CLOUD NATIVE CON Europe 2017

KubeCon A CNCF EVENT

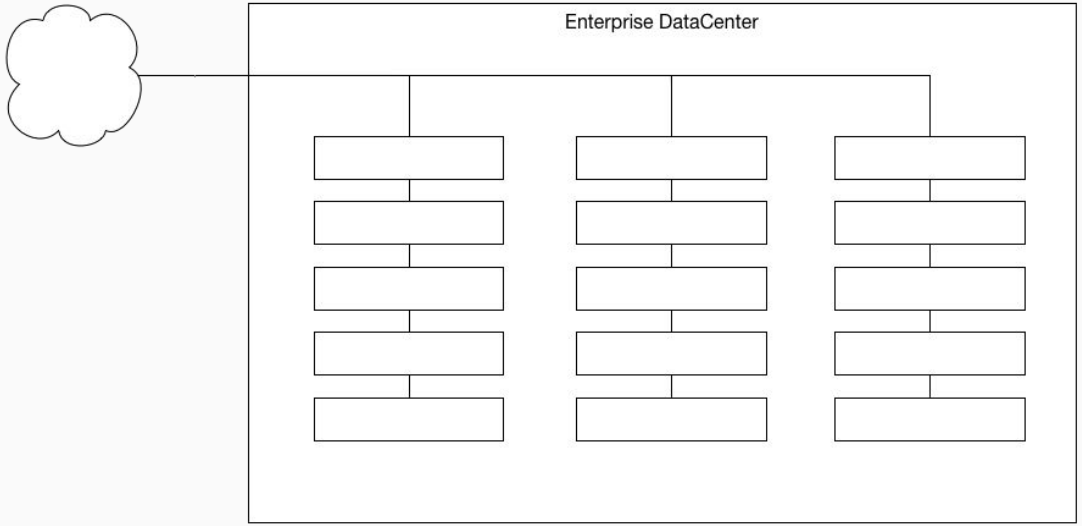
Network independent ACLs

Why Security shouldn't always be tied to your network

Bernard Van De Walle, Aporeto

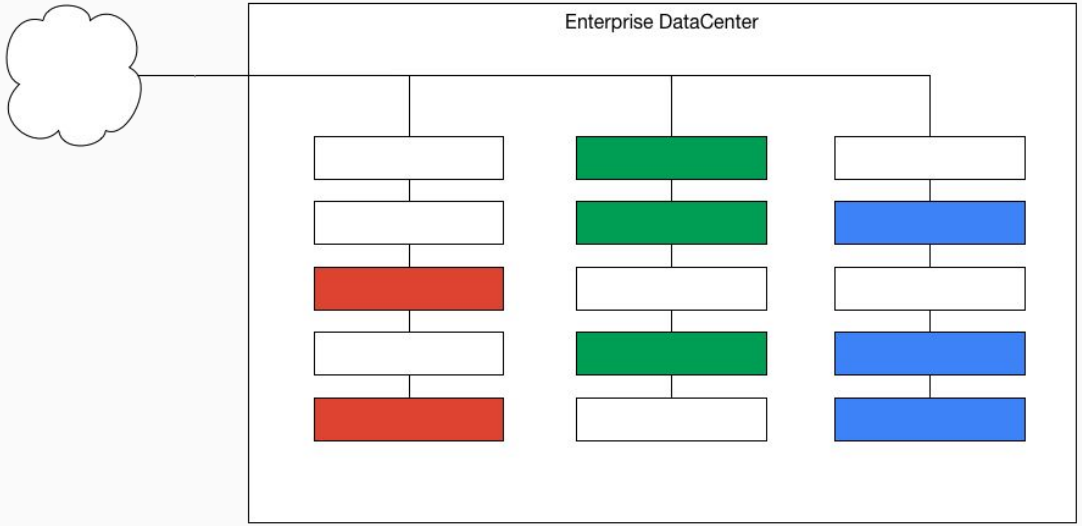
@bvandewa

Once upon a time...



