Istio: Weaving, Observing and Securing Microservices

LIN SUN

Senior Technical Staff Member, IBM

@linsun_unc

The Problem

modern distributed architecture

container based services
deployed into dynamic environments
composed via the network

The Problem

IT’s shift to a modern distributed architecture has left enterprises unable to **connect, observe or secure or control** their services in a consistent way.

Service Mesh

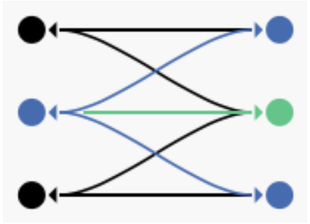
A service mesh provides a **transparent** and **language-independent** network for connecting, observing, securing and controlling the connectivity between services.

Istio

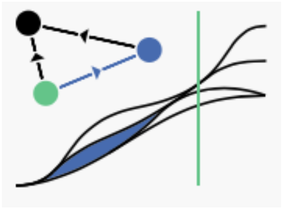
An open service mesh platform to connect, observe, secure, and control microservices.

Istio

An open service mesh platform to connect, observe, secure, and control microservices.

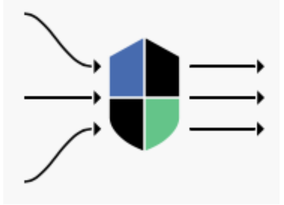


Connect: Traffic Control, Discovery, Load Balancing, Resiliency



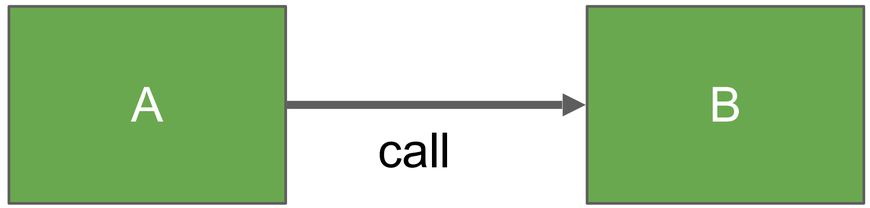
Observe: Metrics, Logging, Tracing

Secure: Encryption (TLS), Authentication, and Authorization of service-to-service communication



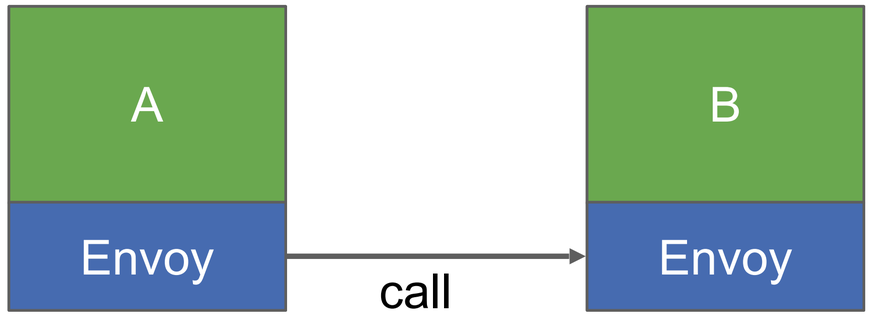
Control: Policy Enforcement

How does it work?



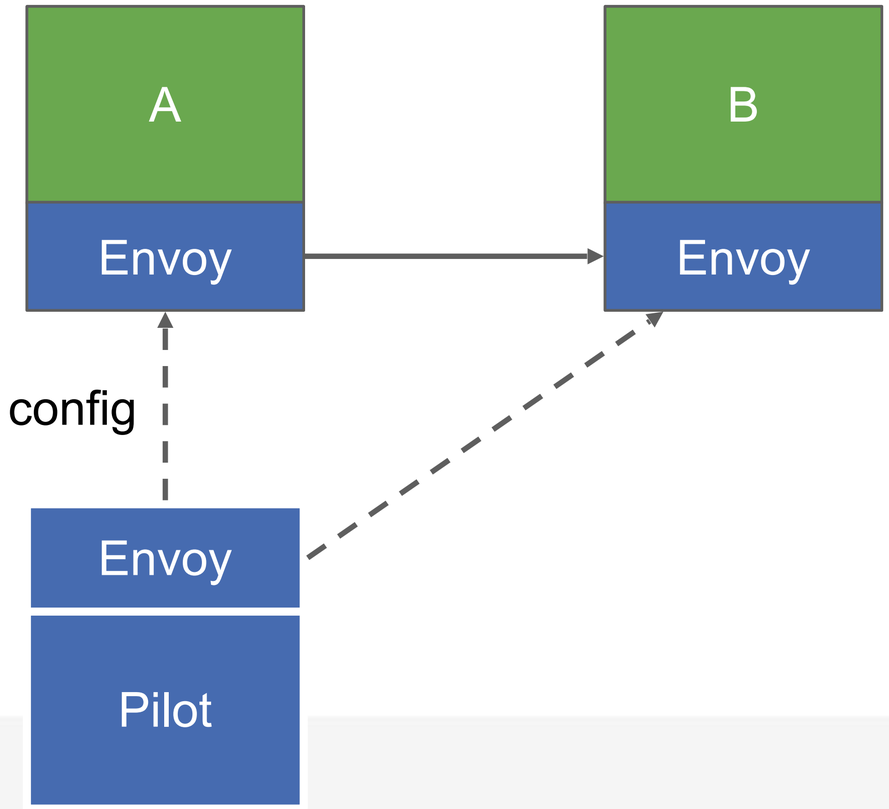
How does it work?

1. Deploy a proxy (Envoy) beside your application (“sidecar deployment”)



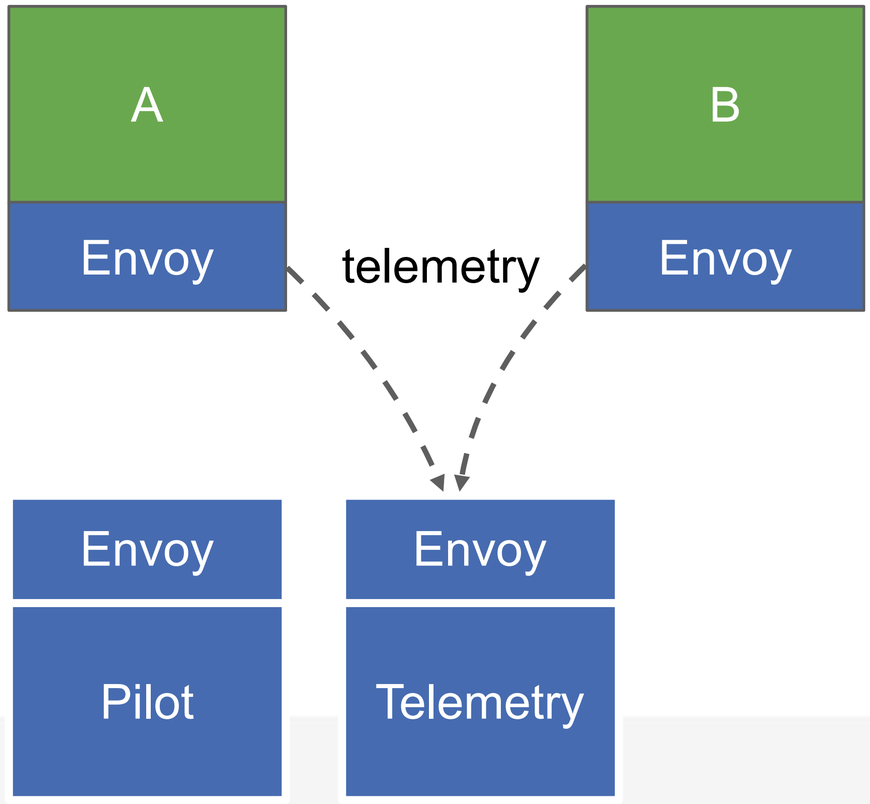
How does it work?

