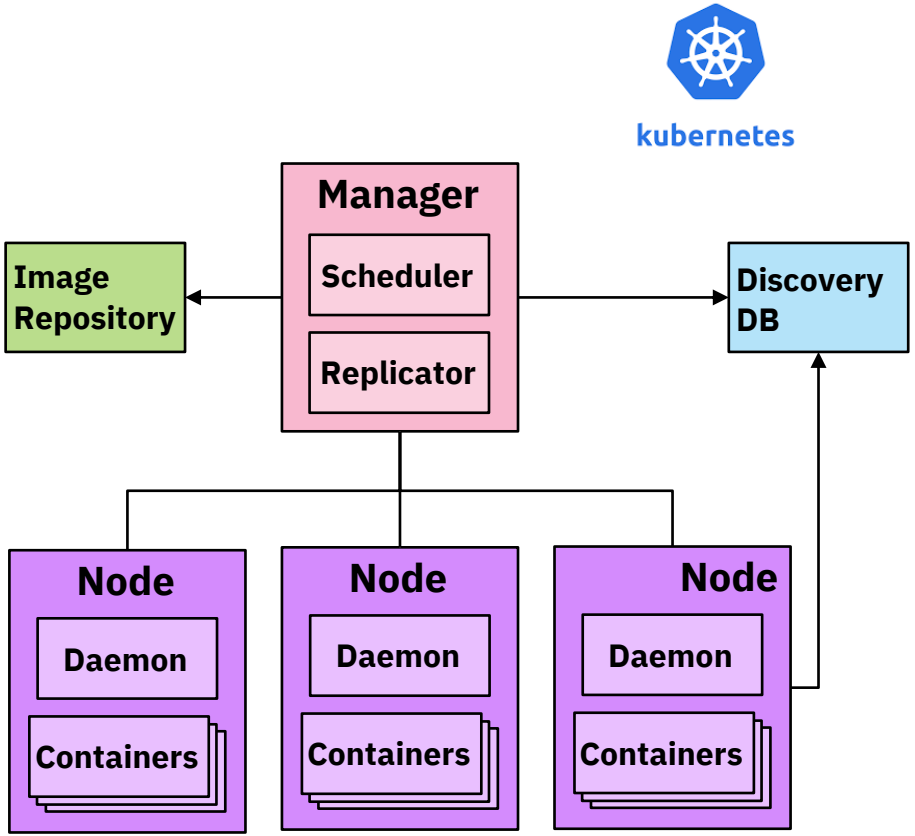
Introduction to Istio

@remkohdev
Remko de Knikker
Developer Advocate

Kubernetes

Benefits:

- Automated scheduling and scaling
- Zero downtime deployments
- High availability and fault tolerance
- A/B deployments



Microservices and 12-Factor

I. Codebase
II. Dependencies
III. Config
IV. Backing Services
V. Build, Release, Run
VI. Processes
VII. Port Binding
VIII. Concurrency
IX. Disposability
X. Dev/Prod Parity
XI. Logs
XII. Admin processes

Microservices Challenges

Traffic Management
- How to do canary testing?
- How to do A/B testing?

Performance, Scalability and Resilience
- How to implement circuit breakers?
- How to test faults in the system?
- How to set rate limits for each service?

Telemetry and Observability
- How to trace a request through my system of multiple services
- How to get centralized logging in a distributed deployment?

Policies and Security
- How to apply policies across services?
- How to manage certificates and trust, and TLS traffic?

Deployment
- How to deploy to multi-cluster environment?
- How to manage complex deployment strategies?

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

← How to solve these problems in a standard way **without changing application code**

What is a service mesh?

A **service mesh** is a network of microservices and the interactions between them. Unlike other systems for managing this communication, a service mesh is a dedicated infrastructure layer.

