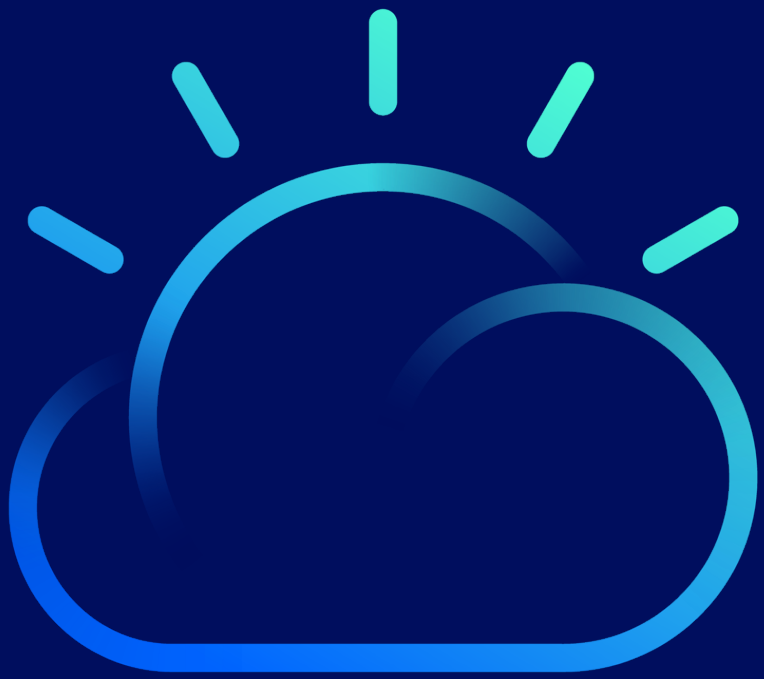
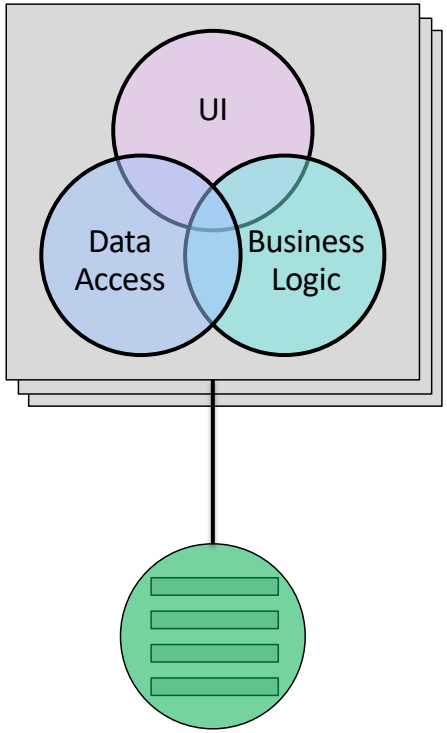
Front Door Architectures

API Connect Istio Integration



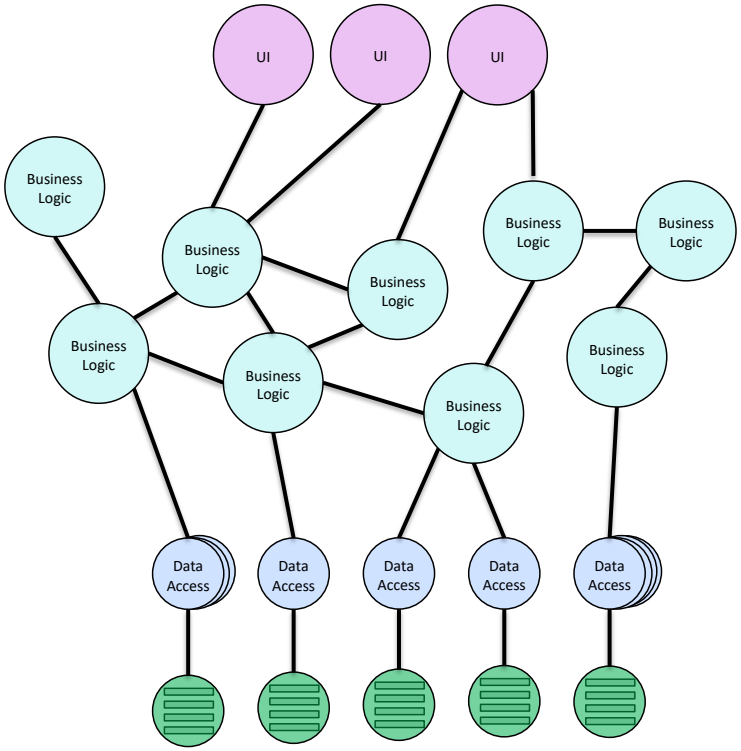
IBM Cloud

Monolithic



versus

Microservices



Weighing the Microservice Investment

Improved delivery velocity and agility

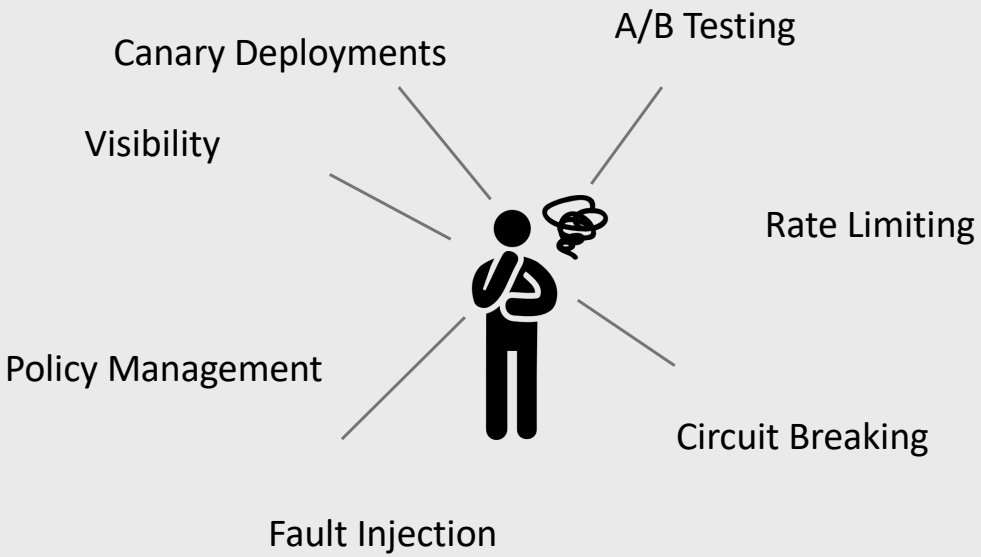
Increased operational complexity

Kubernetes enables the microservice design goals of clean packaging, consistency, scalability and rapid deployment

