Istio – An introduction for developers

WW Developer Advocacy Team

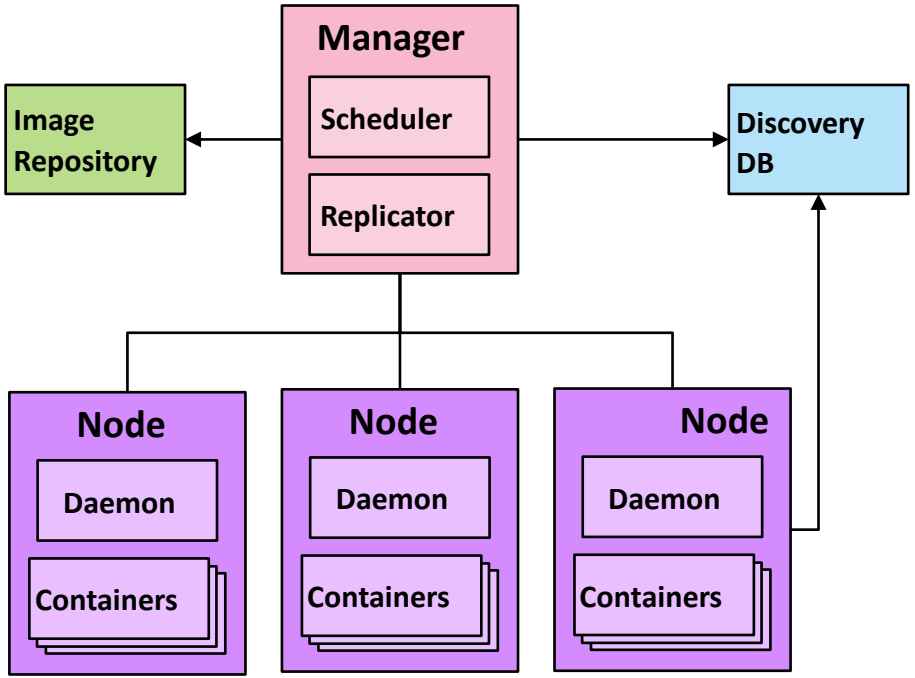
Benefits of Kubernetes

Automated scheduling and scaling

Zero downtime deployments

High availability and fault tolerance

A/B Testing



Microservice Challenges

Controlling Traffic
1. How do I do canary testing?
2. How do I A/B testing?

Resilient Services
1. How do I implement circuit breakers?
2. How do I test faults in the system?
3. How do I limit set rate limits for each service?

Telemetry
1. How can I trace a request through my system of multiple services
2. How can I perform monitoring in a distributed deployment?

Security
1. How can I apply policies across services?

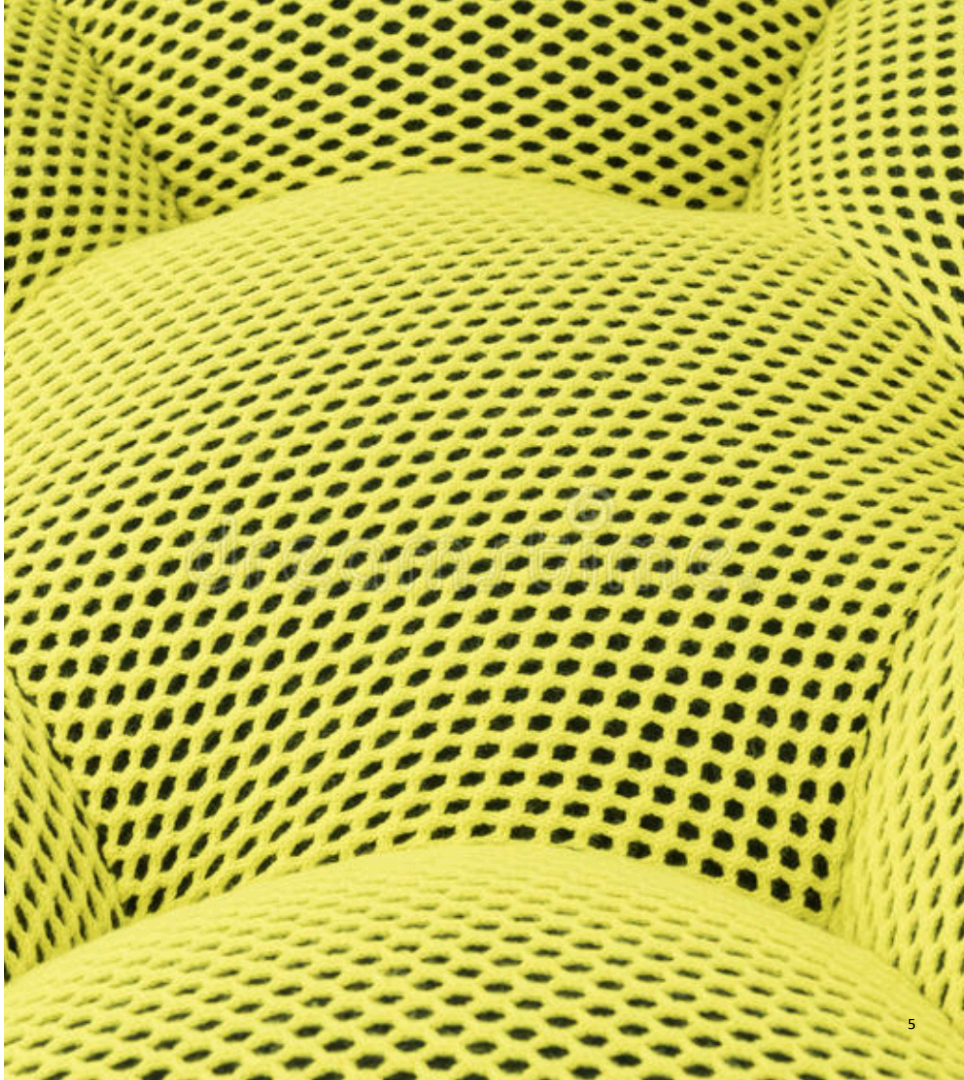
← All of these problems are common across services no matter the runtime.

