

# Isle of Cats: Inside Calico Networking for Kubernetes

Chicago Cloud Native Meetup  
May 20, 2019



© 2018 Tigera, Inc. | Proprietary and Confidential

# WELCOME



## Your Presenter:

Drew Oetzel  
Senior Field Engineer @ Tigera

 [drew@tigera.io](mailto:drew@tigera.io)

 <https://www.linkedin.com/in/andrewoetzel/>



# WHO IS TIGERA?

An 80s hair metal band???

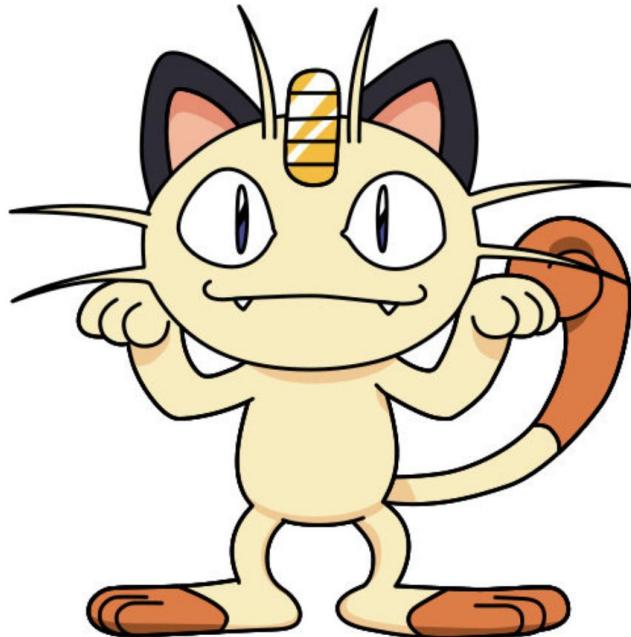


No . . . that's:

**PANTERA**

# WHO IS TIGERA?

A super secret Pokemon?



No, but we do have a conference room named after Meowth!

# WHO IS TIGERA?



**Large cloud &  
enterprise customers**



**Calico: Open Source,  
widely adopted**



**Tigera Secure  
Enterprise Edition**



# STRATEGIC PARTNERSHIPS



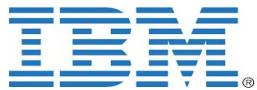
Integrated with ACS Engine, Collaborating on Azure Kubernetes Service (AKS) and Windows



Default network policy for Azure Container Service for Kubernetes (EKS) & Heptio/AWS Quick Start



Partnered to support network policy in Google Container Engine (GKE)



Partner to implement connectivity and security for IBM Kubernetes Service, IBM Cloud Private



OpenShift primed partner. Certified integration with OpenShift Container Platform



Default networking and policy for Docker EE Kubernetes

# WHAT IS CALICO?



**“Networking” for containers.**

**Open source project with worldwide contributors backed by the commercial enterprise Tigera, Inc.**

**Most used CNI plugin for Kubernetes worldwide.**

**[www.projectcalico.org](http://www.projectcalico.org)**

# OK, BUT WHAT CAN CALICO DO FOR ME?



## Efficiently hand out IPs to your pods. (IPAM)

- Can use different subnets per namespace or rack / VPC
- Can assign static IP addresses to pods

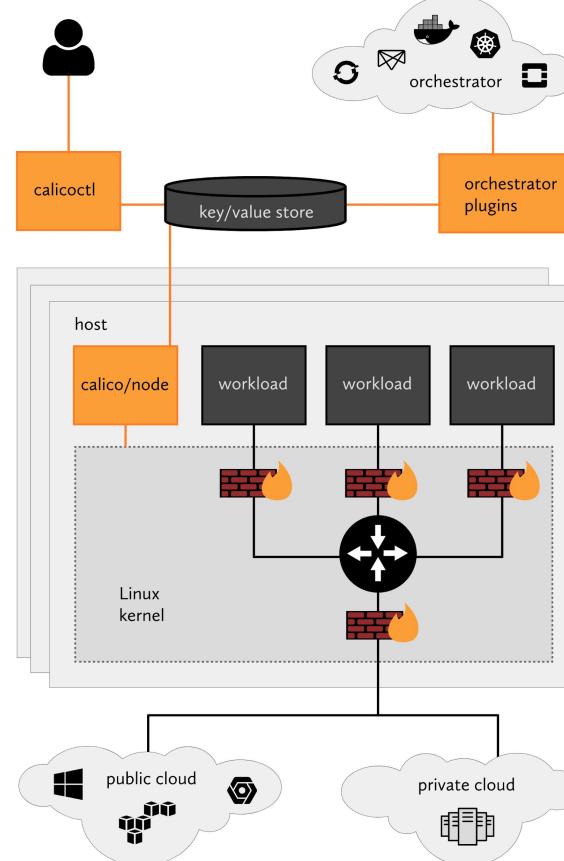
## Secure your inner K8s networking with policies.

- Secure your East/West traffic
- Prevent malicious behavior from spreading
- Block insider threat / insider error

# RUNNING CALICO with KUBERNETES



# HIGH LEVEL CALICO ARCHITECTURE: SUMMARY



# INSTALLING CALICO on KUBERNETES

- Calico is usually the first thing you install right after you setup Kubernetes - in fact the cluster won't be "ready" until you do.
- Calico can be configured to communicate directly with etcd or it can communicate with the Kubernetes API Datastore
- Be sure you select the correct installation yaml from the web page depending on which option you choose

## API Datastore:

- <https://docs.projectcalico.org/v3.7/getting-started/kubernetes/installation/calico#installing-with-the-kubernetes-api-datastore50-nodes-or-less>

## Etcd Directly:

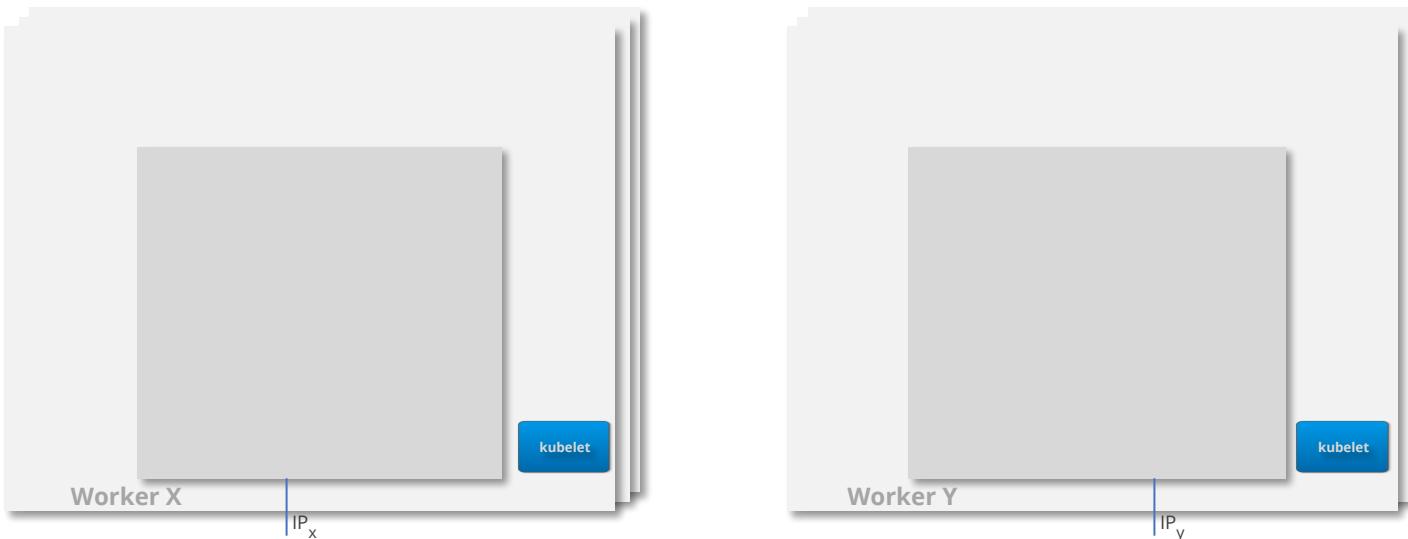
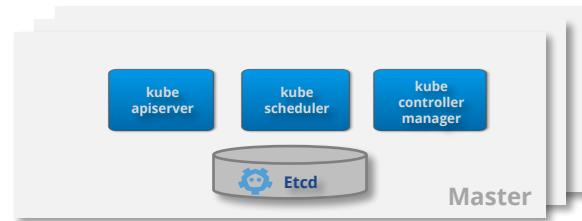
- <https://docs.projectcalico.org/v3.7/getting-started/kubernetes/installation/calico#installing-with-the-etcd-datastore>