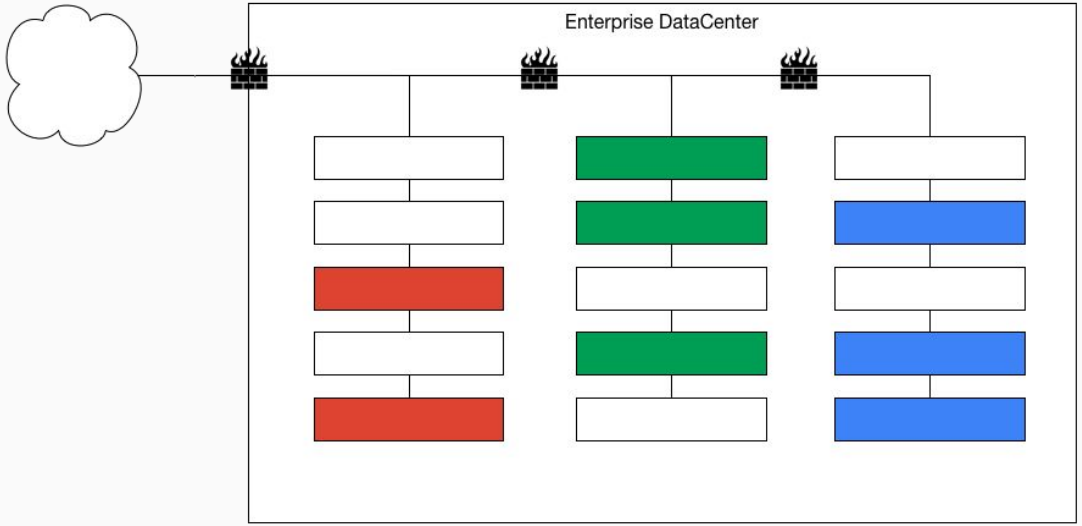
Once upon a time...

Security levels



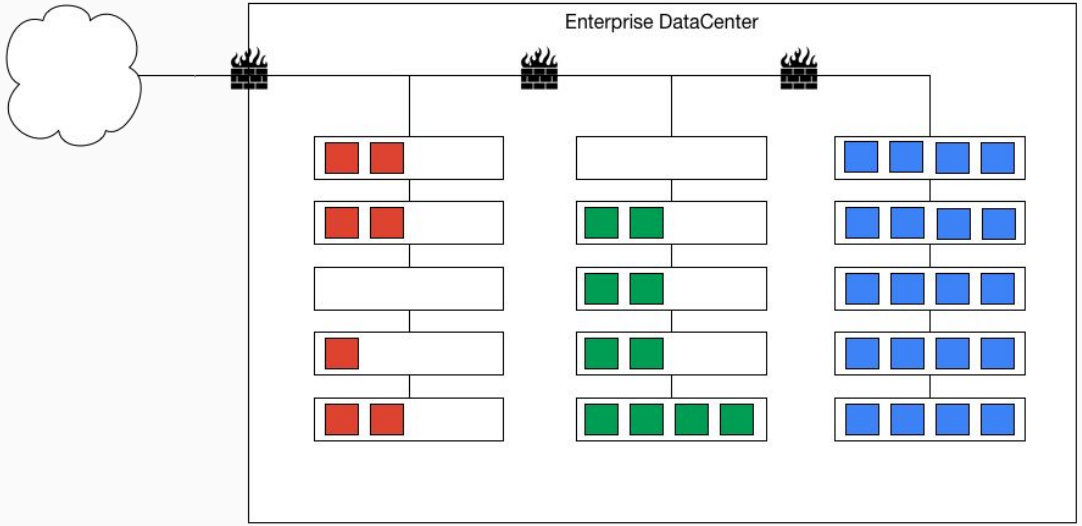
Once upon a time...