Kubernetes alone does not address all of the complexities of the challenge

Microservice Adoption Considerations

Deploying microservice applications is not necessarily easy, the network layer is challenging and tooling is essential

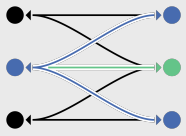


Istio

Connect, secure, control and observe services

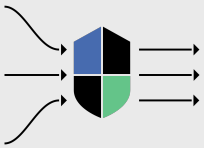
Service mesh describes the **network of microservices** that make up applications and the corresponding **interactions** between them.

Connect
Secure
Control
Observe

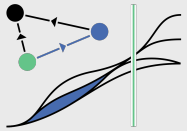


Intelligently control the flow of traffic and API calls between services, conduct a range of tests and upgrade gradually with red / black deployments

Automatically secure your services through managed authentication, authorization and encryption of communication between services



Apply policies and ensure that they are enforced and that resources are fairly distributed among consumers



See what's happening with rich automatic tracing, monitoring and logging of all your services

Istio Core Features and Value

Traffic management
- Easy-to-Configure routing and traffic control
- Simplified configuration of circuit breakers, timeouts, and retries supporting A/B testing, canary and staged rollouts
- High visibility into your traffic

Security
- Free developers to focus on security at the application level
- Istio manages authentication, authorization, and encryption of service communication at scale
- Service communications are secured by default with little or no changes to the application
- Via integration with the platform secure pod-to-pod or service-to-service communication at the network AND application layers

Observability
- Rich tracing, monitoring, and logging provide deep insights into the service mesh
- Understand upstream and downstream performance effects
- Out of the box dashboards provide deep visibility into service usage and performance
- Enables fine-grained control over all interactions between the mesh and infrastructure backends
- Detect, diagnose and fix issues with greater speed and agility

Platform support
- Platform independence
- Deploy across services running in IBM Cloud Private (Kubernetes) and hosted on Virtual Machines

Istio's OOTB Components

A modular set of services/components:

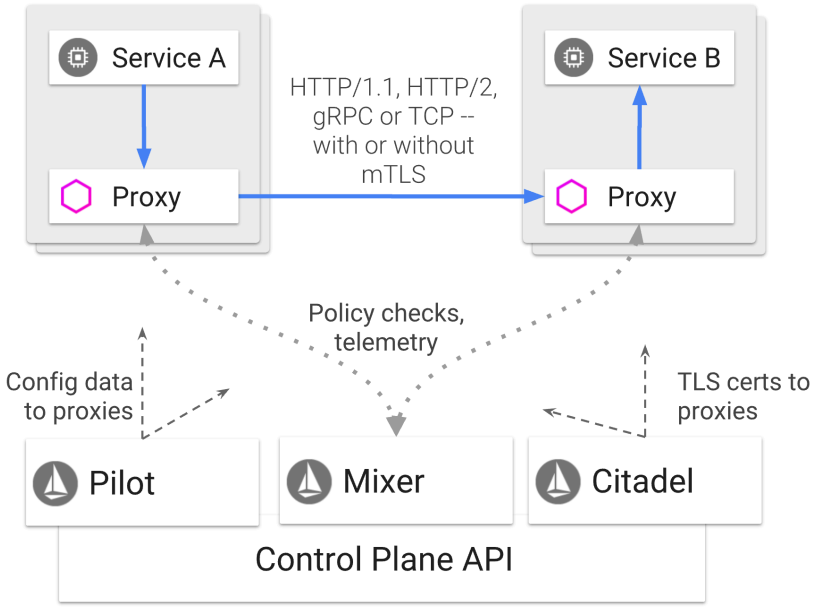
- **Sidecar Proxies (Envoy):** Handles ingress/egress traffic between services in the cluster and from a service to external services transparently
- **Pilot:** Configures the proxies at runtime
- **Mixer:** Enforces ACLs, rate limits, quotas, authentication, request tracing, and telemetry collection
- **Certificate Authority:** Issues and rotates certs for service identities
- **Initializer:** Injects sidecar proxies
- **Ingress:** Manages external access to the services

Istio Architecture
Data Plane & Control Plane

Istio is logically composed from a data plane and a control plane

Data Plane
- Intelligent proxies are deployed as sidecars within the service pods
- The proxies mediate and control communication between microservices
- Proxies interface with the Mixer to provide telemetry data and enforce policy

Control Plane
- Configures the proxies for traffic routing
- Configures Mixers for policy enforcement and telemetry collection