← It would be cool if we could solve these problems in a standard way **without changing application code**

Service Mesh
&
Istio Architecture

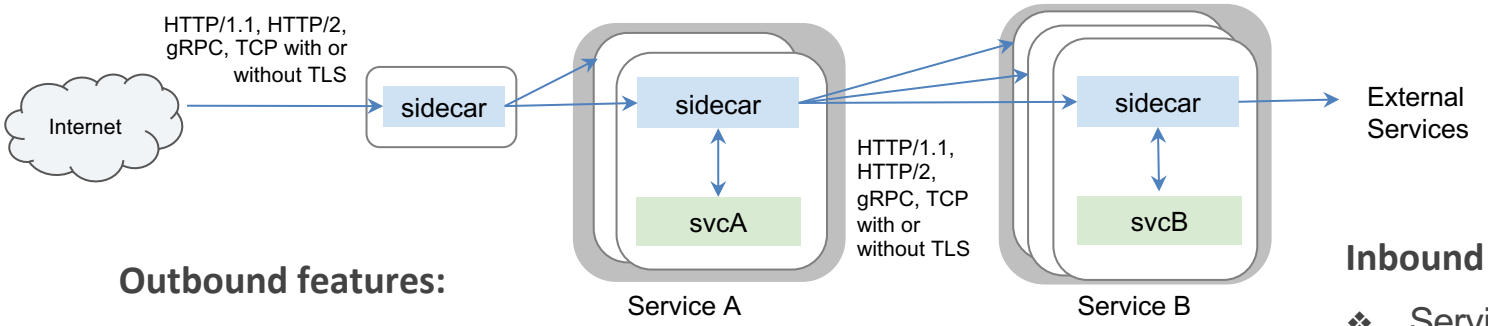
What is a service mesh?

A service mesh provides a **transparent** and **language-independent** way to flexibly and easily manage the **communication** between microservices**.**



Istio is a **service mesh** that supports managing traffic flows between microservices, enforcing access policies, and aggregating telemetry data, all without requiring changes to the microservice code.

Weaving the mesh



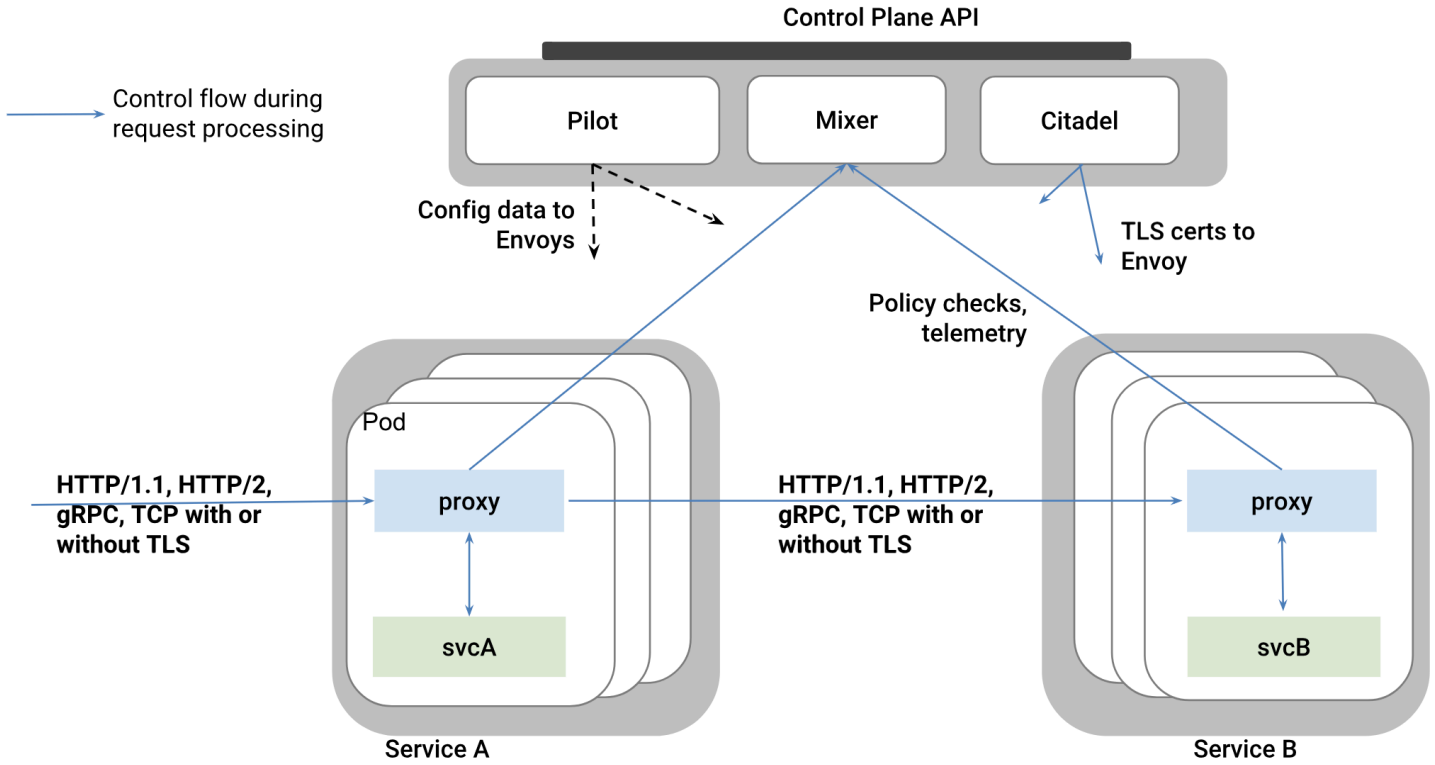
Outbound features:

- Service authentication
- Load balancing
- Retry and circuit breaker
- Fine-grained routing
- Telemetry
- Request Tracing
- Fault Injection

Inbound features:

- Service authentication
- Authorization
- Rate limits
- Load shedding
- Telemetry
- Request Tracing
- Fault Injection