Perimeter security



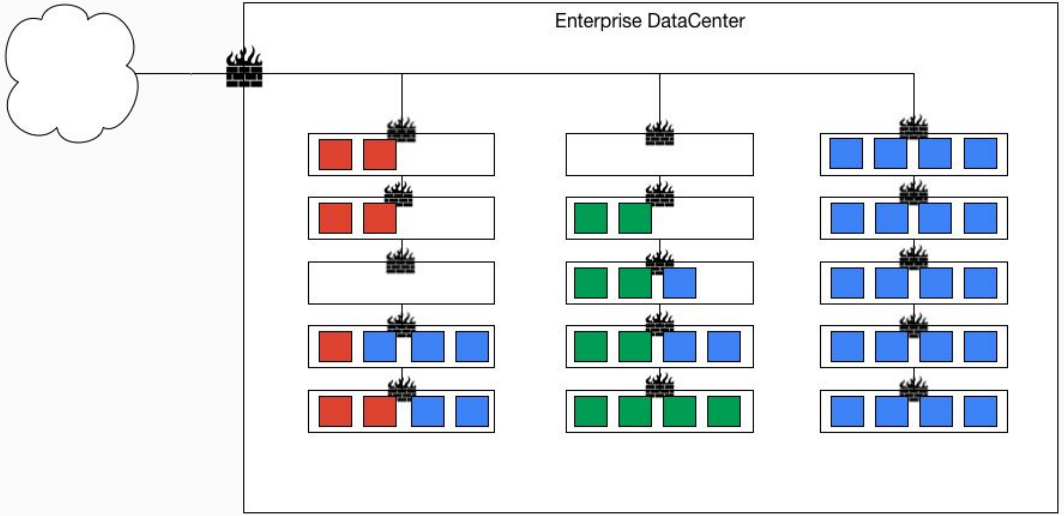
Once upon a time...

Micro-Services



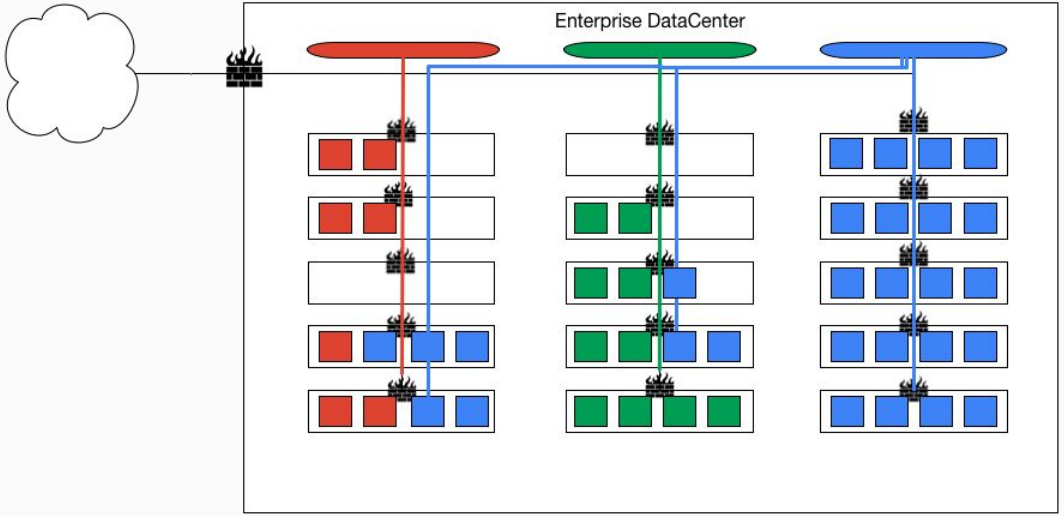
Once upon a time...

Distributed firewalls



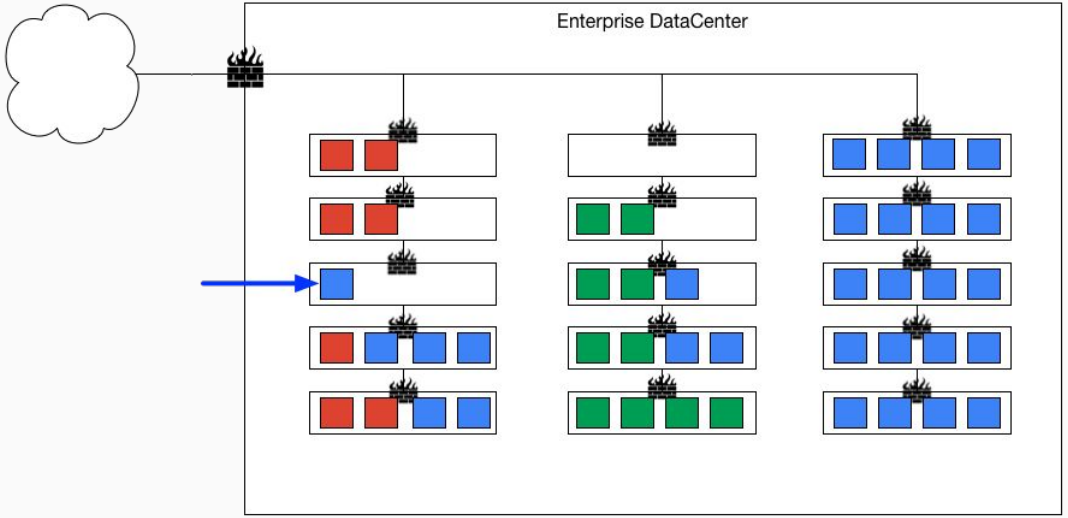
Once upon a time...

