Lessons Learned on Multi-tenancy Controls in Istio

Alex Ly & Will McKinley

Abstract

Abstract:
As Istio adoption becomes mainstream within your organization, new challenges surrounding multi-tenancy and security across multi-cluster will naturally start to grow

- Which group owns what process/workflow?
- Which cluster(s) does each policy affect?
- How to provide control to some groups, while blocking access to others w.r.t. the mesh?
- How does an administrator set this up in a secure fashion?
- How can we stay informed about potential policy violations?
- How can this be fully automated?

In this discussion, Will and Alex will discuss these topics in detail and review strategies and experiences tackling these challenges with some of the largest deployments of service mesh in the world.

Audience / Benefits to Ecosystem

Audience:
The audience for this session would be existing users of Istio looking for best practices/tools/architectures/methodologies to implement multi-tenancy capabilities without compromising security and manageability.

Benefits to the Ecosystem:
While initial technology discovery and MVP of Istio in a single cluster can be relatively simple, management of these lower-level resources at scale can present new complex challenges relative to context, multitenancy and security considerations. As usage and adoption of these common APIs grows, higher-level abstractions are needed to improve the user experience mapping more directly to intent based on business requirements and processes, and letting the system manage the configuration and orchestration of lower-level resources. An API closely aligned with organizational functions providing clear separation of concerns can provide tremendous benefits when working with the complexity of Istio and its constructs.

Challenges

- Which group owns what process/workflow?
- Which cluster(s) does each policy affect?
- How to provide control to some groups, while blocking access to others w.r.t. the mesh?
- How does an administrator set this up in a secure fashion?
- How can we stay informed about potential policy violations?
- How can this be fully automated?