# Calico CNI and IPAM in Action



# Pod Lifecycle

Lets start with baseline K8s nodes



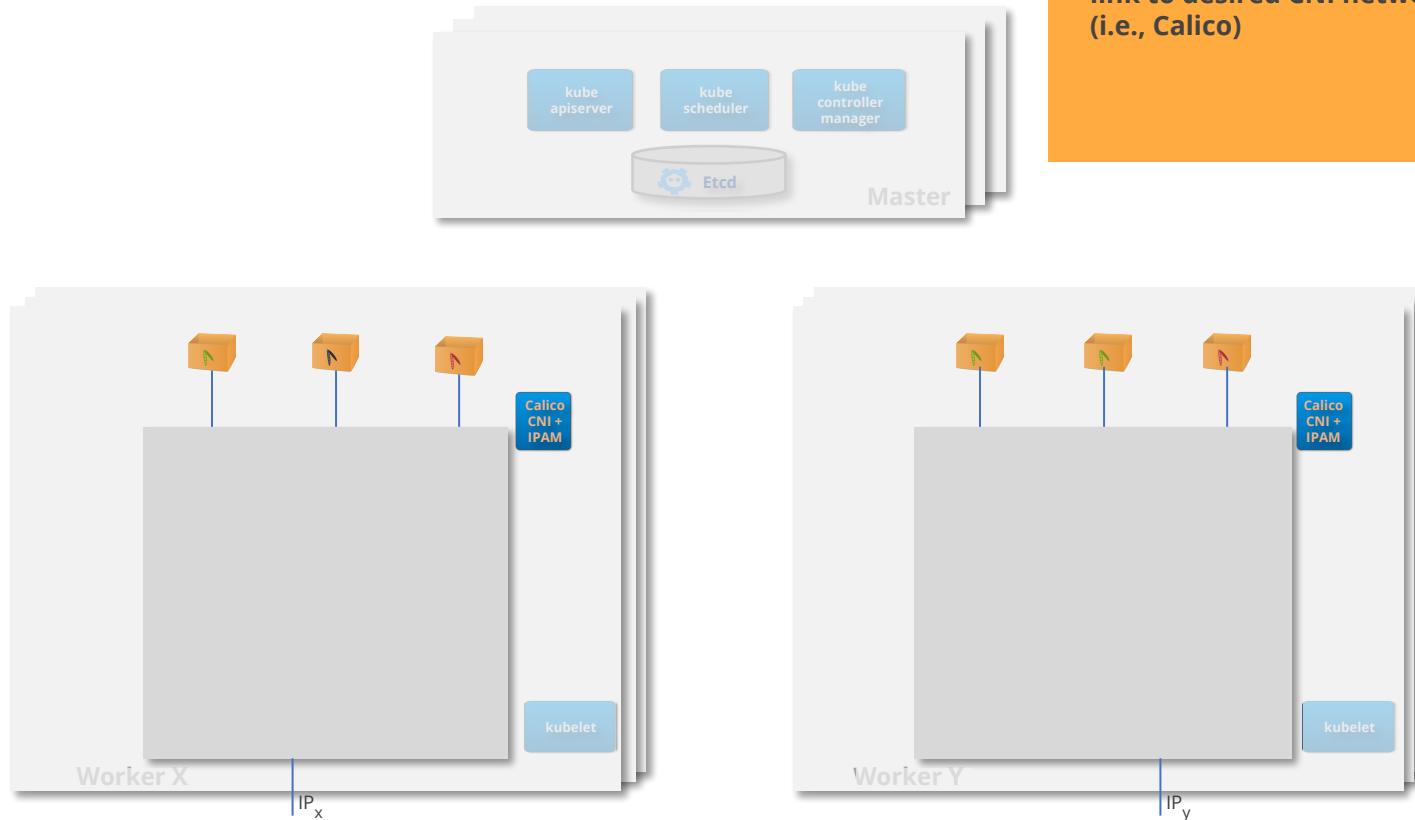
# Pod Lifecycle



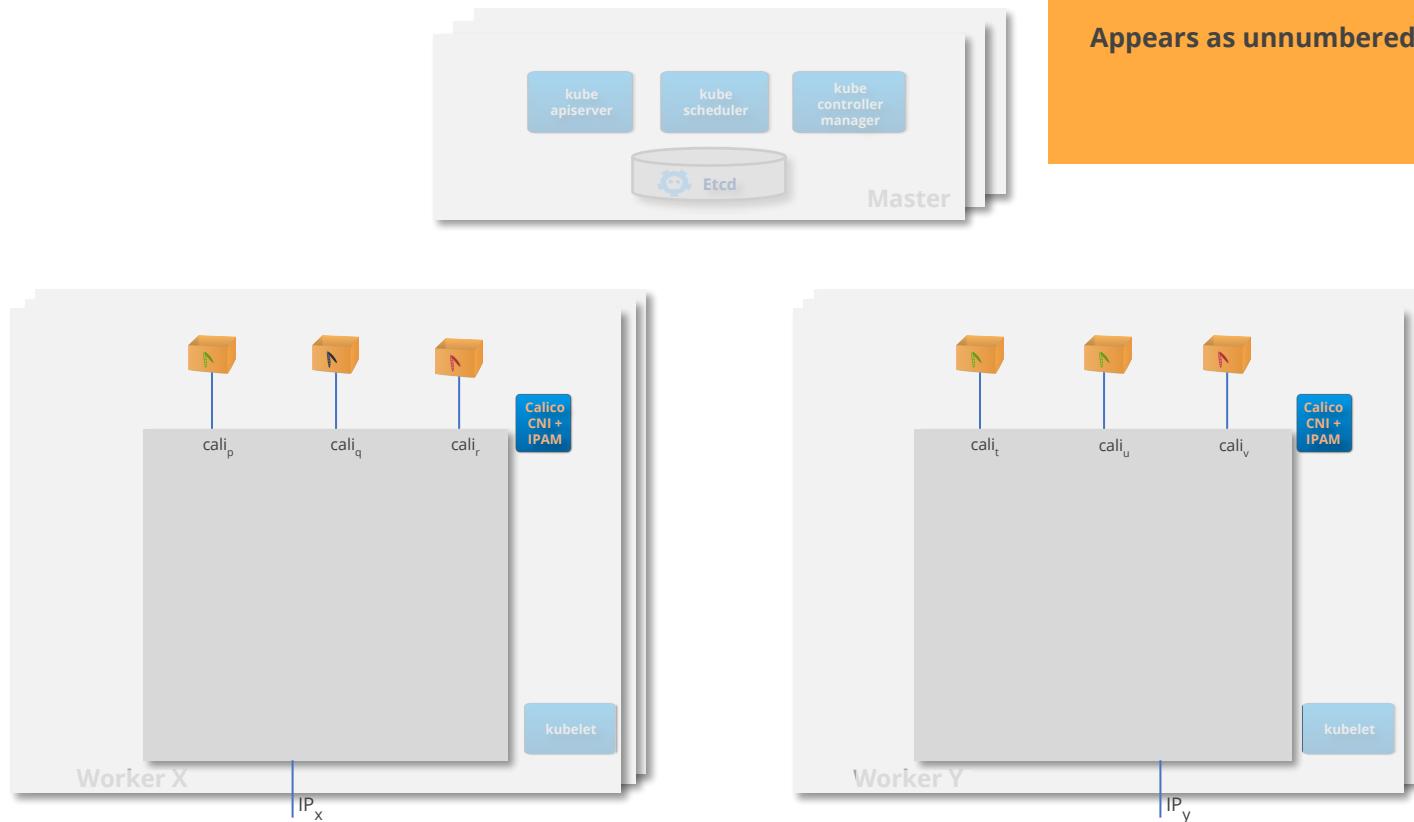