Istio Architecture



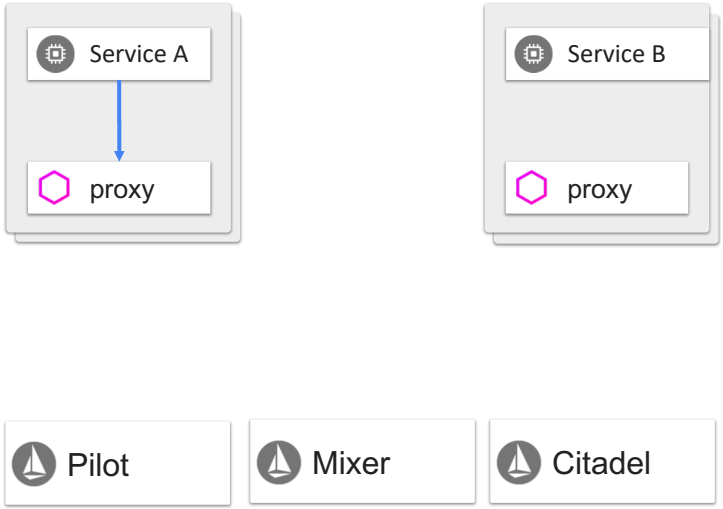
Istio Architecture

Mixer
- Collects telemetry data sent to it from the proxy sidecar containers on each request.
- Platform-independent, very pluggable with various backends.

Pilot
- “Drives” the service mesh.
- Listens for your configuration around traffic routing, circuit-breakers and fault injection.
- Coverts that to low-level routing rules, and distributes those to sidecar containers at runtime.

Citadel
- Does security
- Key and certificate management. Distributes as Kubernetes Secrets.

Life of a request

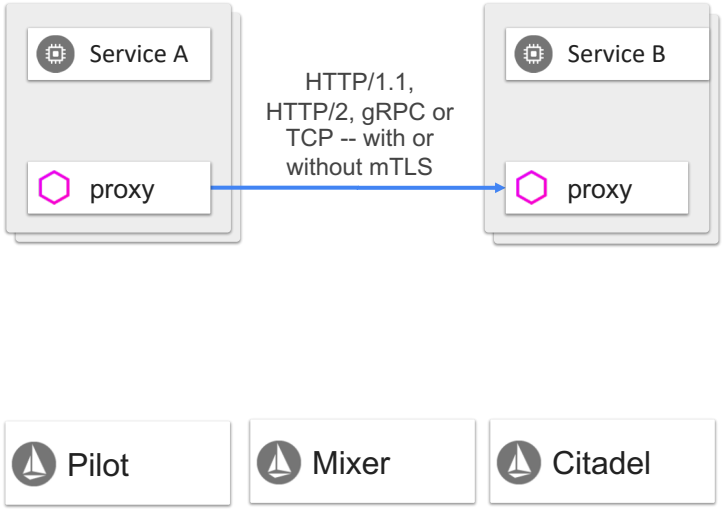


Service A places a call to http://service-b.

Client-side Envoy intercepts the call.

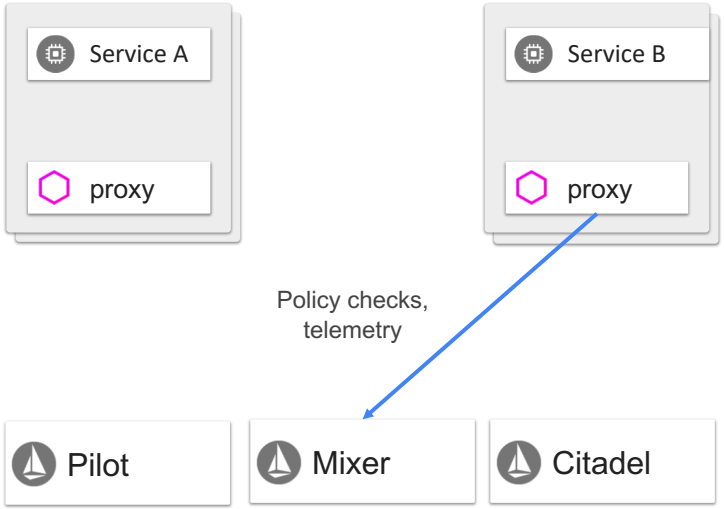
Envoy consults config (previously received from Pilot) to know how/where to route call to service B (taking into account service discovery, load balancing, and routing rules), forwards the call to the right destination.

Life of a request



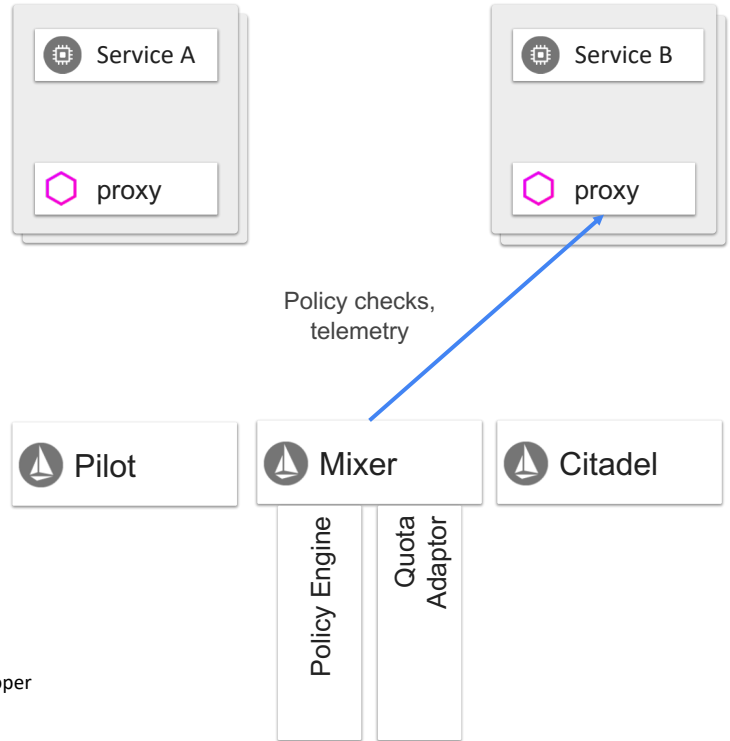
Envoy forwards request to appropriate instance of service B. There, the Envoy proxy deployed with the service intercepts the call.