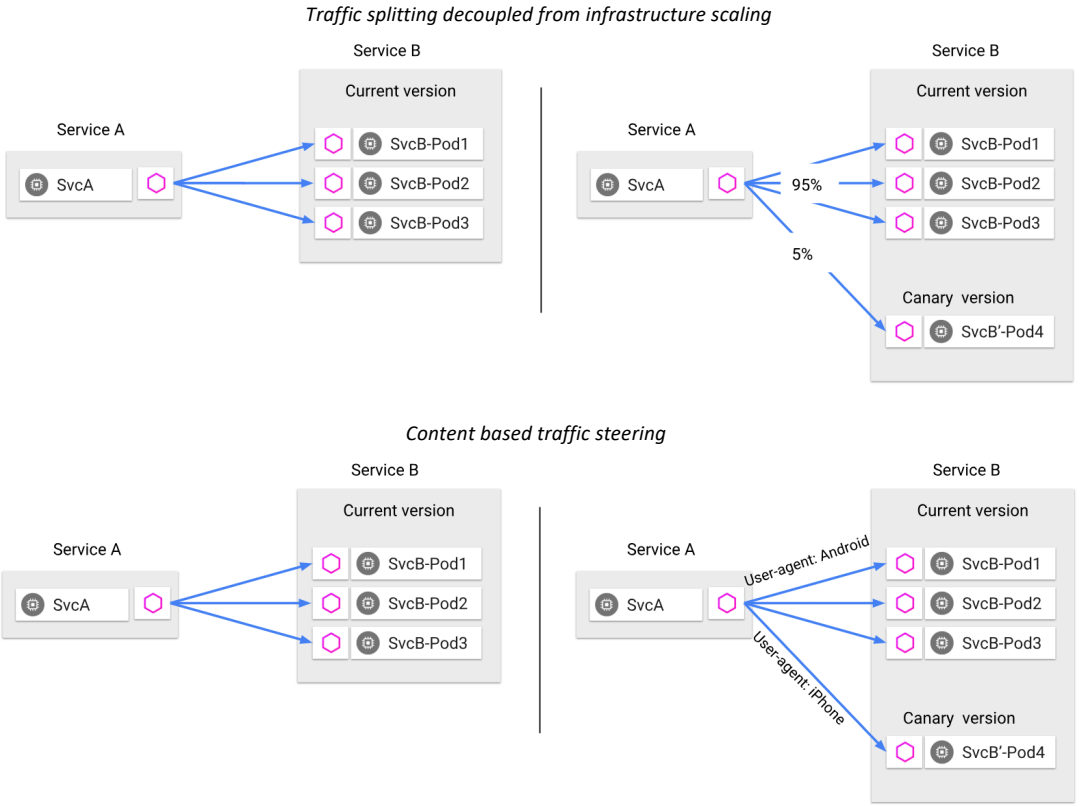


Istio Traffic Management Overview

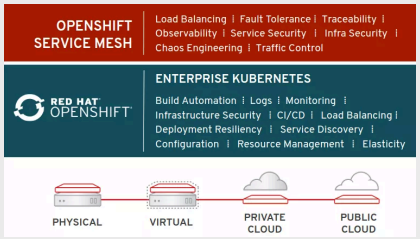
The traffic management model decouples traffic flow and infrastructure scaling giving you the option of specifying via rules and Pilot how traffic should flow

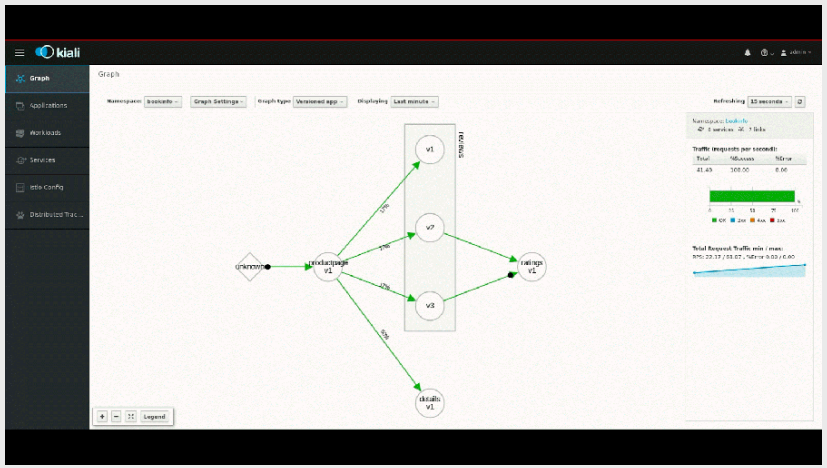
For example, you can direct a percentage of traffic for a particular service to a canary service or only direct to the canary based upon the content of the request

Decoupling traffic flow from scaling of infrastructure allows for traffic management features outside of the application code including failure recovery via timeouts, retries, circuit breakers and fault injection to test failure recovery procedures



OpenShift Service Mesh





- Service Mesh Tech Preview with RHOCP 3.11
- A few limitations: Only supports OCP Software Defined Networking configured as a flat network (no external providers), no federation, no external microservice support
- Forked version of Istio
- Injection is not managed by namespace
- Matching header information via regex has been added
- BoringSSL replaced by OpenSSL
- OpenShift will add two namespaces / projects: istio-operator, istio-system
- Multi-tenancy differences

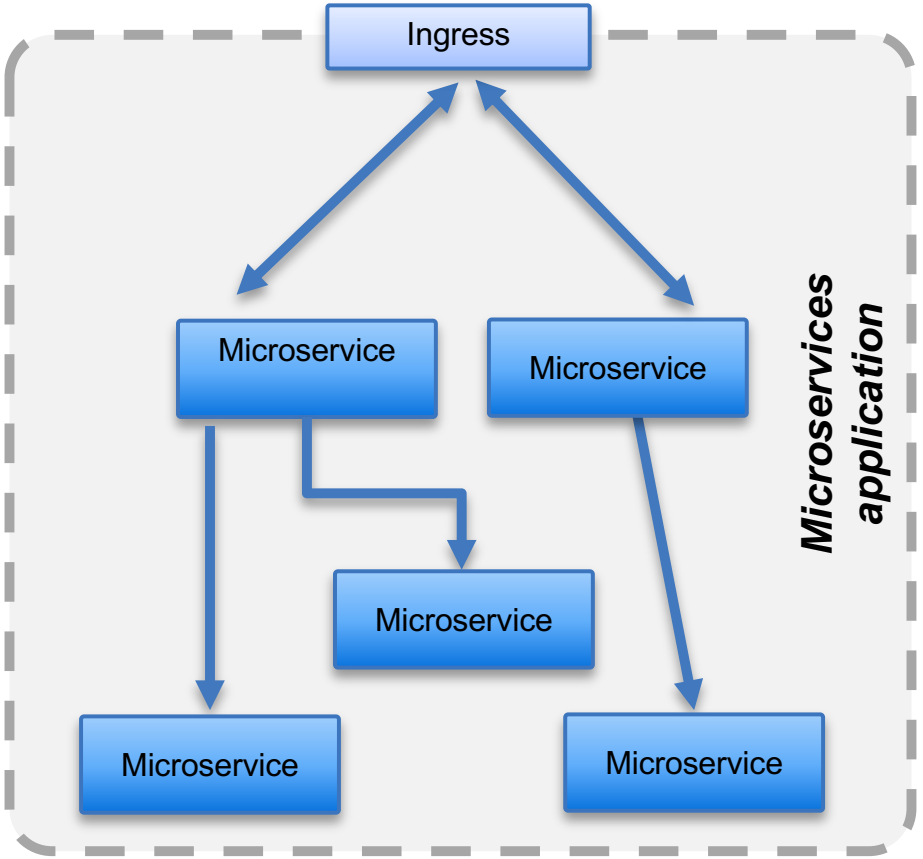
Managing the Interaction Between Microservices