Alex Ly

Field Engineer - North America @ Solo

alex.ly@solo.io

https://www.linkedin.com/in/alexbrucely/

Will McKinley

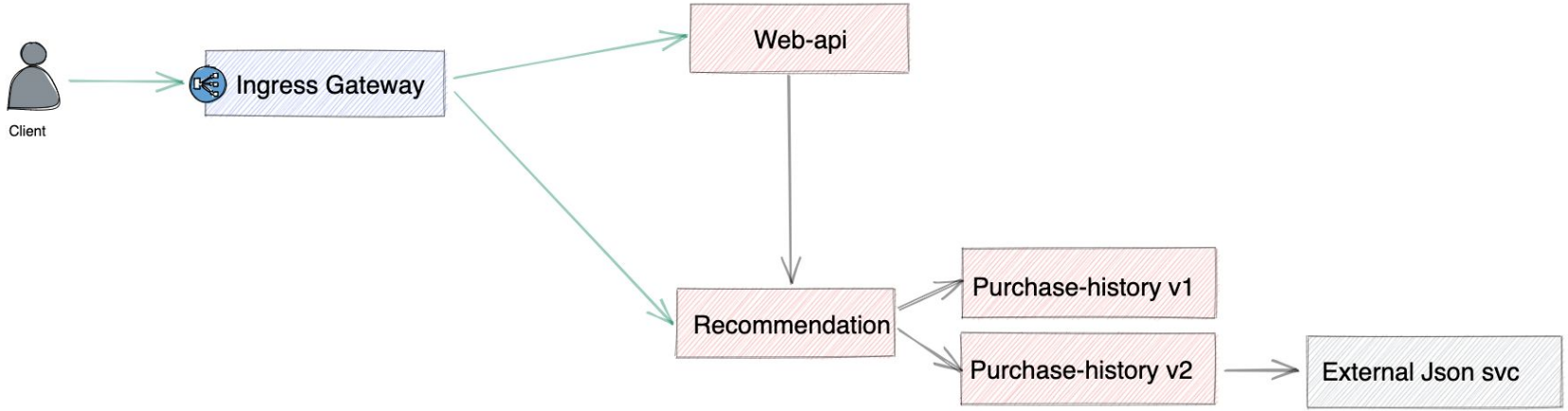
Field Engineer - North America @ Solo

willowmck1

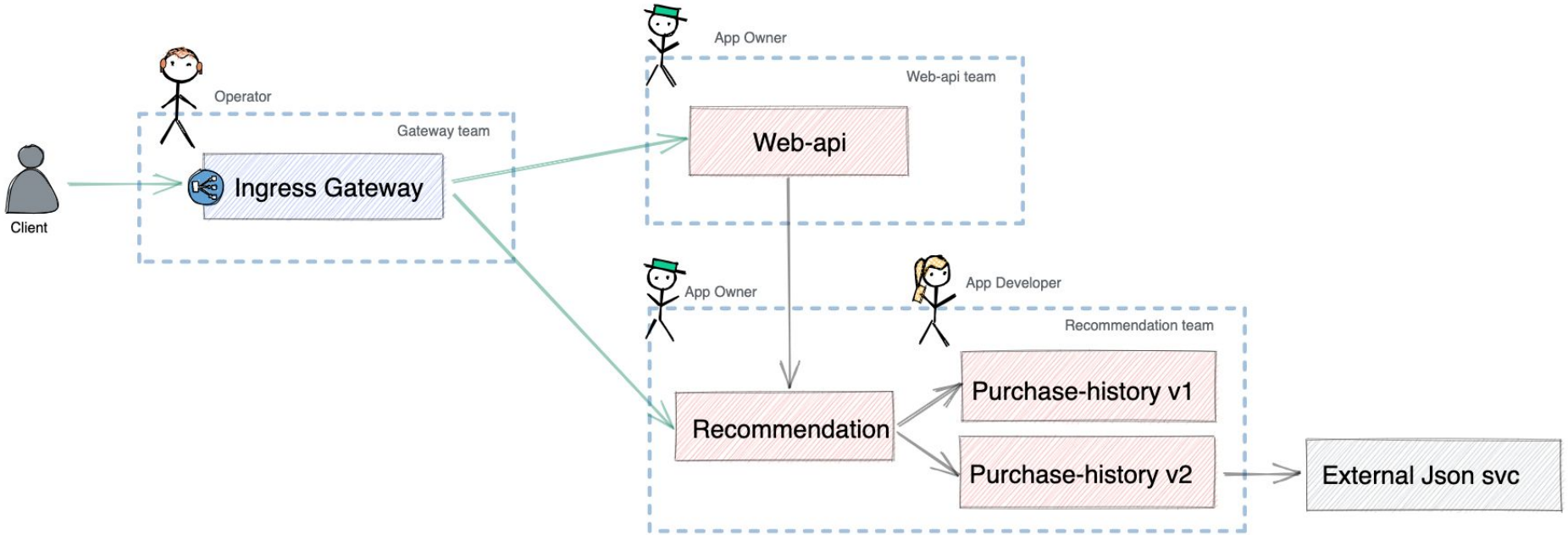
will.mckinley@solo.io

https://www.linkedin.com/in/will-mckinley-470913/

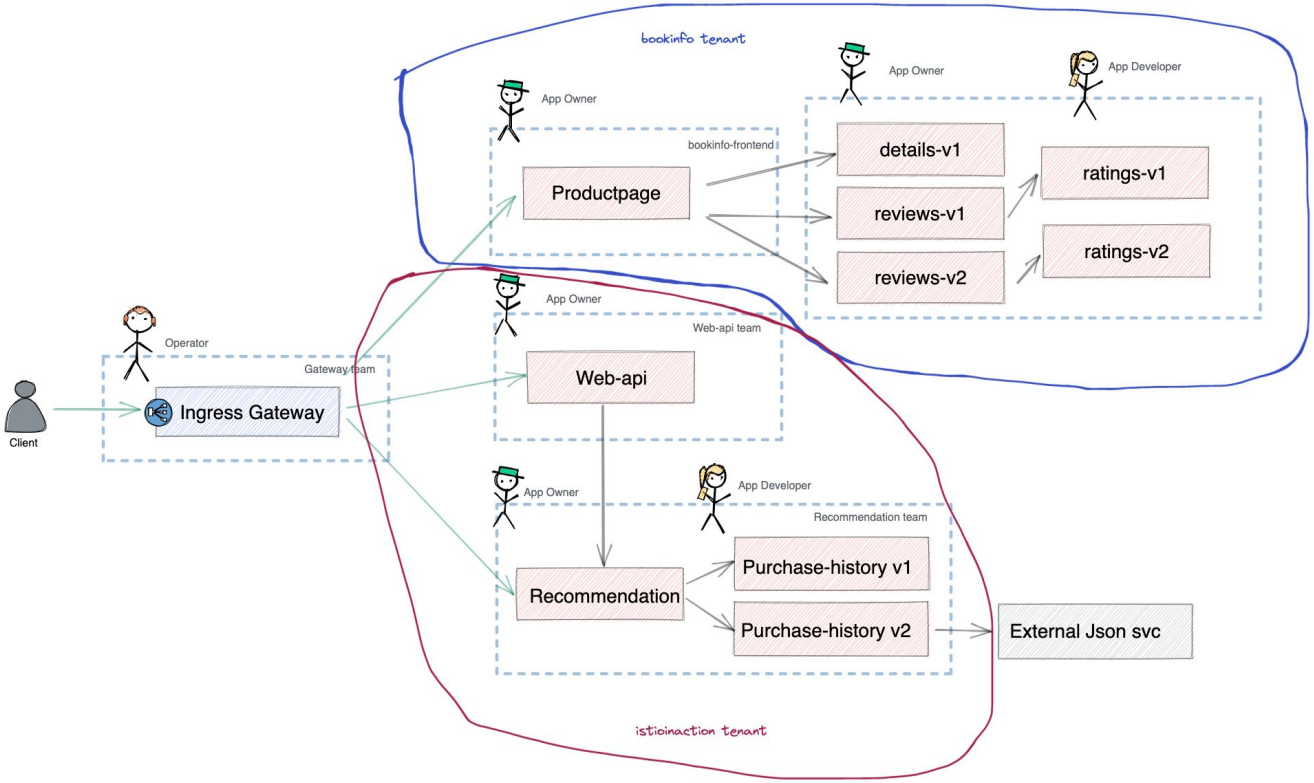
Example App


Client
Ingress Gateway
Web-api
Recommendation
Purchase-history v1
Purchase-history v2
External Json svc

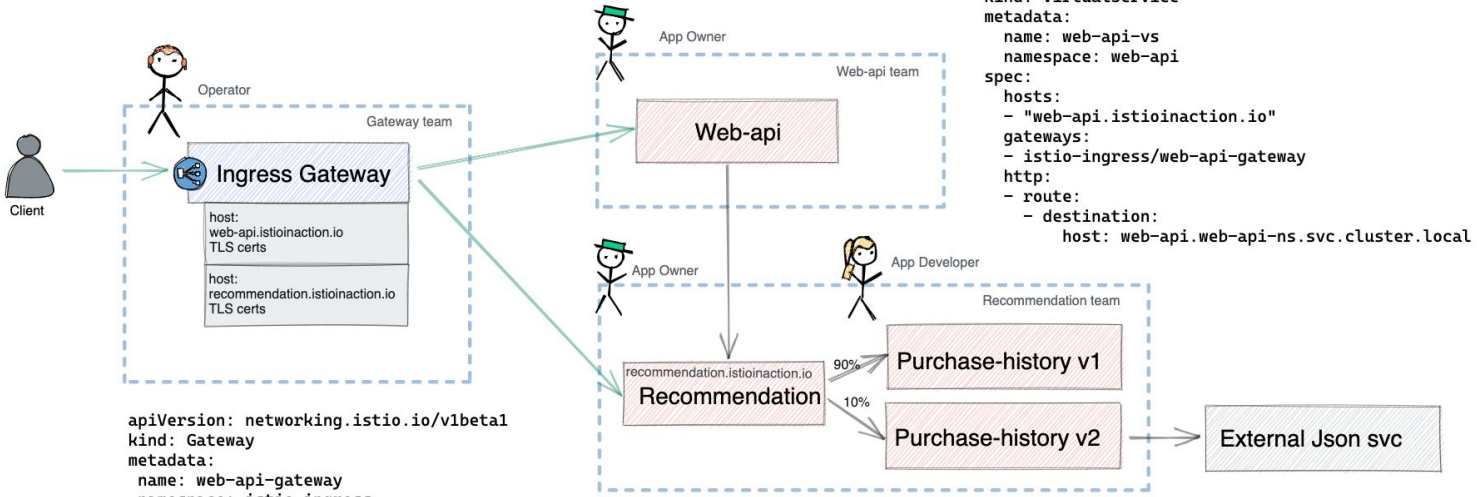
Example App with Group Ownership



Example Multi-tenant Mesh



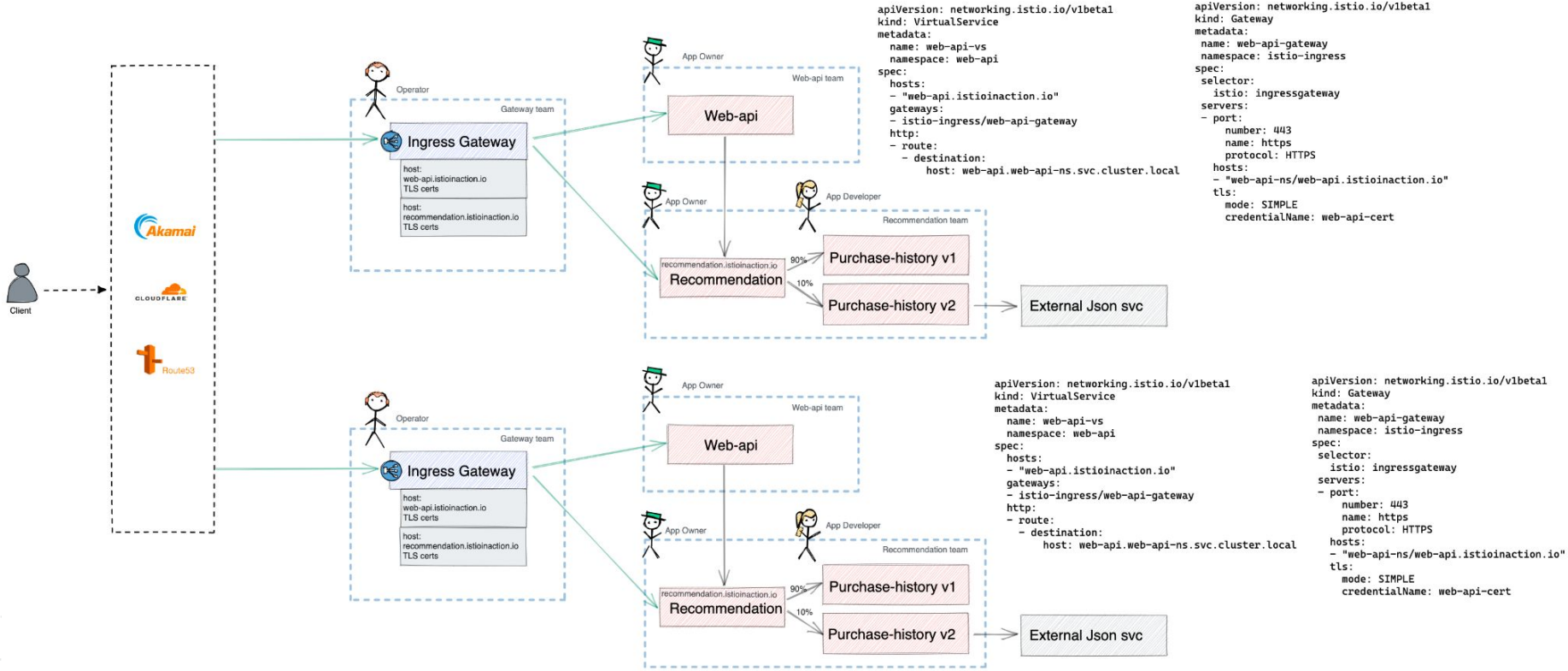
With Istio




```
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
 name: web-api-gateway
 namespace: istio-ingress
spec:
 selector:
   istio: ingressgateway
 servers:
 - port:
     number: 443
     name: https
     protocol: HTTPS
   hosts:
   - "web-api-ns/web-api.istioinaction.io"
   tls:
     mode: SIMPLE
     credentialName: web-api-cert
```

```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: web-api-vs
  namespace: web-api
spec:
  hosts:
  - "web-api.istioinaction.io"
  gateways:
  - istio-ingress/web-api-gateway
  http:
  - route:
    - destination:
        host: web-api.web-api-ns.svc.cluster.local
```


