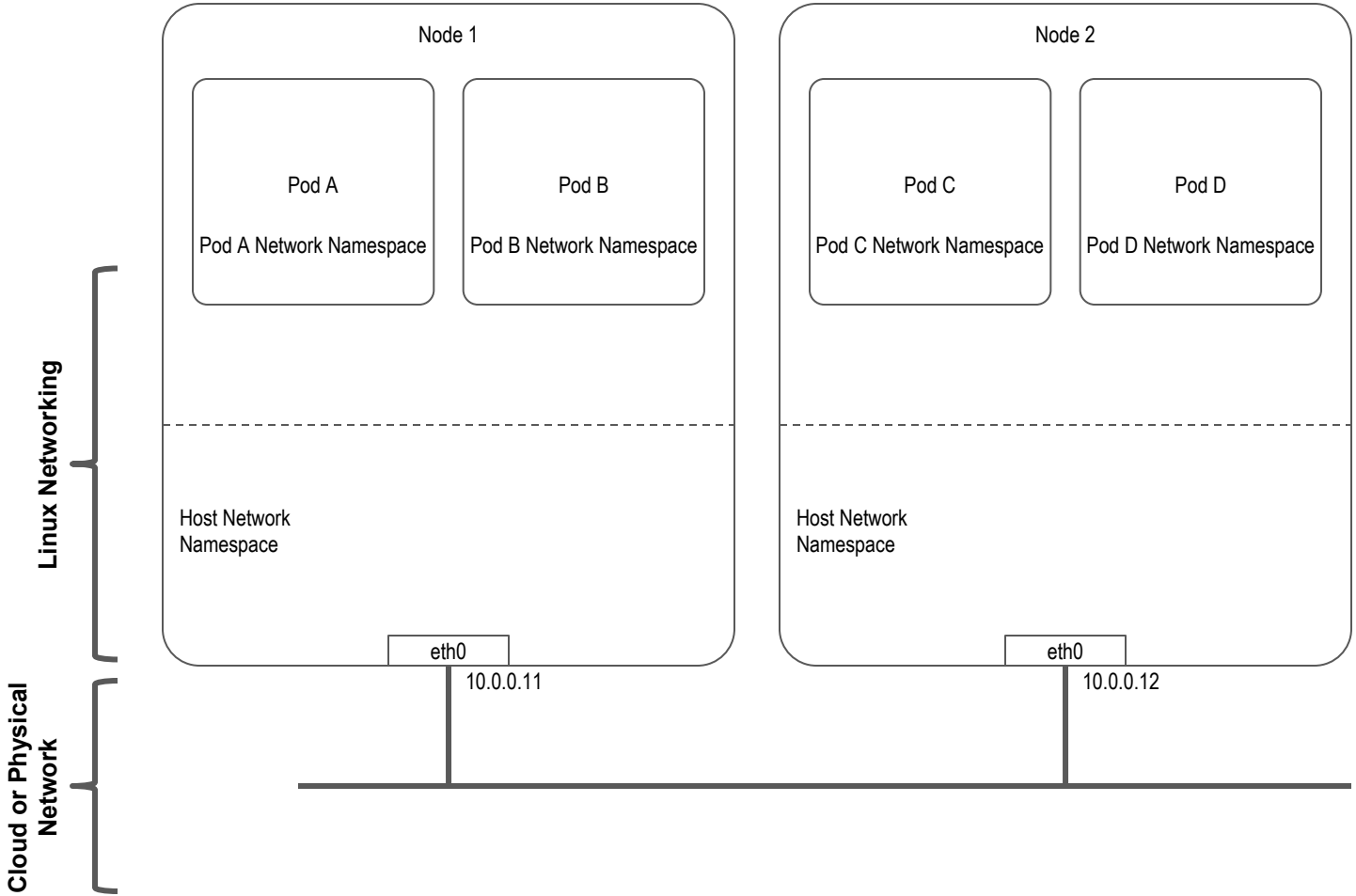
PROJECT CALICO

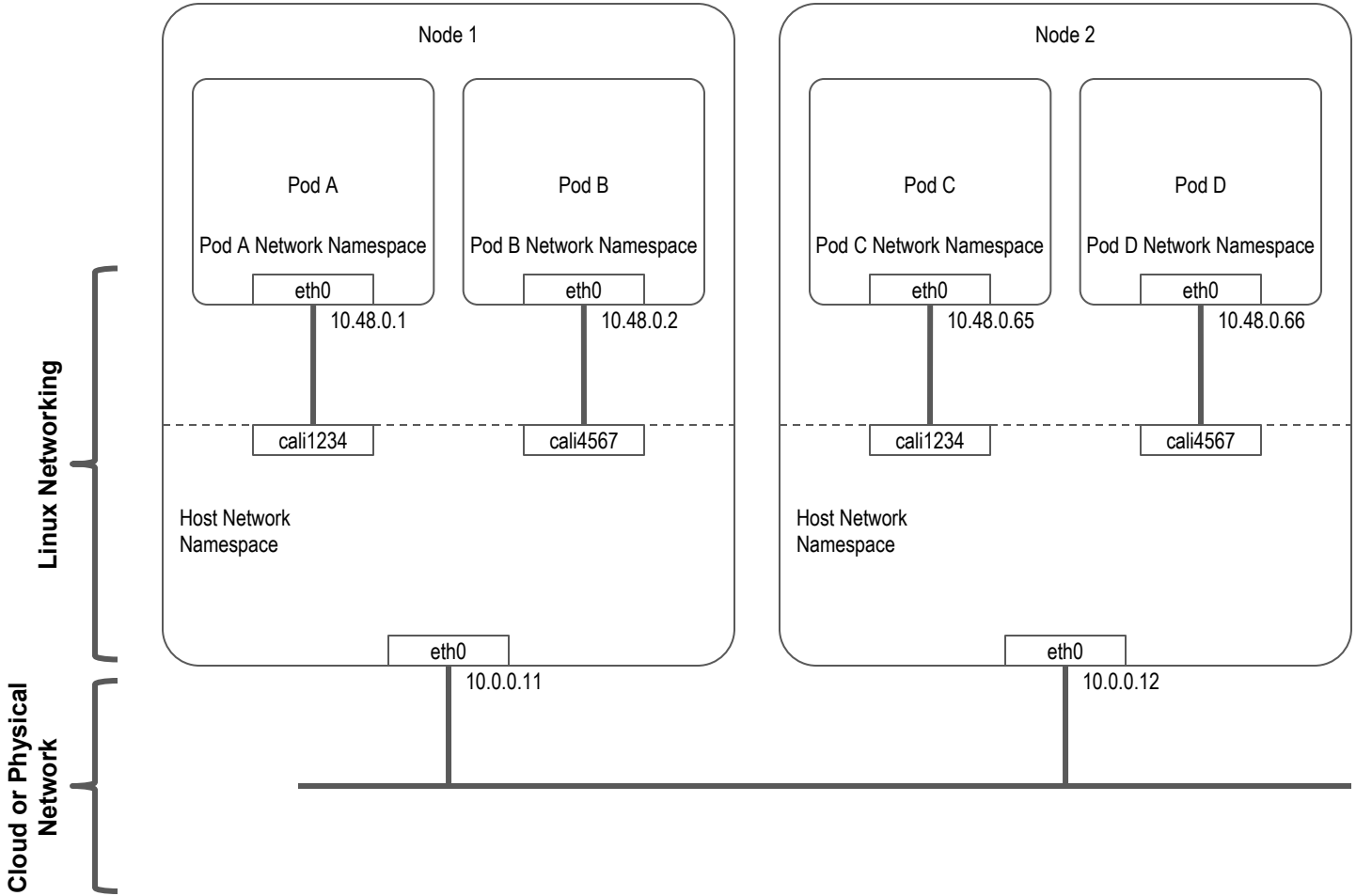
E

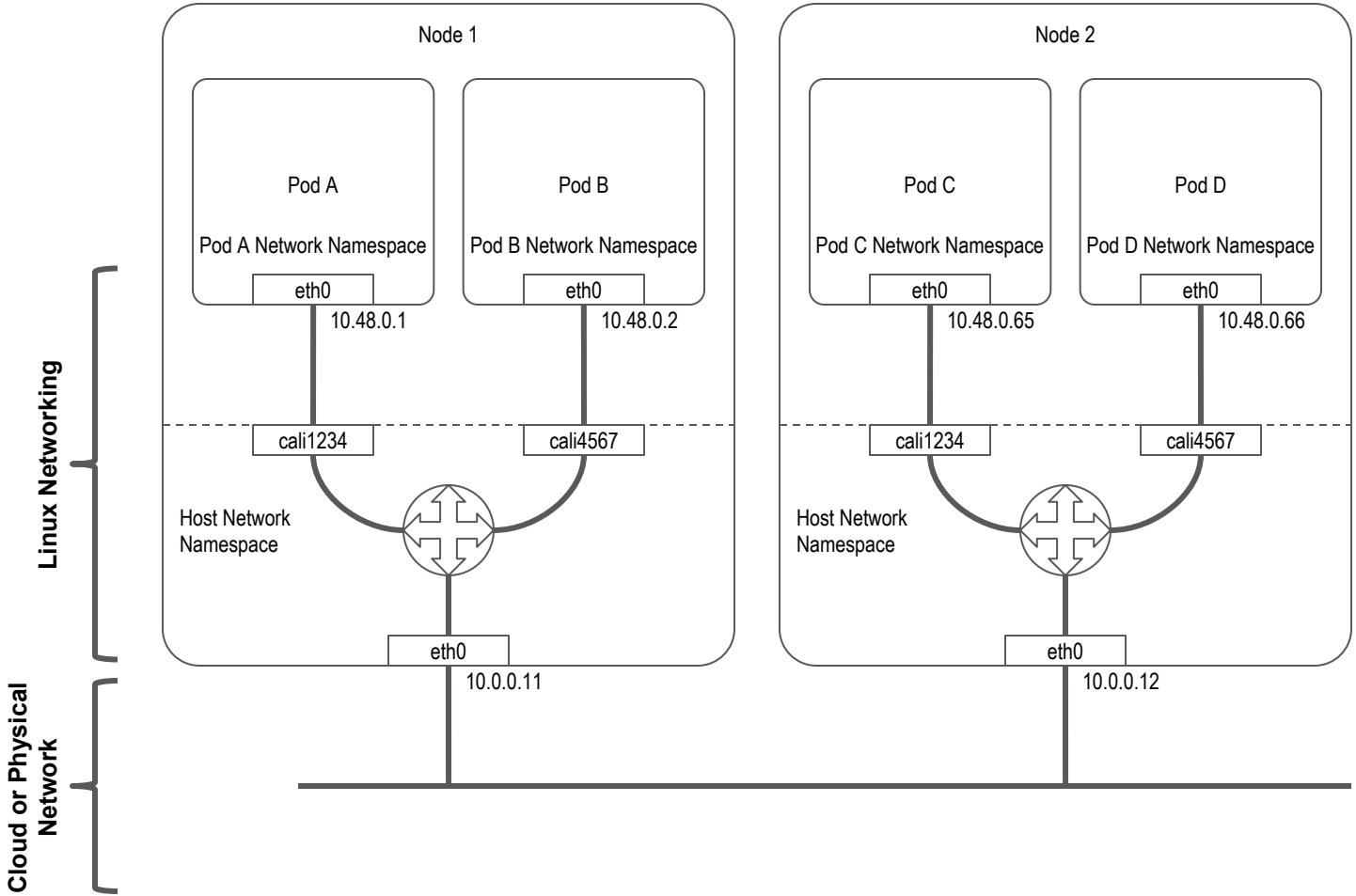
Pod Connectivity

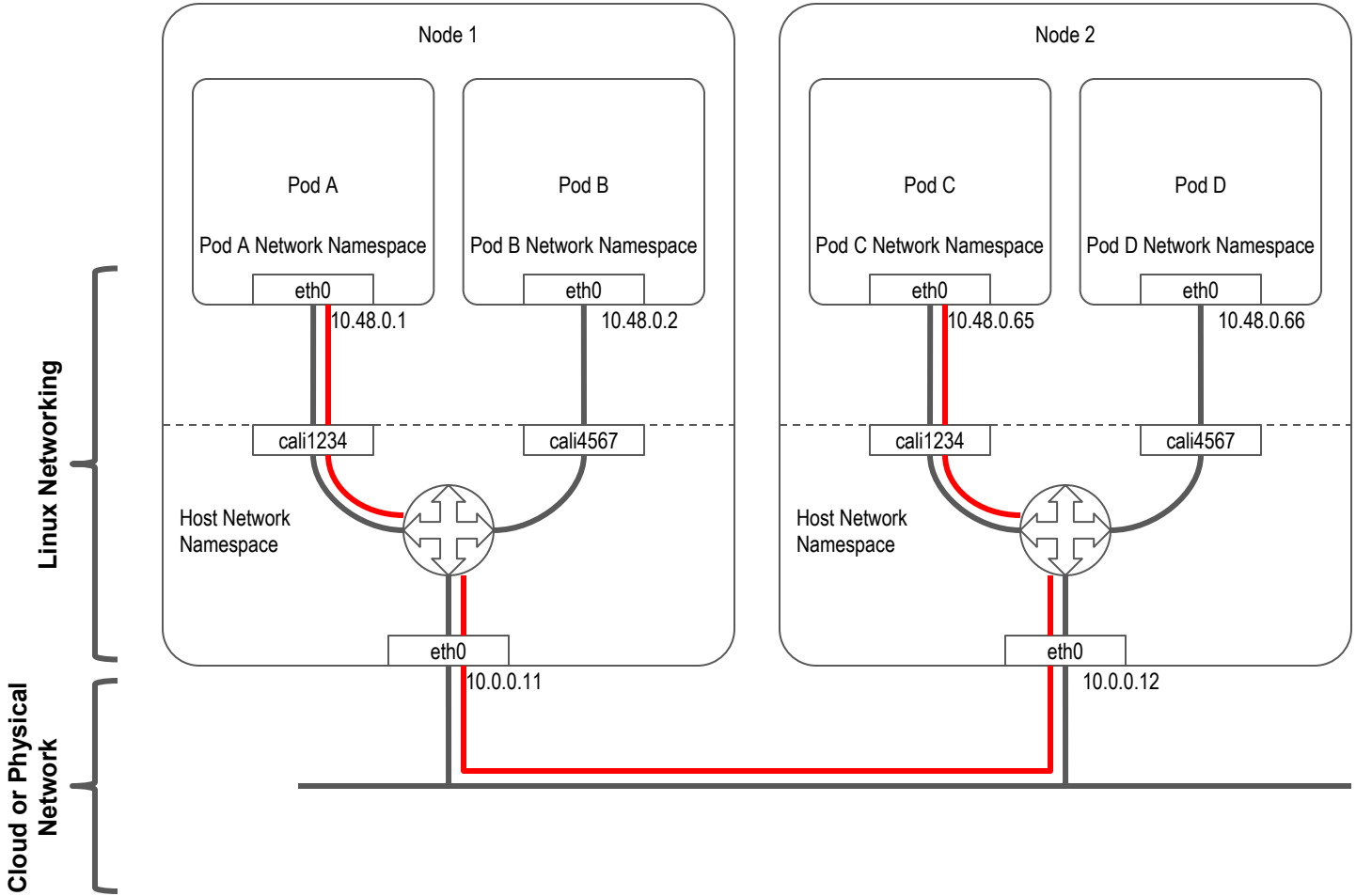
PROJECT CALICO

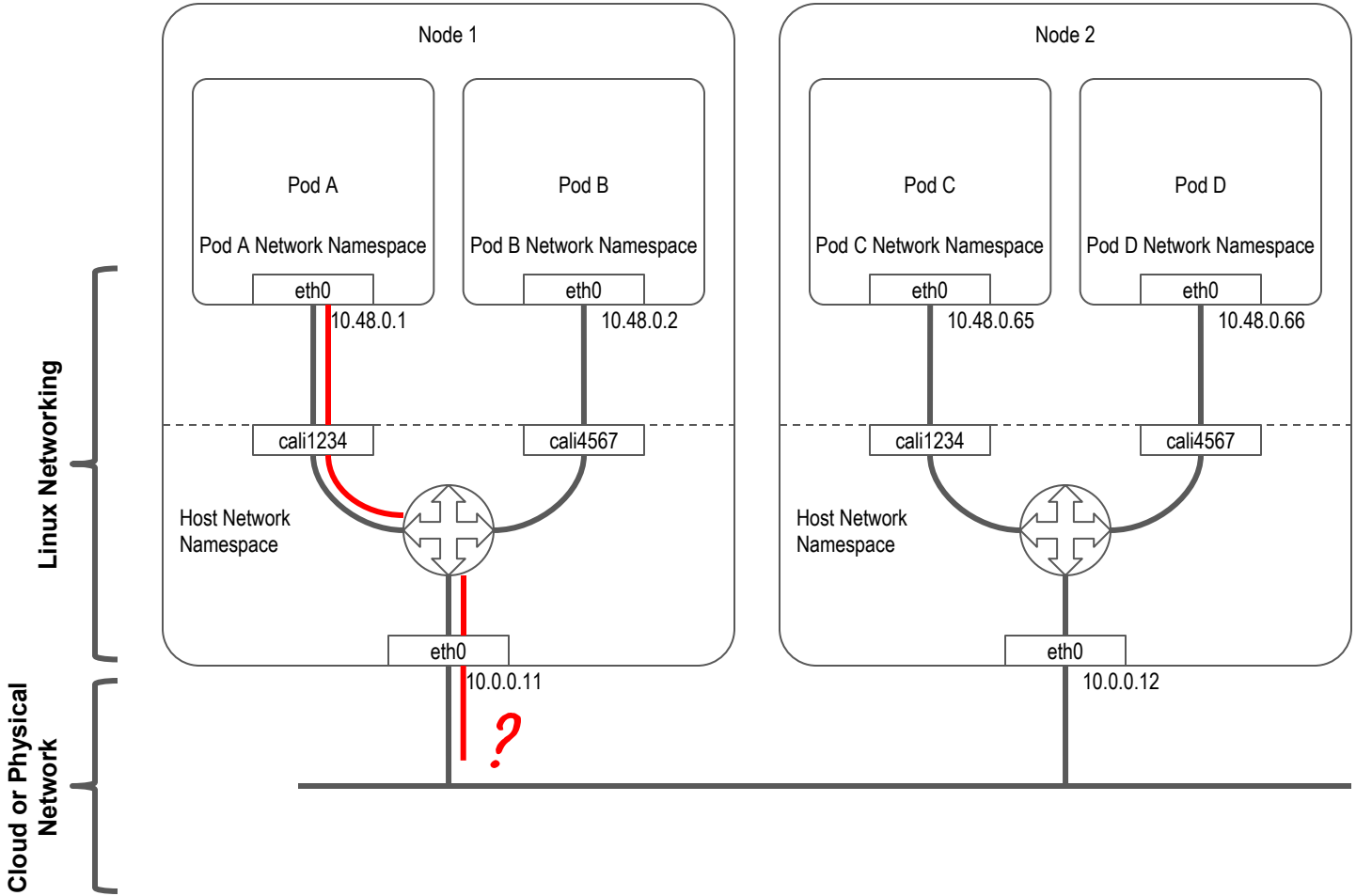

Node 1
Node 2
Pod A
Pod B
Pod C
Pod D
Linux Networking
eth0
eth0
10.0.0.11
10.0.0.12
Cloud or Physical Network

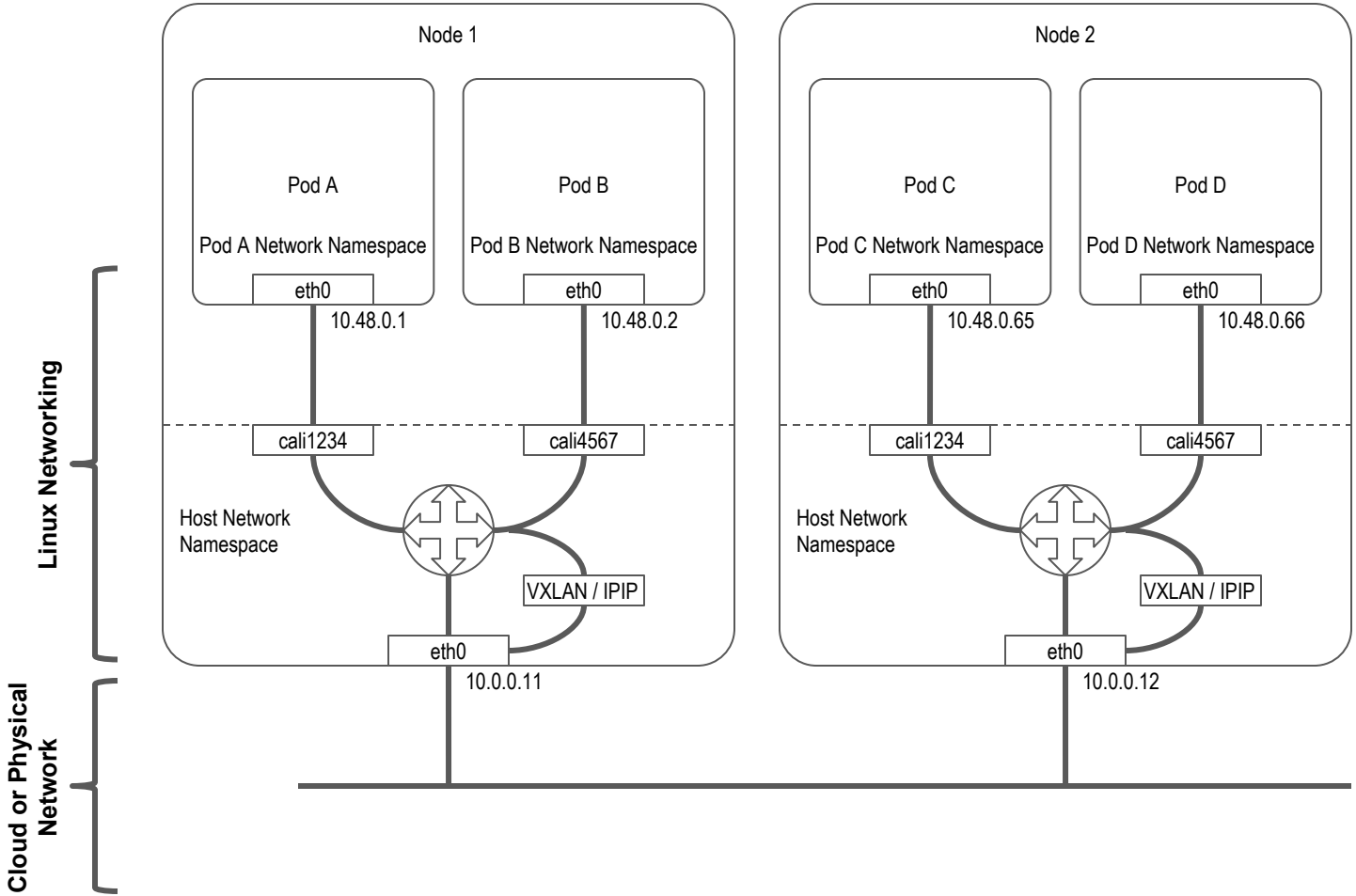

Node 1
Pod A
Pod A Network Namespace
Pod B
Pod B Network Namespace
Node 2
Pod C
Pod C Network Namespace
Pod D