2. Deploy Pilot to configure the sidecars



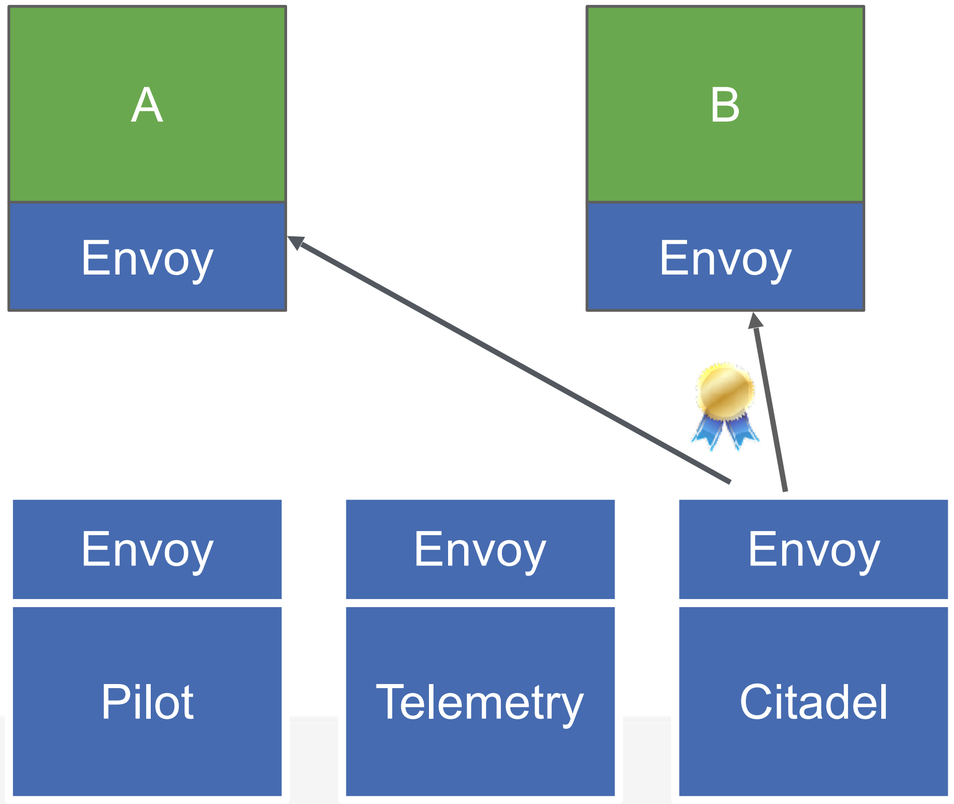
How does it work?

3. Deploy Telemetry to get telemetry



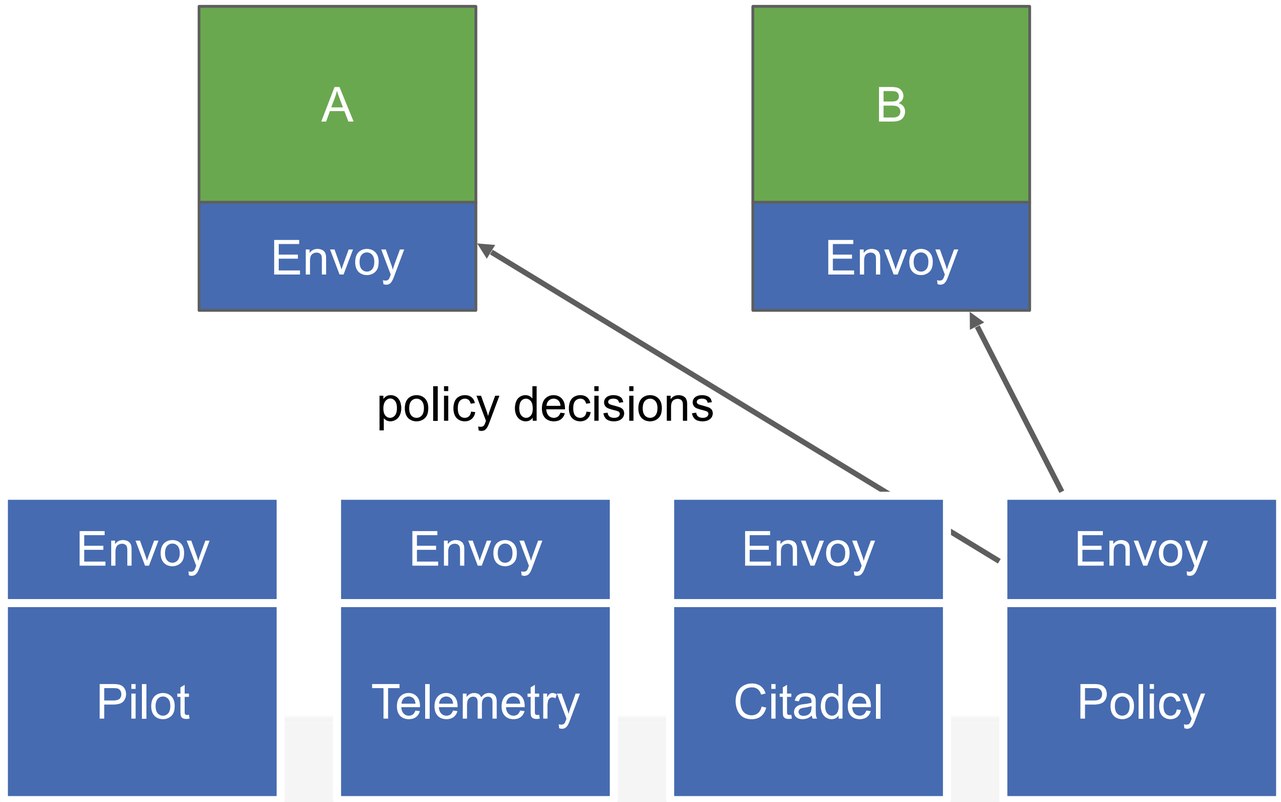
How does it work?

4. Deploy Citadel to assign identities and enable secure communication



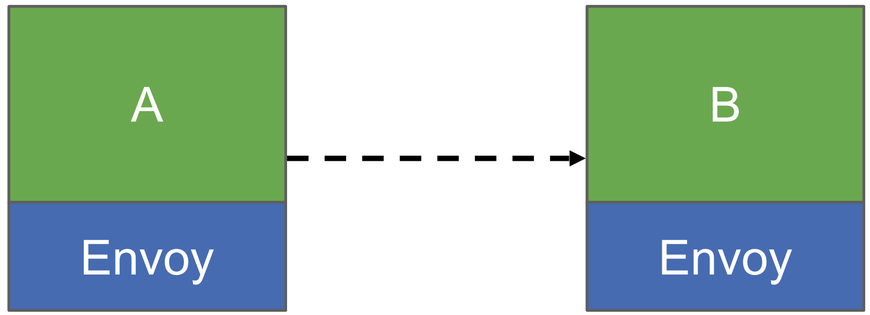
How does it work?

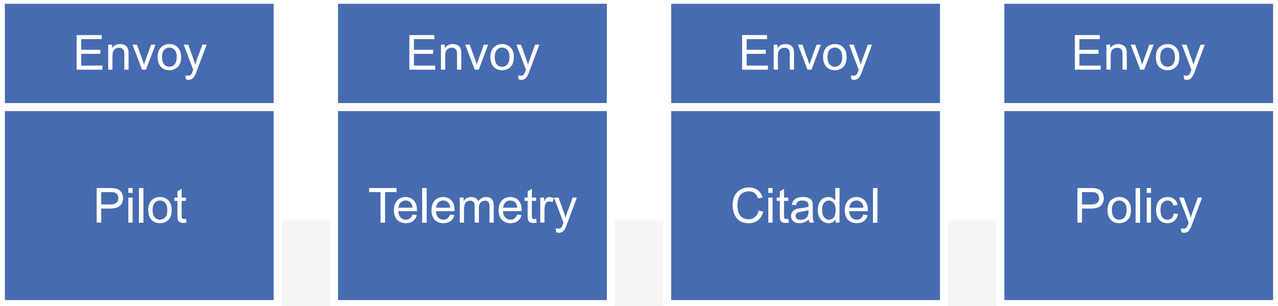
5. Deploy Policy to enforce policies



How does it work?