With Istio Across Multiple Clusters



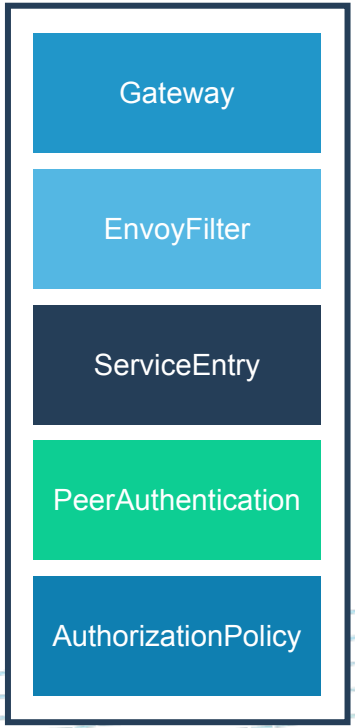
Some issues with ownership...

Example: Platform Owner(s)

Kubernetes Owner, Network Team, or Security Ops

- What service should/should not communicate?
- What calls can be cross organizational boundaries?
- Identity propagation
- Zero Trust
- Setting and Testing Resilience Policies
- High availability of solution
- Rate Limiting
- Observability

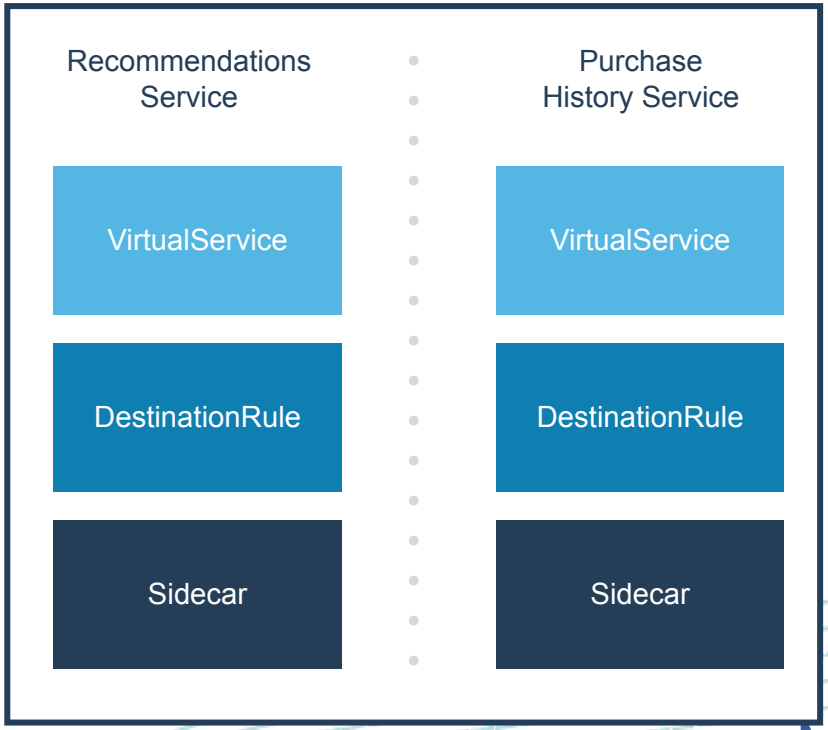
Configured with
these Istio resources



Example: Service Team (Purchase History)

- Service dependencies
- Hostname routing
- CORS
- Traffic splitting
- Rate limiting
- Fault injection
- Circuit breaking

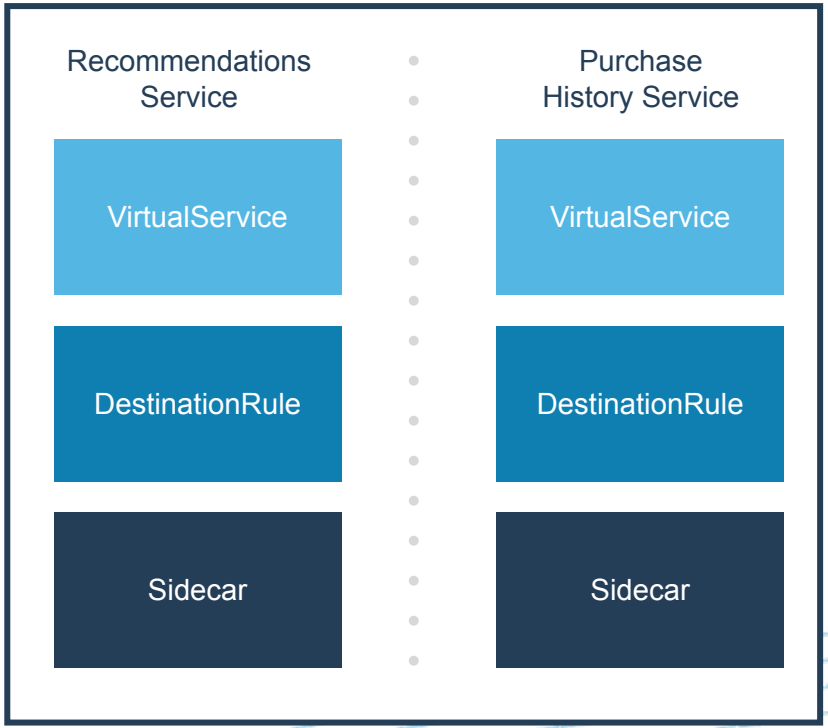
Configured with these Istio resources for each service



Example: Service Team (Recommendations)

- Timeouts
- Retries
- Retry budget
- Locality aware load balancing
- Circuit breaking

Configured with these Istio resources for each service



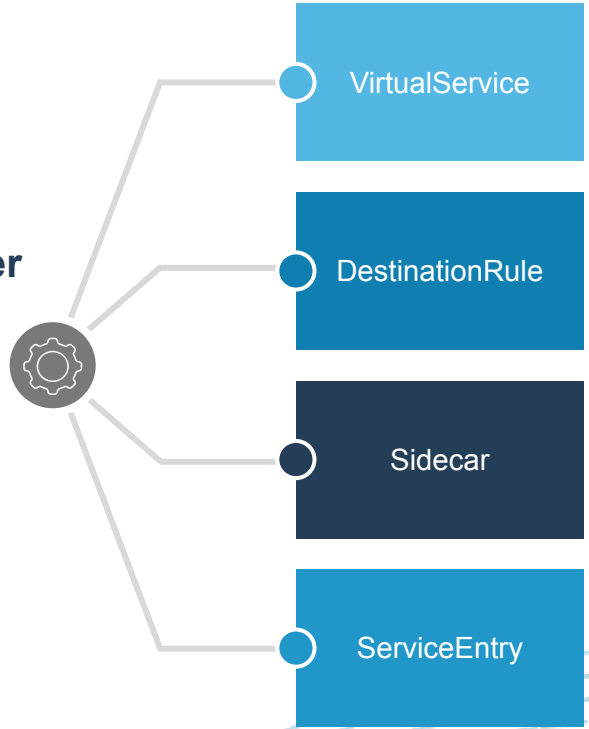
Problem: Contention...Who Owns These Resources?