Kubernetes manages the lifecycle of individual containers

Istio runs on Kubernetes allowing you to manage and associate the interaction between microservices (deployed in containers)

Kubernetes provides routing of microservices but is not concerned with the security or routing requirements between individual microservices

Istio provides a policy-based approach to provide security, app resiliency and dynamic routing between microservices



Managing the Interaction Between Microservices

Deployment in Kubernetes

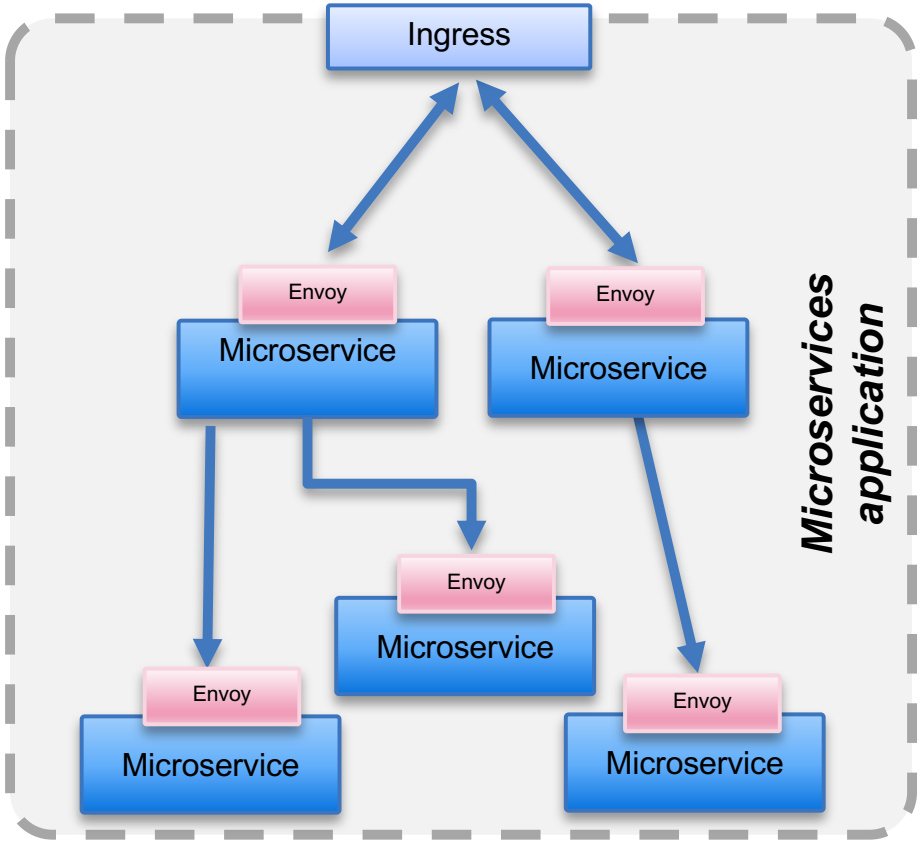
```
NAME                                 READY     STATUS    RESTARTS   AGE
fancave-client-66764c4796-4cr7l      1/1       Running   0          3m
fancave-db-c9d67ccb7-bdxjv           1/1       Running   0          3m
fancave-news-7b577ff4b7-nj2z7        1/1       Running   0          3m
fancave-teams-ab577ytfs-n3rz7        1/1       Running   0          3m
fancave-players-bcfd9bd68-v6lgk      1/1       Running   2          3m
```

Deployment with Istio Sidecars

```
NAME                                 READY     STATUS    RESTARTS   AGE
fancave-client-66764c4796-4cr7l      2/2       Running   0          3m
fancave-db-c9d67ccb7-bdxjv           2/2       Running   0          3m
fancave-news-7b577ff4b7-nj2z7        2/2       Running   0          3m
fancave-teams-ab577ytfs-n3rz7        2/2       Running   0          3m
fancave-players-bcfd9bd68-v6lgk      2/2       Running   2          3m
```

Managing with Policy

```
NAME                                 READY
istio-system   istio-citadel-6b6fdfdd6f-qnk2p
istio-system   istio-policy-67f4d49564-5tx5
istio-system   istio-pilot-6f8d49d4c4-qdbzs
```



API Connect and Istio Comparison

Capabilities of Istio and API Connect

Can Istio replace API Management solutions?

Istio is **NOT** a complete **API Management solution**

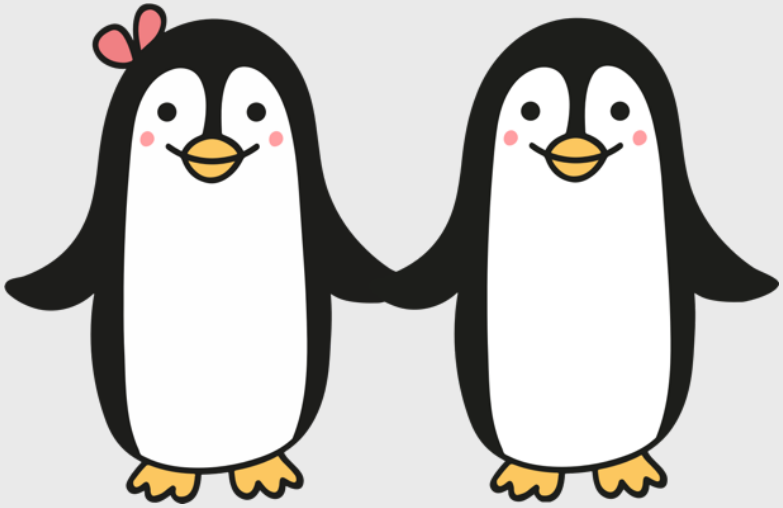
Istio does not provide **API lifecycle**, **socialization** or **comprehensive edge API security**

Can I replace DataPower with Istio / Envoy?

No, they have very different value propositions

Can I use DataPower & Envoy together?