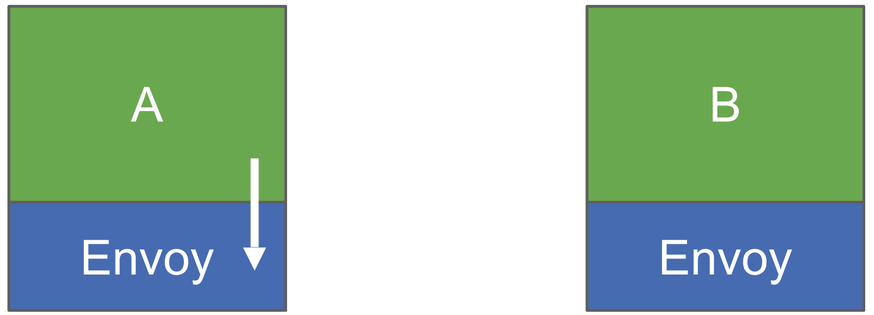
A calls B

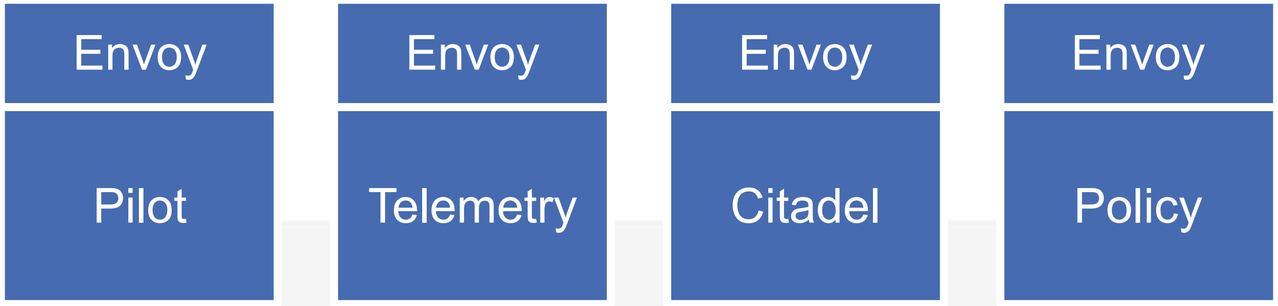




How does it work?

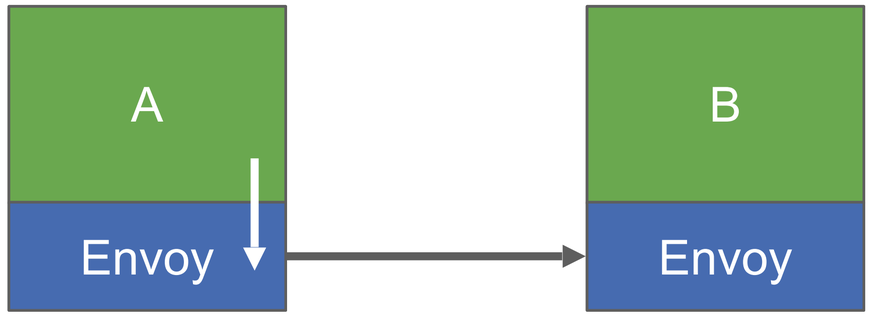
A’s sidecar intercepts the call

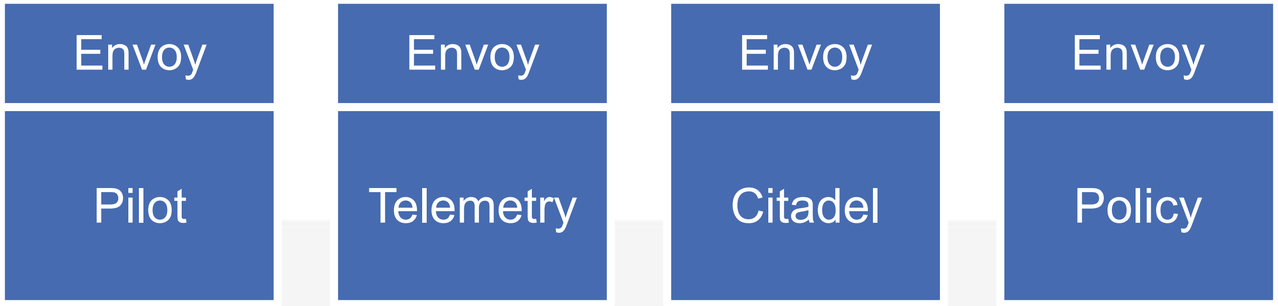




How does it work?

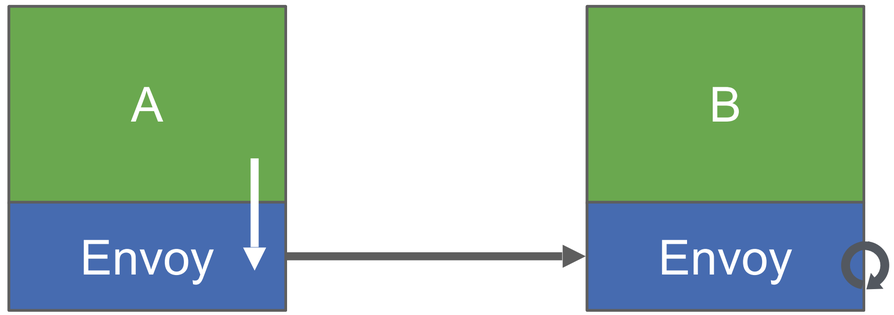
A’s sidecar selects a destination

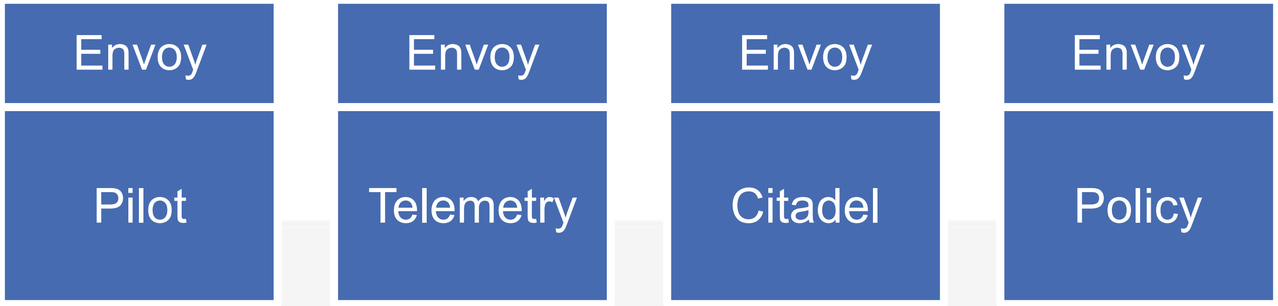




How does it work?

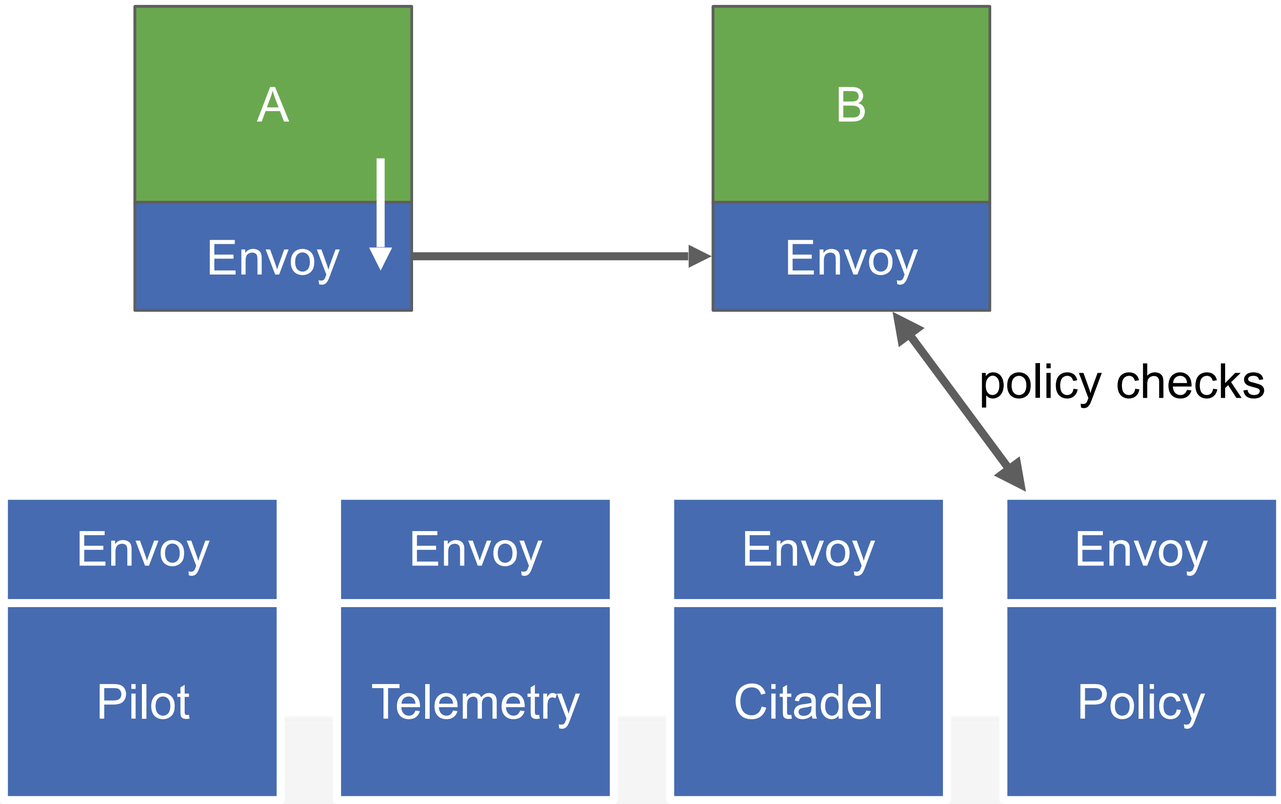
B’s sidecar performs policy checks with local cache