Kubernetes/Platform Owner

Networking Admin

Security Admin

Developers



Examples

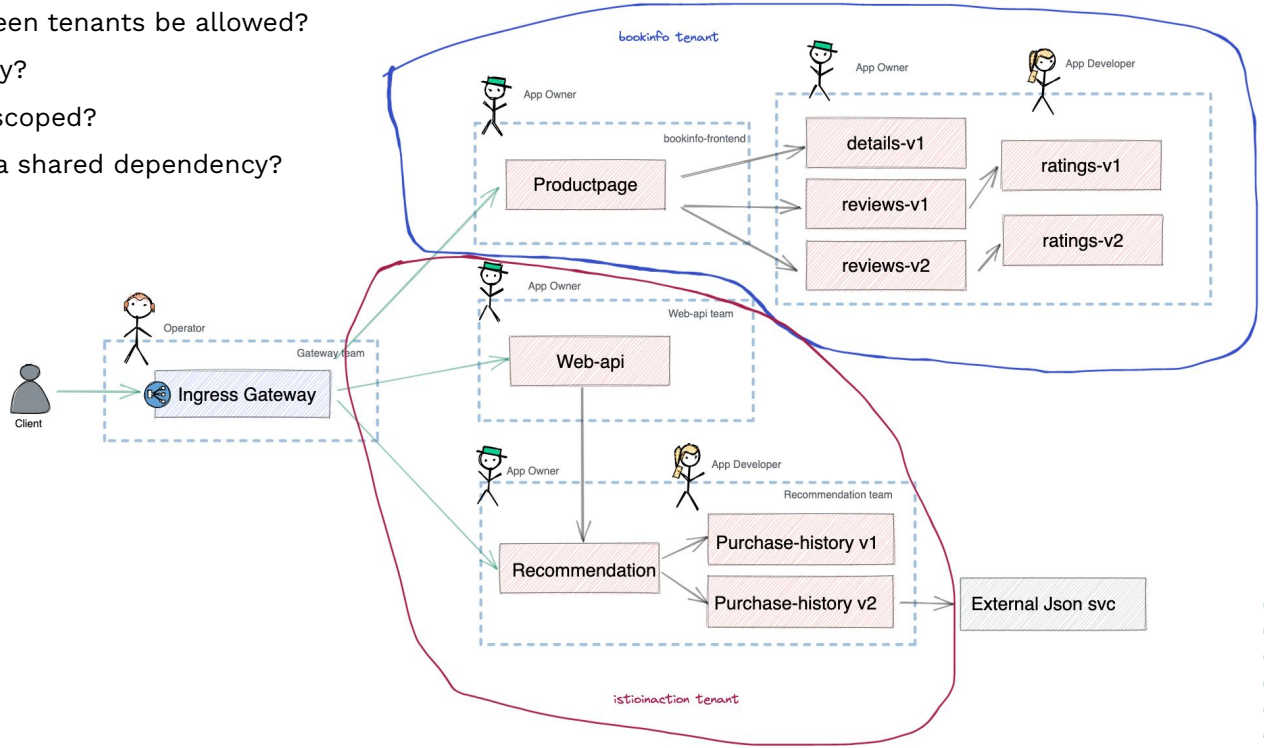
Single Cluster, Multi-tenant

Should communication between tenants be allowed?

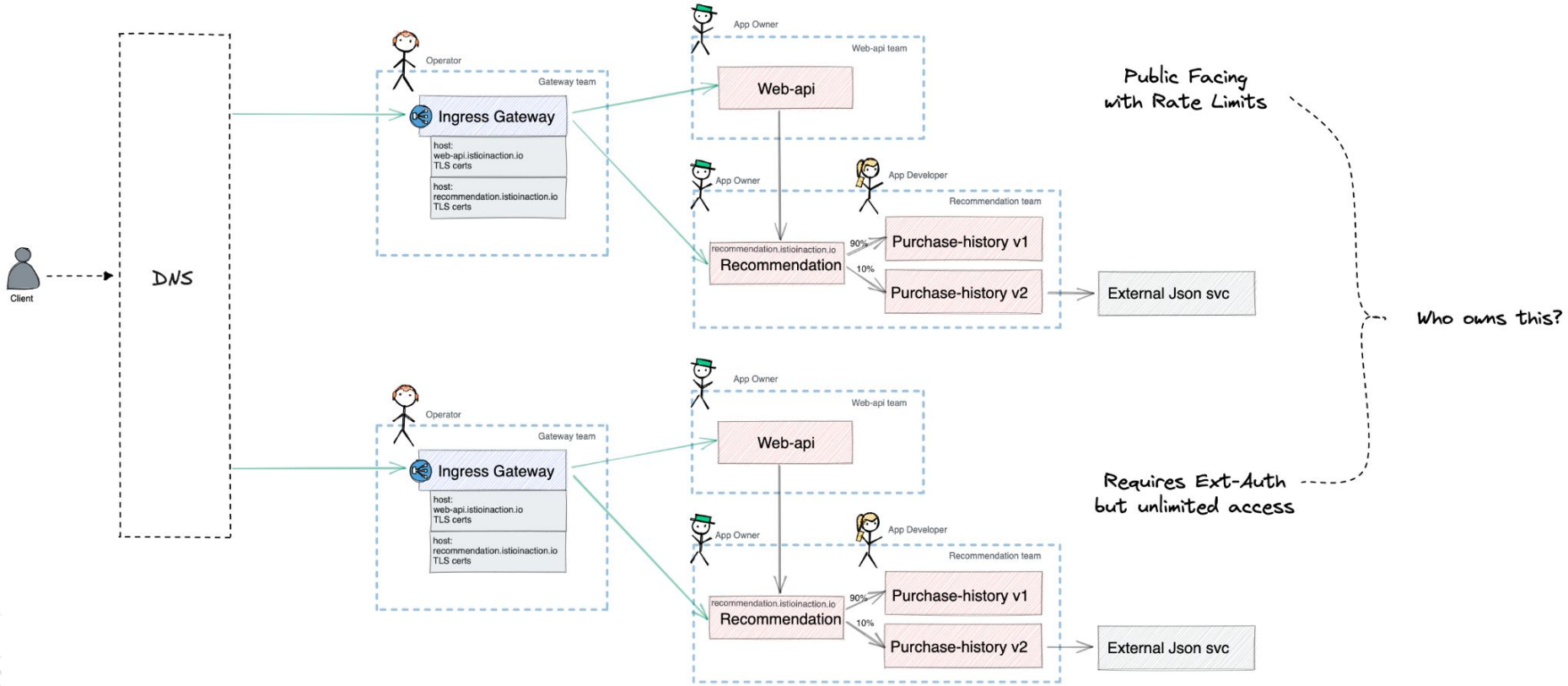
- Who defines the policy?
- How is configuration scoped?
- What if tenants have a shared dependency?

Who owns the Gateway?