Pod D Network Namespace
Linux Networking
Host Network Namespace
Host Network Namespace
eth0
10.0.0.11
eth0
10.0.0.12
Cloud or Physical Network

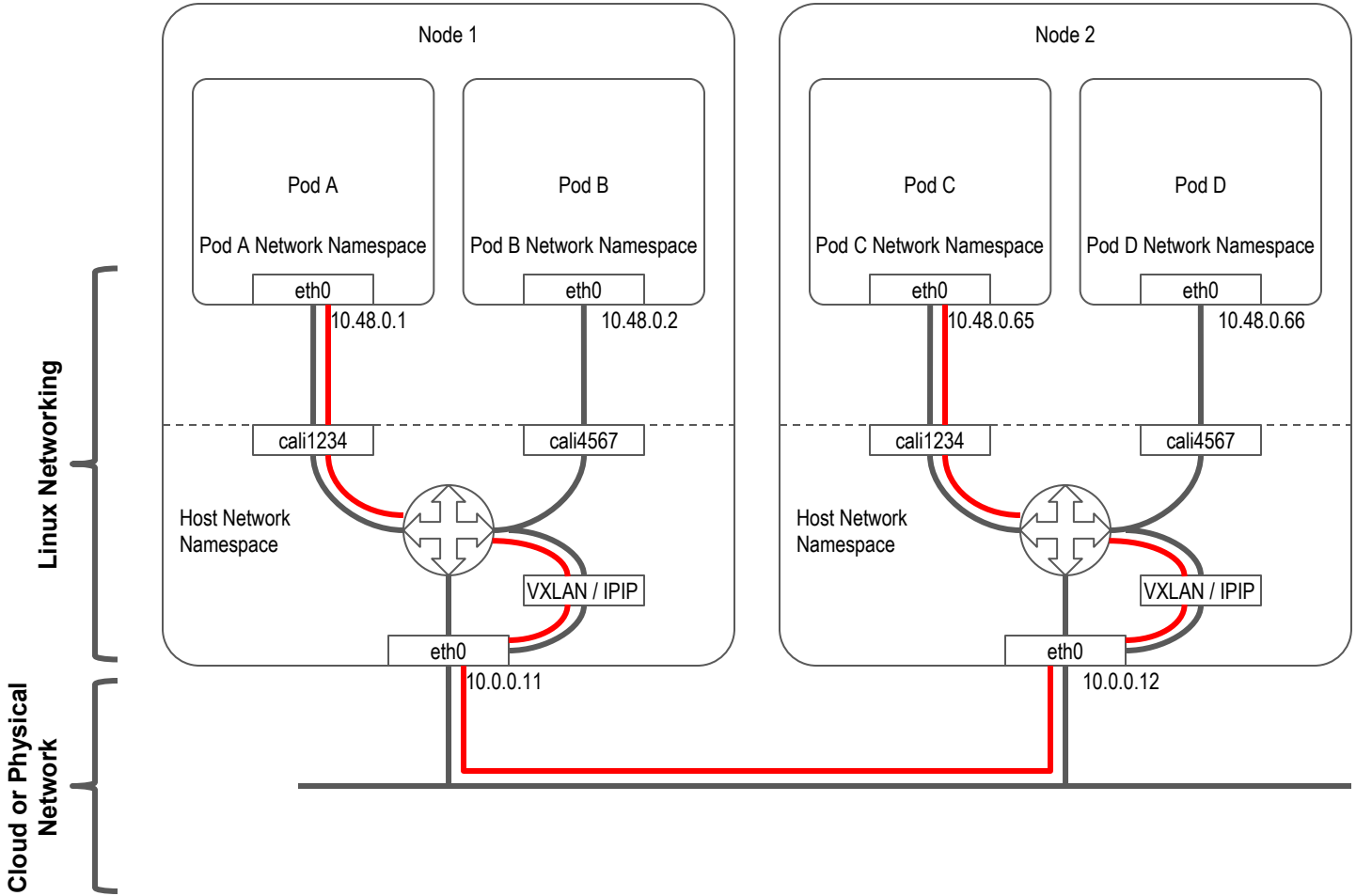

Node 1
Pod A
Pod A Network Namespace
eth0
10.48.0.1
Pod B
Pod B Network Namespace
eth0
10.48.0.2
cali1234
cali4567
Host Network Namespace
eth0
10.0.0.11
Node 2
Pod C
Pod C Network Namespace
eth0
10.48.0.65
Pod D
Pod D Network Namespace
eth0
10.48.0.66
cali1234
cali4567
Host Network Namespace
eth0
10.0.0.12
Linux Networking
Cloud or Physical Network