SDN and VPN solutions



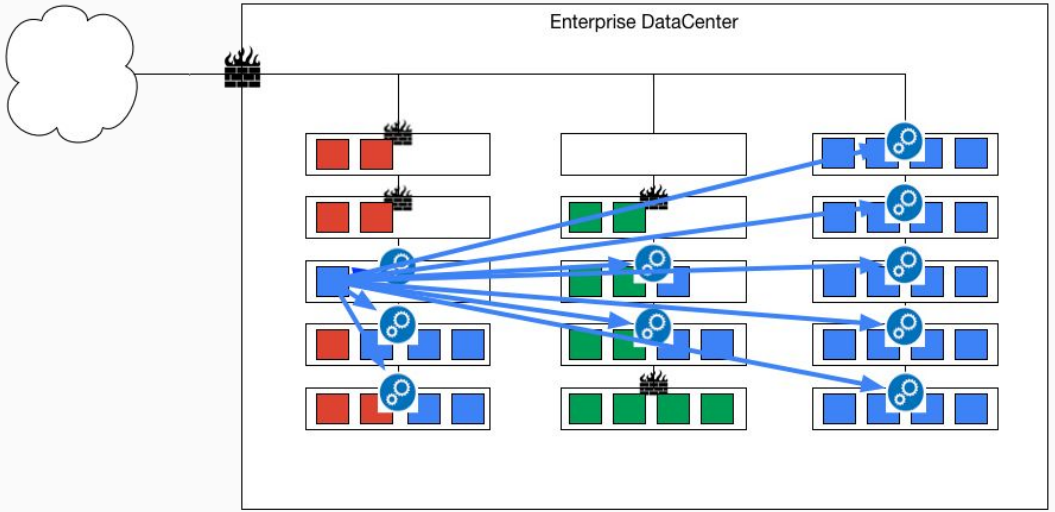
Once upon a time...

Provisioning assets



Once upon a time...

Exponential Complexity



Network

≠

Network security

Zero Trust Networking

Network is insecure by default

Threat model: inside network as insecure as outside network

Network primitives are irrelevant

IP and Port numbers do not carry any information

Flows need to be authorized

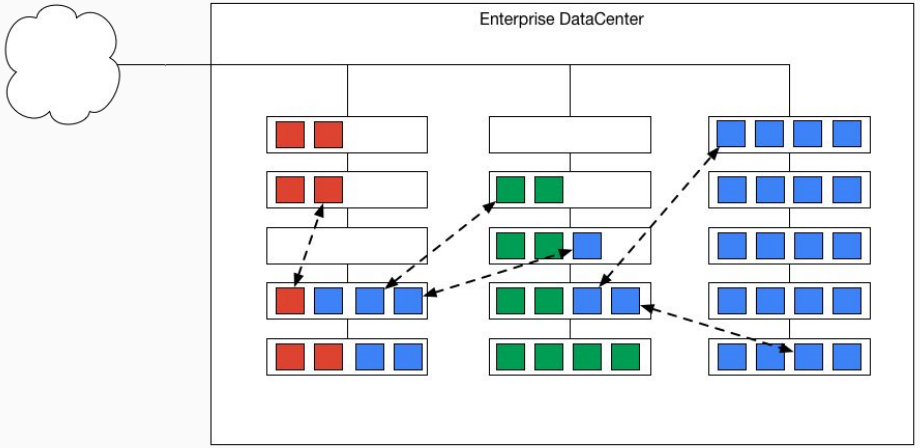
Every connection results from a successful authorization/authentication