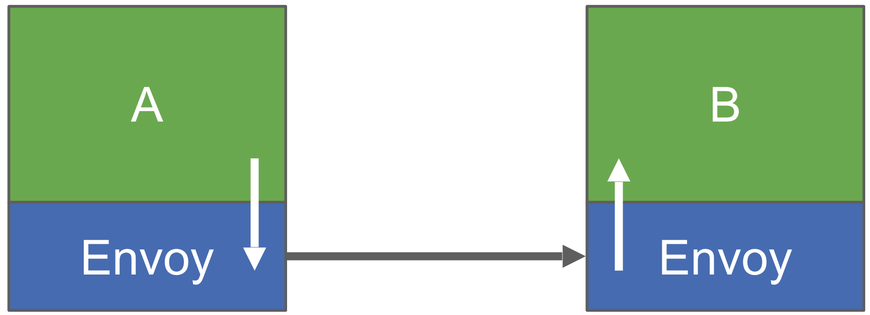
How does it work?

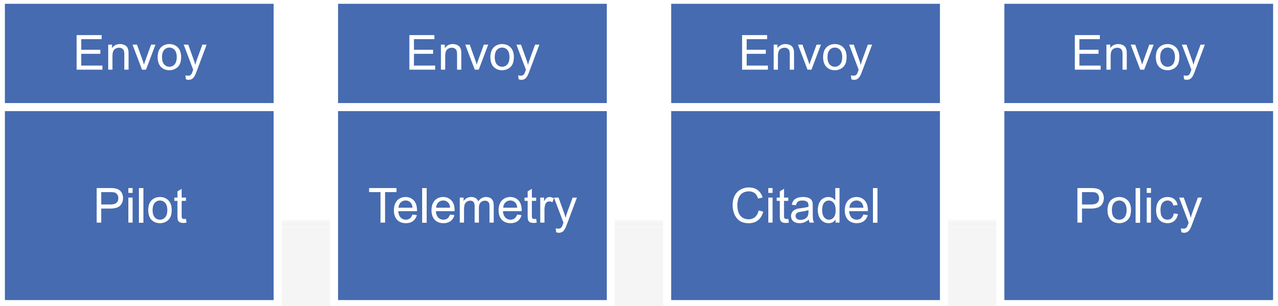
B’s sidecar performs policy checks with Policy component



How does it work?

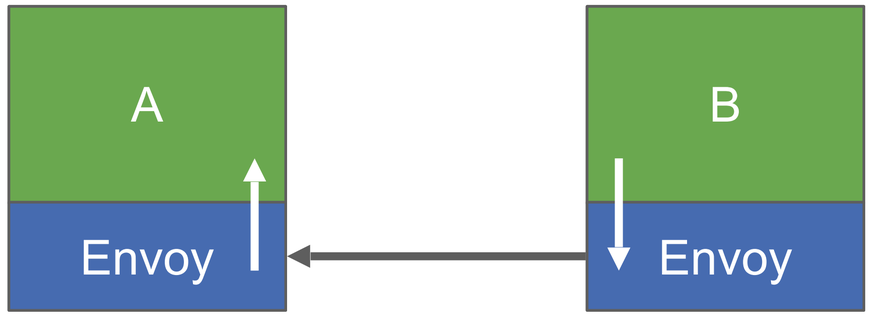
B’s sidecar forwards the call to B

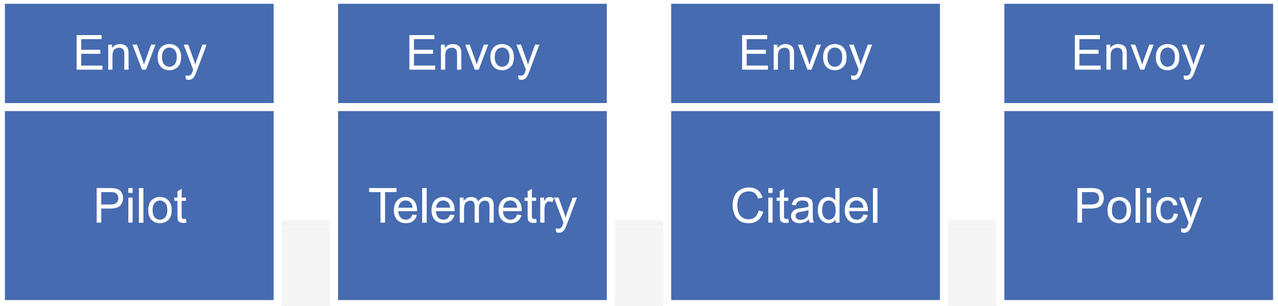




How does it work?