Life of a request



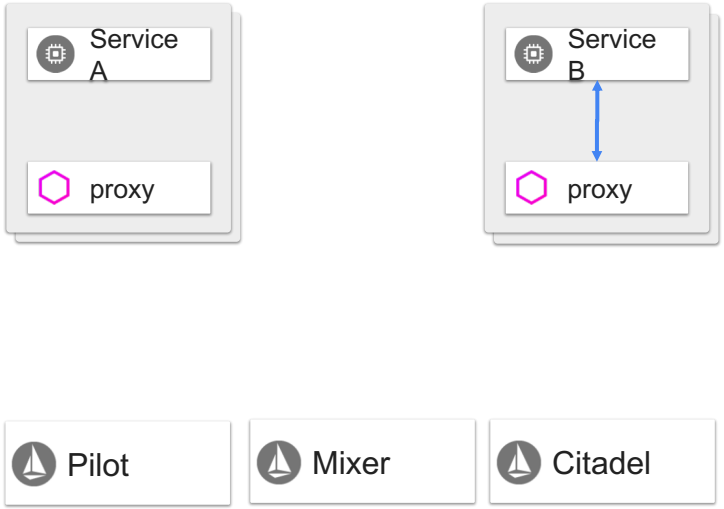
Server-side Envoy checks with Mixer to validate that call should be allowed (ACL check, quota check, etc).

Life of a request



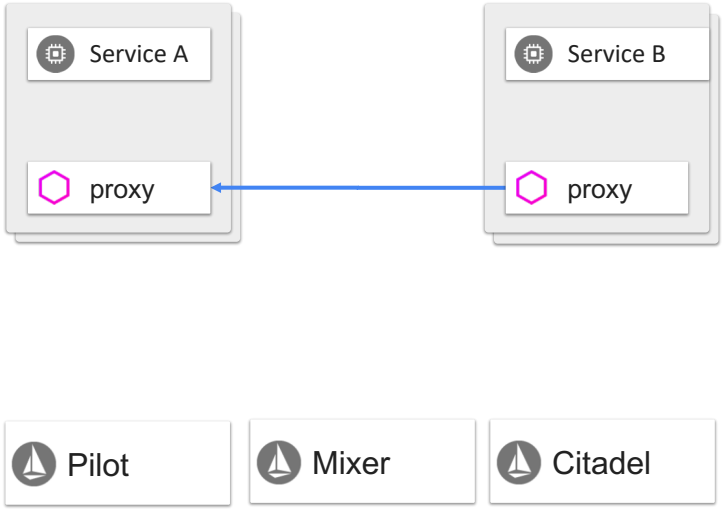
Mixer checks with appropriate adaptors (policy engine, quota adaptor) to verify that the call can proceed and returns true/false to Envoy

Life of a request



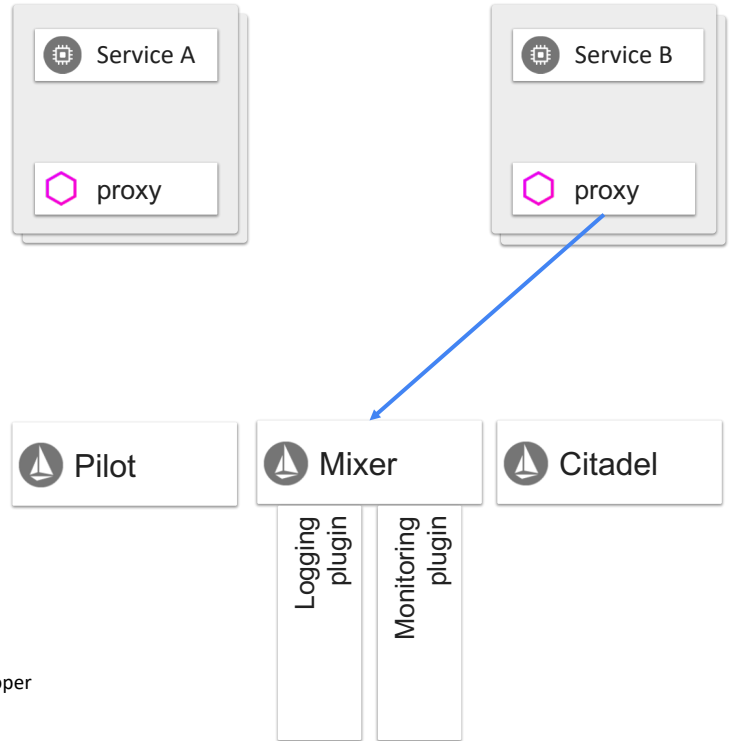
Server-side Envoy forwards request to service B, which process request and returns response

Life of a request



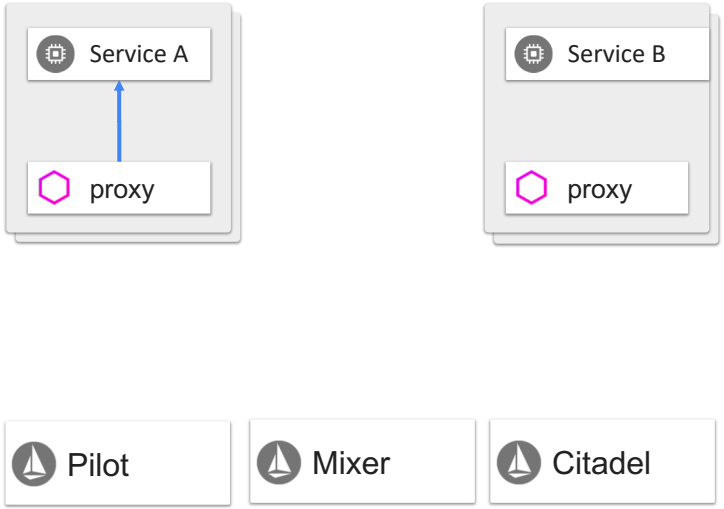
Envoy forwards response to the original caller, where response is intercepted by Envoy on the caller side.