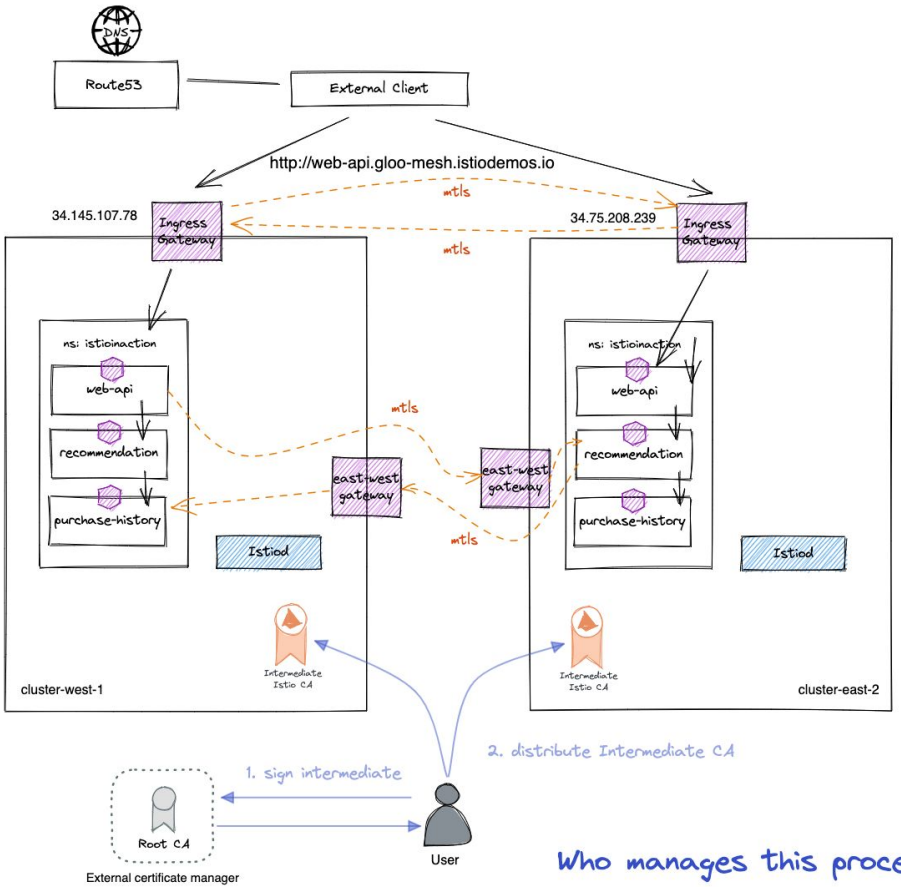
Uniform policy management?



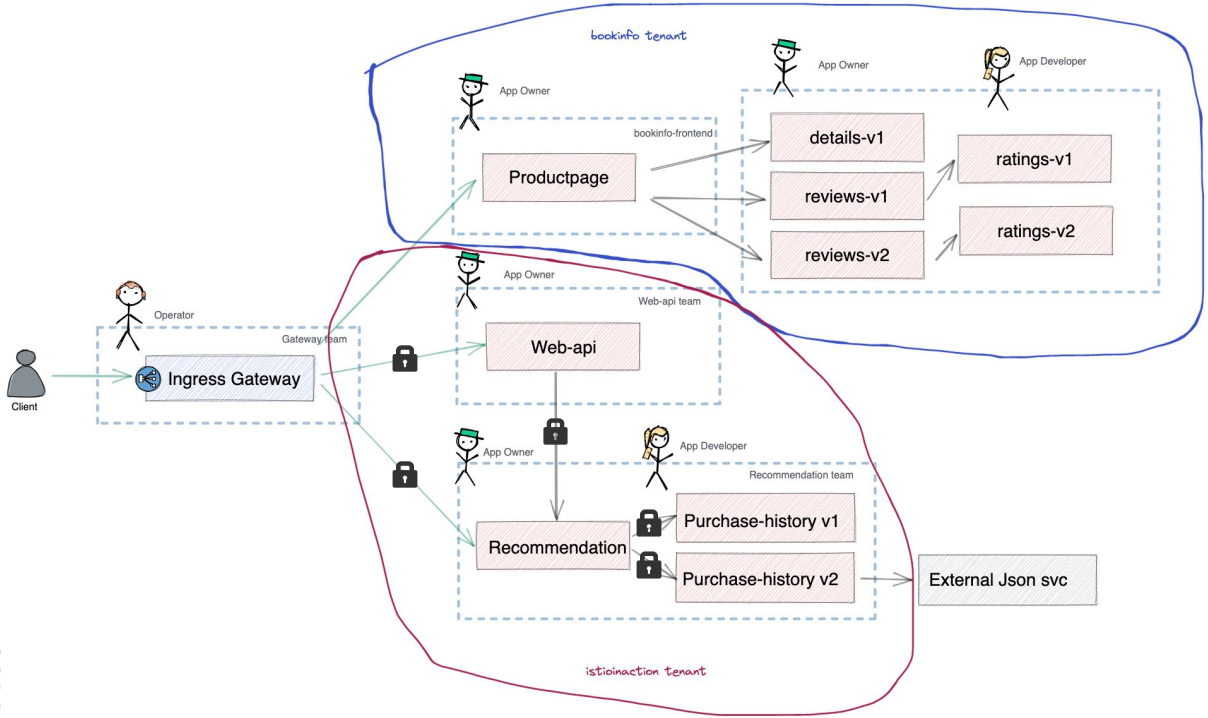
Identical App, similar-environments



Common Root CA

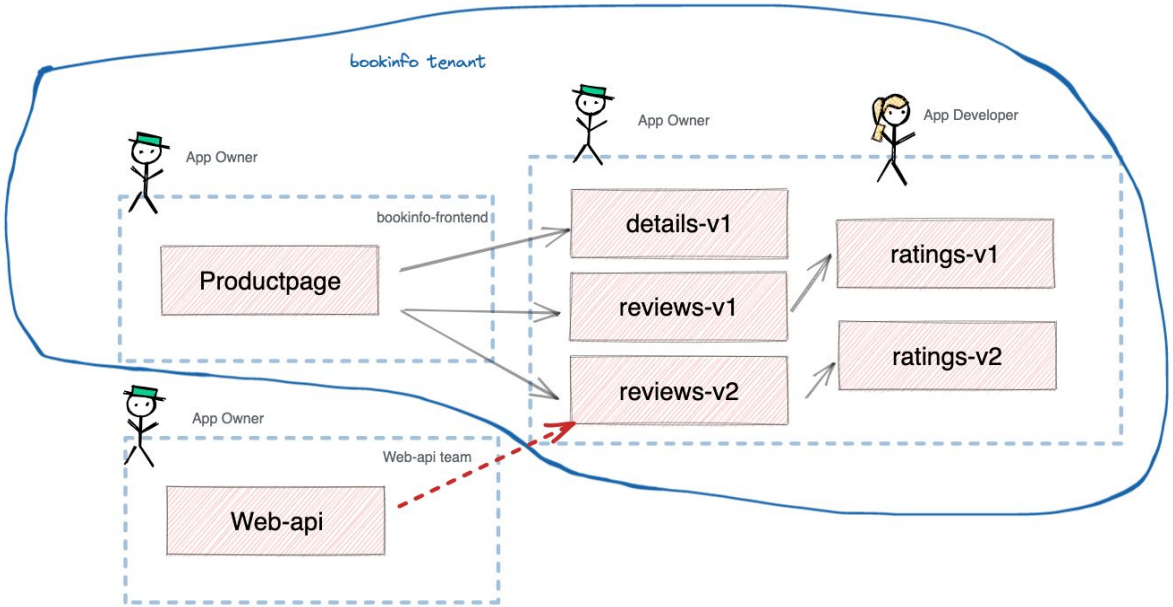


Enforcing Zero-Trust (Istio Only)



Is zero-trust the responsibility of the Platform Owner, the Operator or the App Owner?

Service Isolation



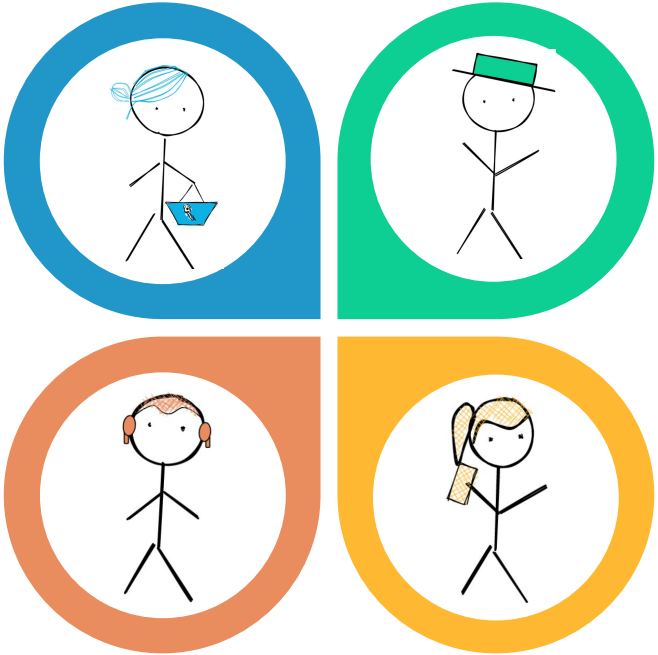
Configuration should be scoped to prevent visibility of other tenants.