Yes, they are complementary and great things happen when they work together

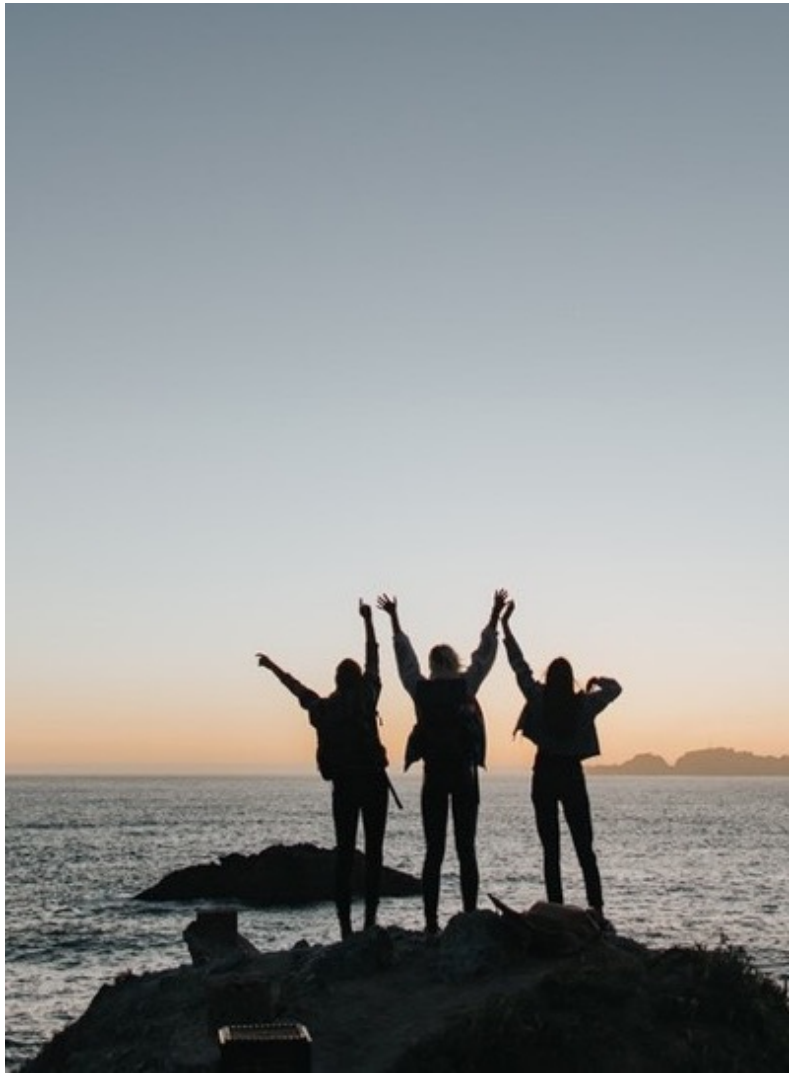


API Management Emphasizes the API Consumer

API management has the goal of greater API control with control of change, consumption and API subscriptions

The goals of Microservice Management are managing service interaction and change (as a collection) over time

API management becomes critical when the organizational distance increases between the API provider and the API consumer



API Economy requires External API Strategy

API changes & versioning requires a controlled communication process especially if there are a large number of public API consumers

APIs must be managed as products since third-party applications are built trusting their availability

API Providers manage changes as part of the API lifecycle: staging, published, replacement (non-breaking), deprecation (if breaking), and finally retirement

Microservices and API Rate Limiting Serve Different Purposes

Rate Limiting of Microservices is to prevent the application from hanging and failing fast to recover quickly

Rate Limiting of APIs is a business requirement to manage the number of API calls, potentially for monetization

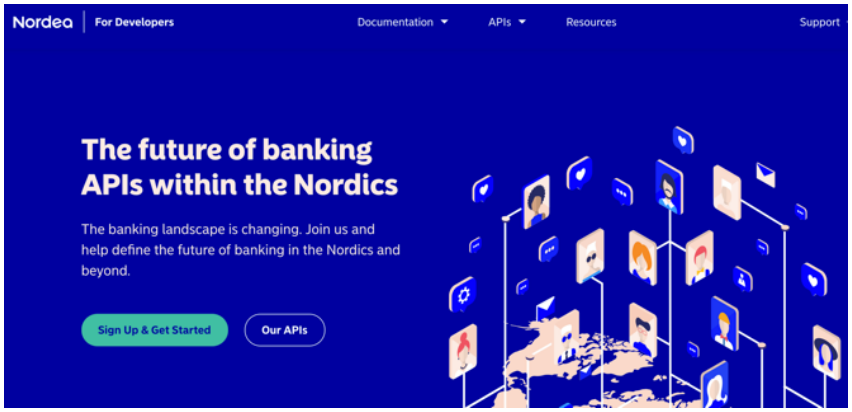
Circuit Breakers in Microservices management provide an additional level of protection to timeout long running microservices and act more resiliently

API Management Provides Developer Portals for Service Discovery

API Management platforms provide a Developer portal so developers can self-discover APIs and invoke them without contacting the API provider

Microservice Management does not have a socialization strategy

Access to the service mesh can be given to services but the discovery and relationship is manually managed



Key Takeaways

API Management is GREAT at

- **Managing API Consumers** and communicating lifecycle changes about the API
- **Securely expose data assets** as APIs to third-party applications
- **Self-service discovery and management** of APIs using Developer Portal