# Pod Lifecycle: CNI



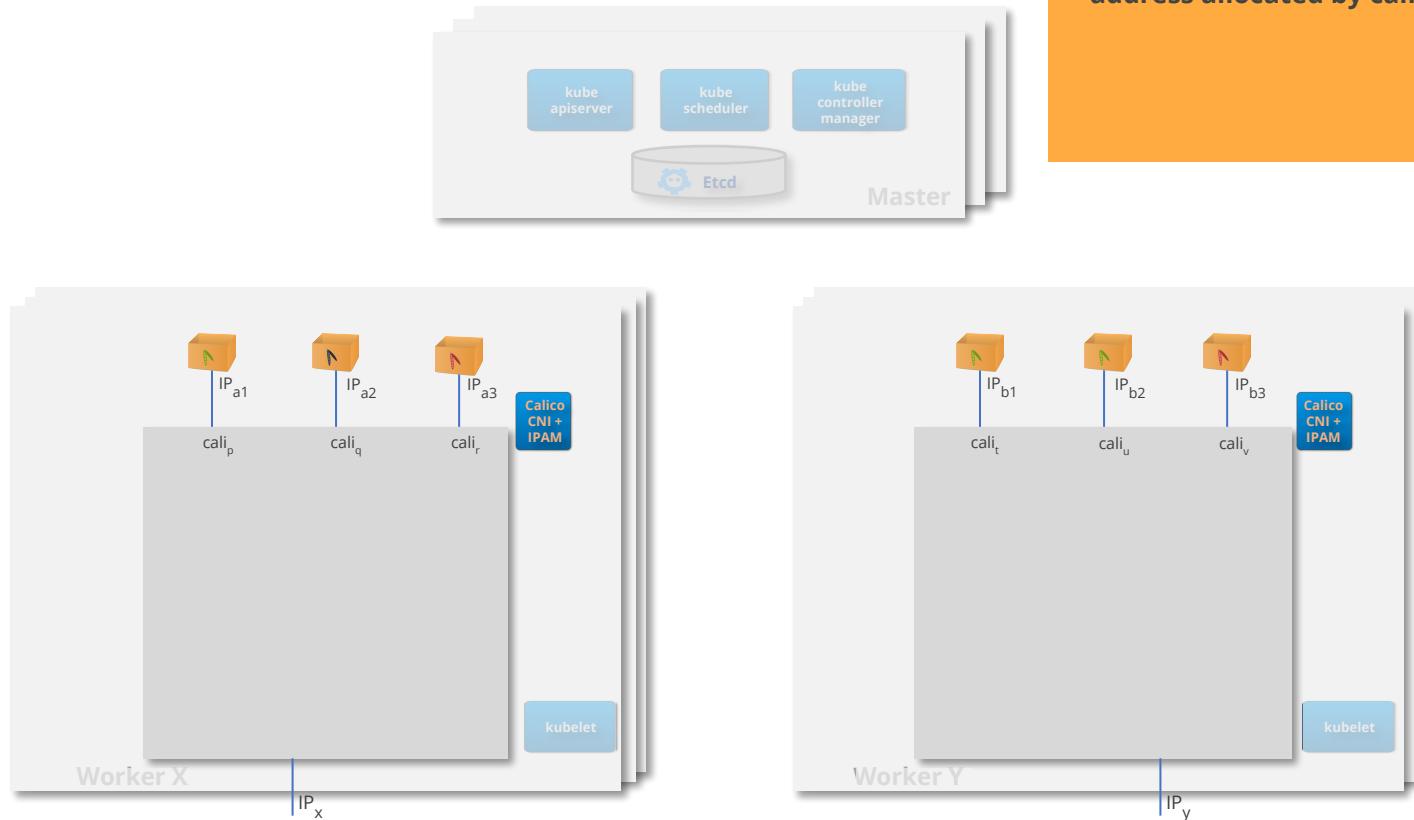
# Pod Lifecycle: CNI



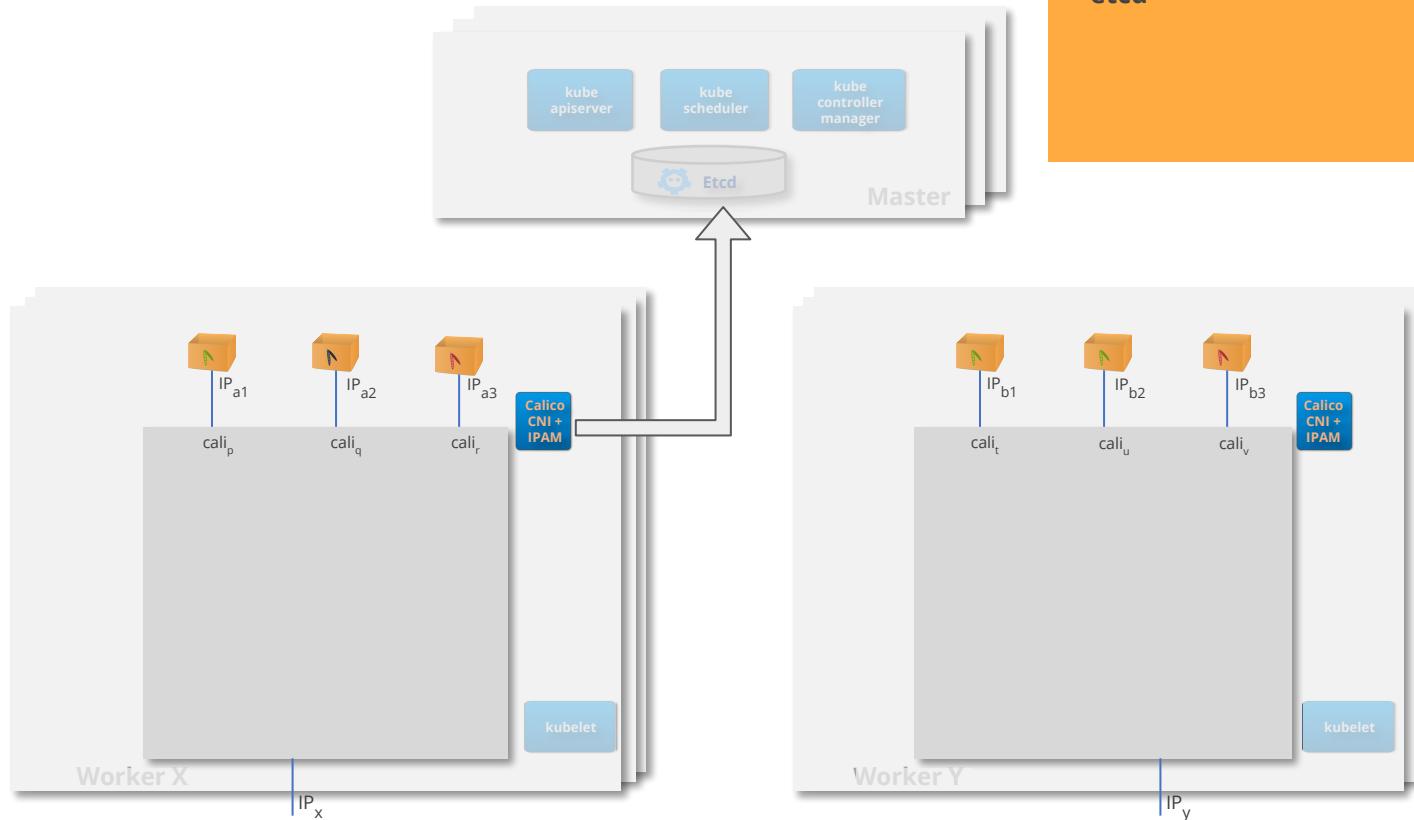
# Pod Lifecycle: CNI



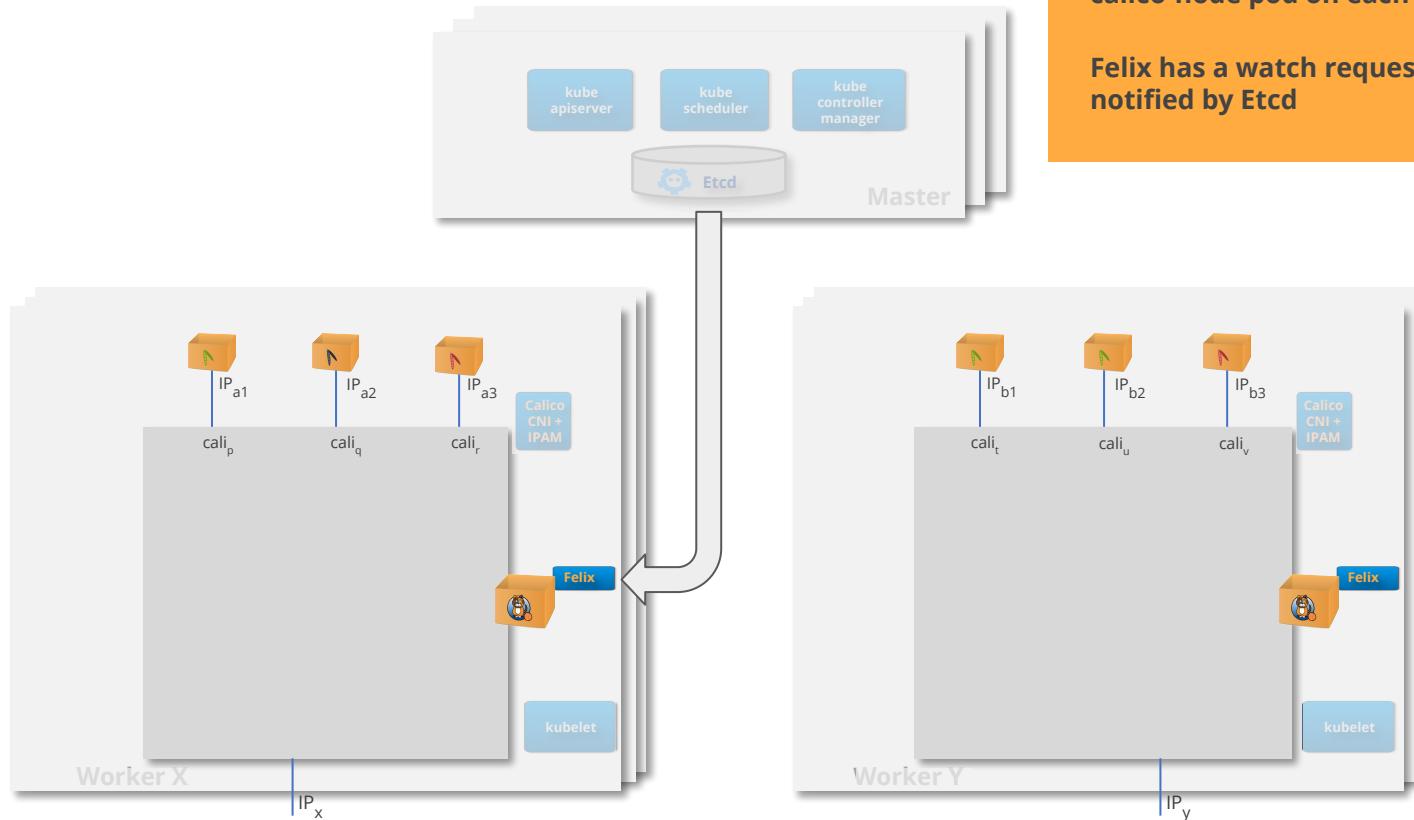
# Pod Lifecycle: Address Block Assignment