Node 1
Node 2
Pod A
Pod A Network Namespace
Pod B
Pod B Network Namespace
Pod C
Pod C Network Namespace
Pod D
Pod D Network Namespace
eth0
10.48.0.1
10.48.0.2
10.48.0.65
10.48.0.66
cali1234
cali4567
Host Network Namespace
10.0.0.11
10.0.0.12
Linux Networking
Cloud or Physical Network

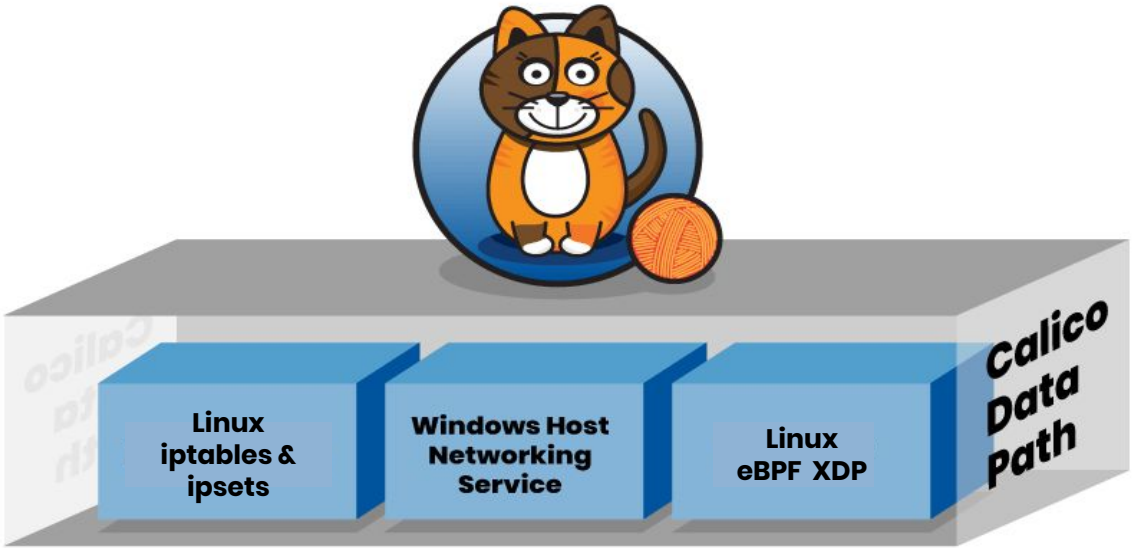

Node 1
Node 2
Pod A
Pod A Network Namespace
Pod B
Pod B Network Namespace
Pod C
Pod C Network Namespace
Pod D
Pod D Network Namespace
eth0
10.48.0.1
10.48.0.2
10.48.0.65
10.48.0.66
cali1234
cali4567
Host Network Namespace
10.0.0.11
10.0.0.12
Linux Networking
Cloud or Physical Network


Node 1
Node 2
Pod A
Pod A Network Namespace
Pod B
Pod B Network Namespace
Pod C
Pod C Network Namespace
Pod D
Pod D Network Namespace
eth0
10.48.0.1
10.48.0.2
10.48.0.65
10.48.0.66
cali1234
cali4567
Host Network Namespace
10.0.0.11
10.0.0.12
?
Linux Networking
Cloud or Physical Network


Node 1
Node 2
Pod A
Pod A Network Namespace
Pod B
Pod B Network Namespace
Pod C
Pod C Network Namespace
Pod D
Pod D Network Namespace
eth0
10.48.0.1
10.48.0.2
10.48.0.65
10.48.0.66
cali1234
cali4567
Host Network Namespace
VXLAN / IPIP
10.0.0.11
10.0.0.12
Linux Networking
Cloud or Physical Network


Node 1
Node 2
Pod A
Pod A Network Namespace
Pod B
Pod B Network Namespace
Pod C
Pod C Network Namespace
Pod D
Pod D Network Namespace
eth0
10.48.0.1
10.48.0.2
10.48.0.65
10.48.0.66
cali1234
cali4567
Host Network Namespace
VXLAN / IPIP
10.0.0.11
10.0.0.12
Linux Networking
Cloud or Physical Network

Dataplane: Leveraging the Underlying Kernel



IPAM Options: Calico-IPAM and Host-local IPAM

Host-local IPAM

- Allocates fixed block size (default: /24) per node
- Multiple IP pools possible, but no advanced features
- Some nodes exhaust addresses while others waste addresses
- Cannot discriminate between pods for IP allocation
- Cannot discriminate between namespaces for IP allocation

Calico-IPAM

- Allocates addresses in blocks (default: /26) as required
- Multiple IP pools with advanced options for users - will be covered in next section
- Calico-IPAM will run out of address space only when every single IP has been assigned to running pods
- Pod annotations can override pool assignment
- Namespace annotations can override pool assignment

CNI Chaining w/ Calico CNI

- Bandwidth-shaping CNI
- Portmap CNI
- Multus & CNI-Genie
- Istio-CNI

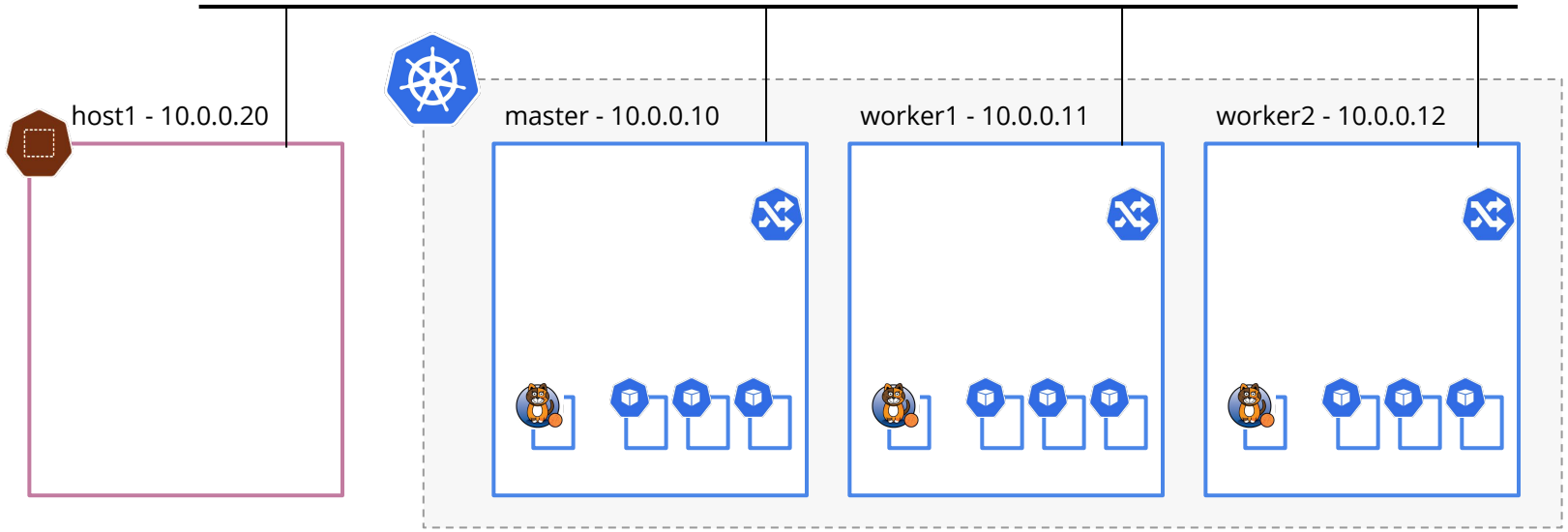
Takeaways!

- Calico provides **scalable pod connectivity** with the option to use encapsulation if desired (**VXLAN or IPIP**), or **native IP without encapsulation** for **simplicity** and operational scale

- Calico can use host-local IPAM similar to other CNI plugins, but can also leverage **Calico-IPAM for advanced controls** over IP allocations and address real-world connectivity requirements

- Calico networking leverages the same underlying technology used for route exchange on the Internet, i.e., **proven at massive scale**

Pod Connectivity Lab: Topology



Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

PROJECT CALICO

F

Advanced Pod Connectivity

PROJECT CALICO

Hosted Cloud Platform Choices: Connectivity and Policy

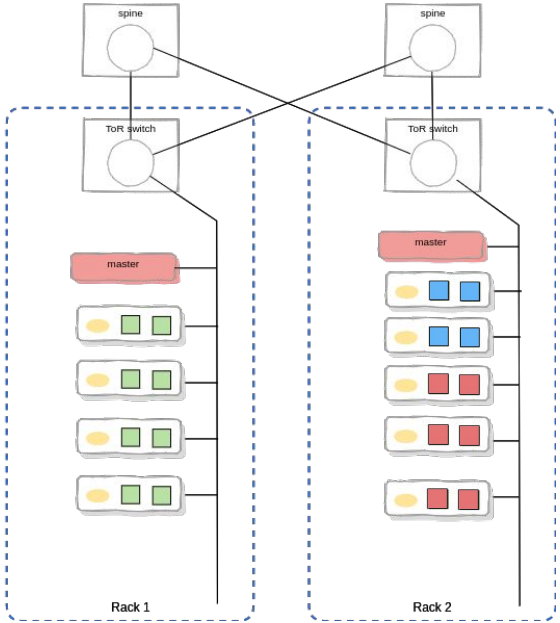
	AWS EKS	Azure AKS	Google GKE	GKE OnPrem	IBM Cloud IKS	IBM Cloud Private (ICP)
Network Policy	Calico	Calico	Calico	Calico	Calico	Calico
CNI	aws-cni	azure-cni	gke-cni	Calico	Calico	Calico
Kube-proxy	IPTables	IPTables	IPTables	IPTables	IPTables IPVS (tech-preview)	IPTables IPVS (tech-preview)

USE CASE: Zone-based Deployment

Rack/Zone Aware IPAM

```
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
   name: zone2-ippool
spec:
   cidr: 10.48.1.0/24
   nodeSelector: zone == "zone2"
```

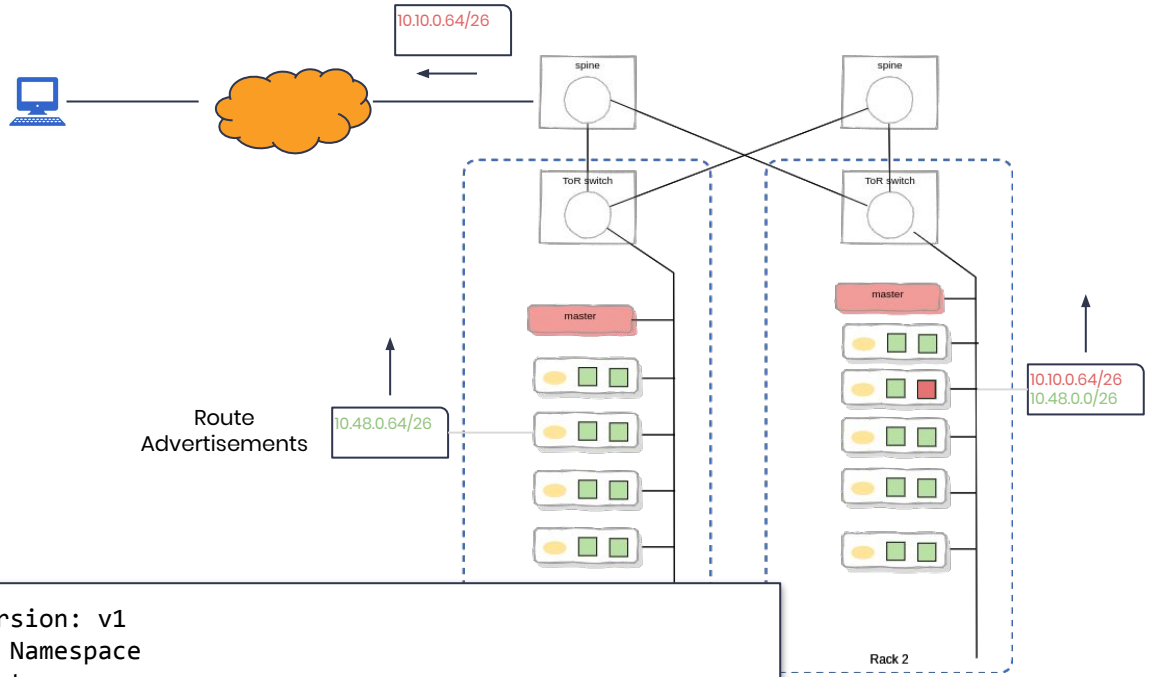
```
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
   name: zone2-ippool
spec:
   cidr: 10.48.2.0/24
   nodeSelector: zone == "zone3"
```