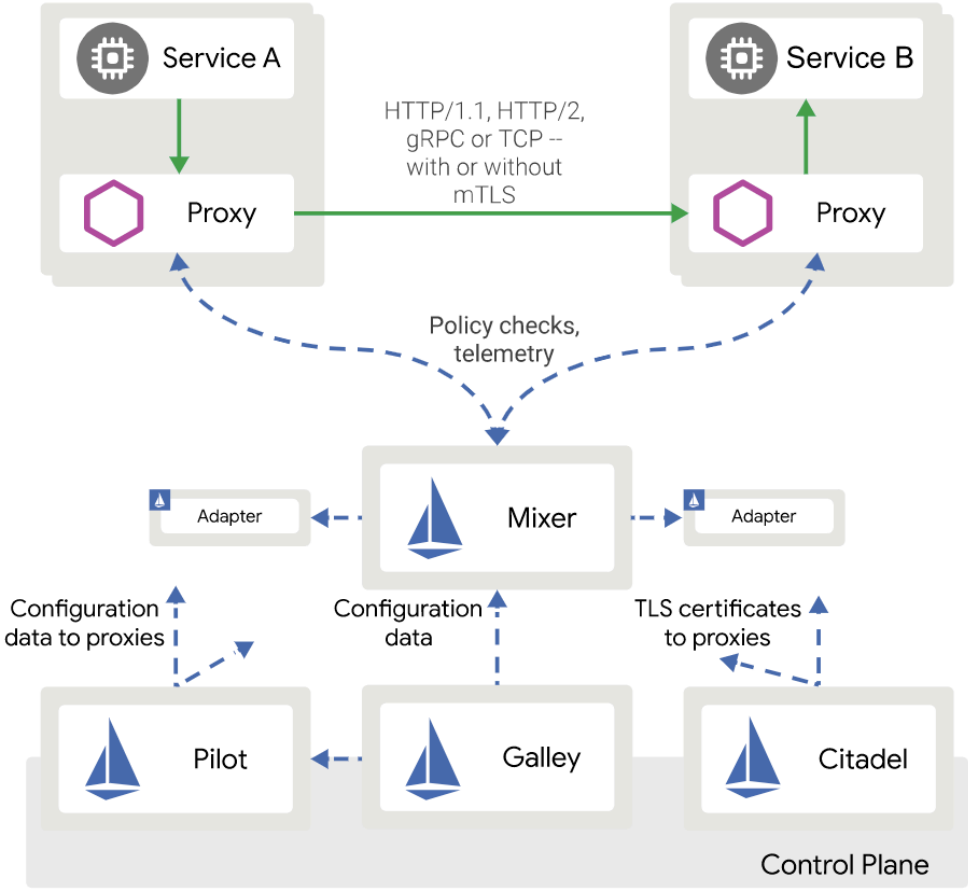
Its requirements can include discovery, load balancing, failure recovery, metrics, and monitoring as well as more complex operational requirements, like A/B testing, canary rollouts, rate limiting, access control, and end-to-end authentication.

Istio

Istio Architecture

The **data plane** is a set of intelligent proxies (Envoy) deployed as sidecars, which mediate and control all network communication between microservices along with Mixer, a general-purpose policy and telemetry hub.

The **control plane** manages and configures the proxies to route traffic. Additionally, the control plane configures Mixers to enforce policies and collect telemetry.



Istio Architecture

Envoy
- L7 Proxy sidecar,
- Mediates all inbound and outbound traffic,

Mixer
- Enforces access control and usage policies,
- Collects telemetry data from the Envoy proxy sidecar on each request.
- Platform-independent, plug-in model, to interface with a variety of host environments and infrastructure backends.

Pilot
- “Drives” the service mesh by providing service discovery across platforms.
- Traffic management capabilities for intelligent routing (A/B tests, canary rollouts).
- Resilience (timeouts, retries, circuit-breakers and fault injection).
- Coverts that to low-level routing rules, and distributes those to sidecar containers at runtime.

Citadel
- Service-to-service and end-user authentication.
- Encrypt traffic.
- Key and certificate management, distributed as Kubernetes Secrets.

Galley
- Configuration validation, ingestion, processing and distribution, from underlying Kubernetes.

Envoy

Istio uses an extended version of the Envoy proxy, https://www.envoyproxy.io/. Envoy is an L7 proxy for the application layer, layer 7 or L7 of the Open Systems Interconnection (OSI) model.