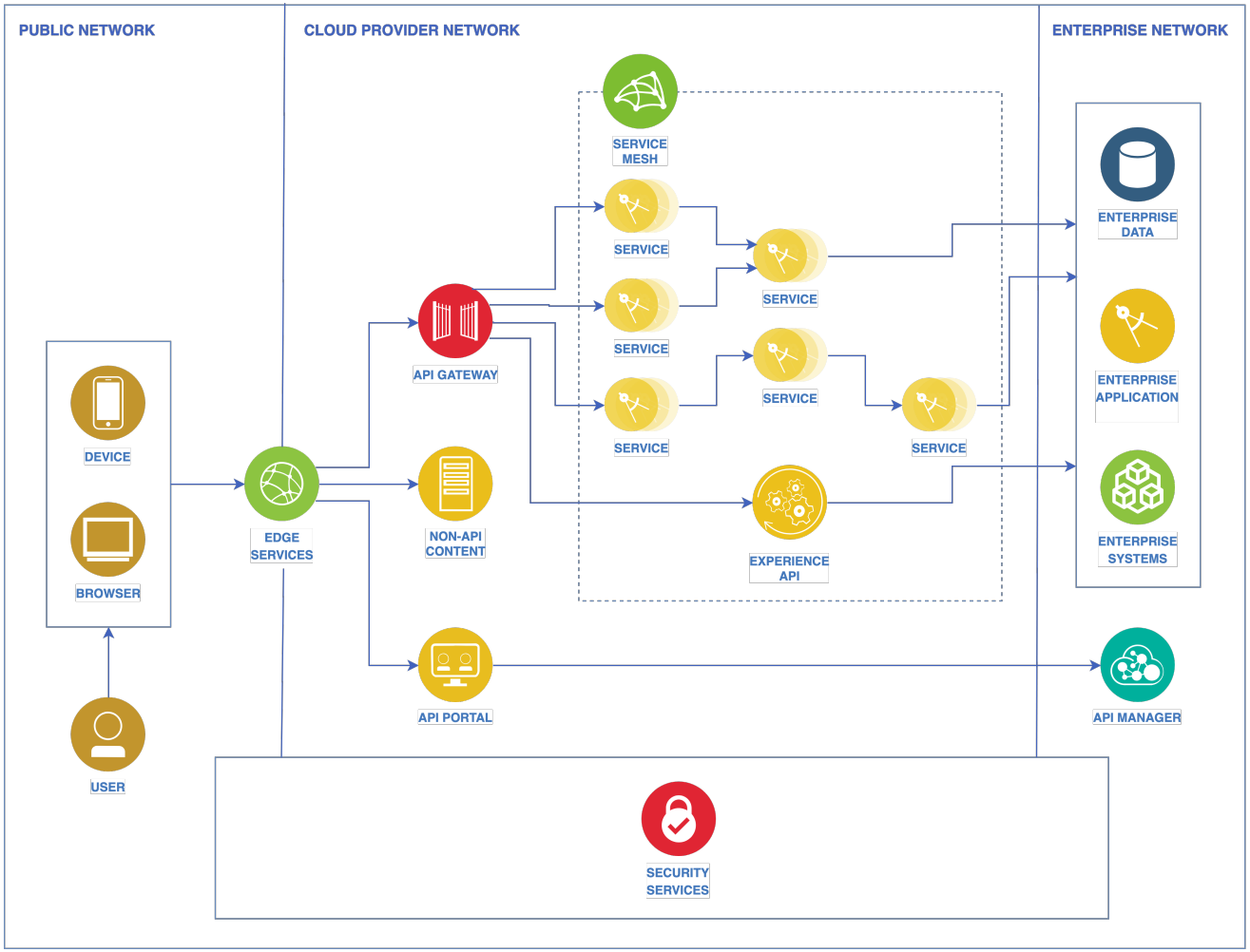
Microservice Management (ISTIO) is GREAT at:

- **Mesh routing and discovery** between Microservices
- **Mesh security** between microservices without impacting performance
- **Preventing microservices from catastrophic** application failures - failing fast to recover quickly
- **Providing visibility** into the service landscape
- **Simplification** of the **developer** experience