Declarative policy language

High-level language to automate policy creation/deployment
(Yet Another Policy Language)

Zero Trust Networking

- Context and Identity used for flow authentication
- Network identity ≠ Endpoint identity
- Secure by default
- Keep the network **simple**



Kubernetes

Zero-Trust networking in Kubernetes

Kubernetes Networking (reachability)

- Based on CNI
- Built-in (GKE, …) or plugin based
- IP doesn’t carry any information

Kubernetes objects

- Associated Identity
  - Name
  - Namespace
  - Labels

```
apiVersion: v1
kind: Pod
metadata:
  name: external
  namespace: demo
  labels:
    role: external
    app: nginx
spec:
  [...]
```

Kubernetes network policies

- White list model
- No default implementation
- Ingress only

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: backend-policy
  namespace: demo
Spec:
  [...]
```

Kubernetes network policies

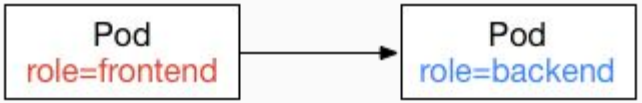
- Explicit activation per **namespace**
- Annotation for activation

```
kind: Namespace
metadata:
  name: demo
  Annotations:
     net.beta.kubernetes.io
     /network-policy: |
       {
         "ingress": {
          "isolation":"DefaultDeny"
         }
       }
```

Kubernetes network policies

- Rules apply to specific Pods
- Pods selected based on labels

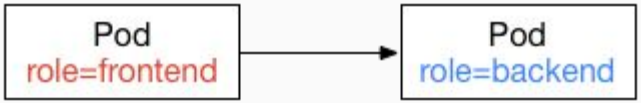
  `role=backend`




```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
      - podSelector:
        matchLabels:
          role: frontend
```

Kubernetes network policies

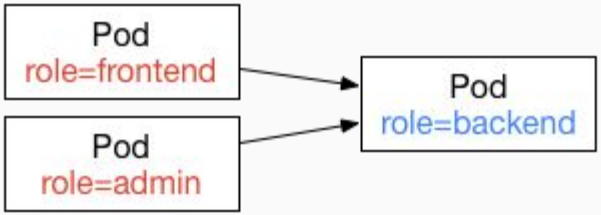
- Rule defines Pods allowed to **send traffic**
- Allowed traffic selected based on labels `role=frontend`




```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
      - podSelector:
        matchLabels:
          role: frontend
```

Kubernetes network policies

- Rules are additive
- Each rule allows additional traffic