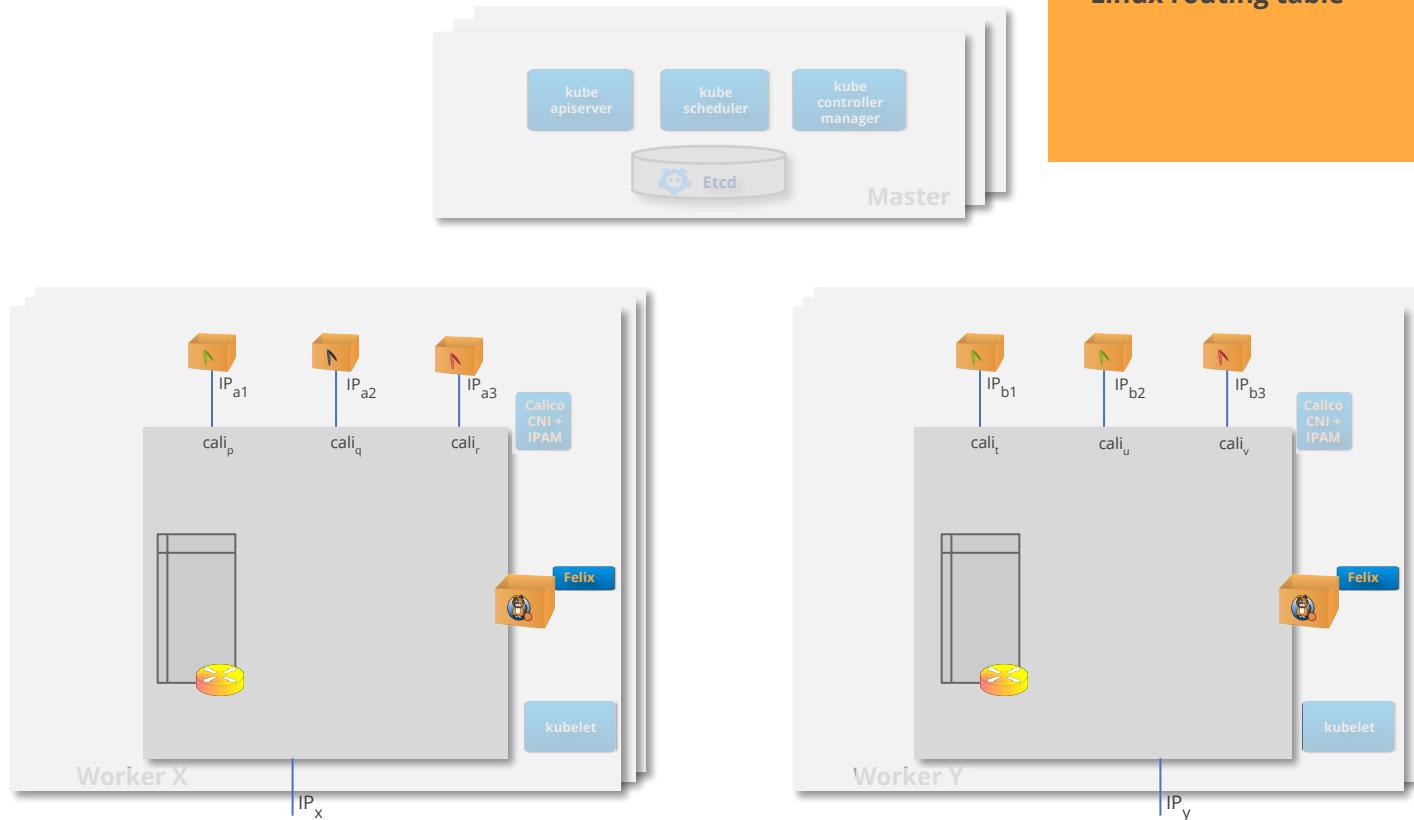
# Pod Lifecycle



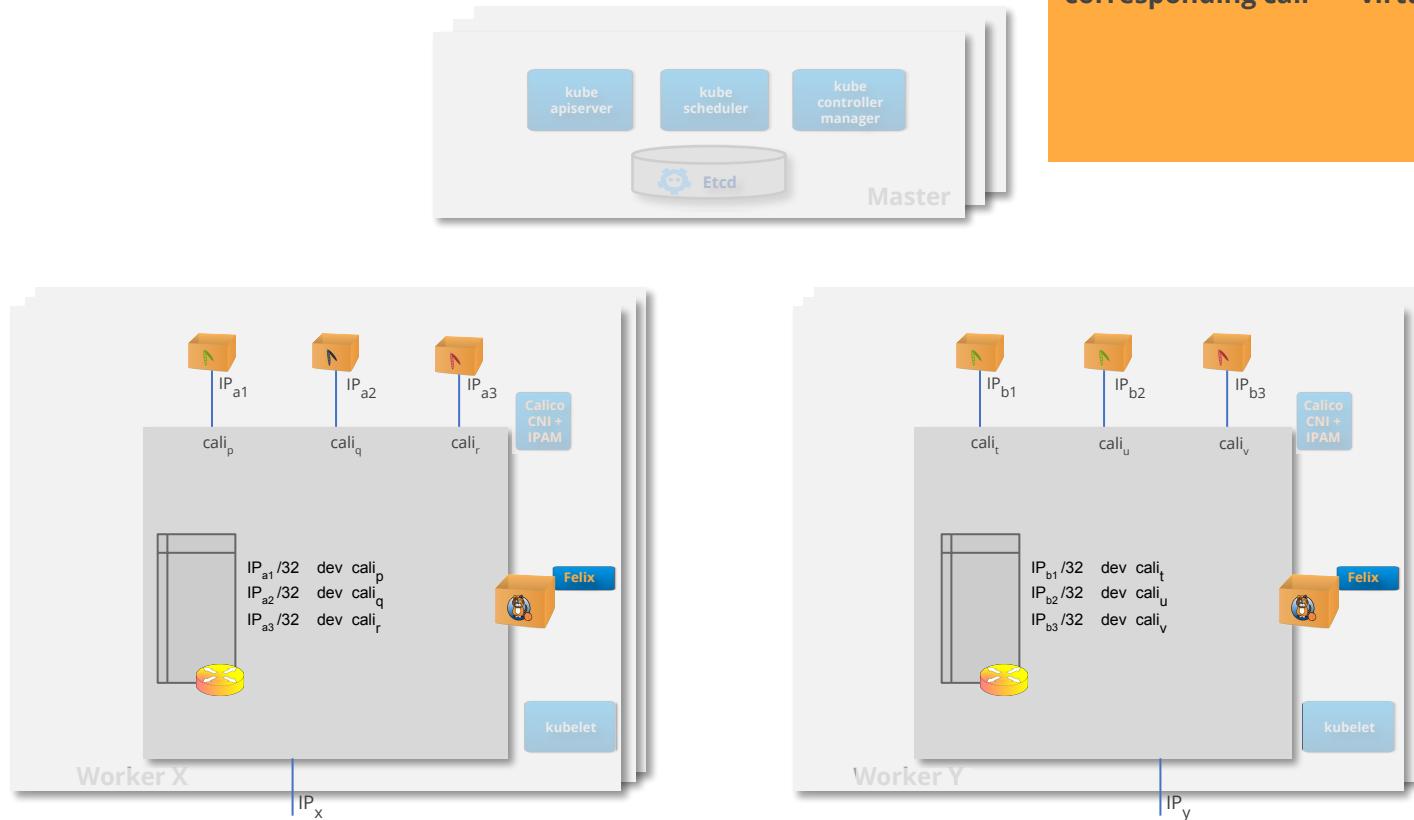
# Pod Lifecycle



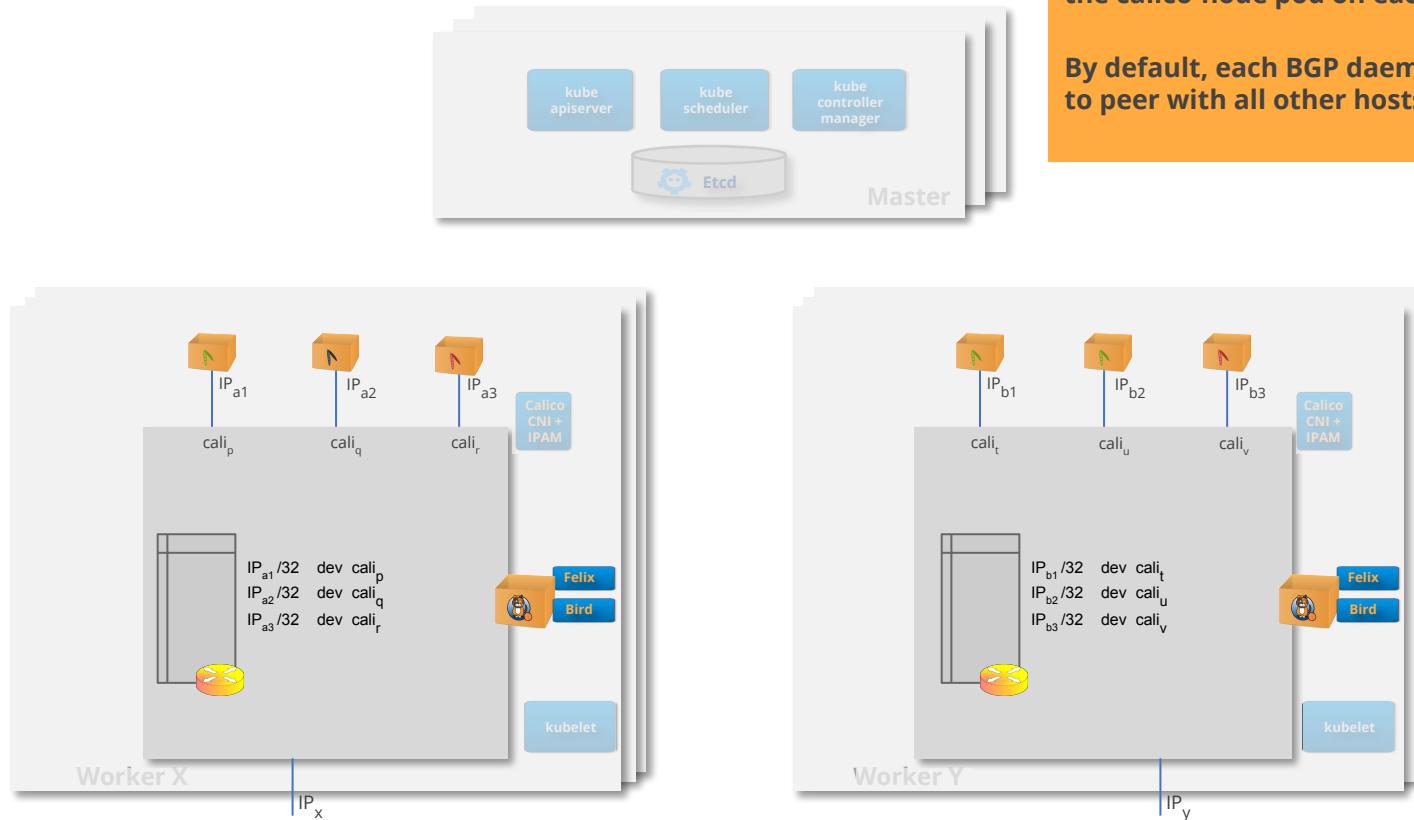
# Pod Lifecycle



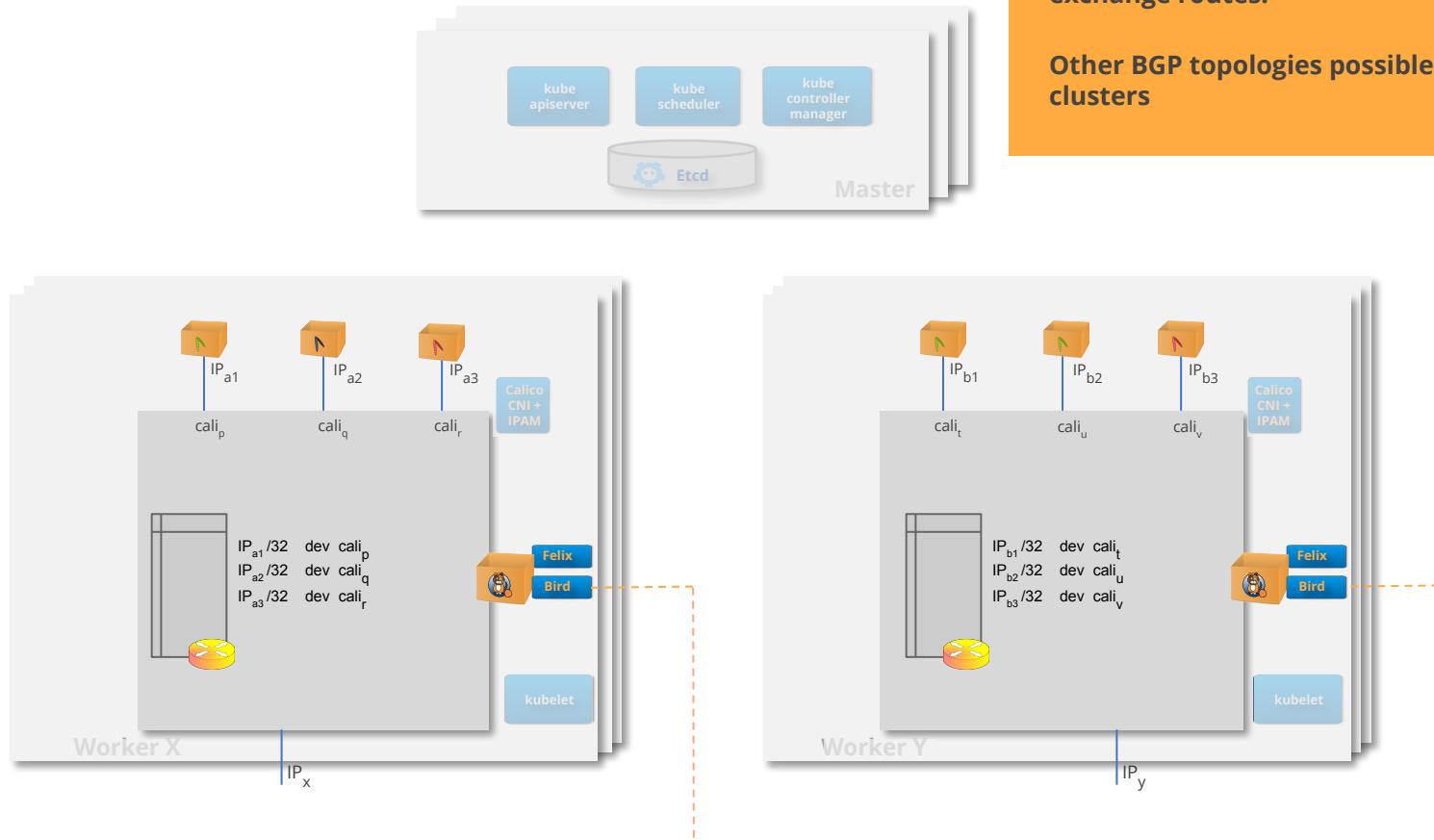
# Pod Lifecycle



# Pod Lifecycle



# Pod Lifecycle



Nodes create a BGP full mesh to exchange routes.

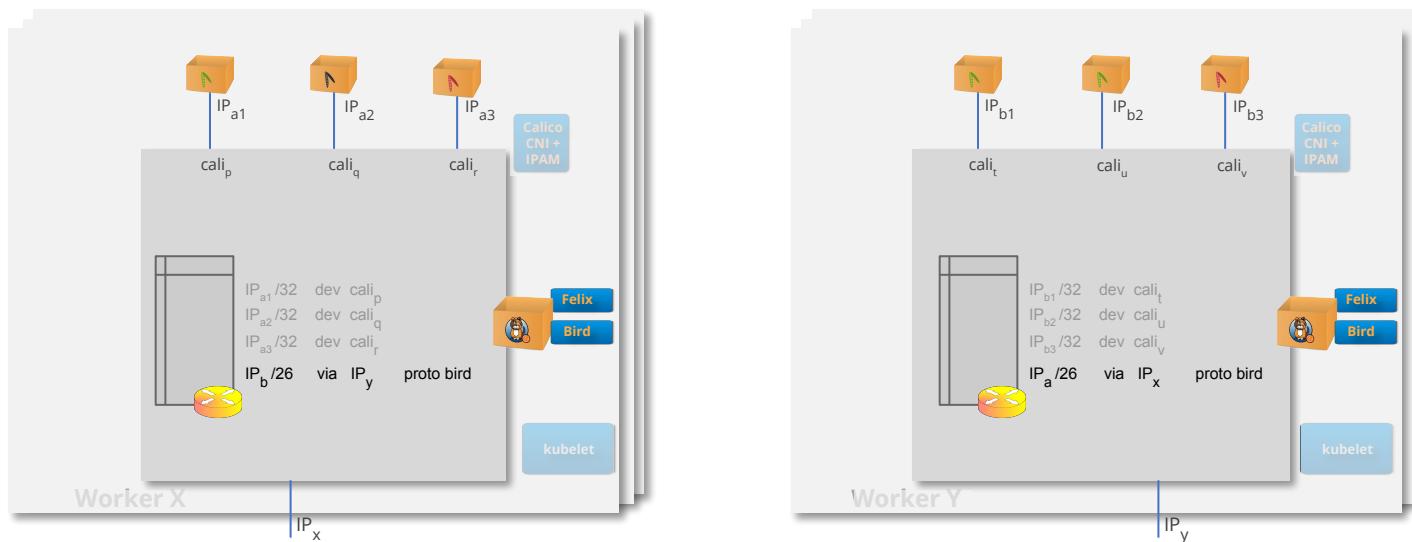
Other BGP topologies possible for large clusters



# Pod Lifecycle

Routes for Pod IP addresses exchanged among nodes via BGP.

Each Linux node simply routes packets using the default routing table.



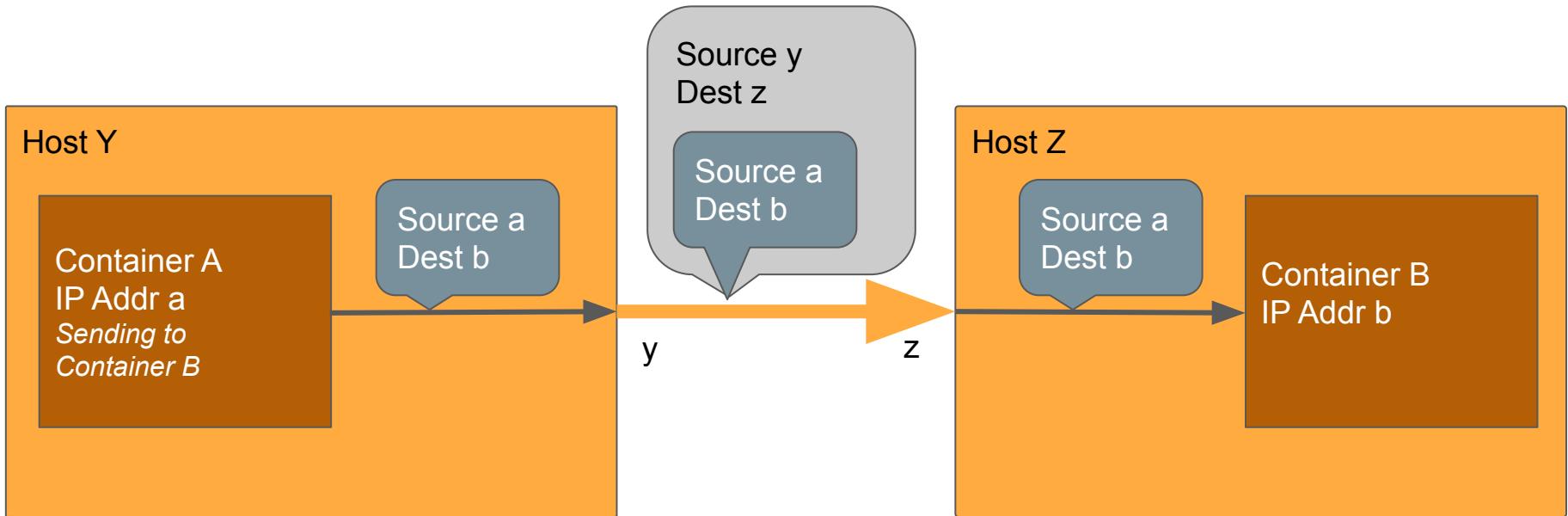
# Overlay vs. Flat Network



# FLAT NETWORKING VS OVERLAY

- > Docker libnetwork, like most virtual networking, uses overlay technology (VXLAN) to transport packets over the underlying network
  - o Advantage: only the host IP address is exposed to the physical network, so only host-to-host IP connectivity is required
  - o Disadvantages:
    - Complexity in troubleshooting (have to look inside packet to find actual src/dest addresses)
    - All traffic in/out of the overlay network must be NAT'd and encapsulated/de-encapsulated
    - Processing overhead & bandwidth inefficiency (inc. MTU considerations)
- > Calico can operate in two modes: Overlay or Non-Overlay (Flat)
  - o Overlay technology used is “IP-in-IP” (or IPIP)
  - o No concept of a “virtual network” - endpoint reachability is separate from network isolation
  - o Docker EE ships with IPIP **enabled by default**