Legend

- Calico-node (daemonset)
- IP from Default IP Pool (10.48.0.0/24)
- IP from IP Pool 2 (10.48.1.0/24)
- IP from IP Pool 3 (10.48.2.0/24)

USE CASE: Differentiated Routability using Namespace Annotations



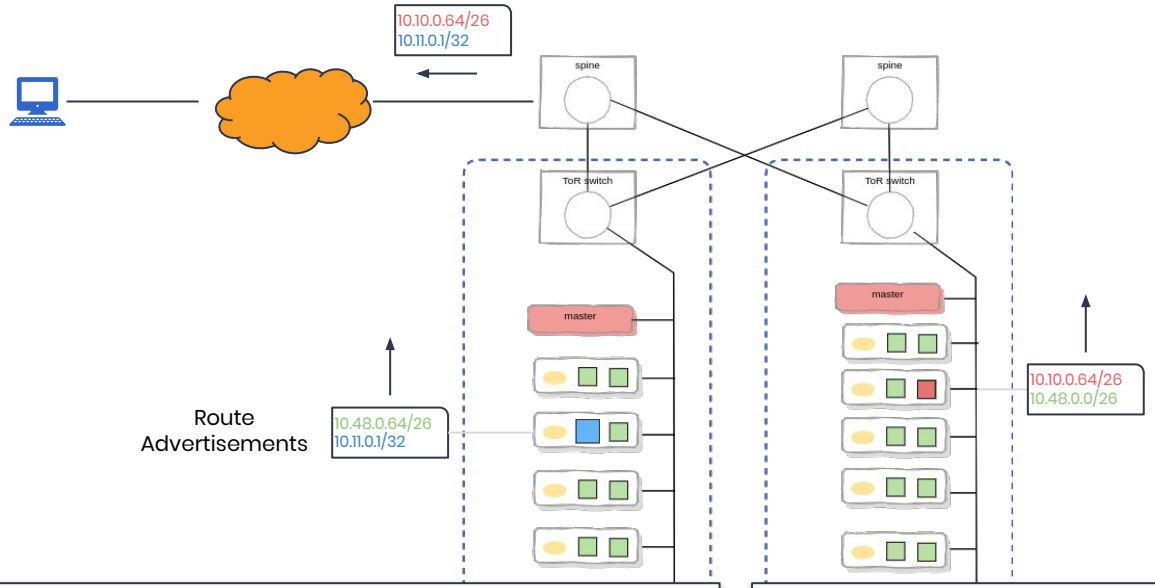
Legend

- Calico-node (daemonset)
- IP from Default IP Pool (10.48.0.0/16)
- IP from External Pool (10.10.0.0/24)

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    cni.projectcalico.org/ipv4pools: '["external-pool"]'
  name: external-ns
```

USE CASE: Externally-routable Pod IP's

Annotations for Pools, IPs, and Floating IPs



Legend

- Calico-node (daemonset)
- IP from Default IP Pool (10.48.0.0/16)
- IP from External Pool (10.10.0.0/24)
- Specific Pod IP or Floating IP

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    cni.projectcalico.org/ipAddrs": '["10.11.0.1"]'
...
```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    cni.projectcalico.org/ipv4pools: '["external-pool"]'
...
```

USE CASE: Other Real-world requirements

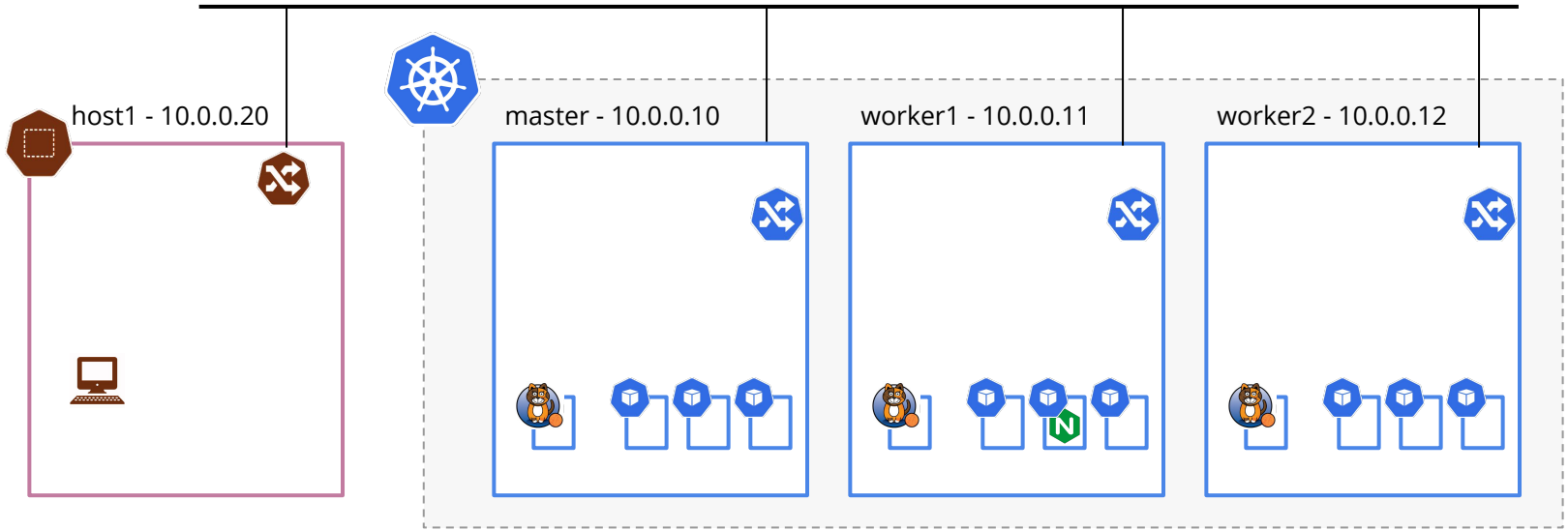
- IPv6 and Dual-Stack
- Windows CNI
- Flannel to Calico Live Migration

Takeaways!

- Calico provides solutions for advanced pod connectivity use cases required in the real world
  - **Zone-based architectures** and **selective addressing to namespaces or pods**
  - **Enabling direct external connectivity** to pods (without NAT) in advanced scenarios

- Allows advanced network configurations leveraging standard approaches used in the Internet (and loved by network engineers)
  - **Scales** easily from small clusters to very large configurations
  - Calico provides **operational simplicity** by automating most controls under the covers, but giving advanced cluster operators the ability to tune if necessary

Advanced Pod Connectivity Lab: Topology



Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

PROJECT CALICO

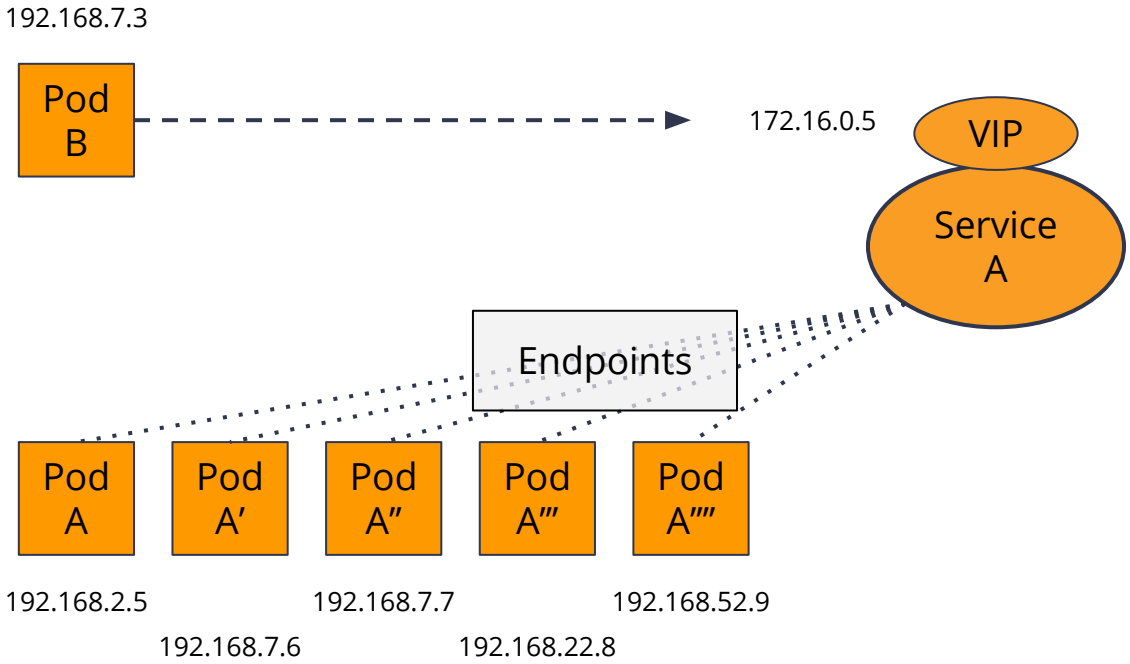
Kubernetes Services

SERVICE RESOURCE DEFINITION

https://kubernetes.io/docs/concepts/services-networking/service

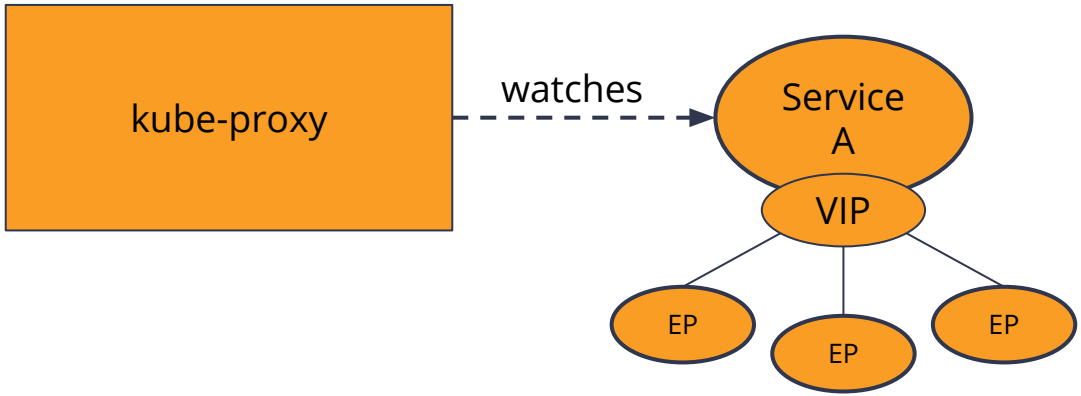
```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