Built-in Envoy features:

- Dynamic service discovery
- Load balancing
- TLS termination
- HTTP/2 and gRPC proxies
- Circuit breakers
- Health checks
- Staged rollouts with %-based traffic split
- Fault injection
- Rich metrics

Mixer

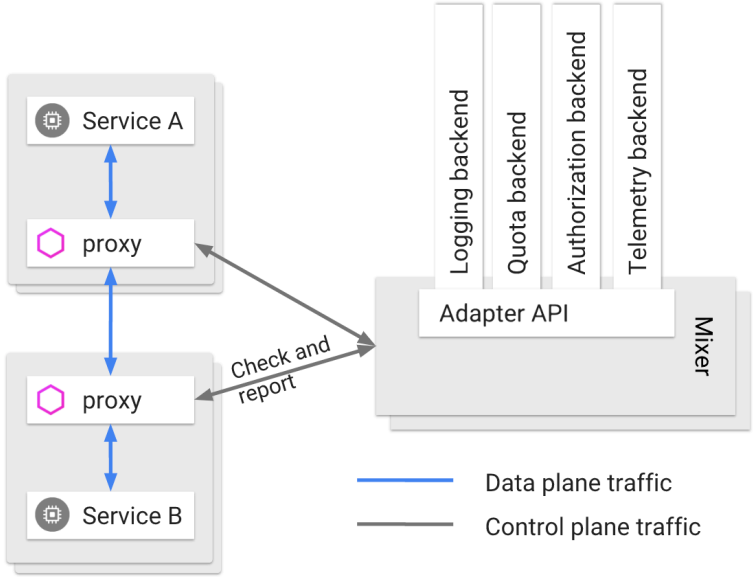
The Envoy sidecar calls Mixer before each request to perform precondition checks, and after each request to report telemetry. One of its key functions is to abstract away the details of different backend systems.

Adapters are individual plug-ins as Go packages.

Templates are used to send data to individual adapters.

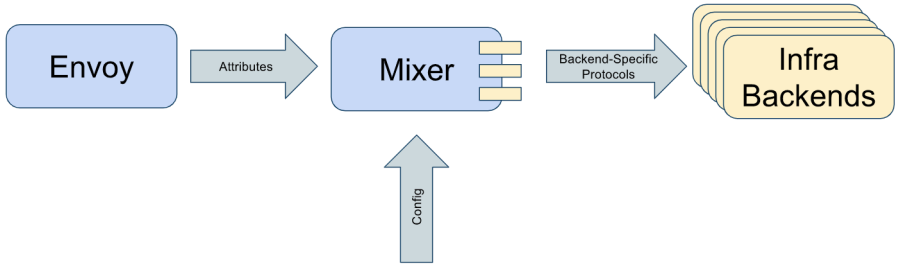
Instances control which data is delivered to individual adapters.

Rules tell Mixer which instances to send to which handler and when.



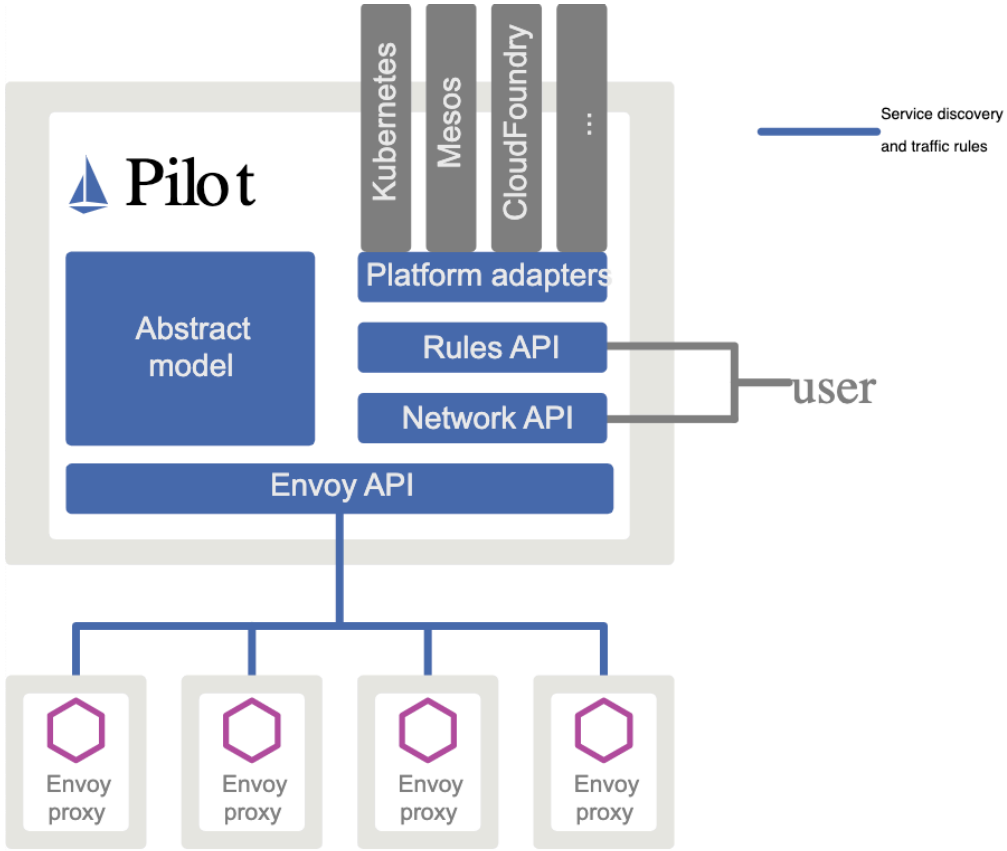
Mixer adapters:

- App Identity and Access
- Datadog
- Fluentd
- Kubernetes env
- Memory quota
- Redis quota
- Stdio
- Zipkin
- Prometheus
- Heapster
- etc



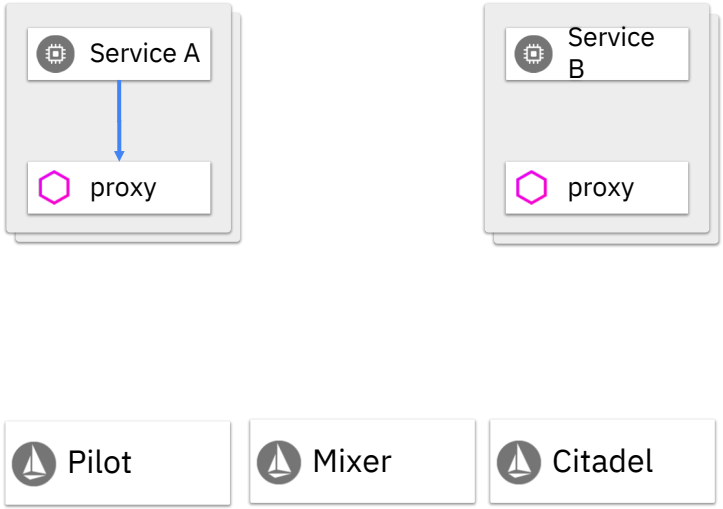
Pilot

Pilot maintains an **abstract model** of all the services in the mesh, to generate appropriate Envoy-specific configurations to let Envoy proxies know about one another in the mesh through the Envoy API. Use the Traffic Management API to instruct Pilot to refine the Envoy configuration.



Lifecycle of a request

Lifecycle of a request

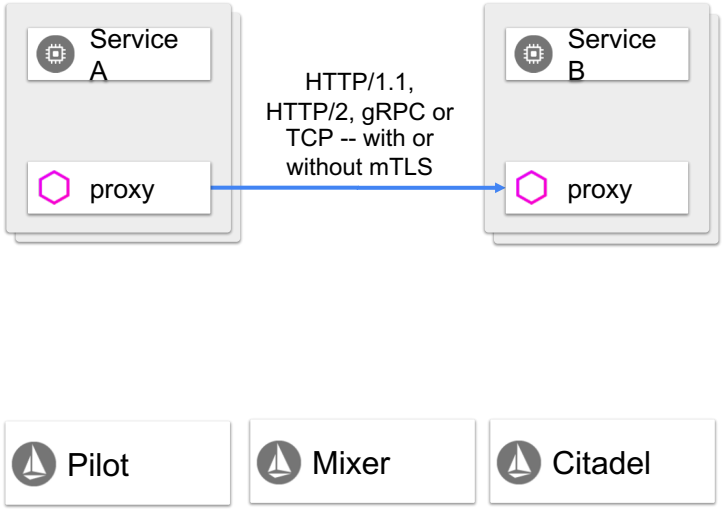


Service A places a call to http://service-b.