# IP-IN-IP VISUALIZED

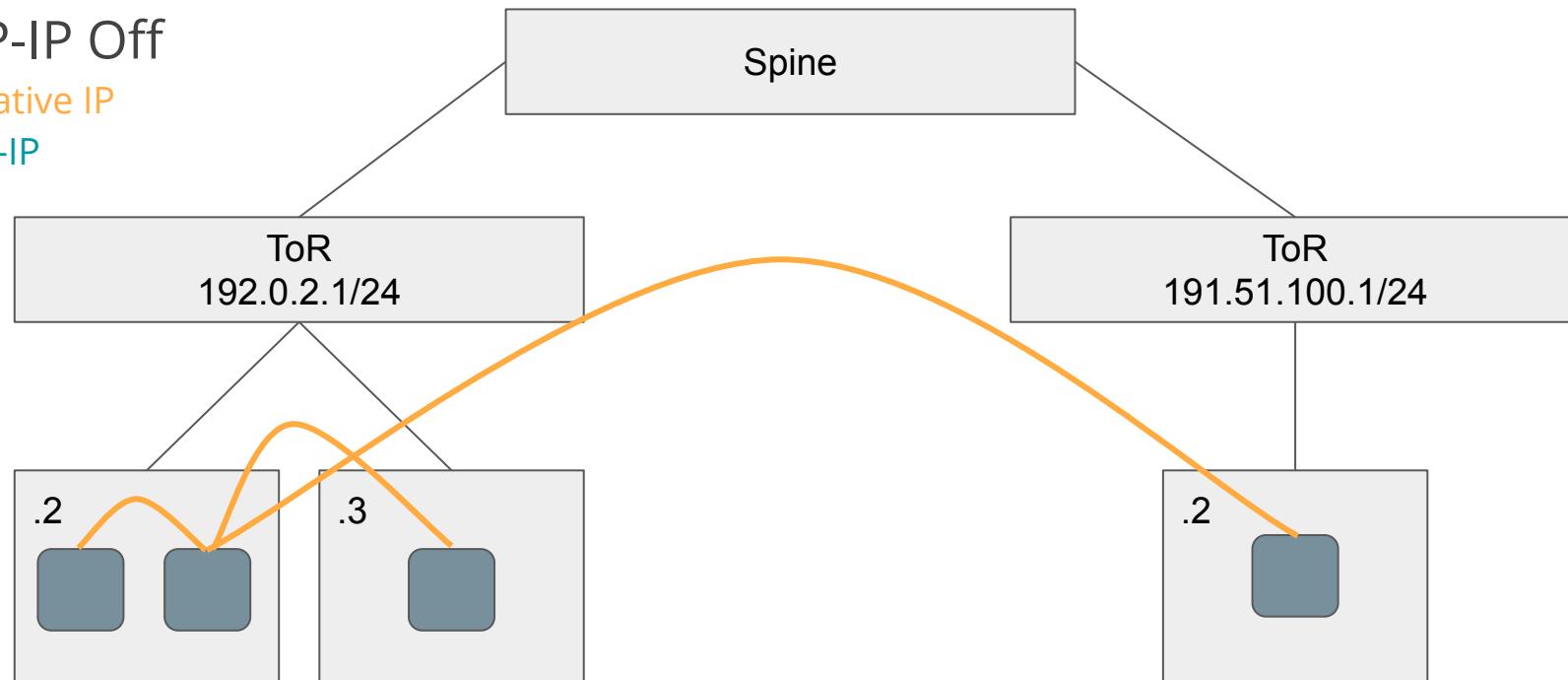


# CALICO IP-IN-IP OPTIONS

IP-IP Off

Native IP

IP-IP

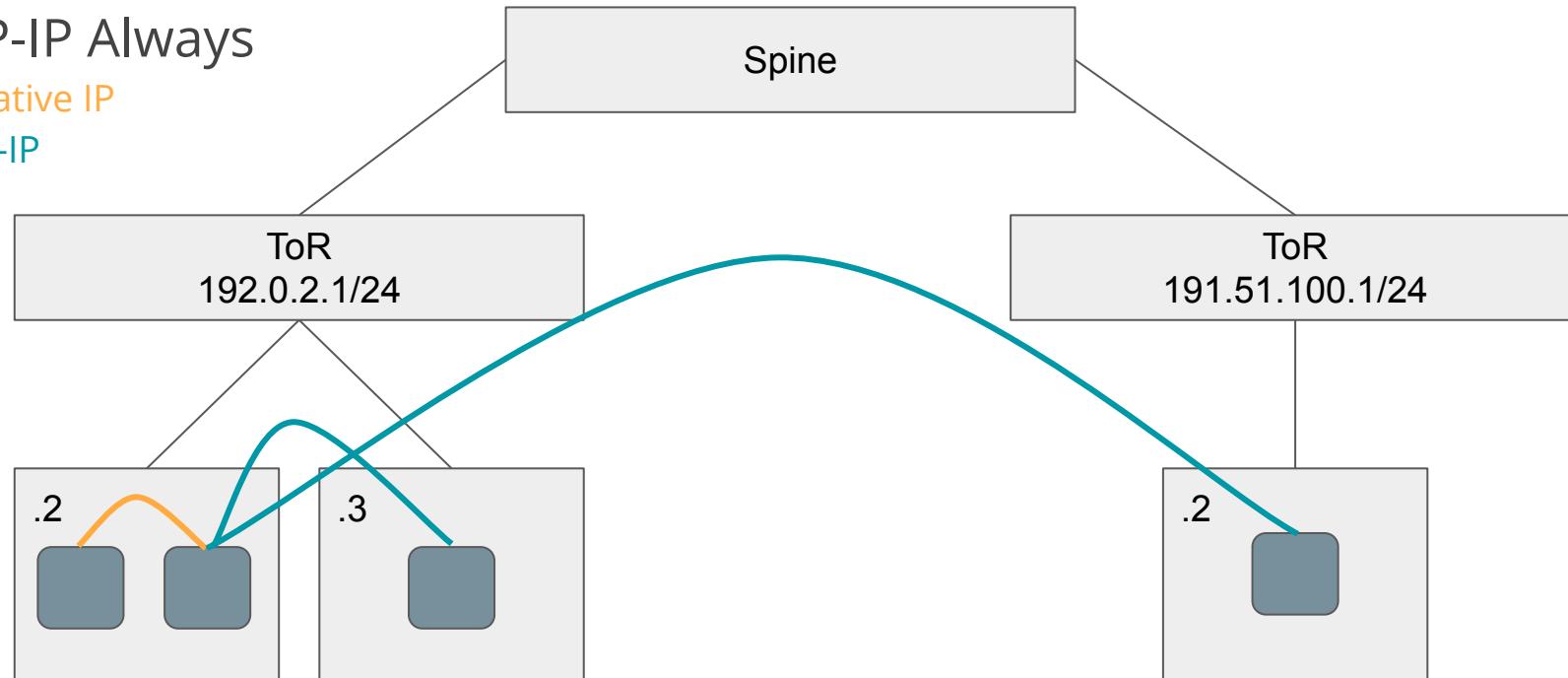


# CALICO IP-IN-IP OPTIONS

IP-IP Always

Native IP

IP-IP

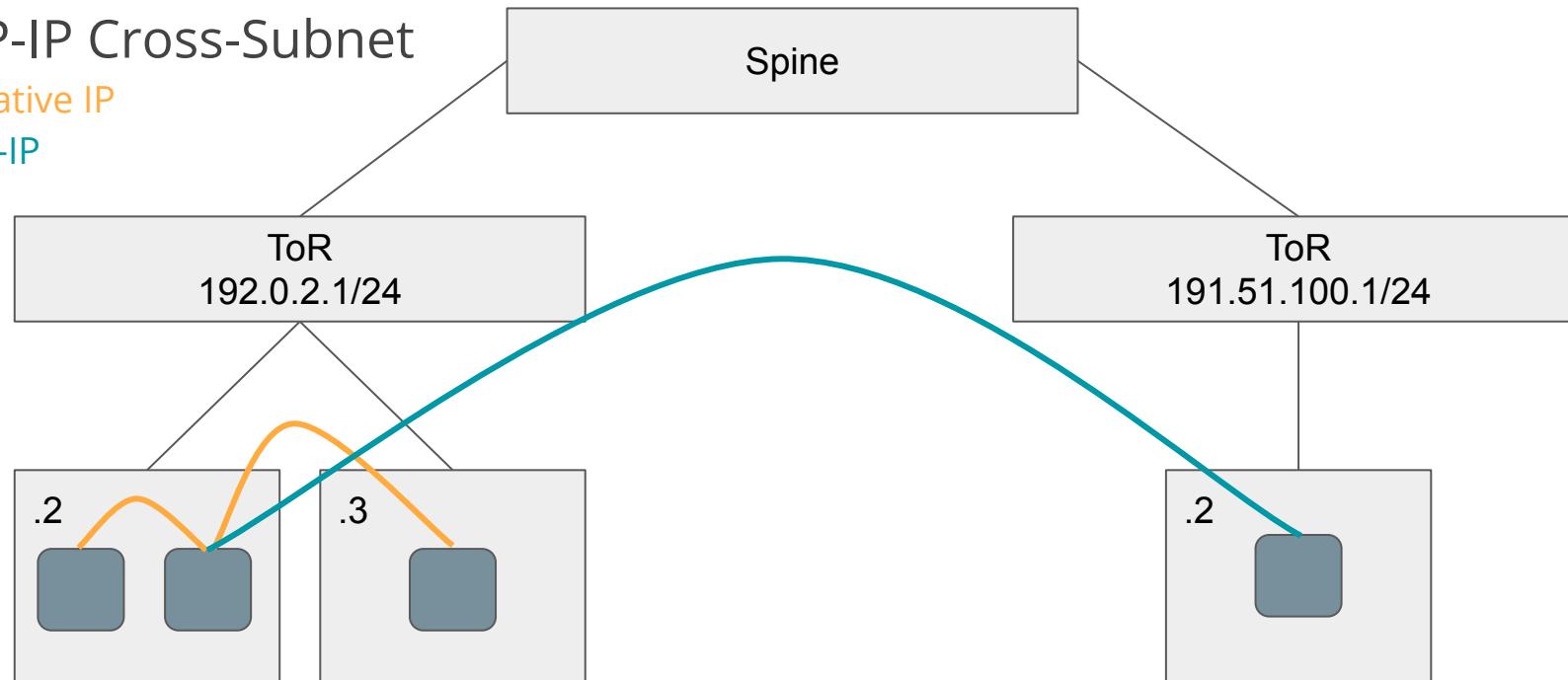


# CALICO IP-IN-IP OPTIONS

## IP-IP Cross-Subnet

Native IP

IP-IP

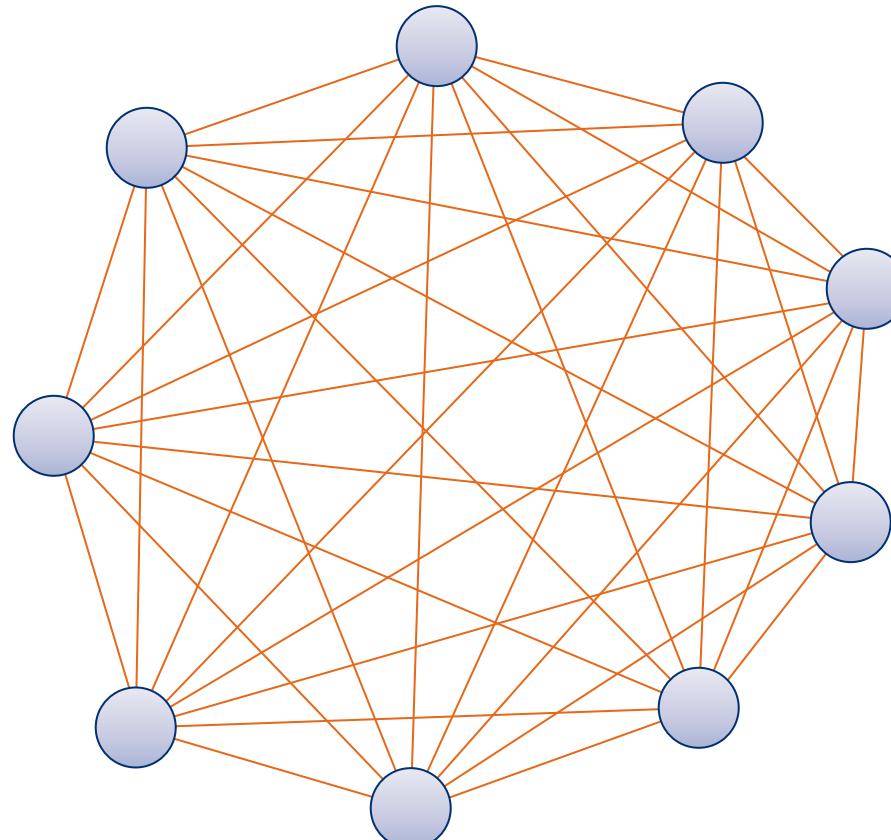