LOGICAL SERVICES AND ENDPOINTS

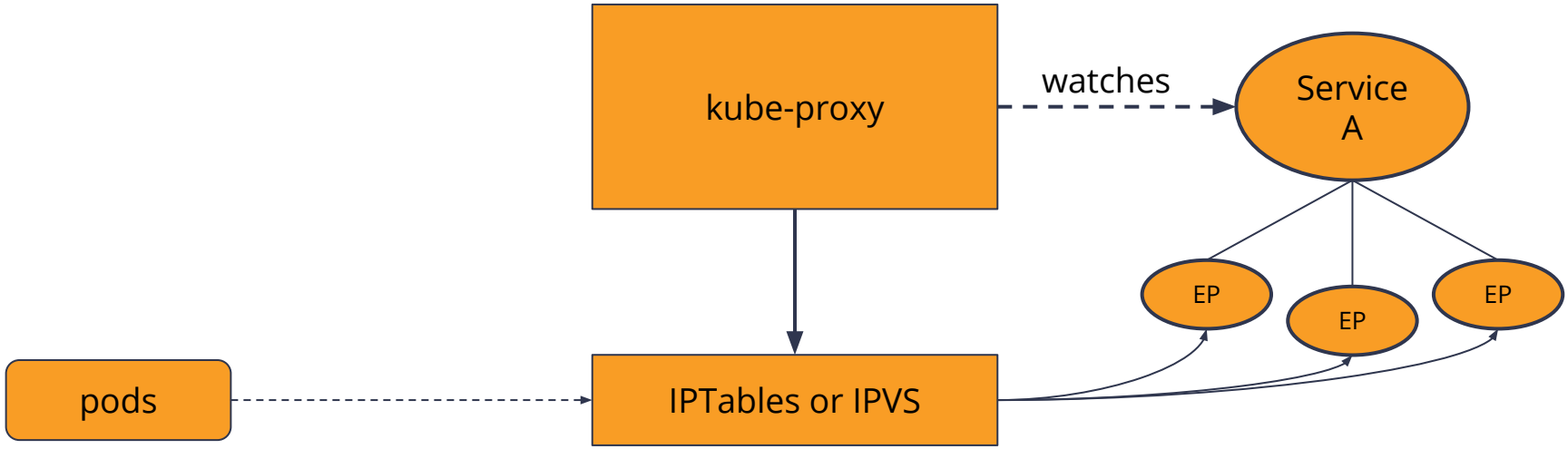


KUBE-PROXY

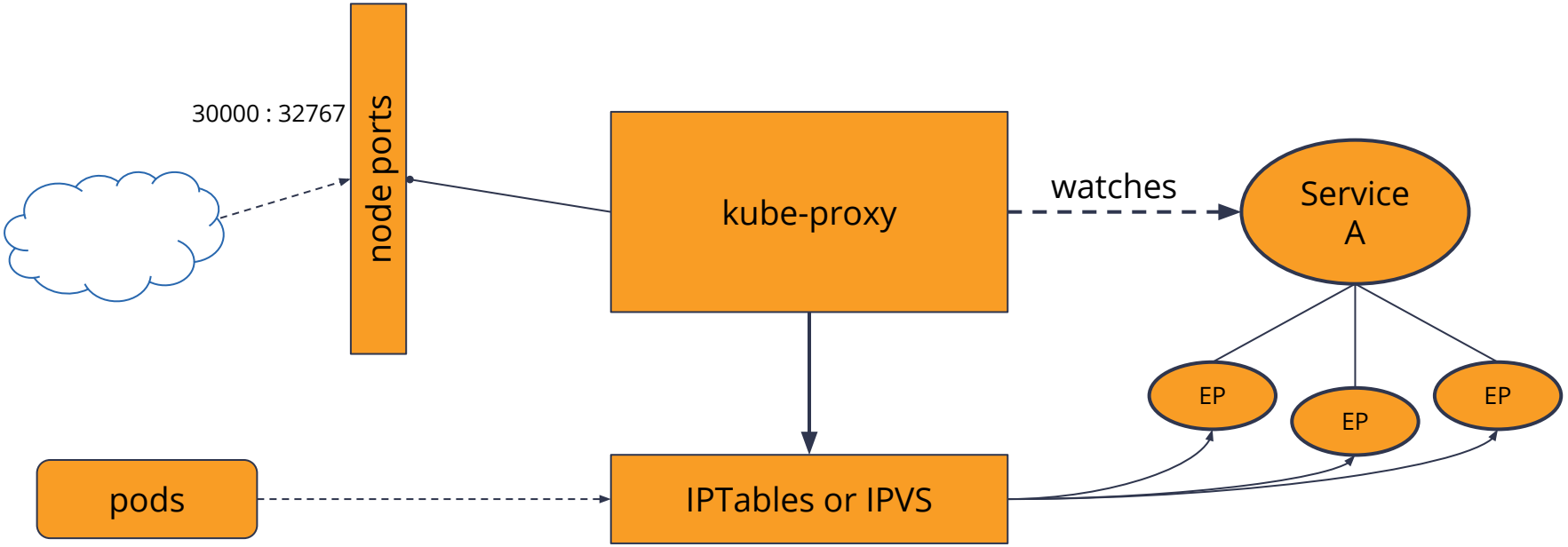
KUBE-PROXY



KUBE-PROXY



KUBE-PROXY



82

SERVICE TYPES

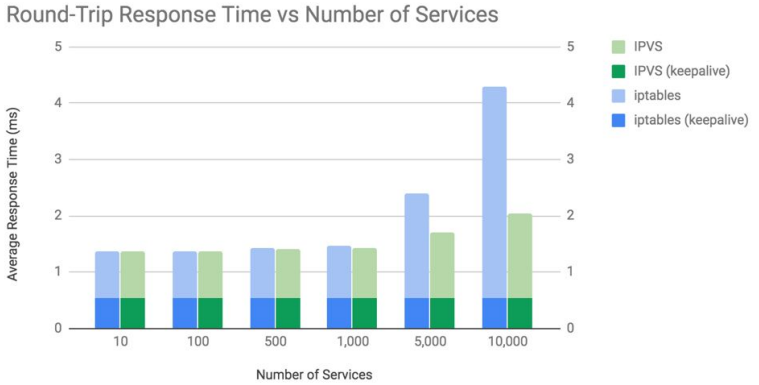
None used to track service endpoints but not load balance. Typically used by operators/controllers and other automation integrations.

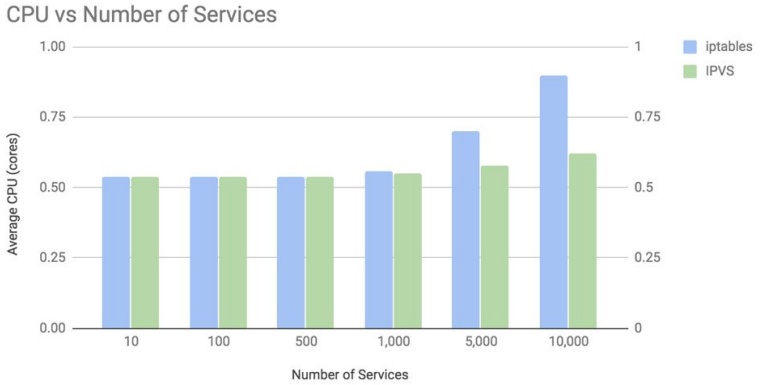
Cluster IP assigned to service; kube-proxy instantiates SNAT and DNAT rules to translate pod traffic destined to cluster vip to a pod IP address and redirect traffic

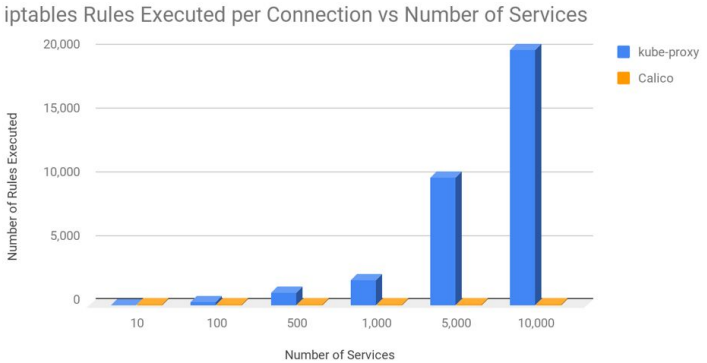
Node Port (optionally) assigned to service (default range 30000-32767); static port exposed on each node allowing access from outside the cluster

Load Balancer automatically creates rules on (supported) Cloud Providers load balancer