Client-side Envoy intercepts the call.

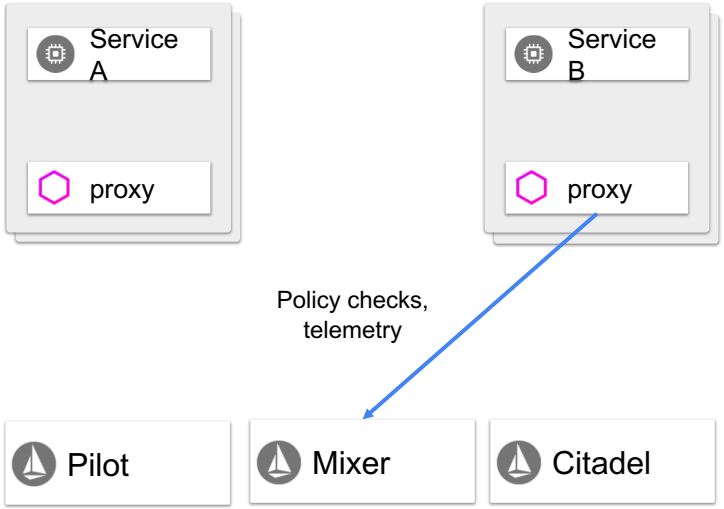
Envoy consults config (previously received from Pilot) to know how/where to route call to service B (taking into account service discovery, load balancing, and routing rules), forwards the call to the right destination.

Lifecycle of a request



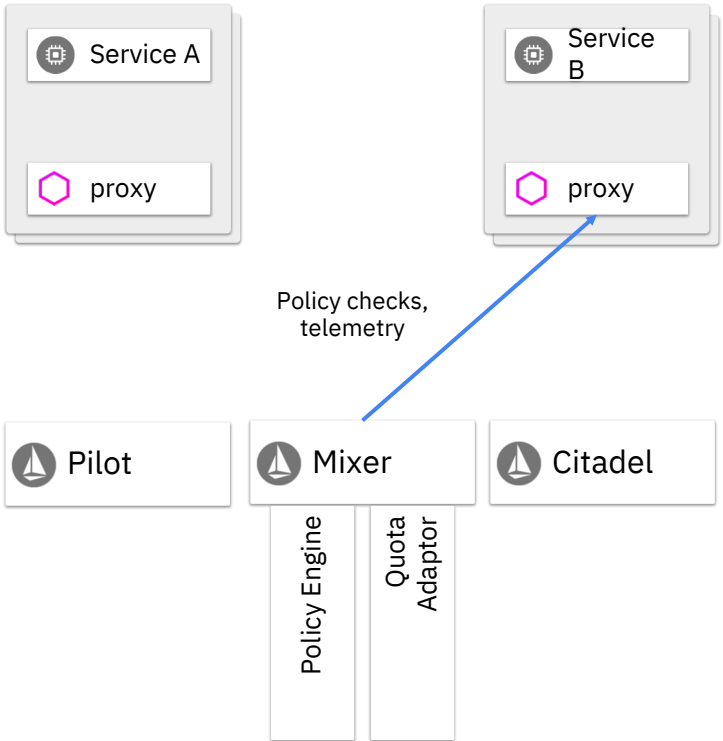
Envoy forwards request to appropriate instance of service B. There, the Envoy proxy deployed with the service intercepts the call.

Lifecycle of a request



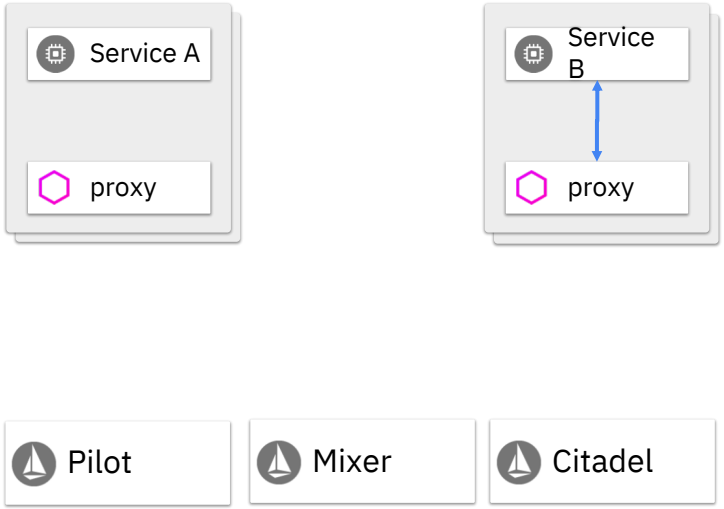
Server-side Envoy checks with Mixer to validate that call should be allowed (ACL check, quota check, etc).