# Declarative Network Policy

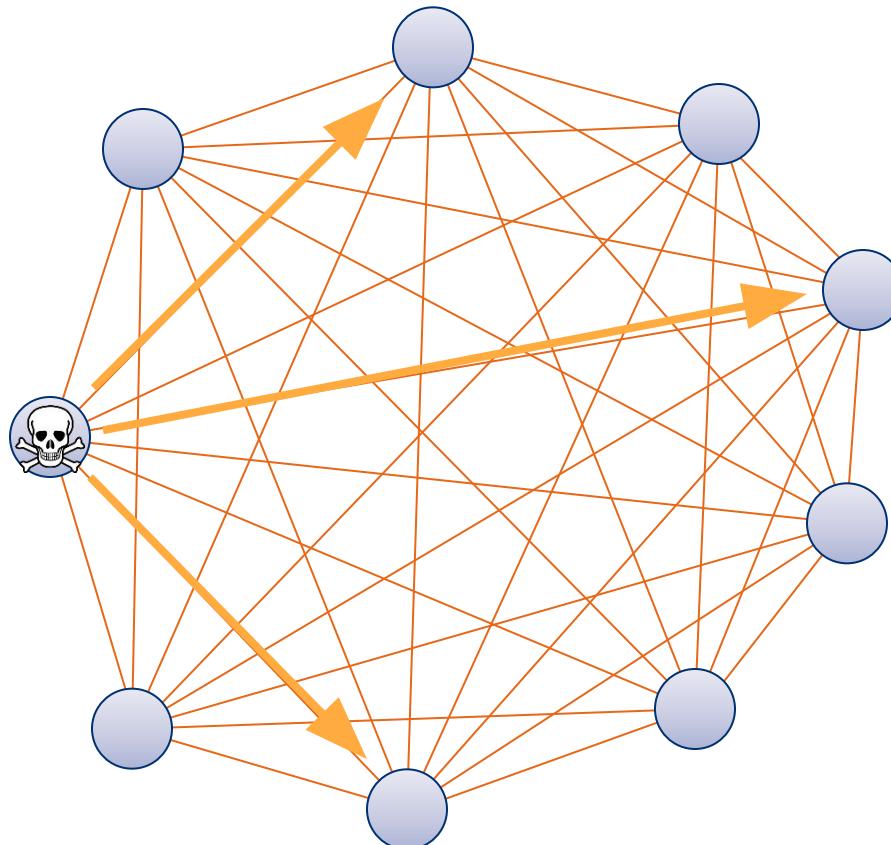


# MOTIVATION FOR NETWORK POLICY

Consider a Kubernetes cluster of  $n$  services. Of the  $n^2$  possible connections between all your services, only a tiny, tiny fraction of them are actually useful for your application.

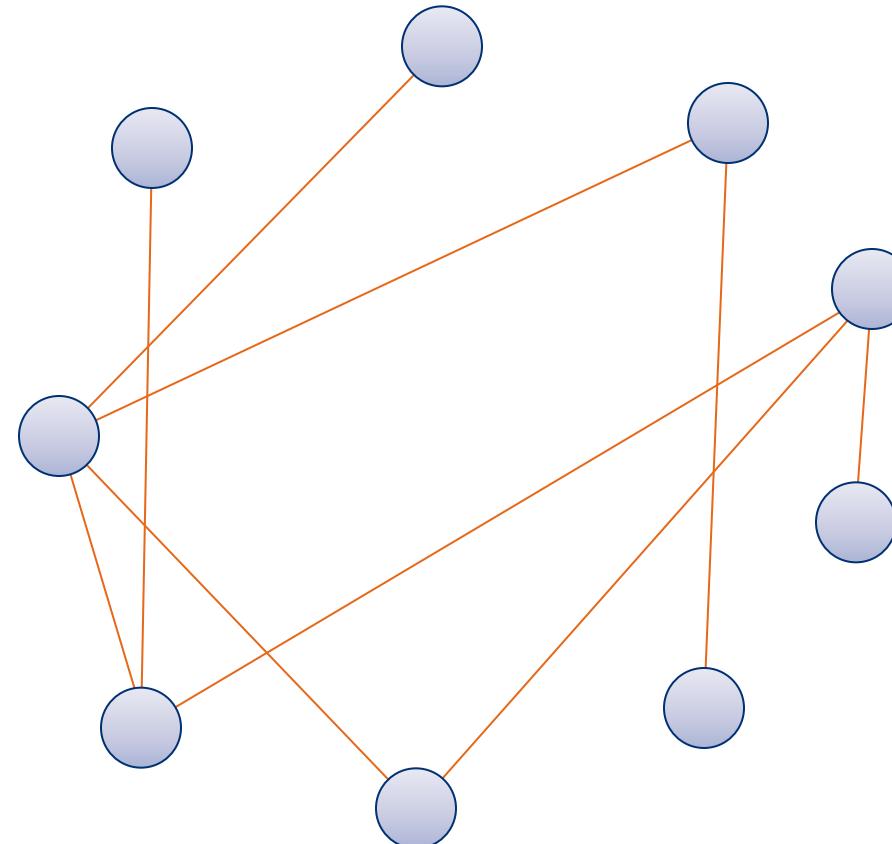


The rest are only useful for an attacker.