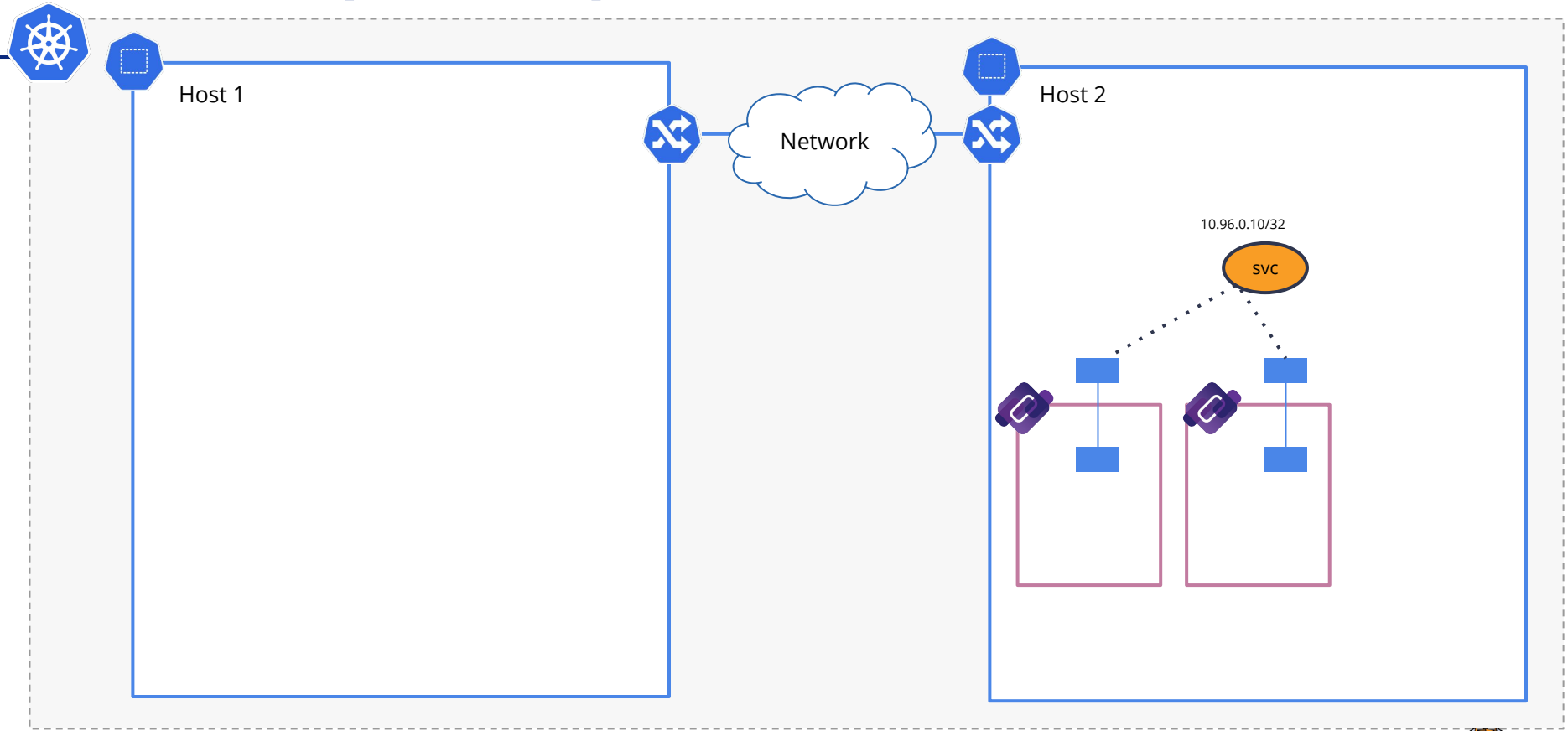
Kube-proxy: IPVS vs. IPTables



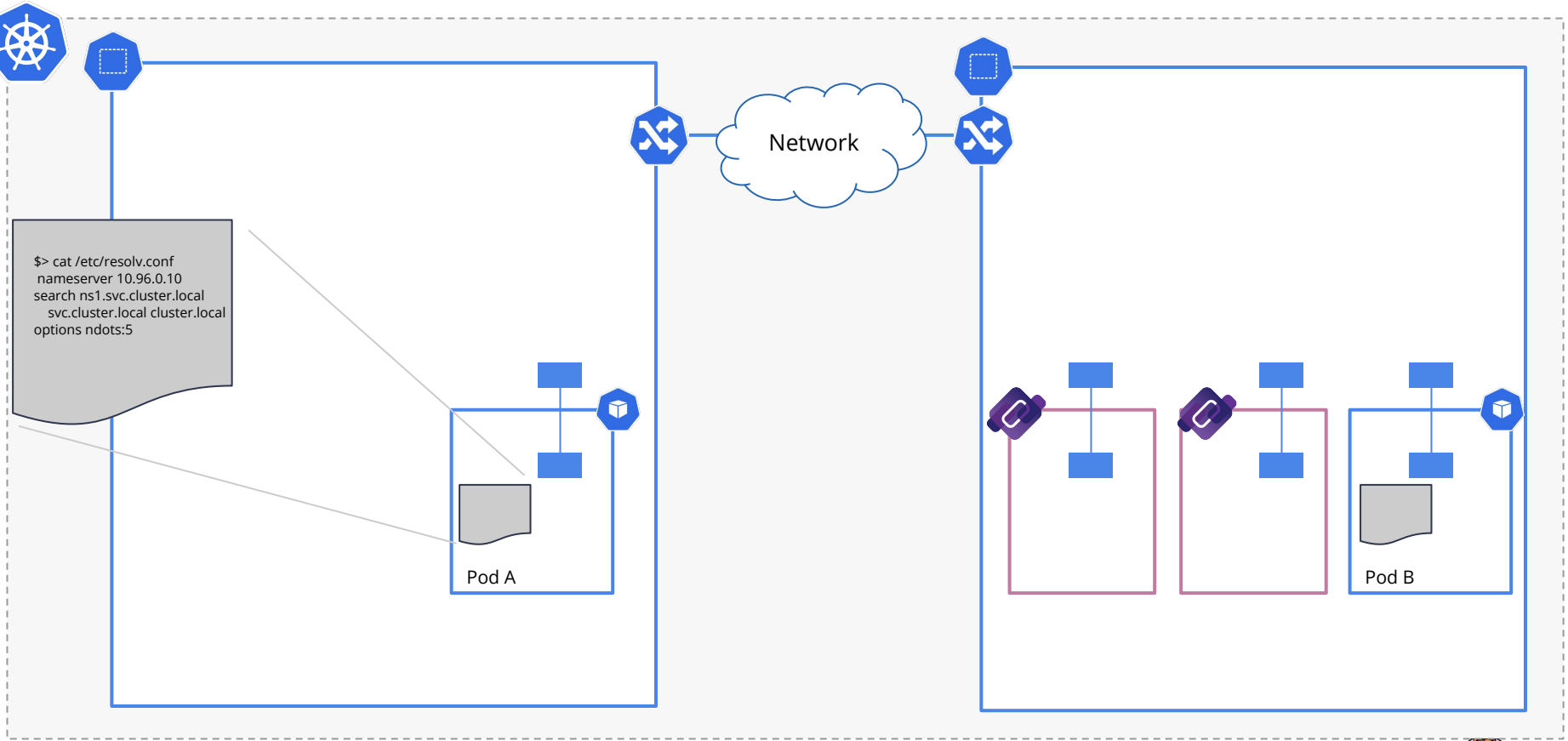




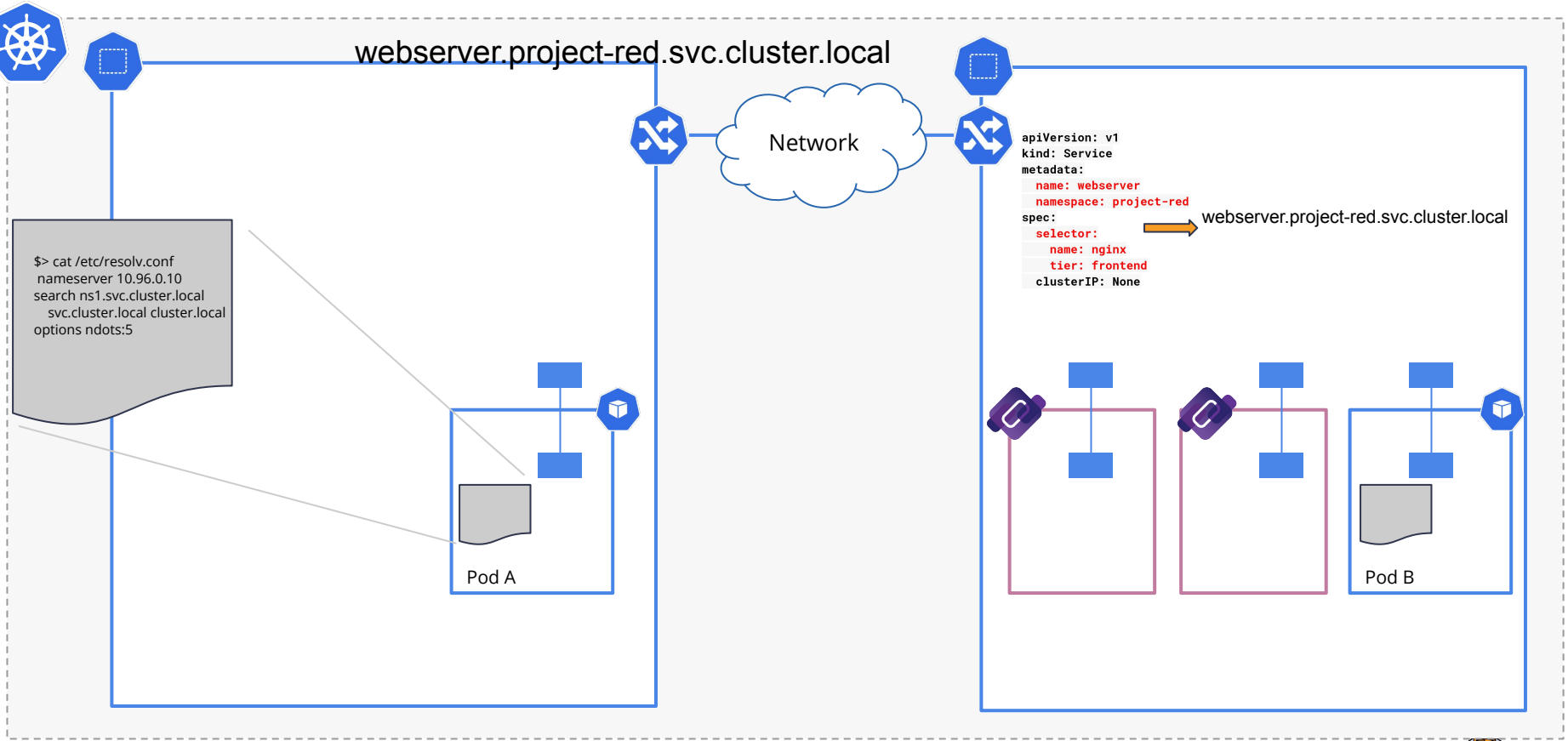
Kube-DNS (CoreDNS)



Kube-DNS (CoreDNS)



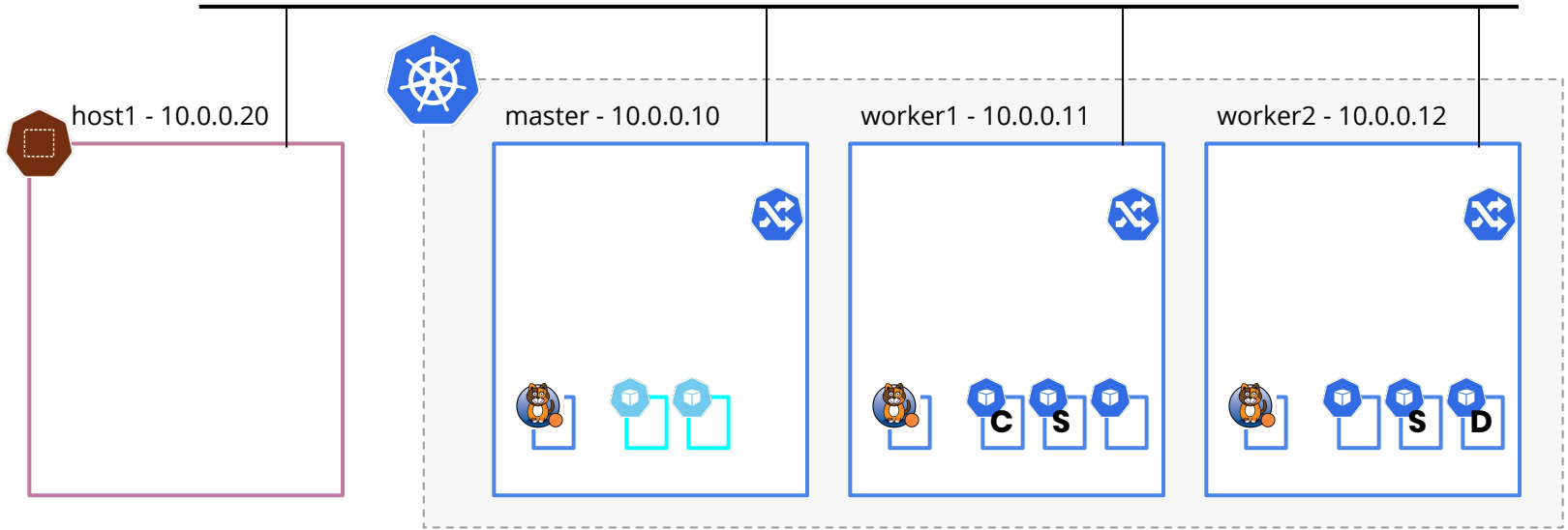
Kube-DNS (CoreDNS)



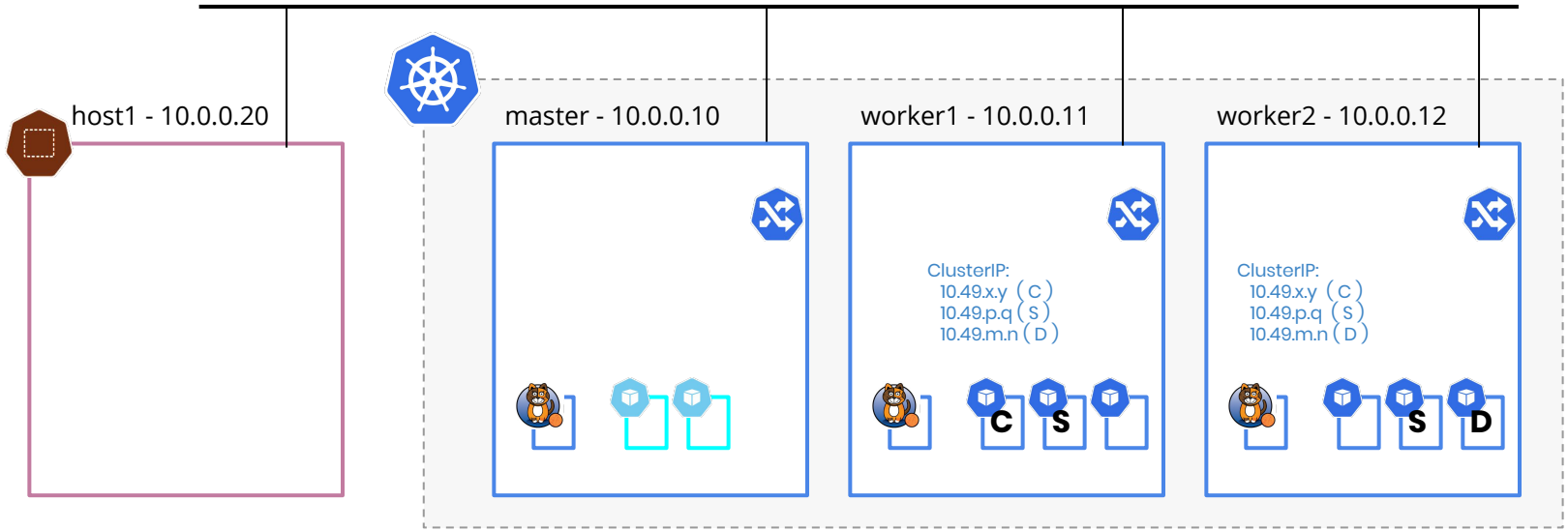
Takeaways!

- Kubernetes services provide a variety of abstractions, leveraging kube-proxy and other constructs
- IPTables kube-proxy works well for anything besides the largest clusters
- **IPVS kube-proxy** provides **better scale for the largest clusters** (e.g., around ~5000 -10000+ services with 2-3 pods each)
- Important to understand the differences between various Kubernetes service types, and Operations in production

Kubernetes Services Lab: Topology



Kubernetes Services Lab: Topology



Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

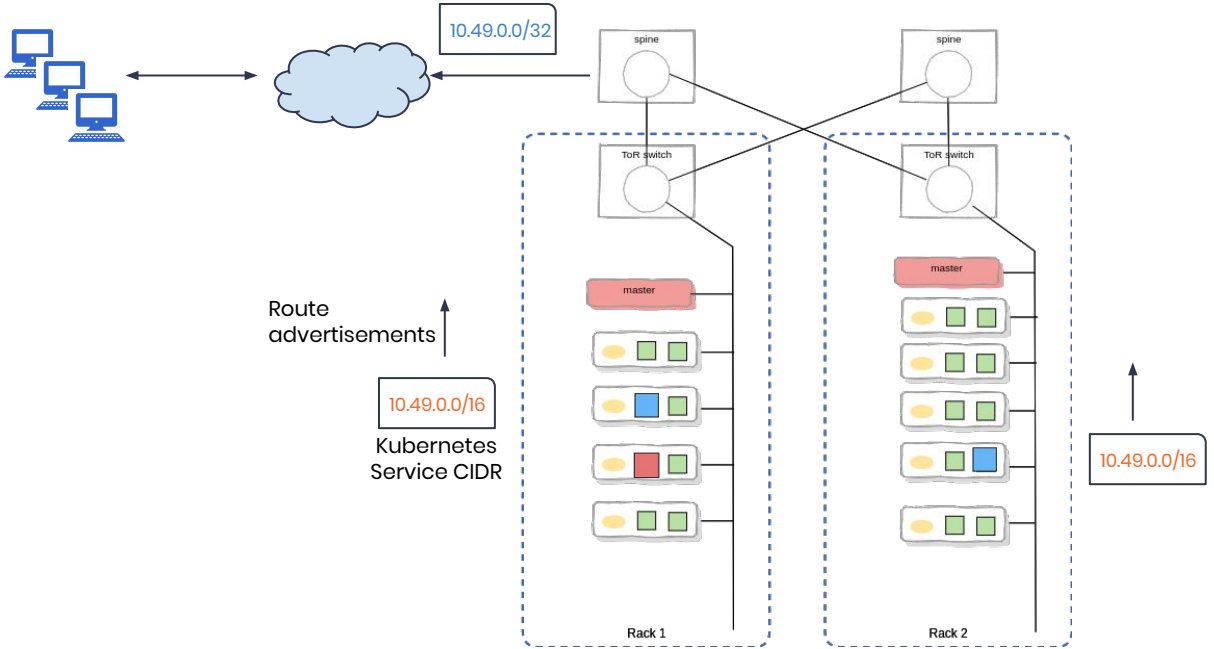
E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