Lifecycle of a request



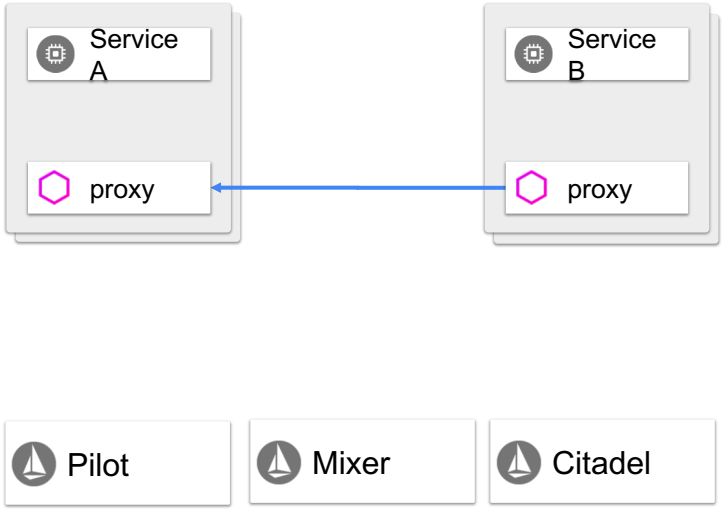
Mixer checks with appropriate adaptors (policy engine, quota adaptor) to verify that the call can proceed and returns true/false to Envoy

Lifecycle of a request



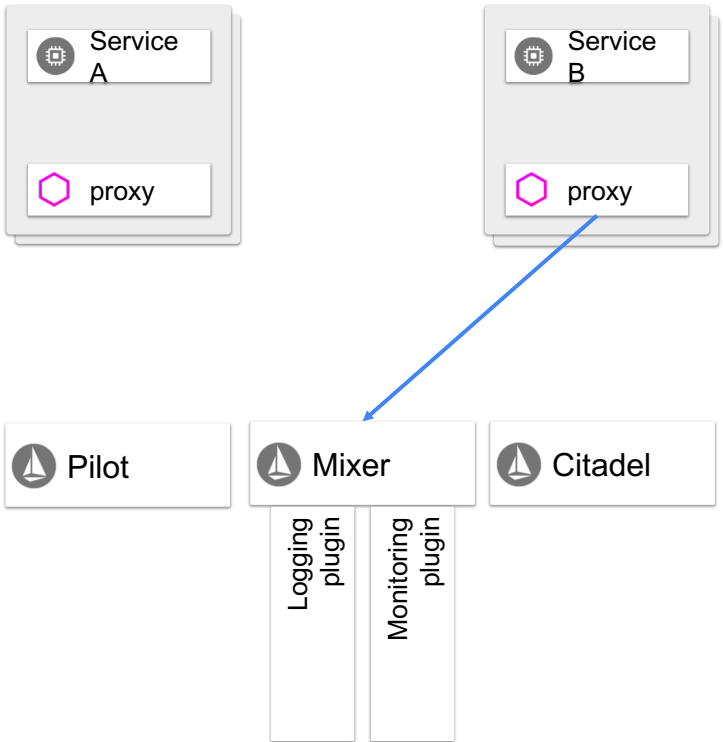
Server-side Envoy forwards request to service B, which process request and returns response