Life of a request



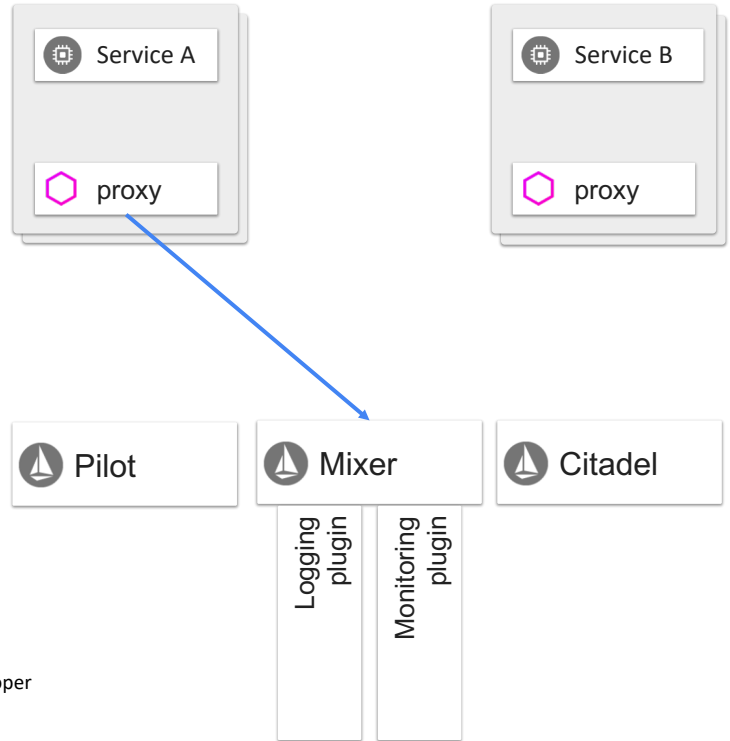
Envoy reports telemetry to Mixer, which in turn notifies appropriate plugins

Life of a request



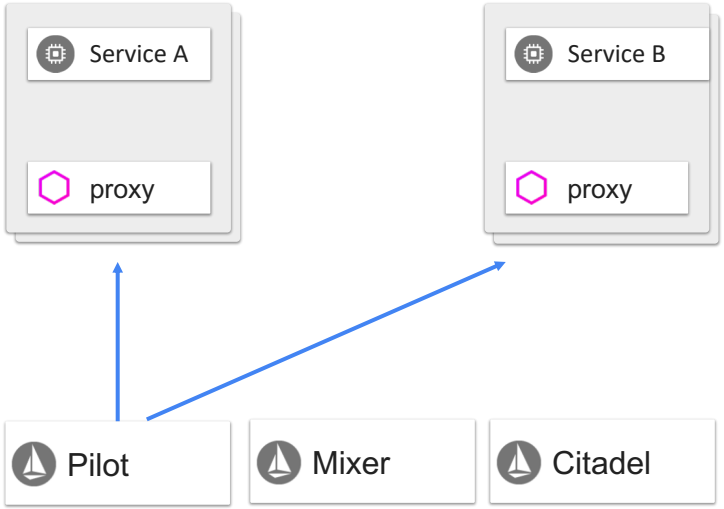
Client-side Envoy forwards response to original caller.

Life of a request



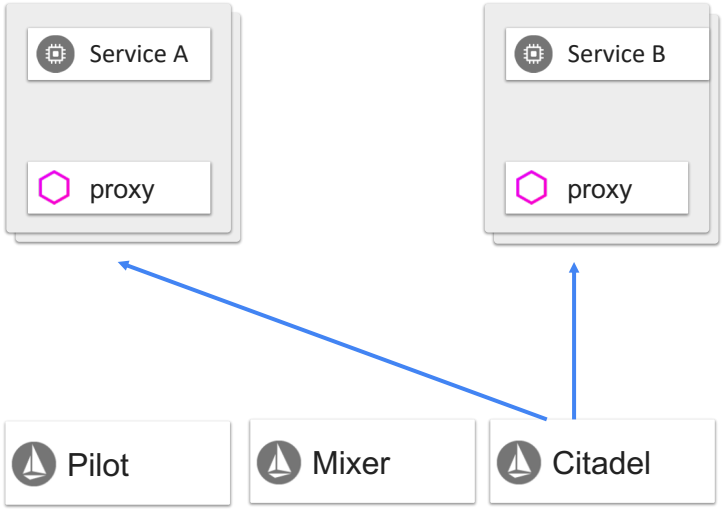
Client-side Envoy reports telemetry to Mixer (including client-perceived latency), which in turn notifies appropriate plugins

At Anytime



Pilot listens to your configuration, such as routing rules, circuit breaking and fault injection. Converts to low-level config (routing rules) and distributes to proxy side cars

At Anytime



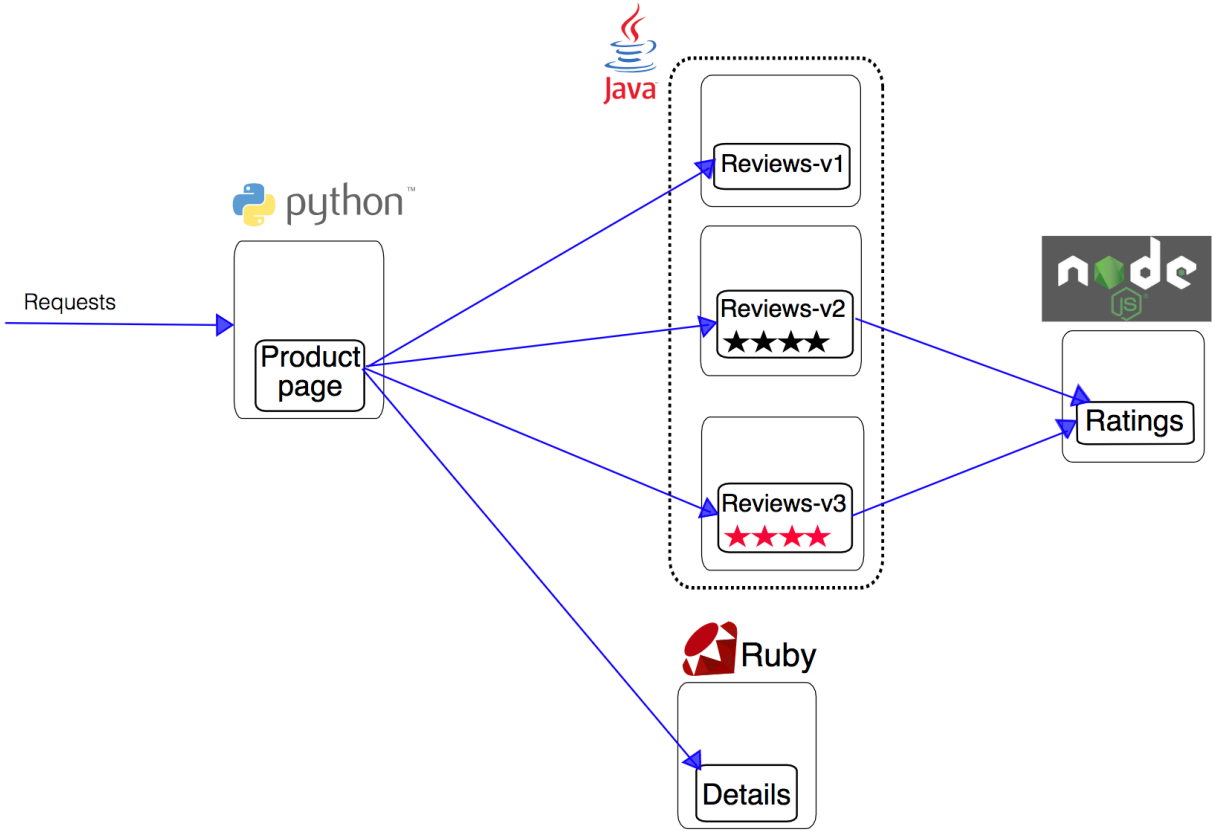
Citadel distributes keys and certificates as Kubernetes Secrets available to sidecar containers