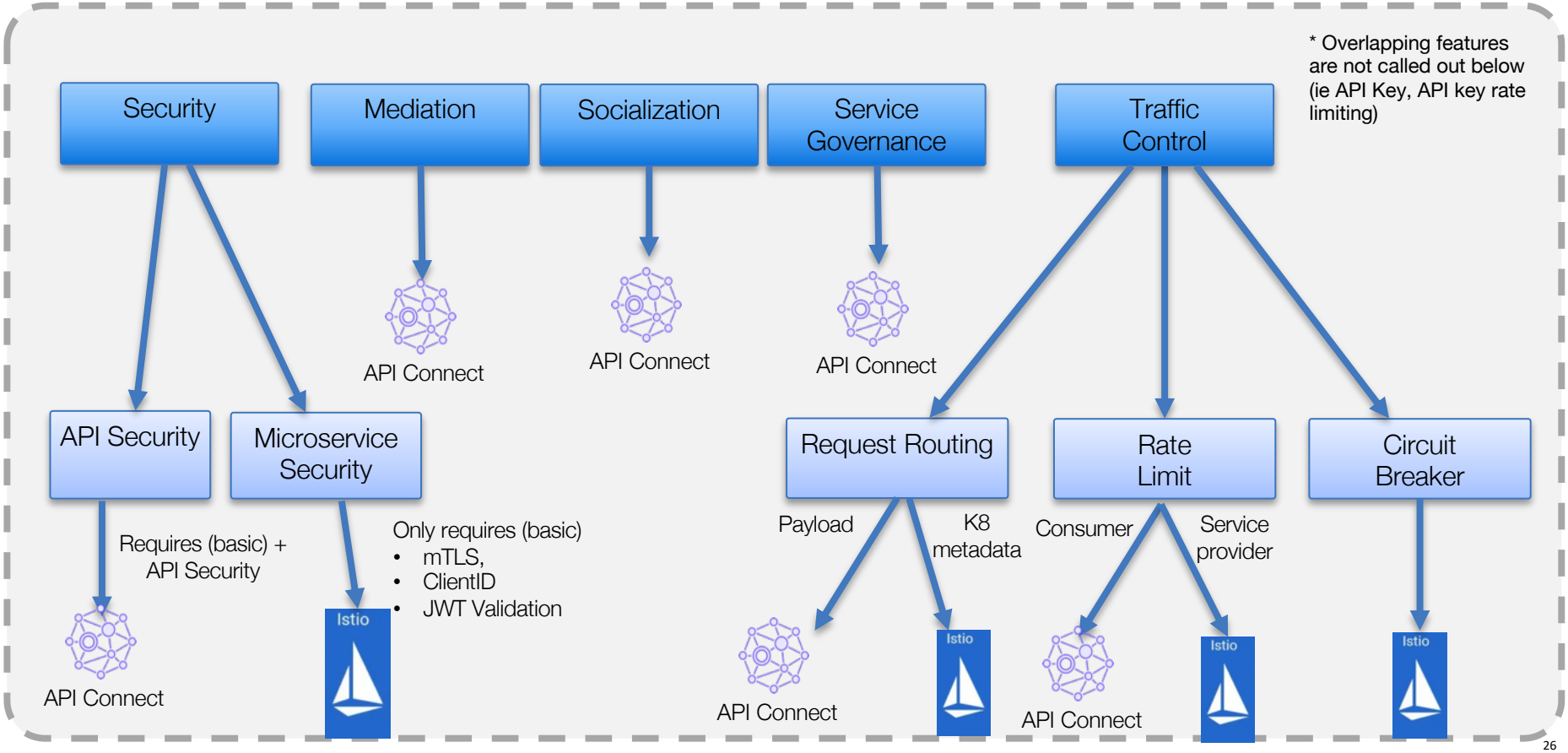
API Connect Istio Enablement

Reference Architecture

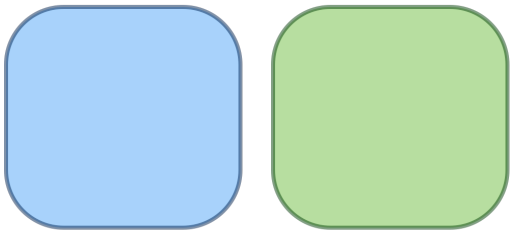
API Connect & Istio Reference Architecture



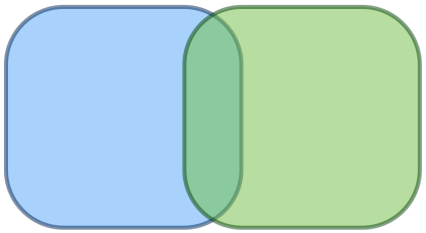
API vs Microservices Management Guidance