Lifecycle of a request



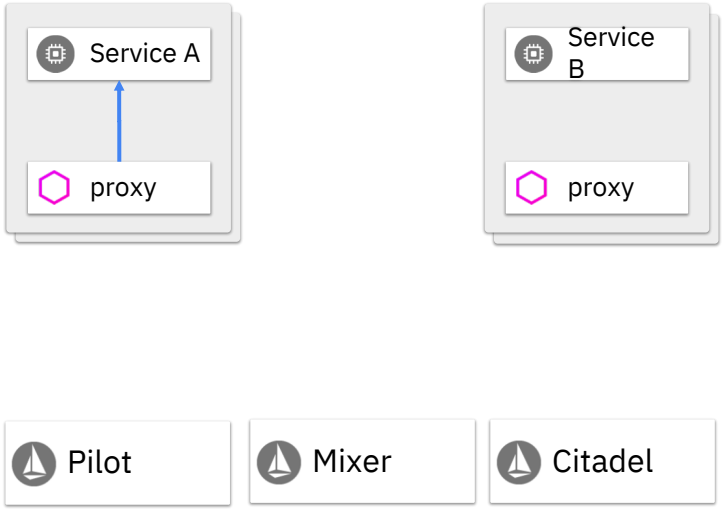
Envoy forwards response to the original caller, where response is intercepted by Envoy on the caller side.

Lifecycle of a request



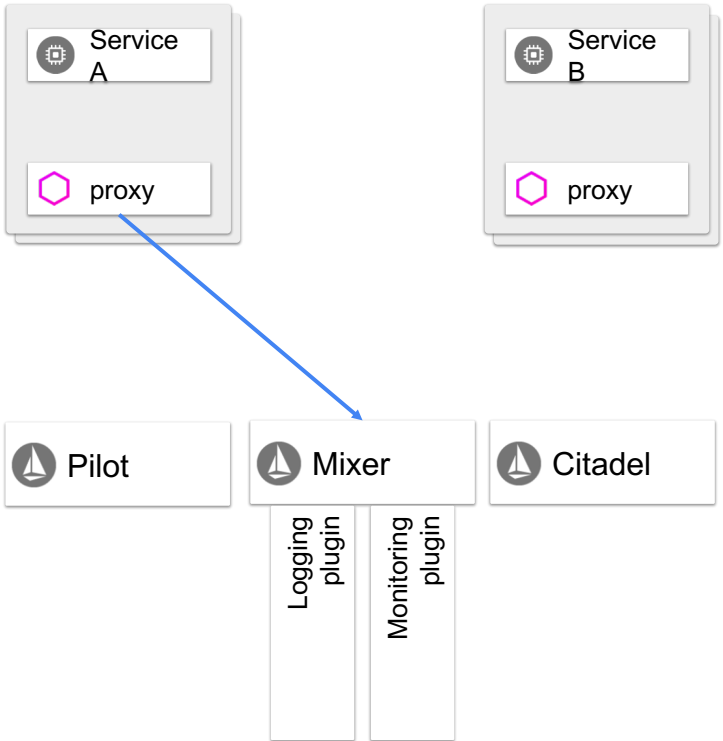
Envoy reports telemetry to Mixer, which in turn notifies appropriate plugins

Lifecycle of a request



Client-side Envoy forwards response to original caller.