Istio Proxy Sidecar

- To create a service mesh with Istio, you update the deployment of the pods to add the Istio Proxy (based on the Lyft Envoy Proxy) as a side car to each pod. The Proxy is then run as a separate container that manages all communication with that pod.
  - Manual Approach
  - **Automated Approach**

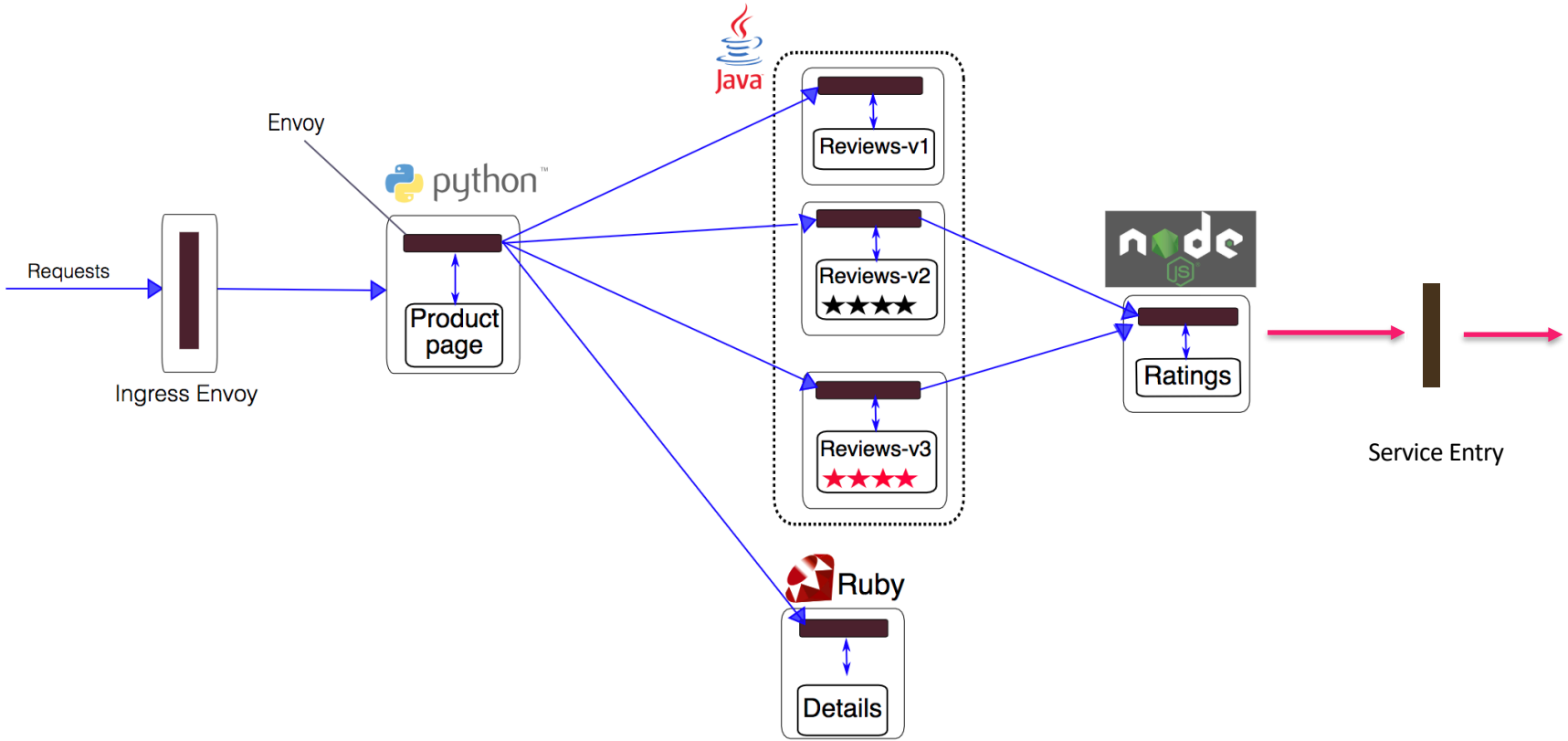
- Istio modifies the individual pods and not deployments

Bookinfo Sample Microservices Application (without Istio)



Bookinfo Application without Istio

Bookinfo Sample Microservices Application (with Istio)