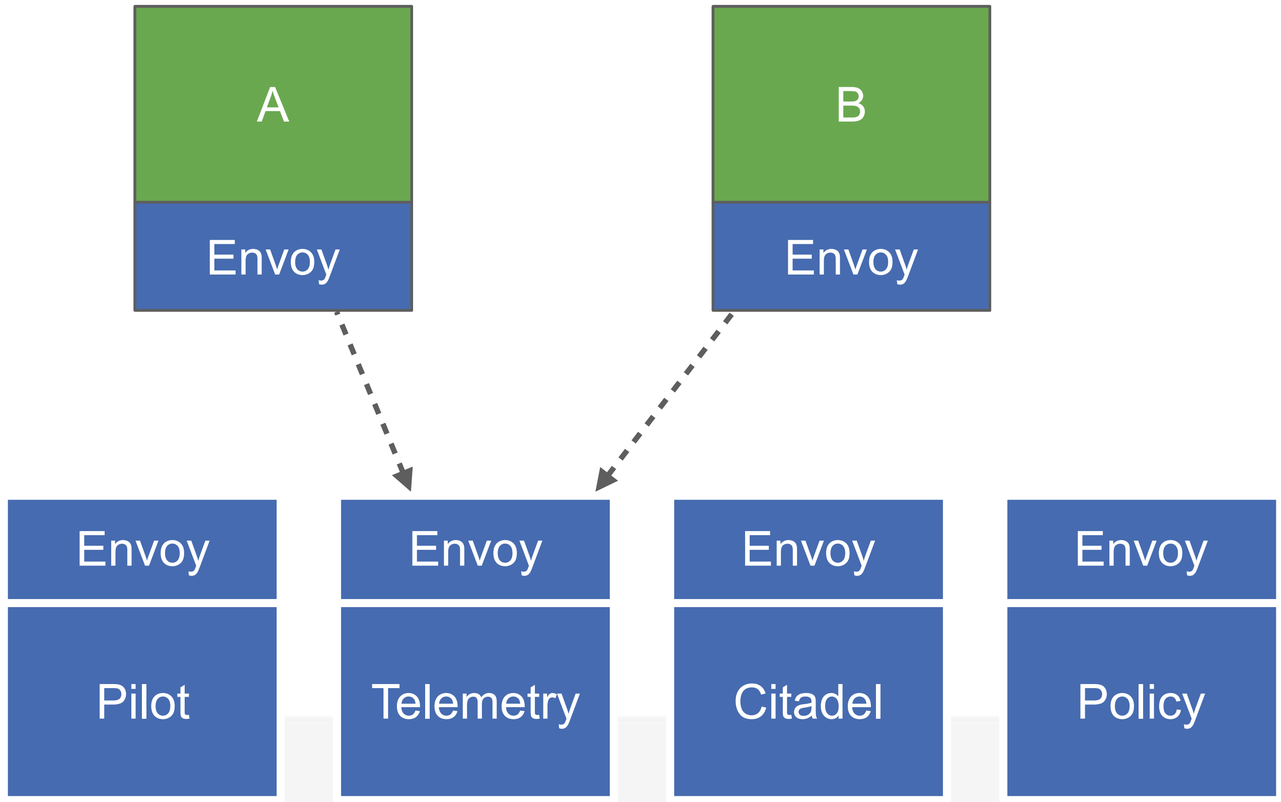
B’s response is sent back



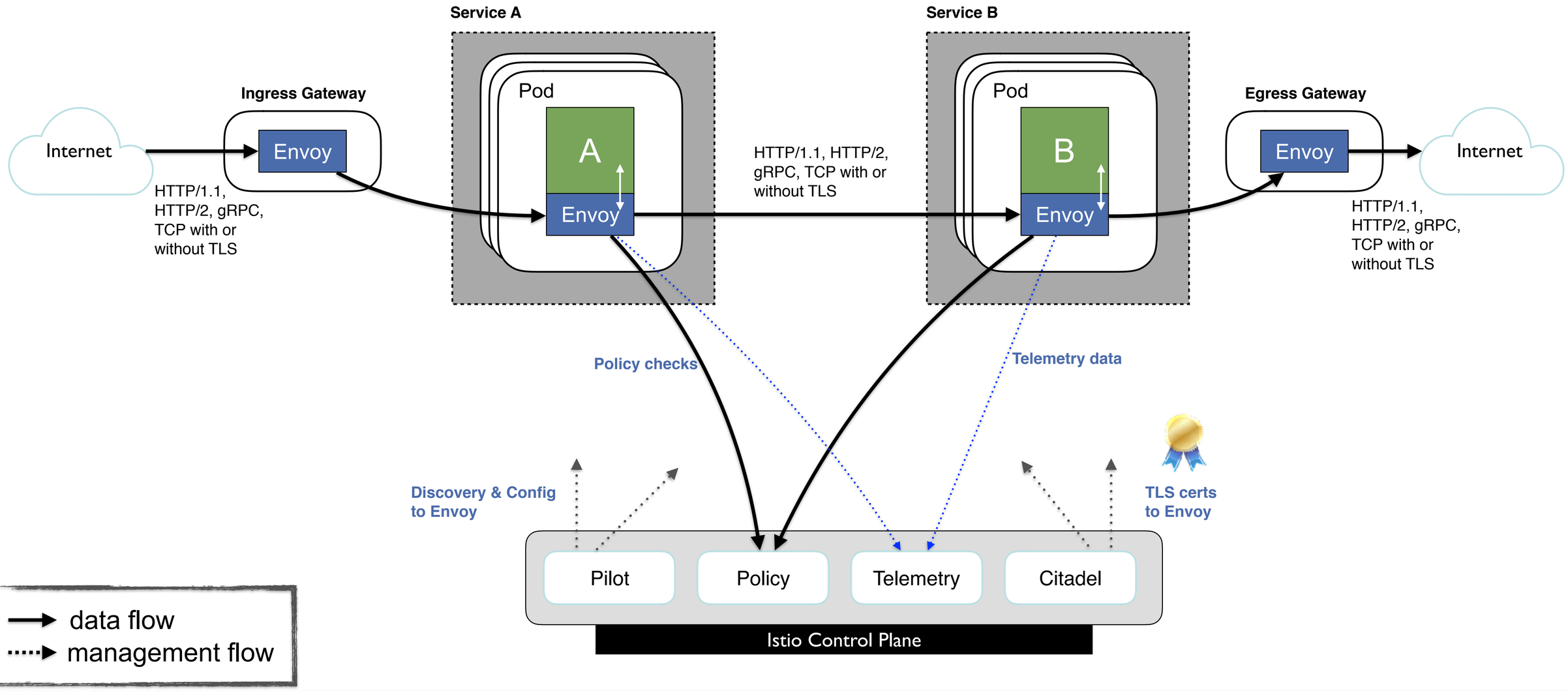


How does it work?

Both sidecars report telemetry data



Wire it up together

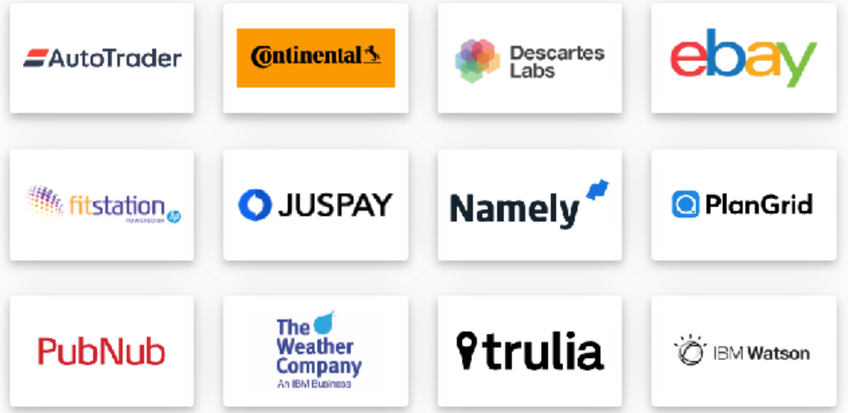


Where we are

Istio 1.0

- After ~2 years of work
- ~200 developers
- IBM, Google, VMWare, Cisco, Red Hat, Tigera, others…
- Many adapters
- Many customers

Istio Partners



Istio in Action

Istio 1.0 highlights

- Flexible and highly configurable setup
- Minimal Istio
- Revamped traffic management model reaches Beta
- Multicluster support reaches Beta
- Incremental mTLS support
- Out of process Mixer adapter support
- Optimized authorization policies implementation
- Scalability and performance improvements
- 1.0.x addresses some critical issues found in 1.0

Istio has many features

Provides a network for **services**:

- Security
- Policy Enforcement
- Resiliency
- Traffic Control
- Observability

Key Features:

- Service authn and identity
- Authorization
- Rate limiting
- Load balancing / shedding
- Retries and circuit breaking
- Fine-grained routing
- Metrics and logs generation
- Request tracing
- Fault injection

The Weather Company

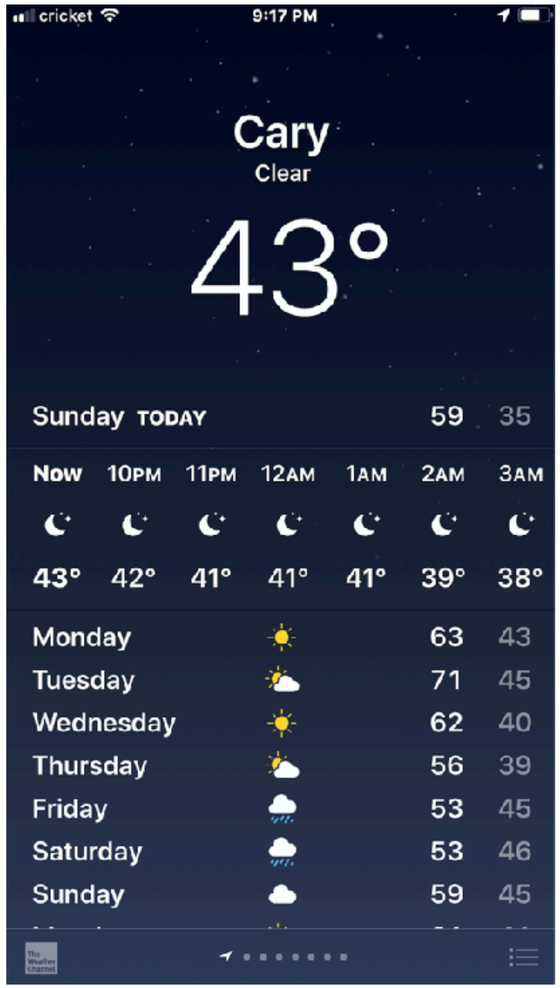
The Weather Company
An IBM Business

Why Istio?