```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
      - podSelector:
        matchLabels:
          role: admin
```

Implementations

Tied to networking backend

Because Policing is based on IPs

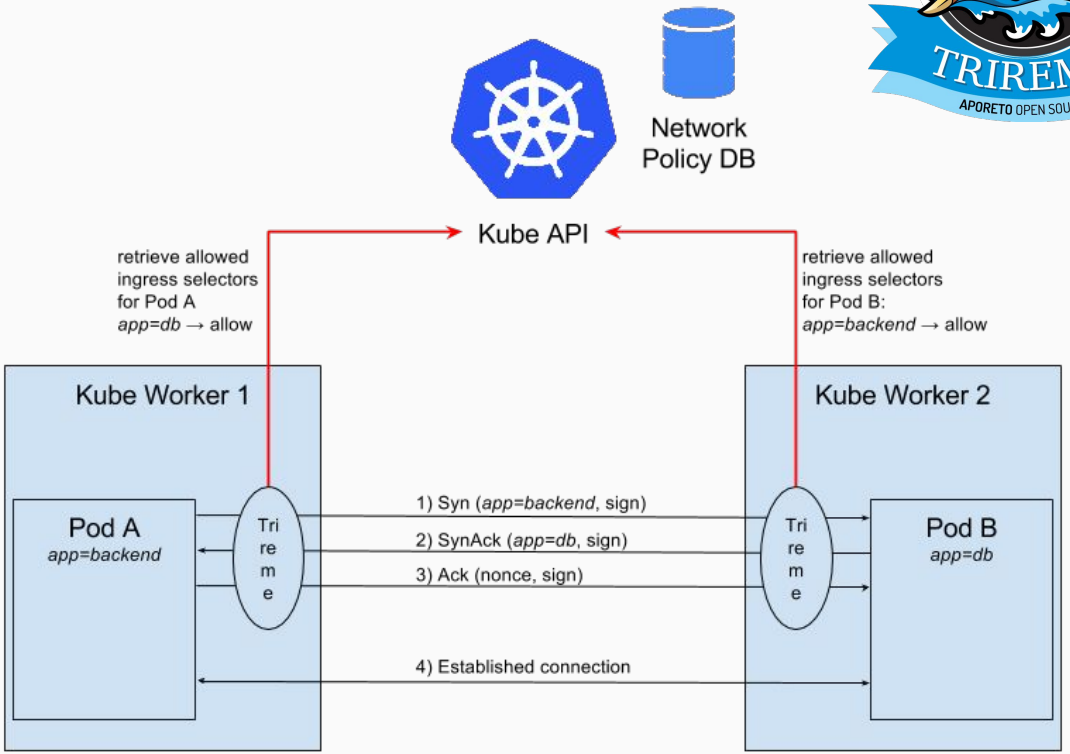
Trireme

https://github.com/aporeto-inc/trireme-kubernetes

- Identity is the pod label
- **IP** irrelevant. Network independent
- Compatible with any Networking backend

E2E authentication

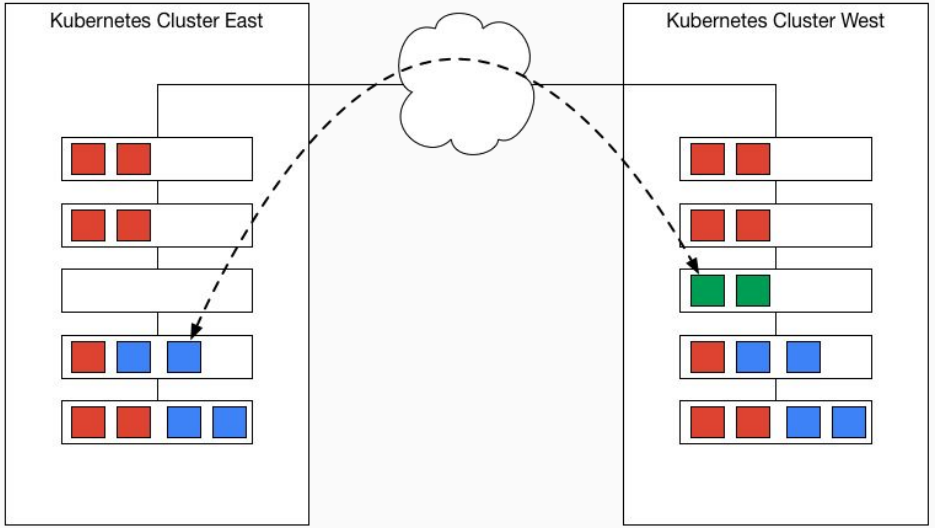
- Identity added on TCP flows handshake
- Identity signed



“Demo Time”

Cluster federation

With Zero-Trust Networking



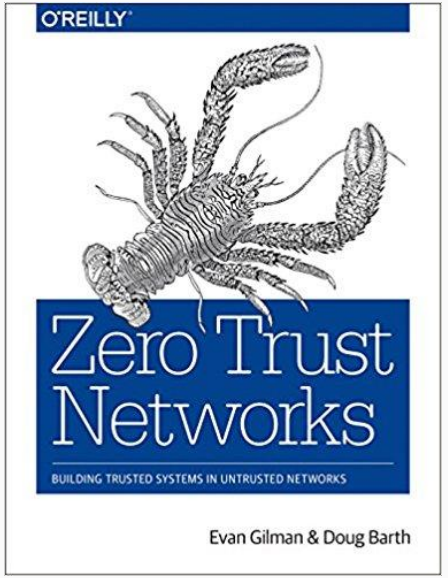
Network reachability

≠

Network security

More about zero-trust

- Encryption
- Visibility
- Auditing



Thanks!

@bvandewa

Trireme on Github:
https://github.com/aporeto-inc/trireme-kubernetes

Demo code and slides:
https://github.com/bvandewalle/kubecon-zerotrust