Bookinfo Application

Control Ingress Traffic

- **Gateway** describes a load balancer operating at the edge of the mesh receiving incoming or outgoing HTTP/TCP connections.
- **VirtualService** defines a set of traffic routing rules to apply when a host is addressed. Each routing rule defines matching criteria for traffic of a specific protocol.
- Istio ingress controller gets exposed as a normal kubernetes service load balancer on port 80.

```
kubectl get svc -n istio-system
```

https://istio.io/docs/tasks/traffic-management/ingress/

Istio Examples

Request Routing

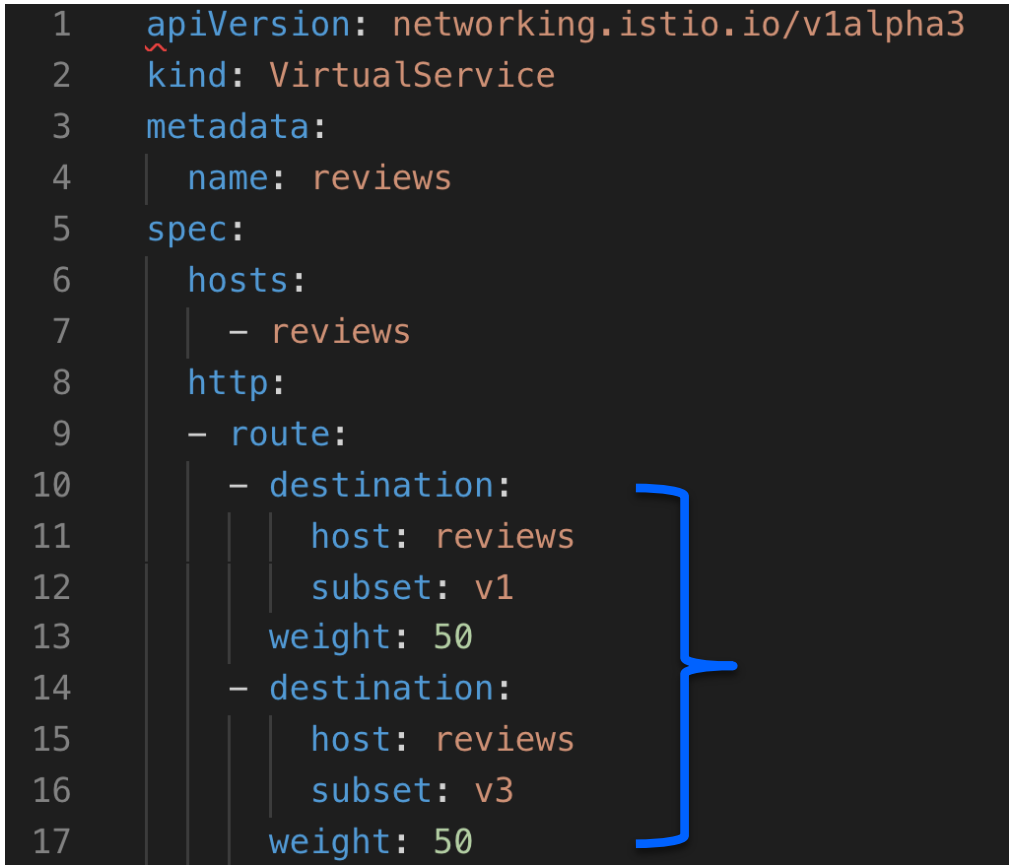
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
    - reviews
  http:
  - match:
    - headers:
        end-user:
          exact: jason
    route:
    - destination:
        host: reviews
        subset: v2
  - route:
    - destination:
        host: reviews
        subset: v1
```

Canary Testing

Route user:jason to reviews:v2

Others still get reviews:v1

Traffic Shifting



```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
    - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
      weight: 50
    - destination:
        host: reviews
        subset: v3
      weight: 50
```

50% -> v1

50% -> v3

Rate Limits

```yaml
apiVersion: "config.istio.io/v1alpha2"
kind: memquota
metadata:
  name: handler
  namespace: istio-system
spec:
  quotas:
  - name: requestcount.quota.istio-system
    maxAmount: 5000
    validDuration: 1s
    # The first matching override is applied.
    # A requestcount instance is checked against override dimensions.
    overrides:
    # The following override applies to 'ratings' when
    # the source is 'reviews'.
    - dimensions:
        destination: ratings
        source: reviews
      maxAmount: 1
      validDuration: 1s
    # The following override applies to 'ratings' regardless
    # of the source.
    - dimensions:
        destination: ratin
      maxAmount: 100
```

5000 requests per 1s

ratings: 100 requests per 1s

Circuit Breaking

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: httpbin
spec:
  host: httpbin
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 1
      http:
        http1MaxPendingRequests: 1
        maxRequestsPerConnection: 1
    outlierDetection:
      consecutiveErrors: 1
      interval: 1s
      baseEjectionTime: 3m
      maxEjectionPercent: 100
```

Max 1 concurrent connection & request

Delay Injection

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - match:
    - headers:
        end-user:
          exact: jason
    fault:
      delay:
        percent: 100
        fixedDelay: 7s
    route:
    - destination:
        host: ratings
        subset: v1
  - route:
    - destination:
        host: ratings
        subset: v1
```

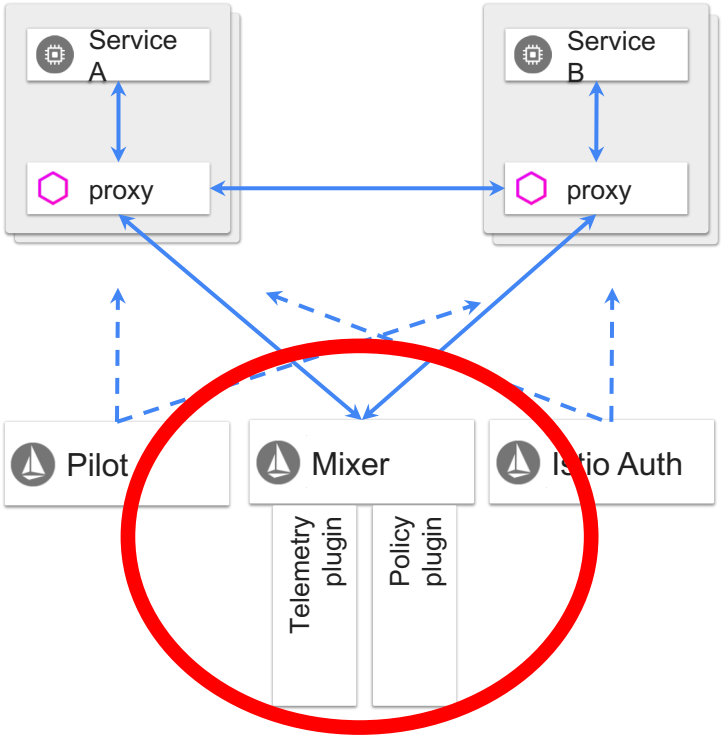
Inject 7 second delay

Fault Injection

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - match:
    - headers:
        end-user:
          exact: jason
    fault:
      abort:
        percent: 100
        httpStatus: 500
    route:
    - destination:
        host: ratings
        subset: v1
  - route:
    - destination:
        host: ratings
        subset: v1
```

jason: Return with Error 500

Istio Telemetry (v1)