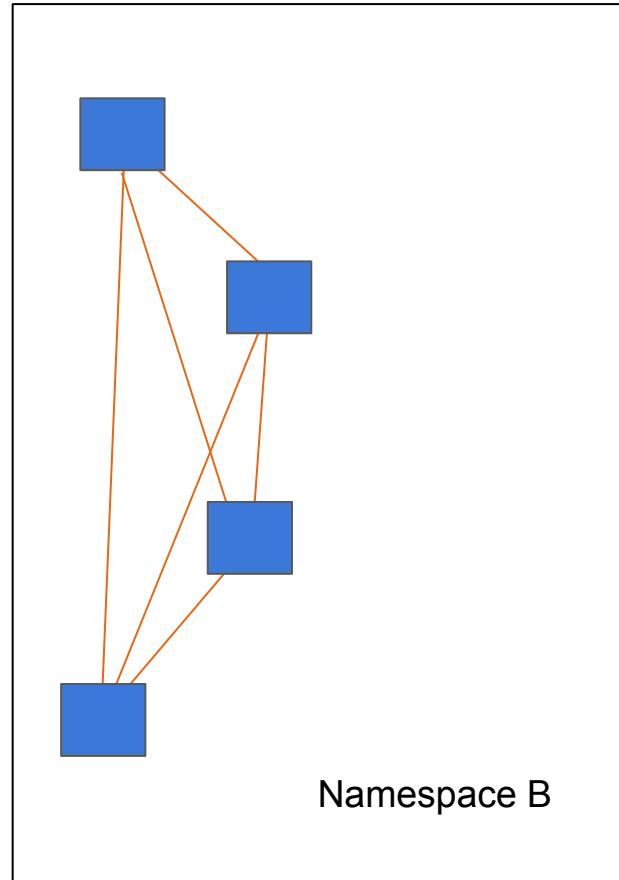
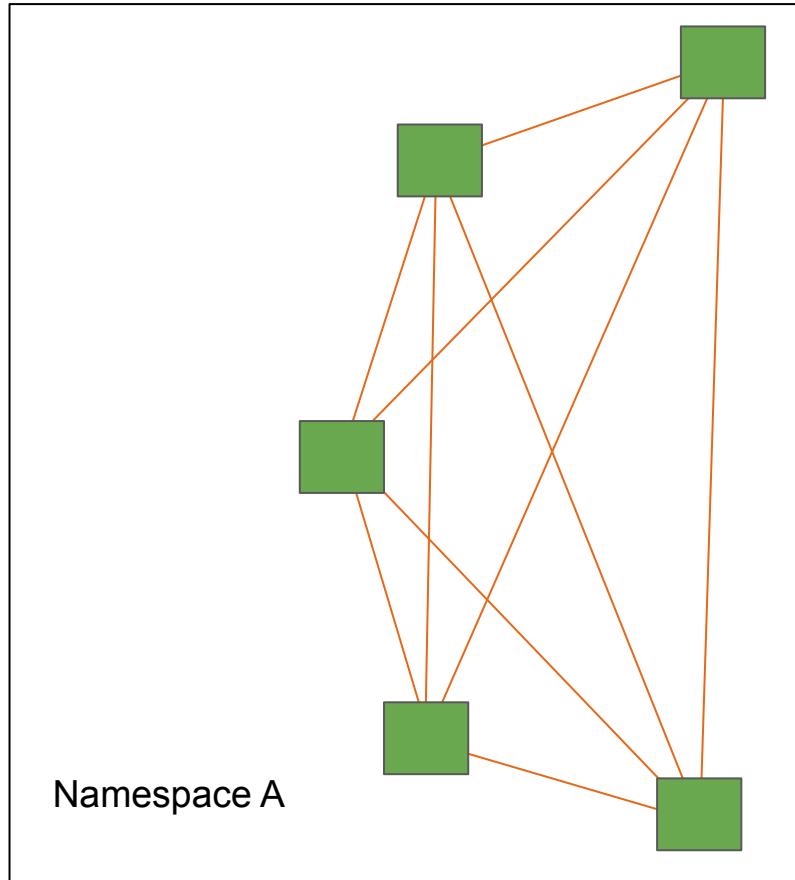


We know that our frontend load balancers don't connect directly to the backend database. dev containers do not connect to prod, and so on.

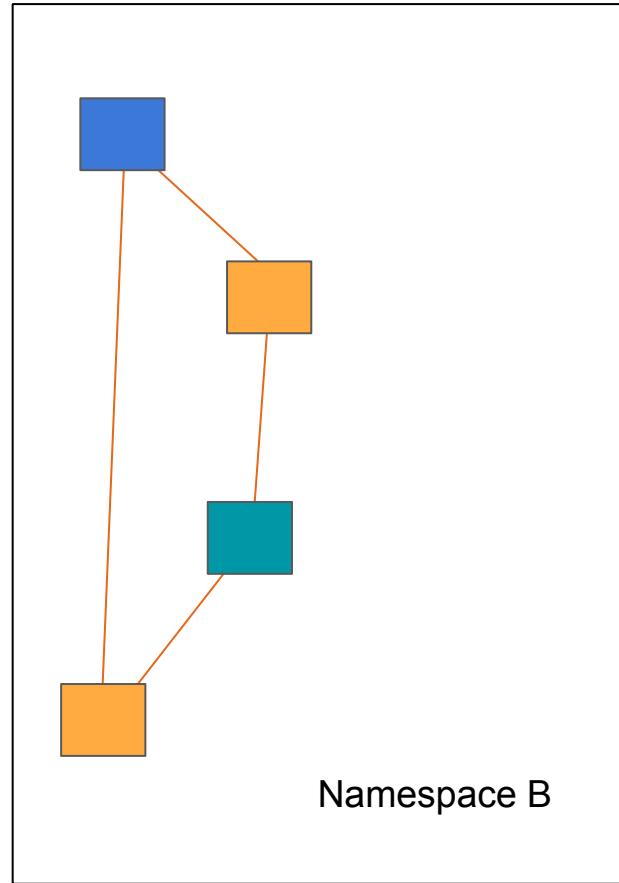
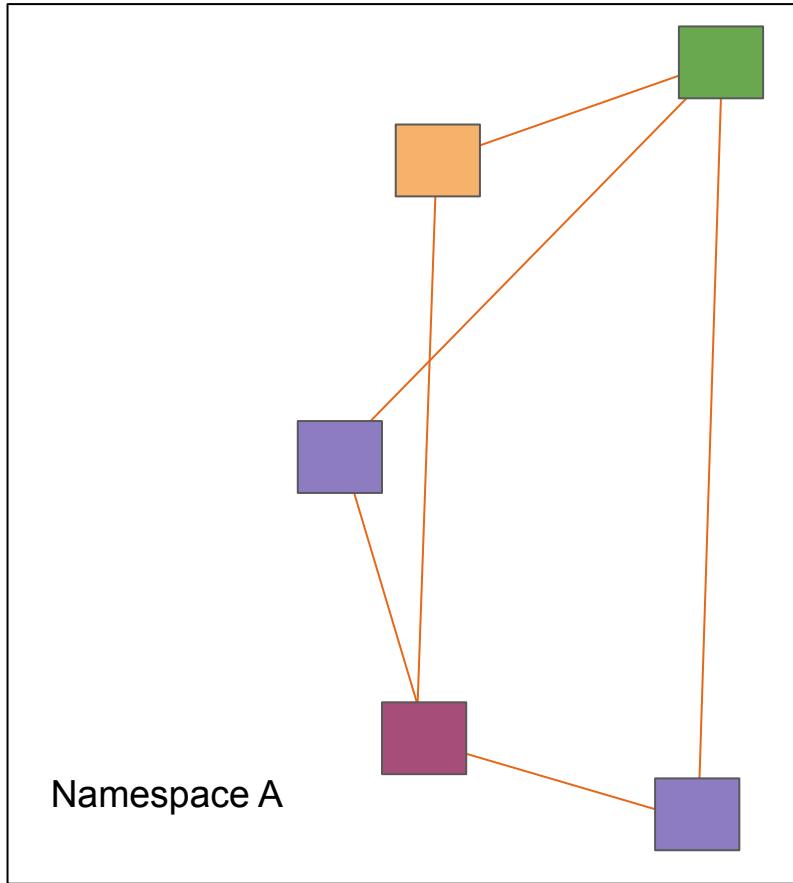
Each connection path that we can eliminate will reduce complexity and improve security.



## Namespace Isolation



## Custom Isolation



# EXAMPLE NETWORK POLICY USE CASES

- > Stage separation - isolate dev / test / prod instances
- > Translation of traditional firewall rules - e.g. 3-tier
- > “Tenant” separation - e.g. typically use namespaces for different teams within a company - but without network policy, they are not network isolated
- > Fine-grained firewalls - reduce attack surface within microservice-based applications
- > Compliance - e.g. PCI, HIPAA
- > Any combination of the above

# KEY KUBERNETES CONCEPT: LABELS

- Labels are **key/value pairs** that are attached to objects, such as pods.
- Labels are intended to be used to specify identifying attributes of objects that are **meaningful and relevant to users**, but do not directly imply semantics to the core system.
- Labels can be used to **organize and to select subsets of objects**.
- Labels can be attached to objects at **creation time** and subsequently added and **modified at any time**.
- Each Key must be unique for a given object.

```
"labels": {  
    "key1" : "value1",  
    "key2" : "value2"  
}
```

Source: <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/>

# Thank you!

## Now labs!

See Github to get started.