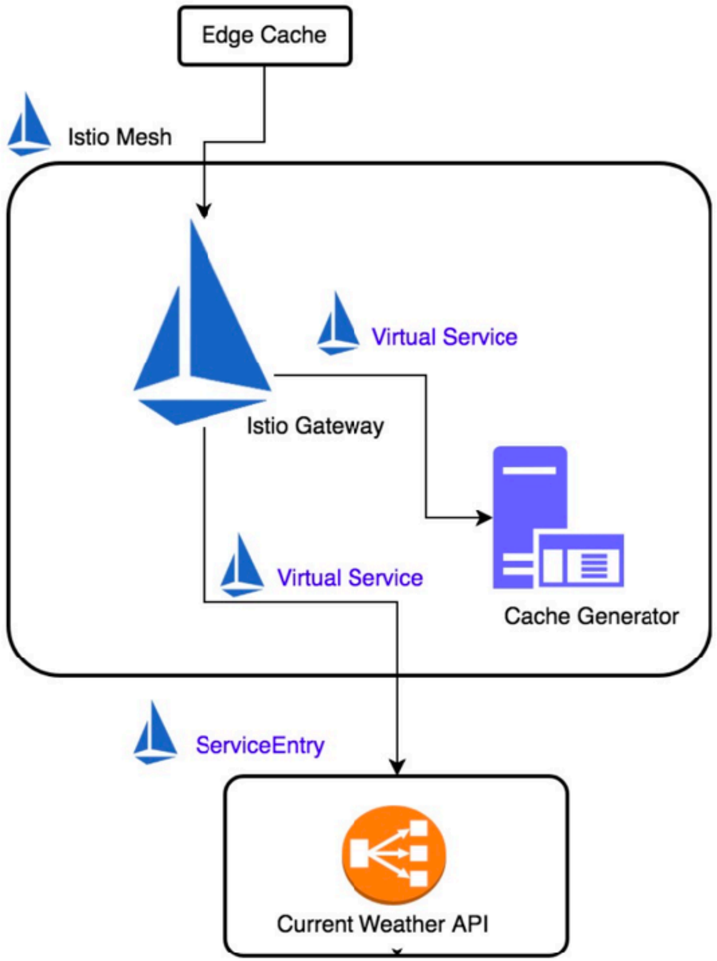
- Single platform for many tools
- Significantly less code to write
- Enterprise support and testing
- Cloud agnostic
- Cloud Native

Journey

- Using Istio for months
- api.weather.com not in production yet
- Some smaller APIs using Istio in production
- Actively working on 1.0 support
- Exploring traffic routing/shifting support



The Weather Company



Istio allows us to …

- Resolve issues faster
- Mitigate backend issues
- Be more transparent
- Reduce use of custom code
- Make route changes at will

Nick Nellis talk: https://www.youtube.com/watch?v=0fKi3NeCsSE

Service Entries

```
apiVersion:
networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
 name: current-weather
spec:
 hosts:
 - current-weather.internal-
weather.com
 ports:
 - number: 80
   protocol: http
   name: http
```

- 40 service entries
- Dynamically updateable
- Can use mTLS
- Gateways
- Virtual Services