Lessons Learned

Map to Enterprise Personas

Pam
Platform Admin
Owner of fleet of clusters
Sets organizational policies

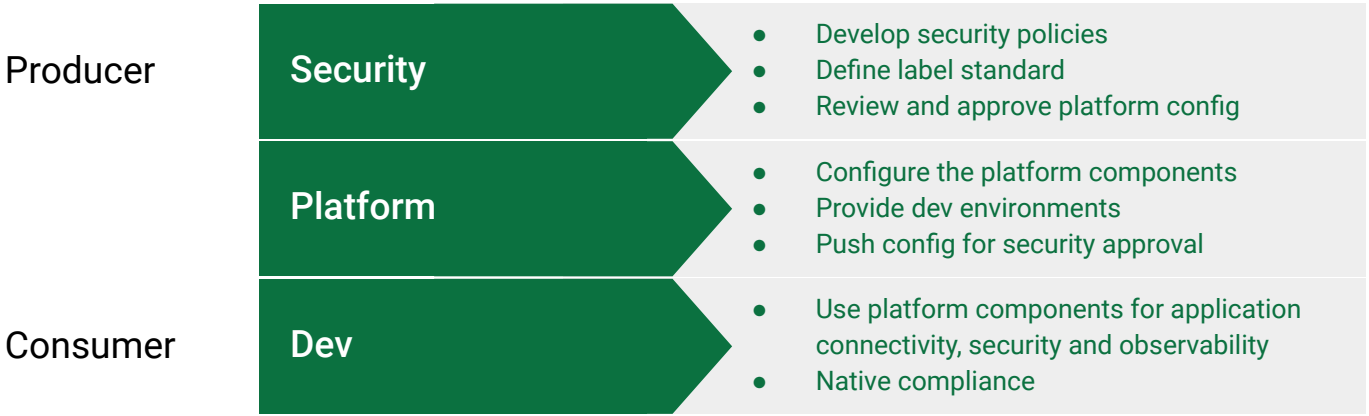
Oliver
Operator
Operate cluster and service for service owners



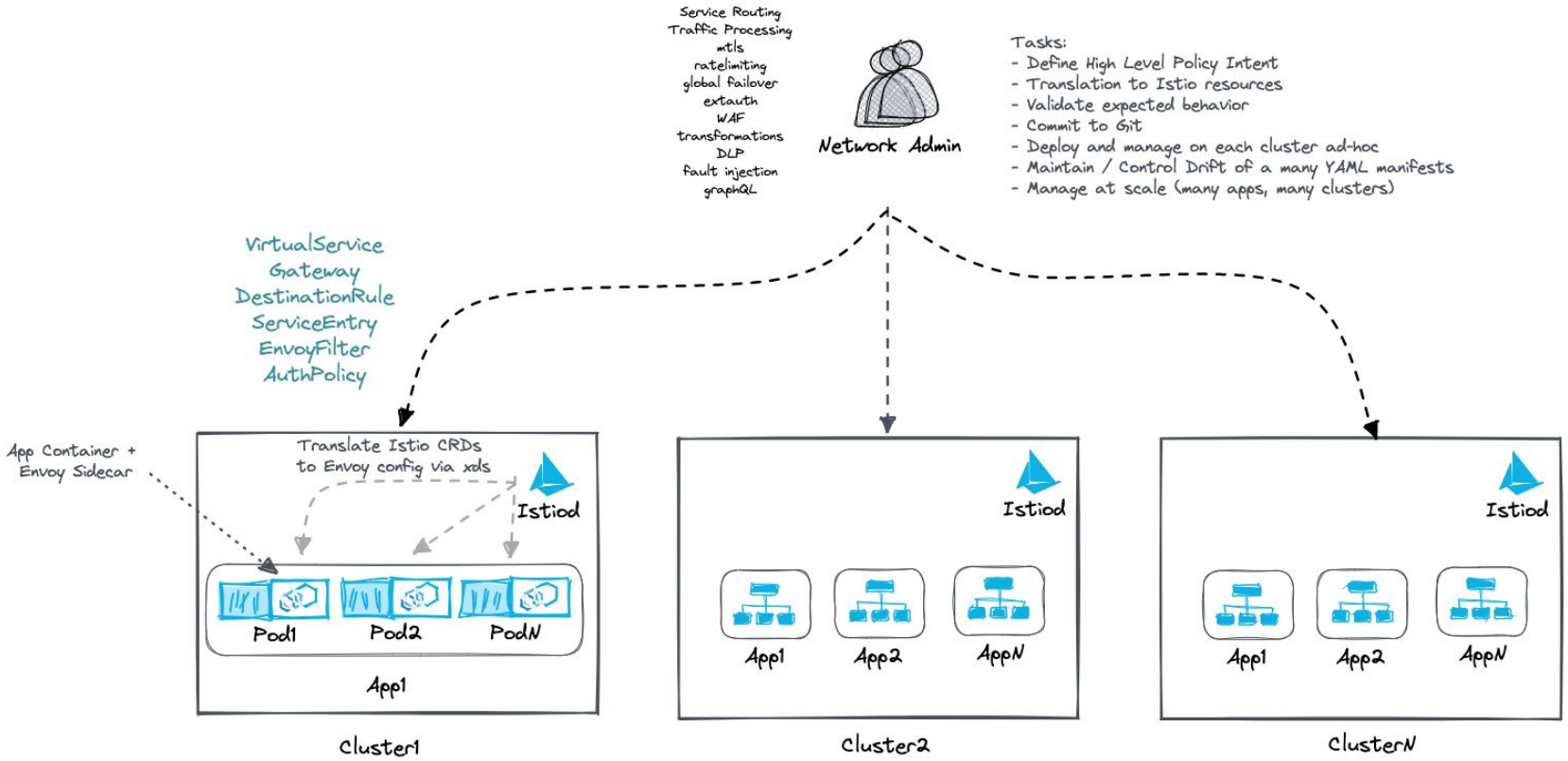
Arjay
Application Owner
Onboard developers within the team
Understands team dependencies