Telemetry Tools

Metrics

Service Mesh Visualization

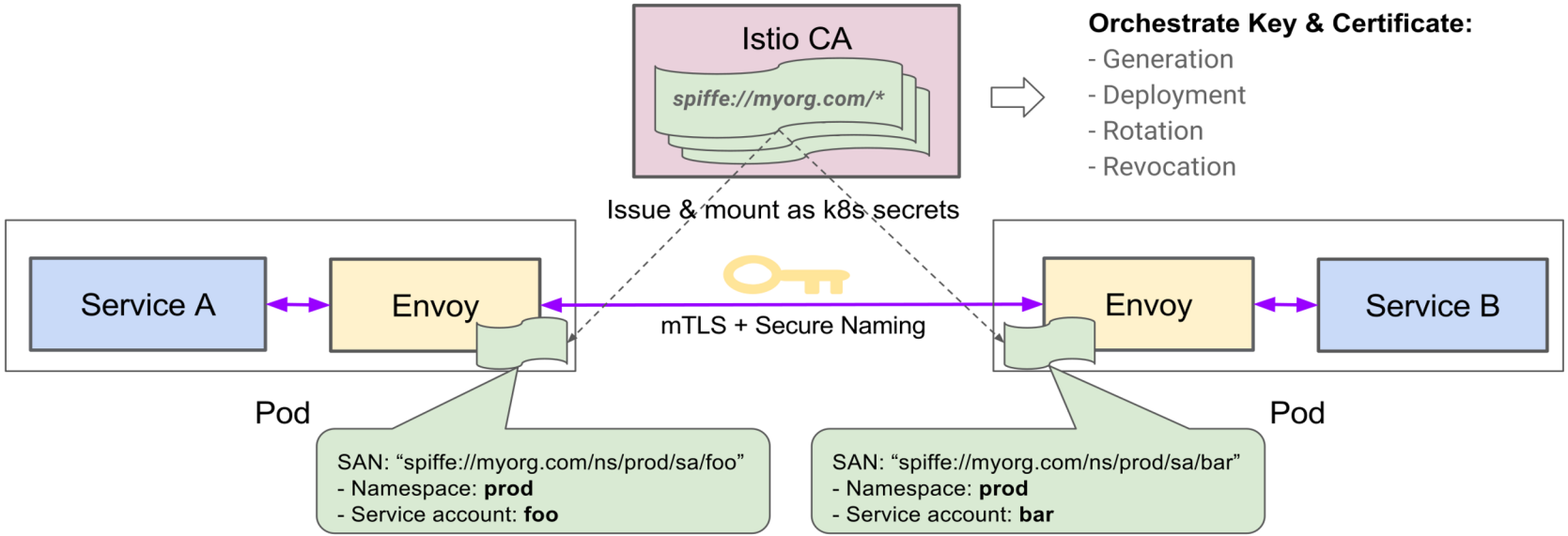
Dashboard for Metrics

Request Tracing

Securing Microservices

- Verifiable identity
- Secure naming / addressing
- Traffic encryption
- Revocation

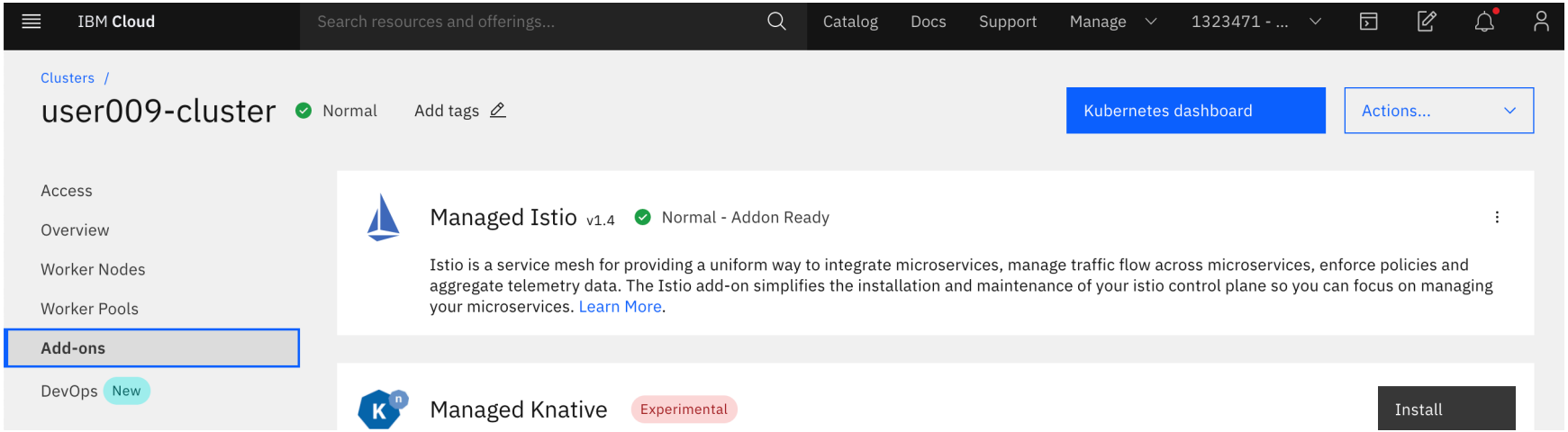
Istio Citadel



Resources

Istio documentation
https://istio.io/docs/

Istio in IBM products
https://www.ibm.com/cloud/info/istio



Lab

https://ibm.biz/BdzP4i