Nick Nellis talk: https://www.youtube.com/watch?v=0fKi3NeCsSE

Routing - Cache Intercept

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
 name: current-weather-cache
spec:
 gateways:
 - istio-gateway
 hosts:
 - api.weather.com
 http:
 - route:
   - destination:
       name: cache-generator
  - match:
   - uri:
       exact: /v3/wx/observations/current
```

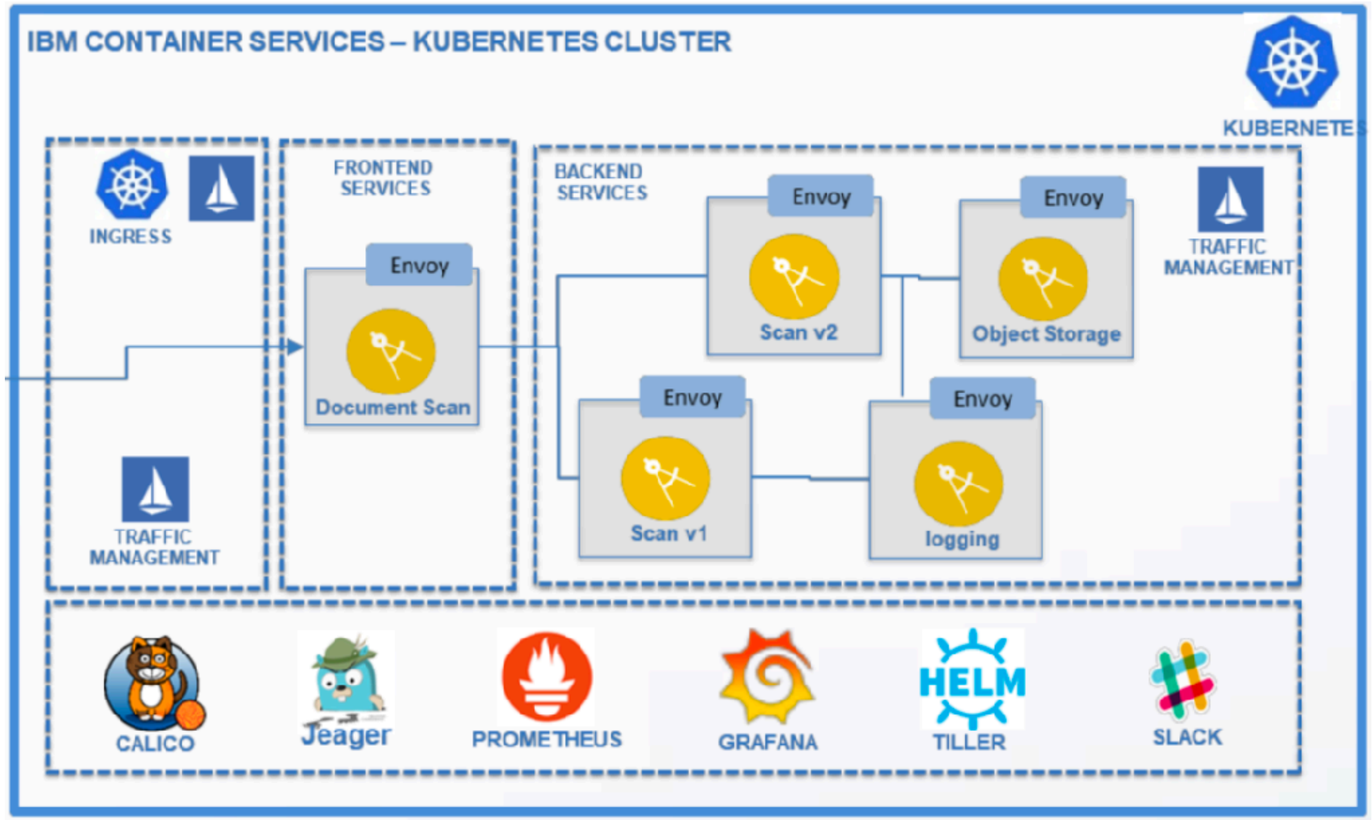
- Virtual service is new route rule
- Dynamically updateable
- Gateway selector

Nick Nellis talk: https://www.youtube.com/watch?v=0fKi3NeCsSE

Large Airline

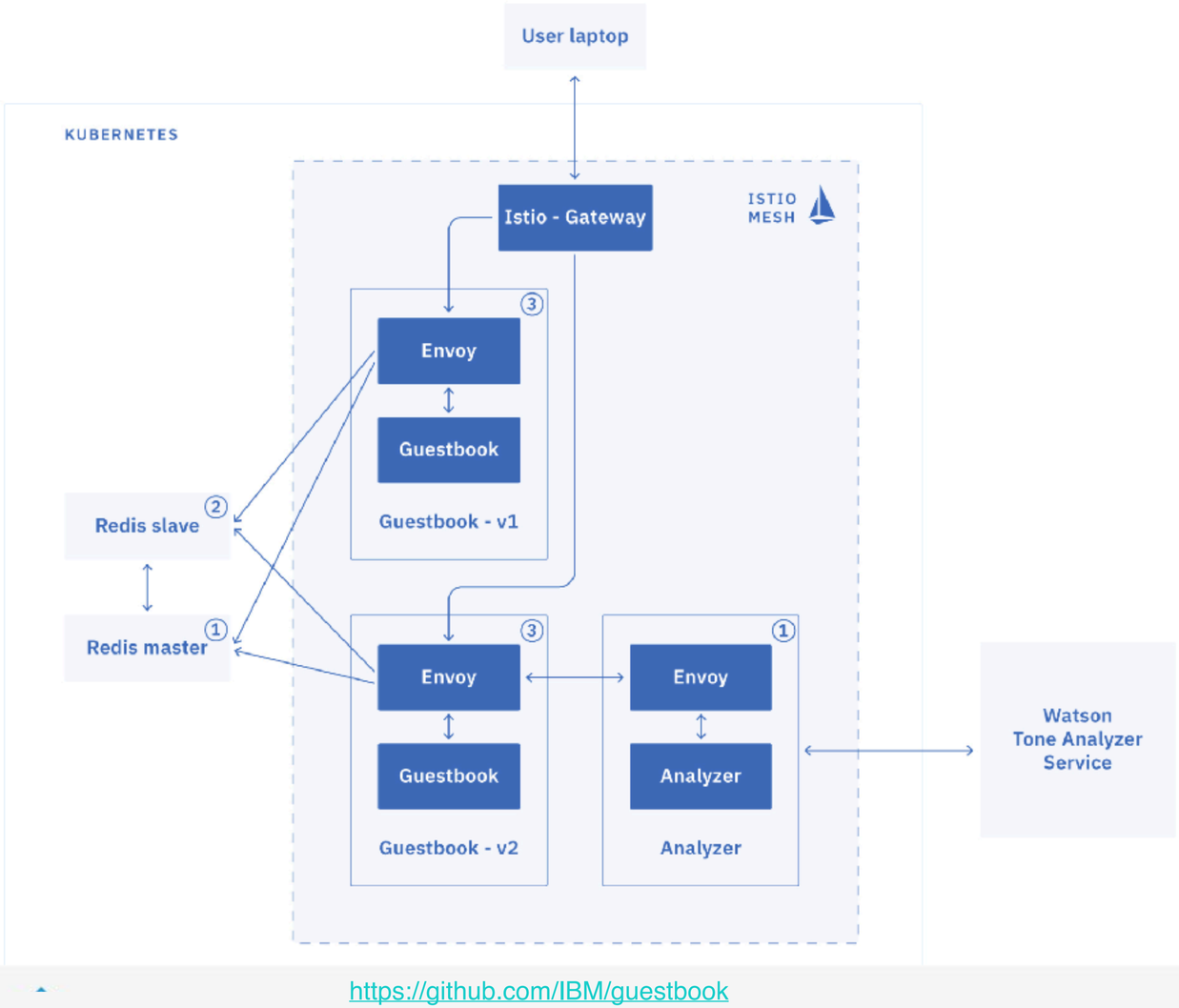
Leverage Traffic Management:

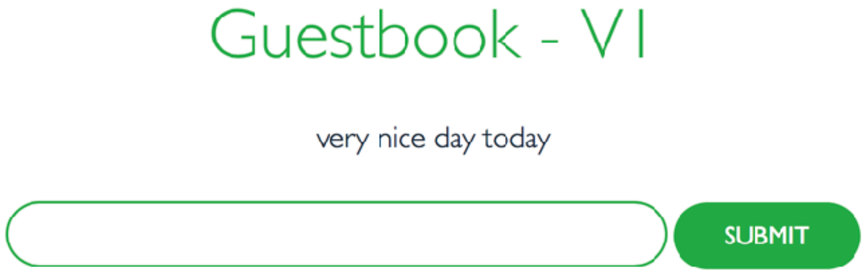
- Request Routing
- Circuit Breaking
- Route Rule
- Traffic mirroring
- Egress-Ingress Policies

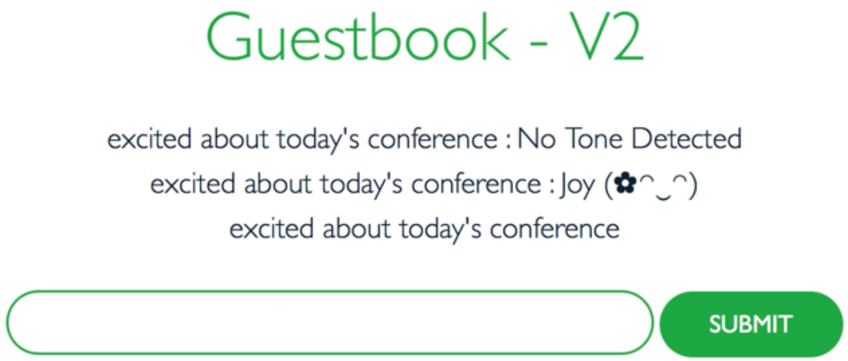


Let us see it live!

The Guestbook Application







https://github.com/IBM/guestbook

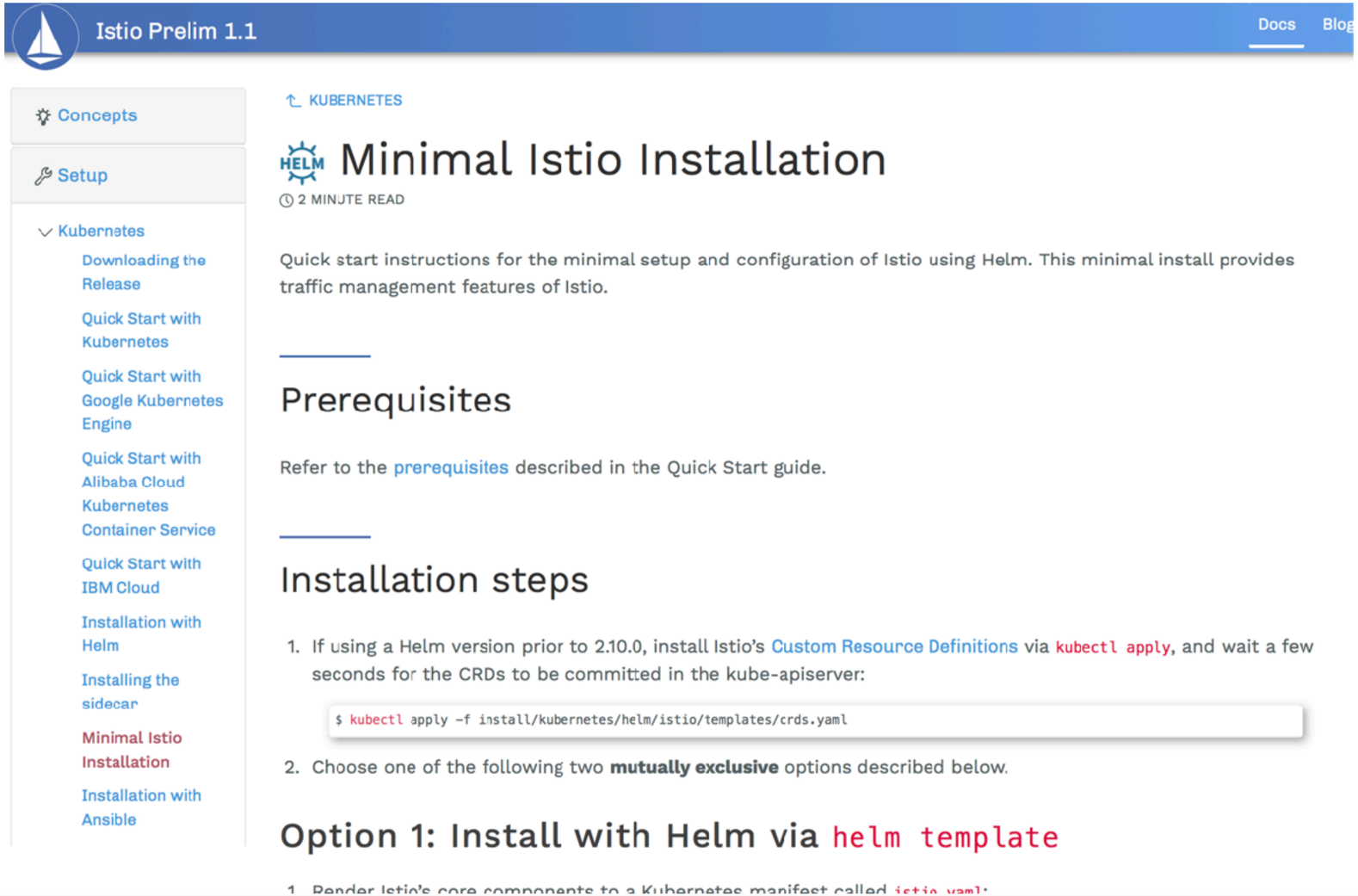
What You Have Seen