H

Advanced Services

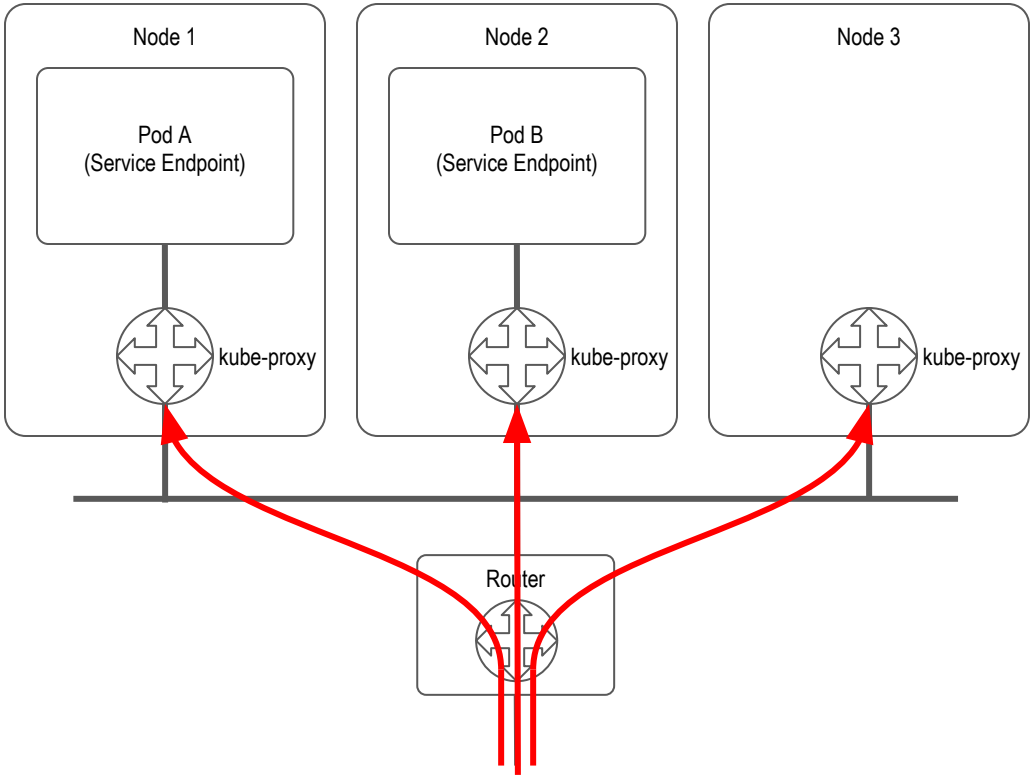
PROJECT CALICO

USE CASE: Routable Service Cluster IP Range (with ECMP Anycast for native LB & Failover)

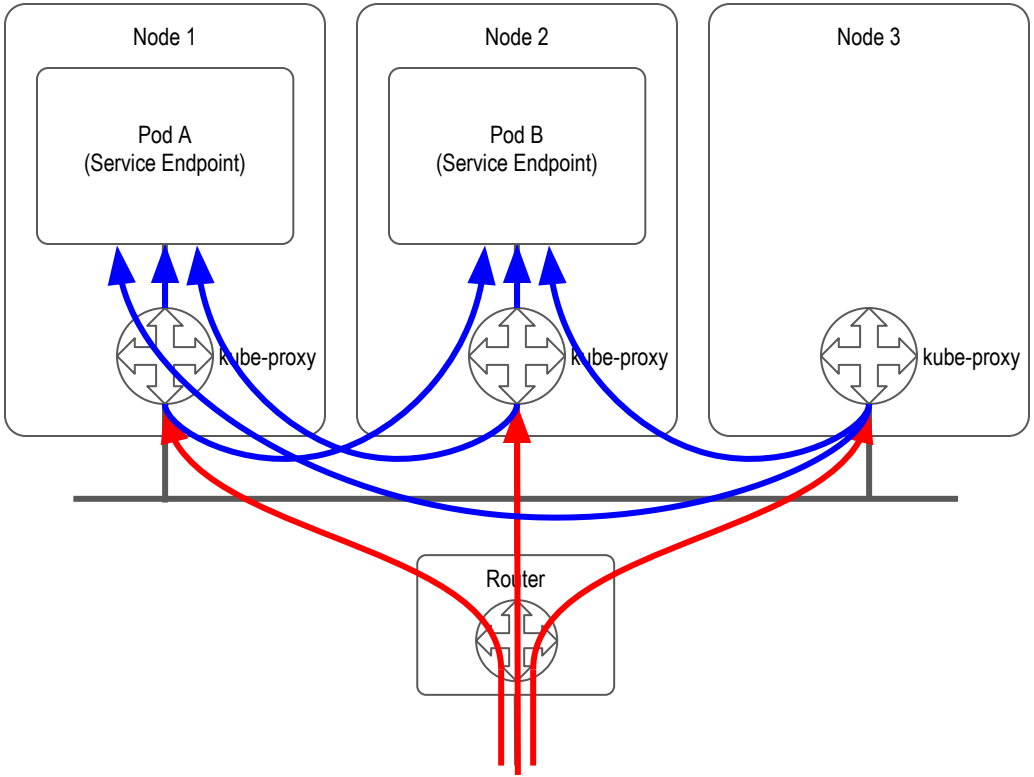



```
apiVersion: projectcalico.org/v3
kind: BGPConfiguration
metadata:
  name: default
spec:
  serviceClusterIPs:
  - cidr: "10.49.0.0/16"
```

Kube-Proxy and ECMP to Service Cluster IP Range

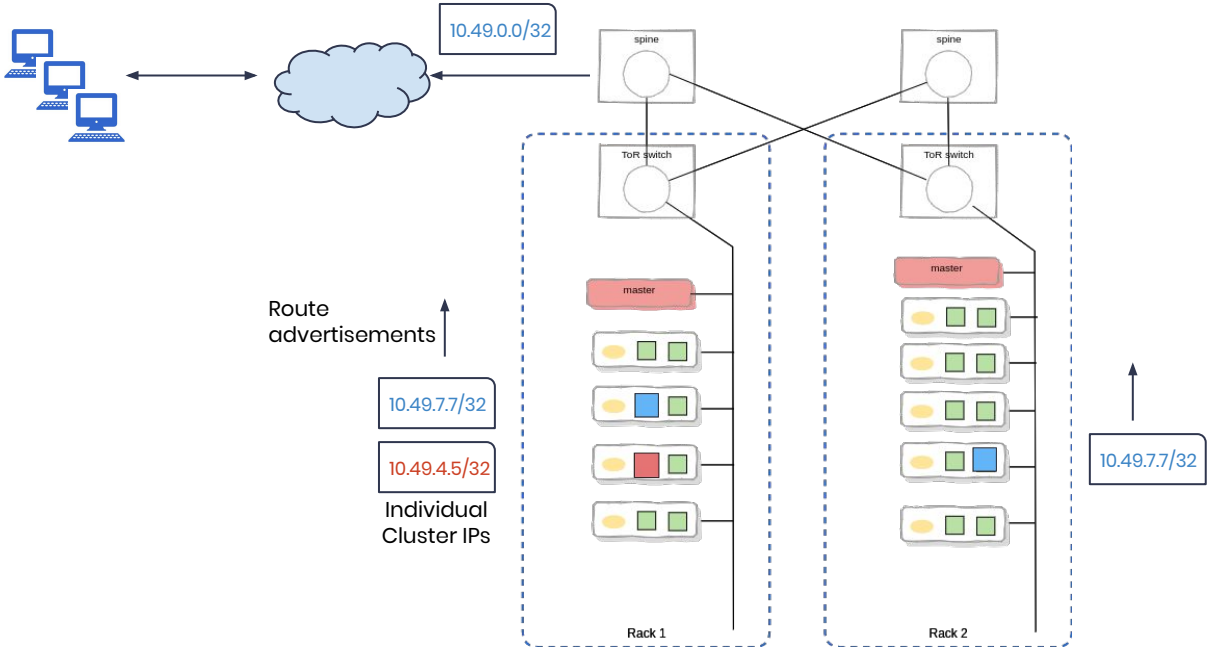


Kube-Proxy and ECMP to Service Cluster IP Range



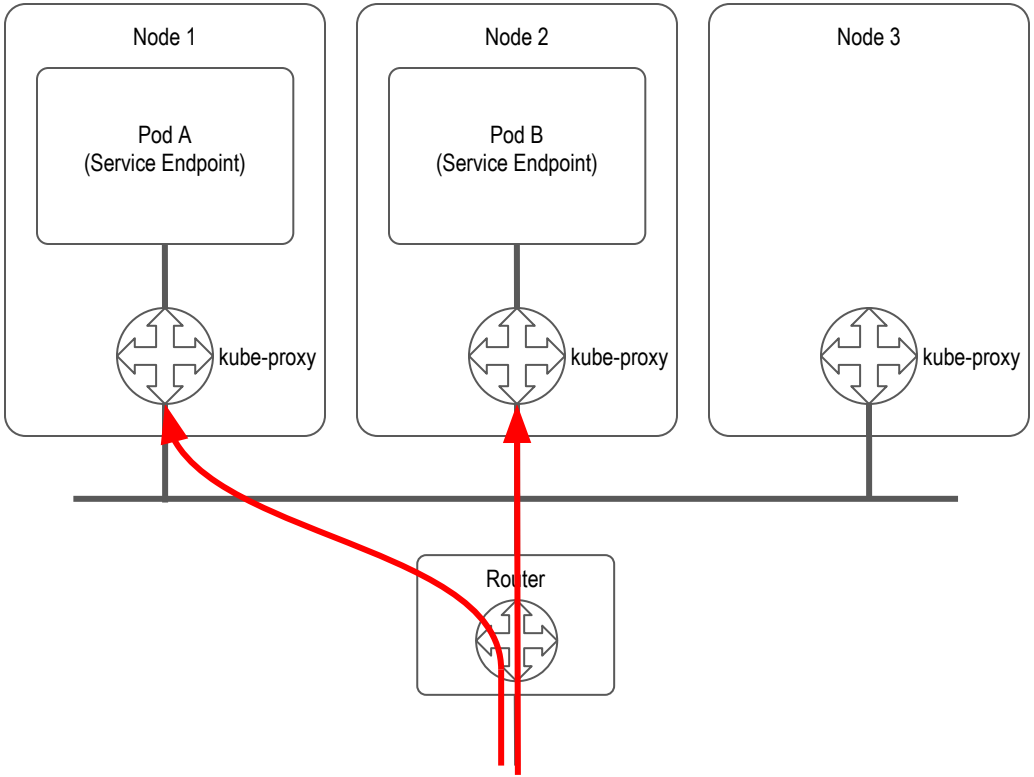
Same behavior as NodePorts

USE CASE: Routable Service Cluster IP (with ECMP Anycast for native LB & Failover)

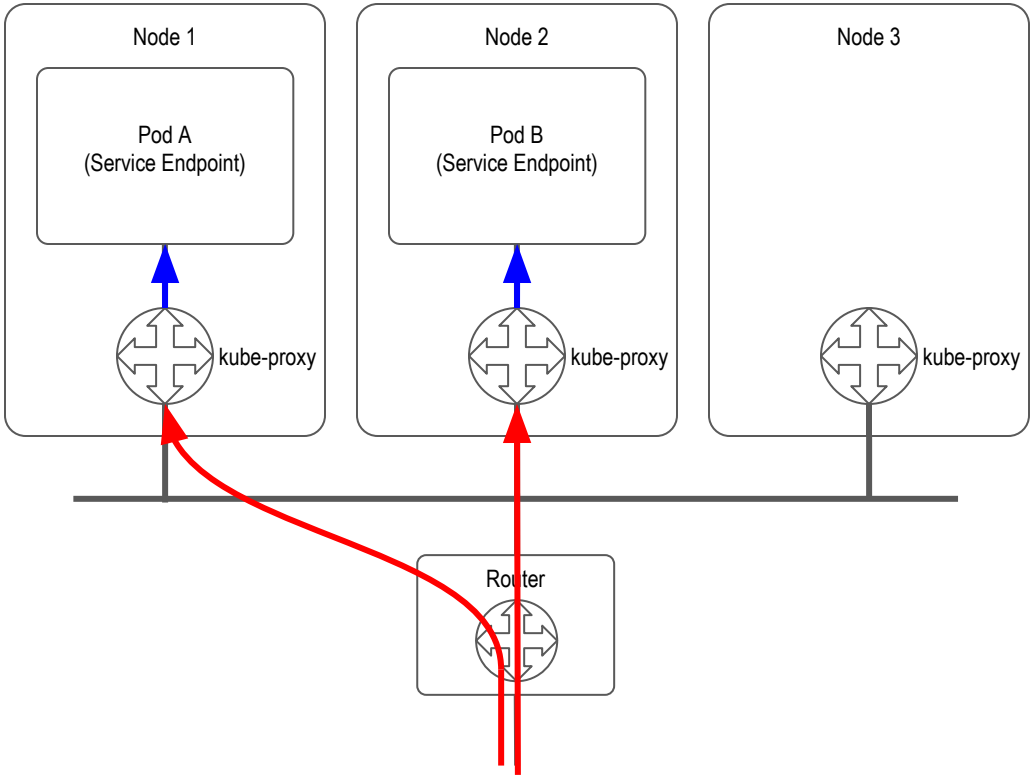



```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: customer
  name: customer
  namespace: yaobank
spec:
  clusterIP: 10.49.4.5
  externalTrafficPolicy: Local
  ports:
  - name: http
    nodePort: 30180
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: customer
  type: NodePort
```