Alice
Application Developer
Write code and develop services

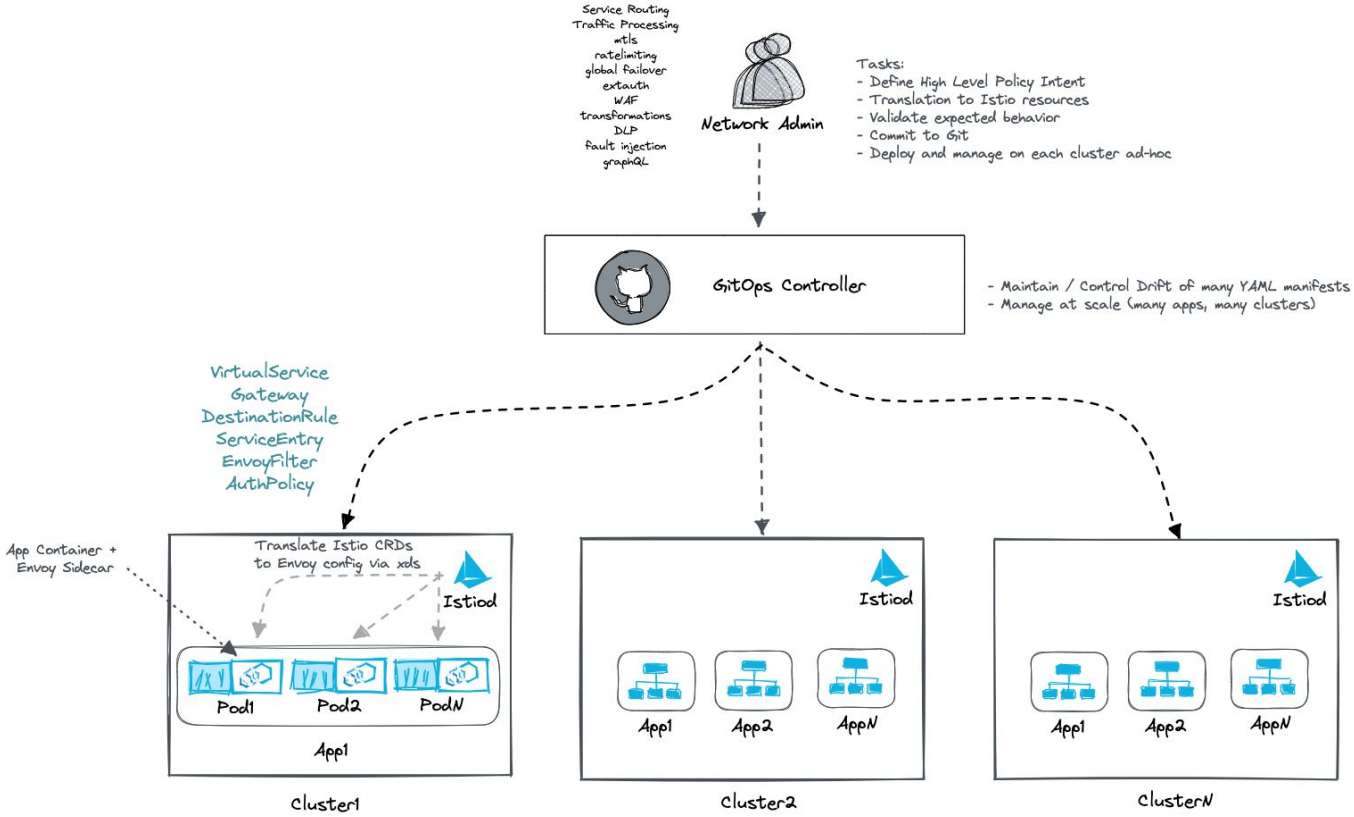
Develop an Operating Model



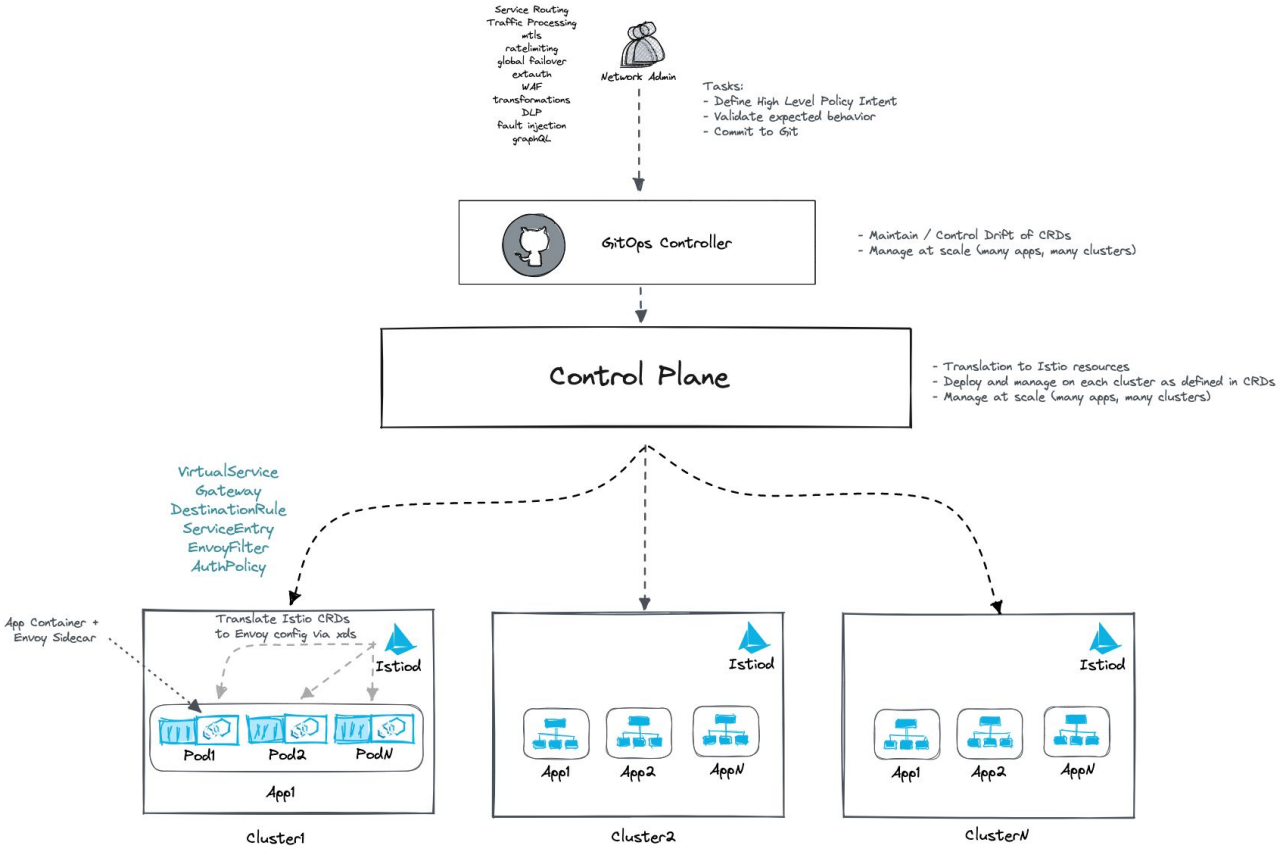
Who is the Control Plane?