- Service entry
- Ingress gateway
- Traffic management
- Distributed tracing
- Istio Mesh Metrics

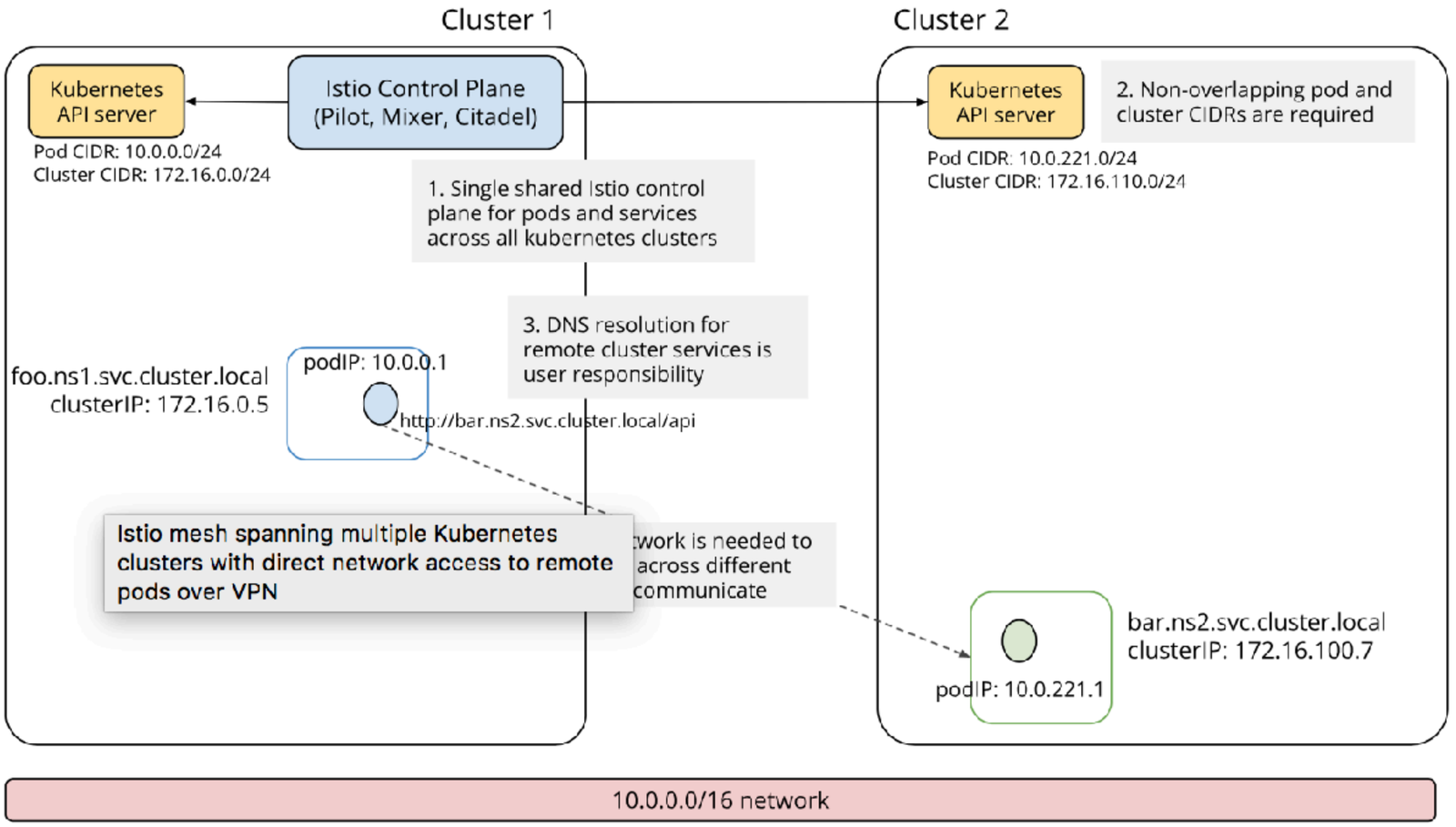
Istio 1.1 and beyond

- Make Istio easier to use and adopt
- Incrementally adopt Istio
- Ability to run mixer per node or sidecar
- Support non routable L3 network among pods for multiclusters and multiple environments
- Continued Performance and Scalability Improvement
- Integration with API management
- Various bug fixes!

Minimal Istio

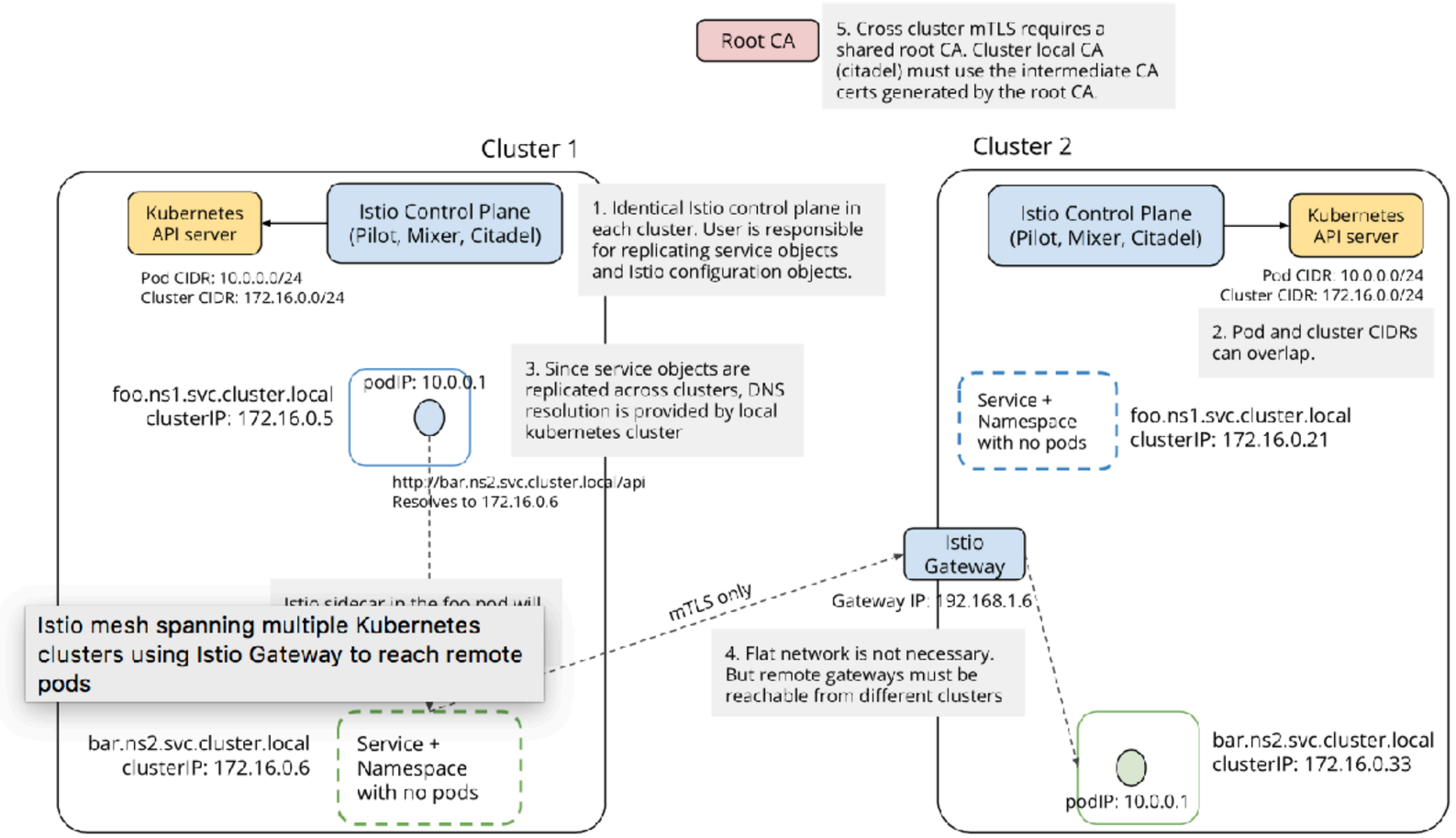


Istio Multicluster Single Control Plane



Istio mesh spanning multiple Kubernetes clusters with direct network access to remote pods over VPN

Federated Istio Multicluster via Gateway



Istio mesh spanning multiple Kubernetes clusters using Istio Gateway to reach remote pods


Try Istio today! Thanks!
Contact us:
istio-user@googlegroups.com
istio-dev@googlegroups.com
@IstioMesh
More information:
istio.io
istio.io/about/community/
github.com/istio