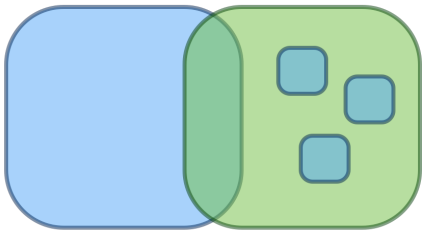
API Connect Istio Mesh



Complementary Capability

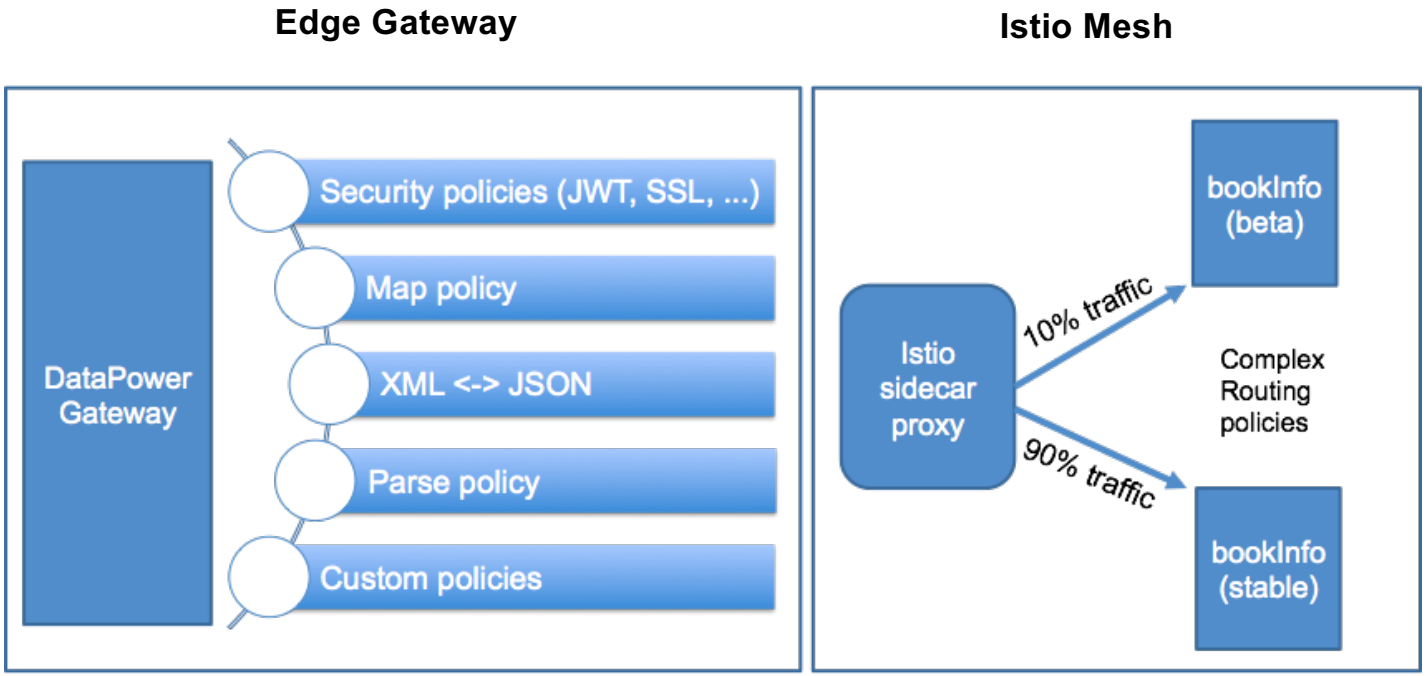


Ingress Overlap



Augmented Mesh

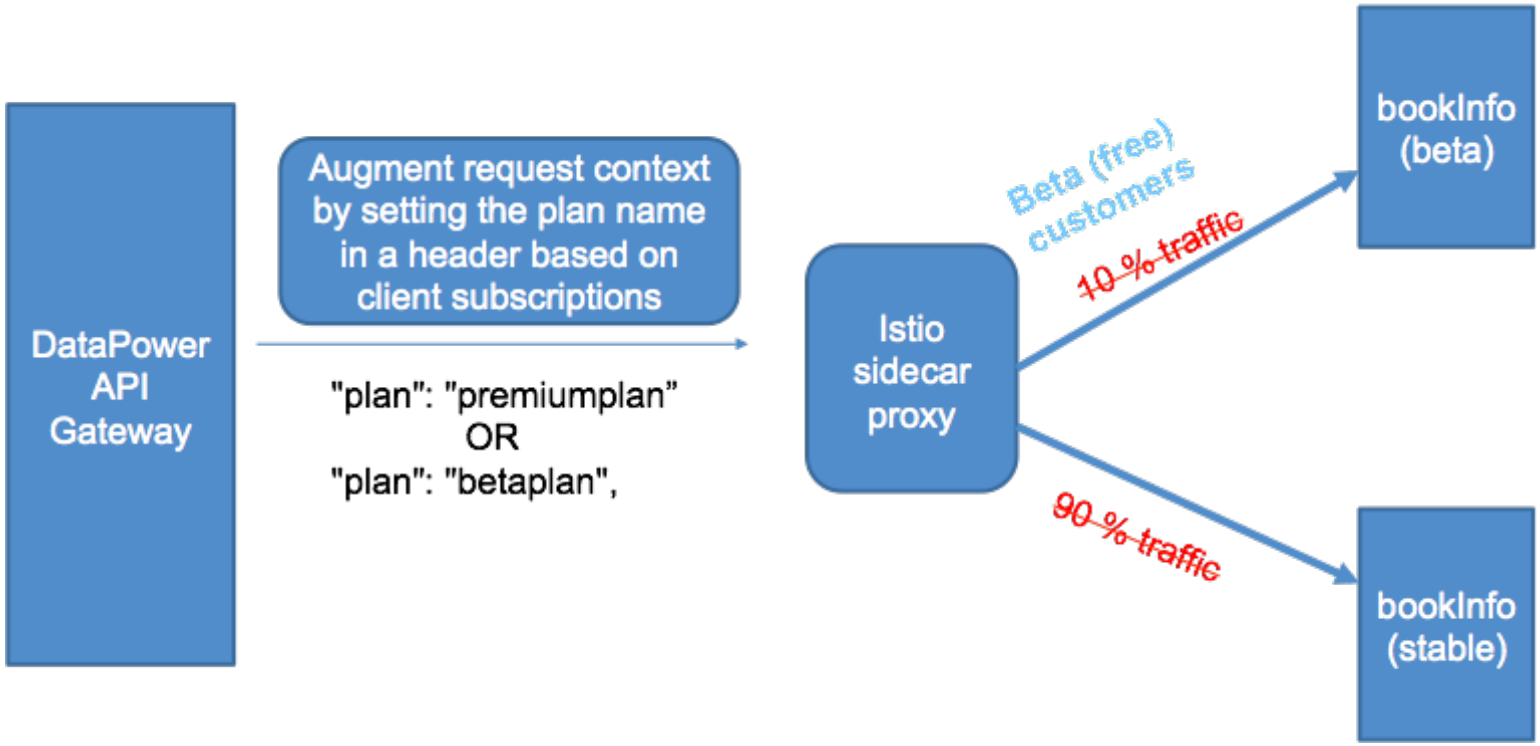
API Connect Istio Enablement



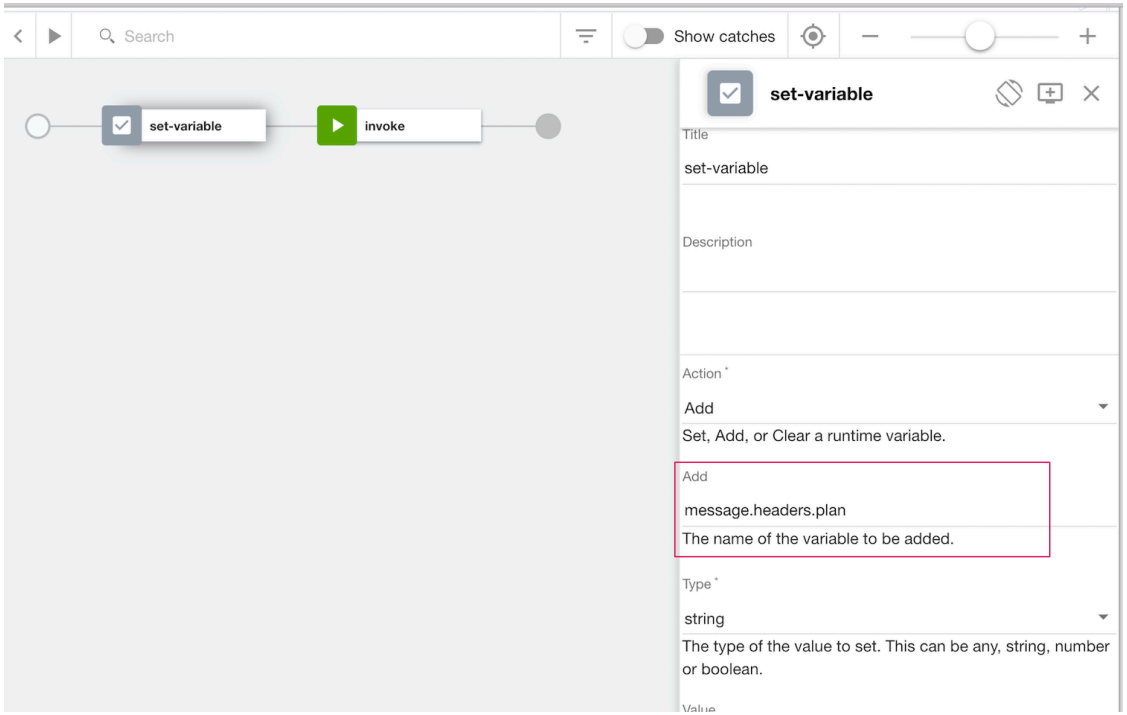
API Connect Istio Demo

Context Augmentation & Plan Based Routing

Context Augmentation & Plan Based Routing



API config with context augmentation



Istio plan based policy

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
  - bookinfo
  http:
  - match:
    - headers:
        plan:
          exact: premium-plan
    route:
    - destination:
        host: bookinfo
        subset: StableVersion
  - route:
    - destination:
        host: bookinfo
        subset: BetaVersion
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: bookinfo
spec:
  host: bookinfo
  subsets:
  - name: StableVersion
    labels:
      version: v1
  - name: BetaVersion
    labels:
      version: v2
  - name: v3
    labels:
      version: v3
```