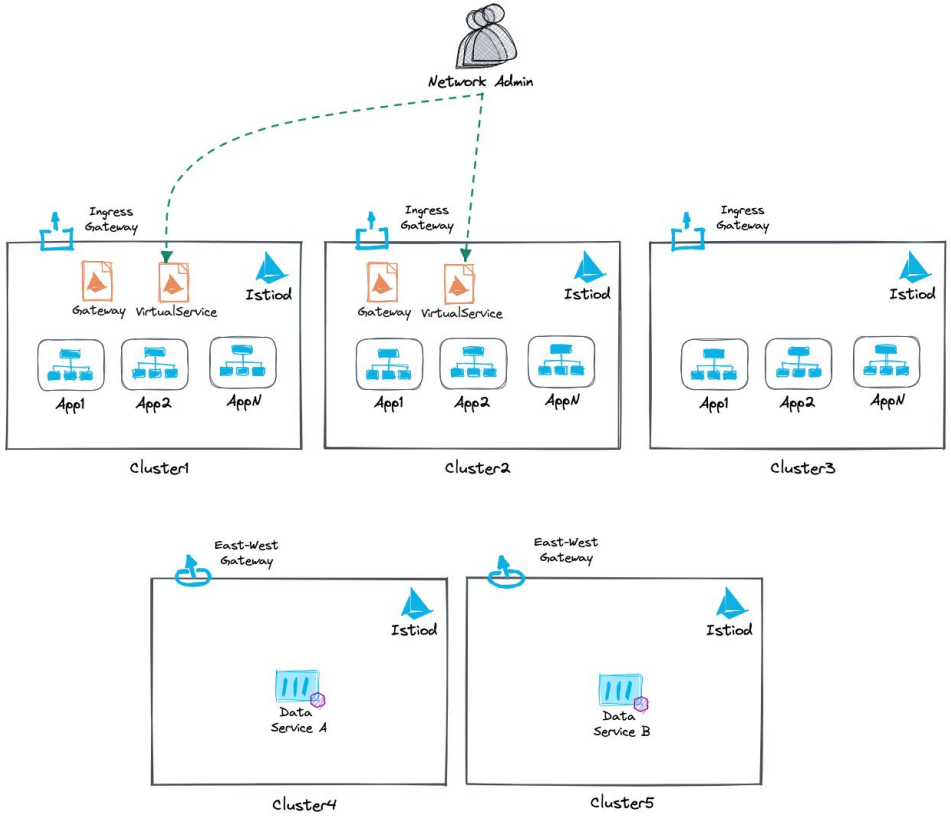
Gitops is a good place to start



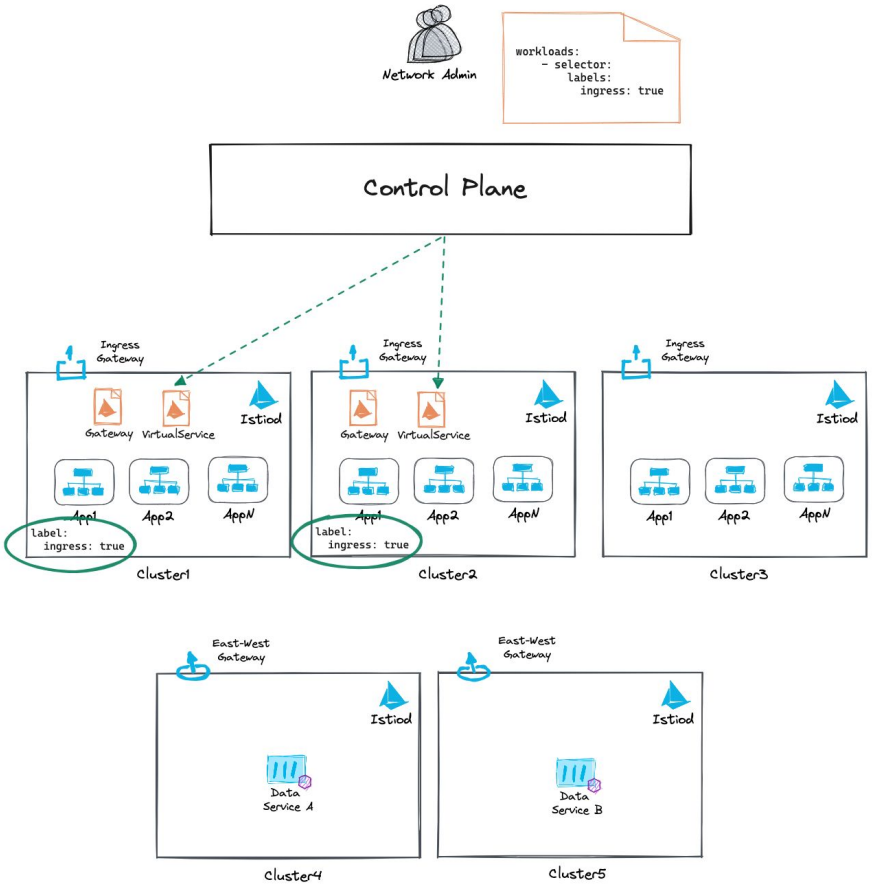
But GitOps alone is not enough



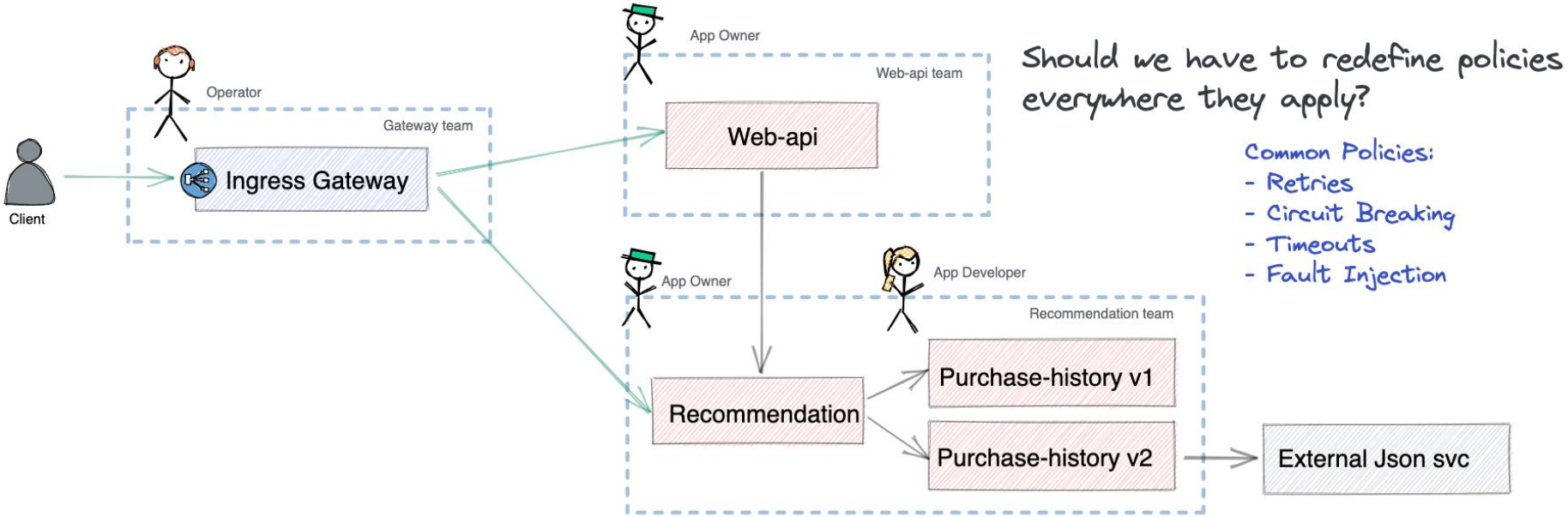
Orchestration is tedious at scale



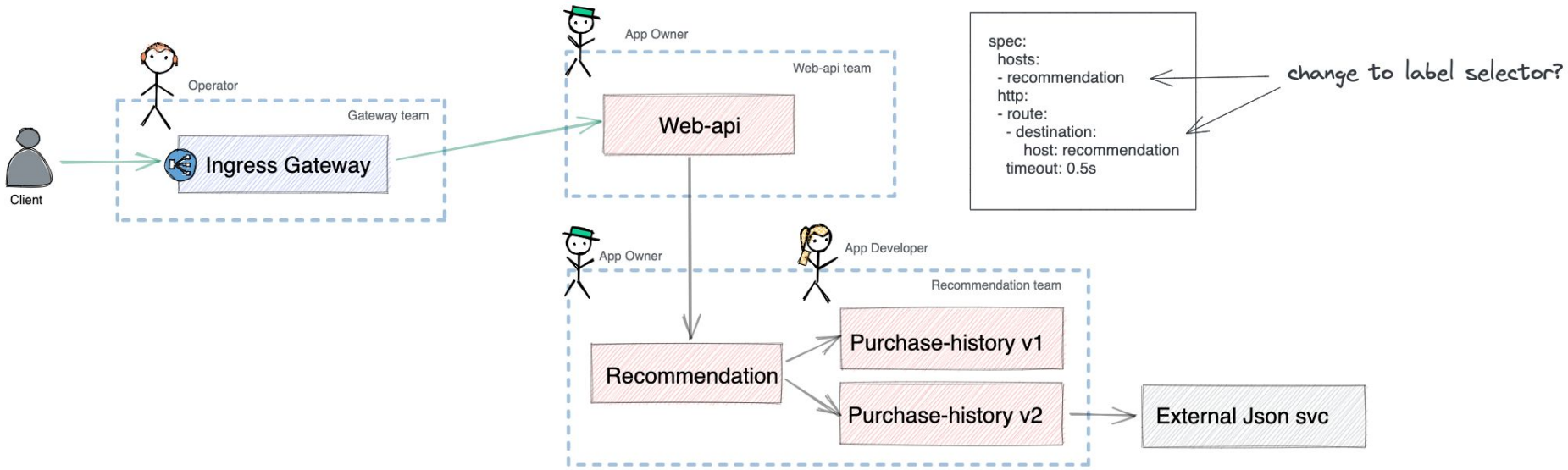
Usage of Labels and Label Selectors



Uniform Policies



Usage of Labels and Label Selectors (Policy)



Thank you!

https://solo.io
https://slack.solo.io

#IstioCon