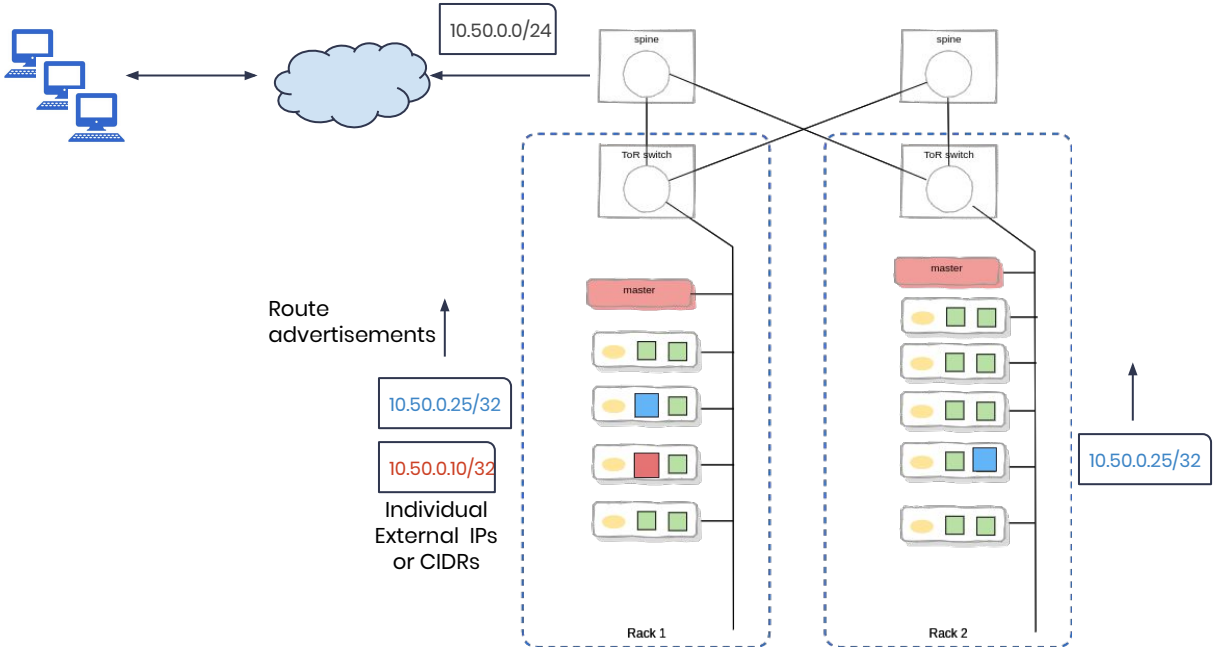
Kube-Proxy and ECMP to Service with externalTrafficPolicy:Local



Kube-Proxy and ECMP to Service with externalTrafficPolicy:Local

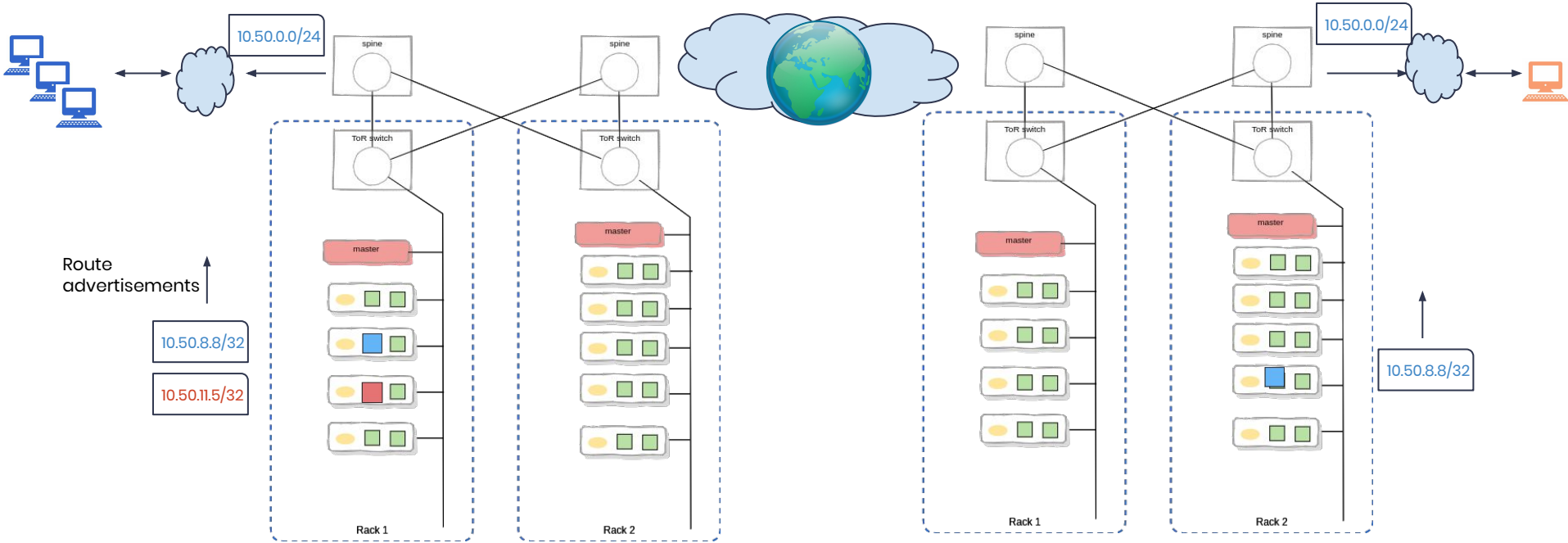


USE CASE: Routable Service External IP's (with ECMP Anycast for native LB & Failover)




```
apiVersion: projectcalico.org/v3
kind: BGPConfiguration
metadata:
  name: default
spec:
  serviceClusterIPs:
  - cidr: "10.49.0.0/16"
  serviceExternalIPs:
  - cidr: "10.50.0.10/32"
  - cidr: "10.50.0.25/32"
```

USE CASE: Zone-aware Service Advertisements (with ECMP Anycast for native LB & Failover)



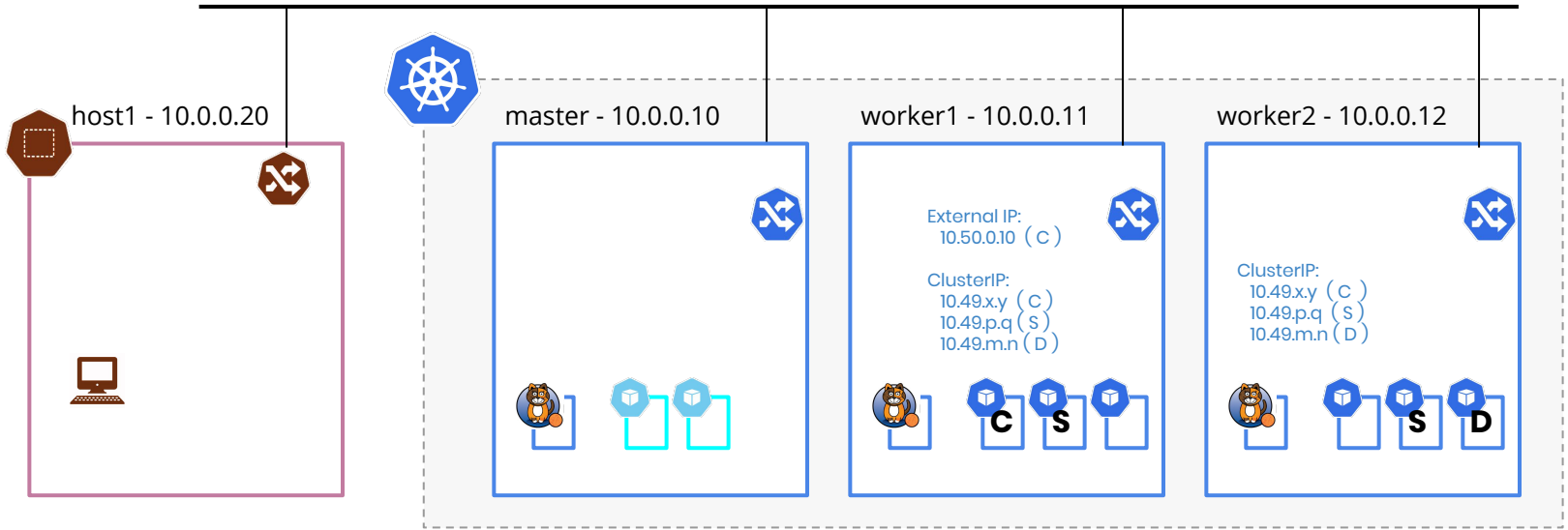
Takeaways!

- Calico allows **service Cluster IP's to be advertised**, **allowing access from outside the cluster** using ECMP anycast

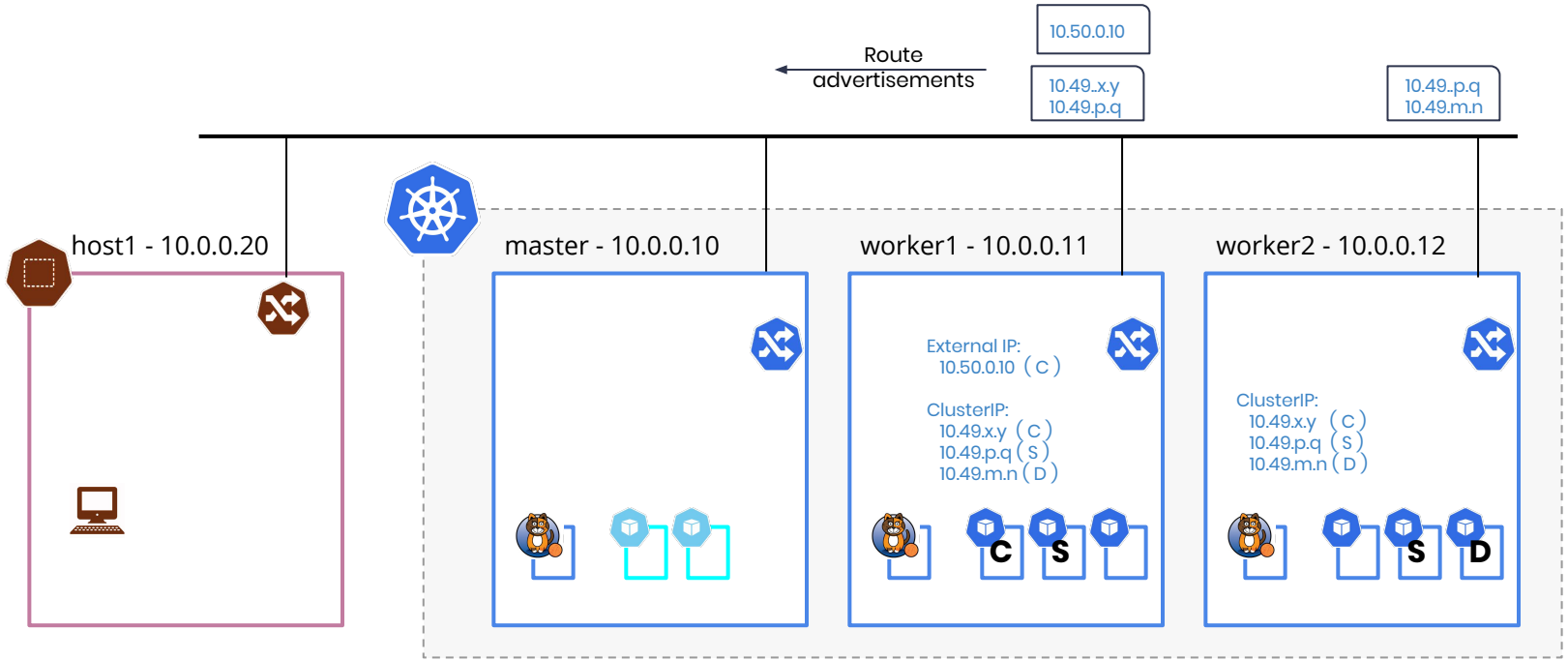
- Calico allows **service External IP's** to be advertised, allowing access from outside the cluster using ECMP anycast

- Calico works with **standard upstream kube-proxy**, both IPVS kube-proxy and IPTables kube-proxy

Advanced Services Lab: Topology



Advanced Services Lab: Service IP Advertisement



Agenda

A. Welcome to CalicoCon
B. State of the ‘K8s Talk
C. Policy Fundamentals Lab
D. Adv. Policy w/ Calico Lab

Lunch Break

E. Pod Connectivity Lab
F. Adv. Pod Connectivity w/ Calico Lab
G. K8s Services Lab
H. Adv. Services w/ Calico Lab
I. Wrap up

Wrap Up

PROJECT CALICO

Focus for CalicoCon Workshops Today

Education	Lab / Sandbox	Pilot / Pre - Production	Prod - Initial Apps	Prod - Mass Migration
Get basic cluster working with networking	Network policy Policy best practices Managing trust across teams	Understand networking at scale Integrate with enterprise network	Understand services at scale Select best options for service types Integrate services with rest of infrastructure	
000 Lab	**100 & 200 Lab**	**300 & 400 Lab**	**500 & 600 Lab**	

Help support Project Calico

Community repo: github.com/projectcalico/community

👀 Join the monthly community meetings

- Hear what's new
- Sneak peeks of new features
- Demos
- Calico Heroes - who's been hitting it out of the park for Calico
- Q & A / Stump the maintainer

📣 Write a blog or mention us in a talk

⭐ Star Calico on github (github.com/projectcalico/calico)

🛠️ Become a contributor

- Community repo has what you need, including good first-time issues

Follow-up Materials

This evening you will receive:

- A PDF copy of these slides
- A survey (please take the time to fill this out)

Sponsored by Tigera

Inventors and lead maintainers of open source Project Calico

Hiring in all departments: San Francisco, San Jose, Vancouver

Products and Services

PROJECT CALICO

Networking
& Network Policy

CALICO ESSENTIALS

Kubernetes Journey
Acceleration

CALICO ENTERPRISE

Enterprise Teams,
Visibility,
Network Security,
Compliance

CalicoCon 2019

Thank you for being a part of it!