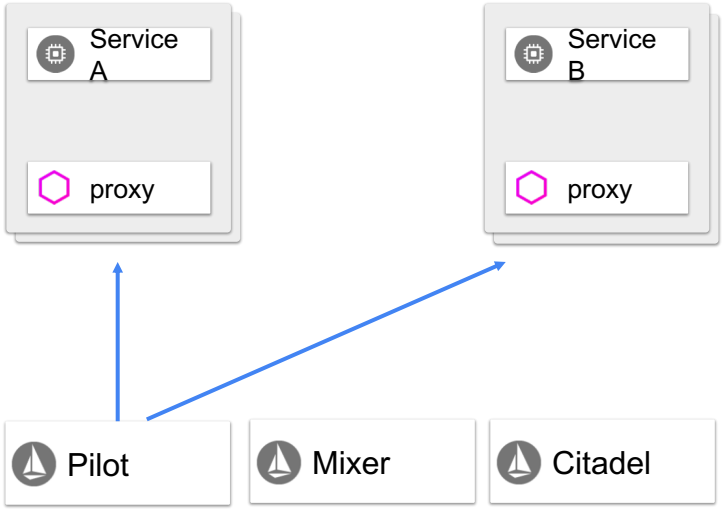
Lifecycle of a request



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

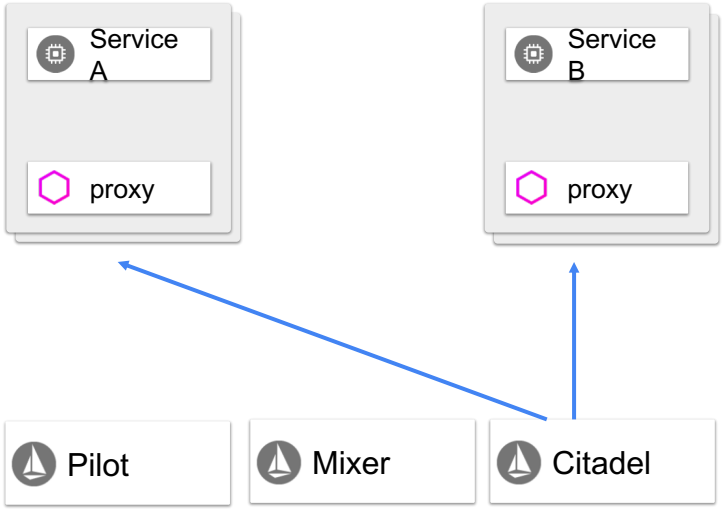
At Anytime

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



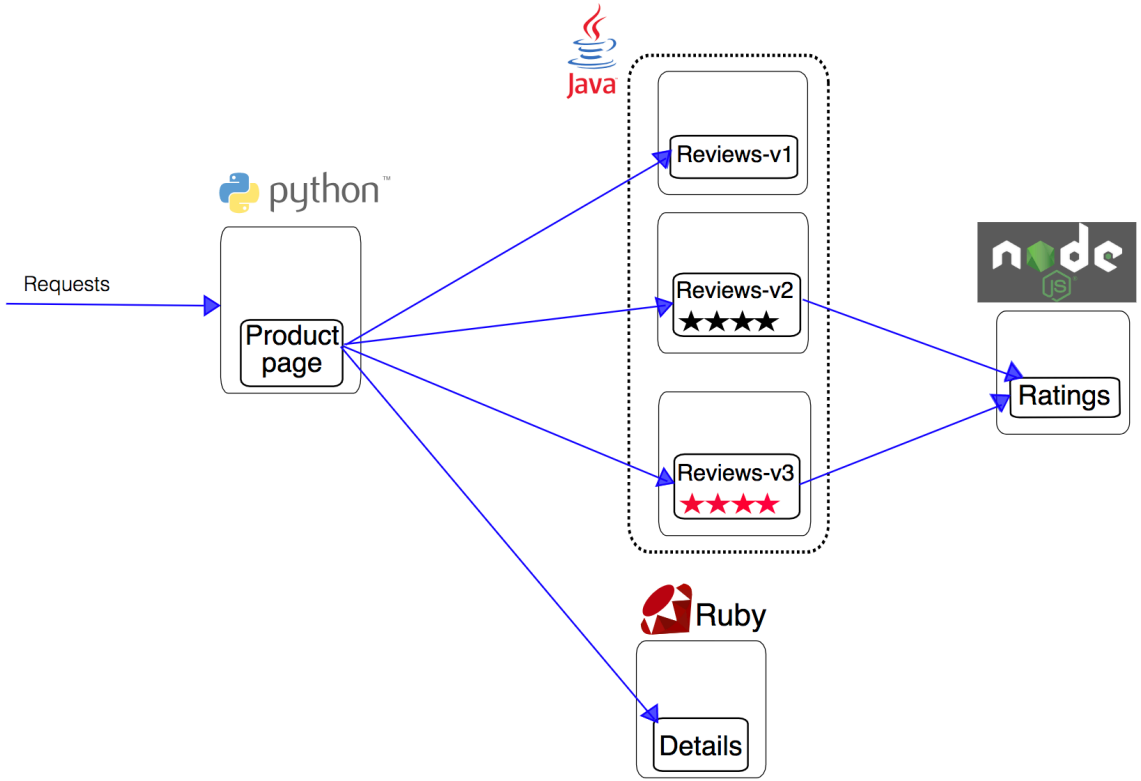
Citadel distributes keys and certificates as Kubernetes Secrets available to sidecar containers

Envoy 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

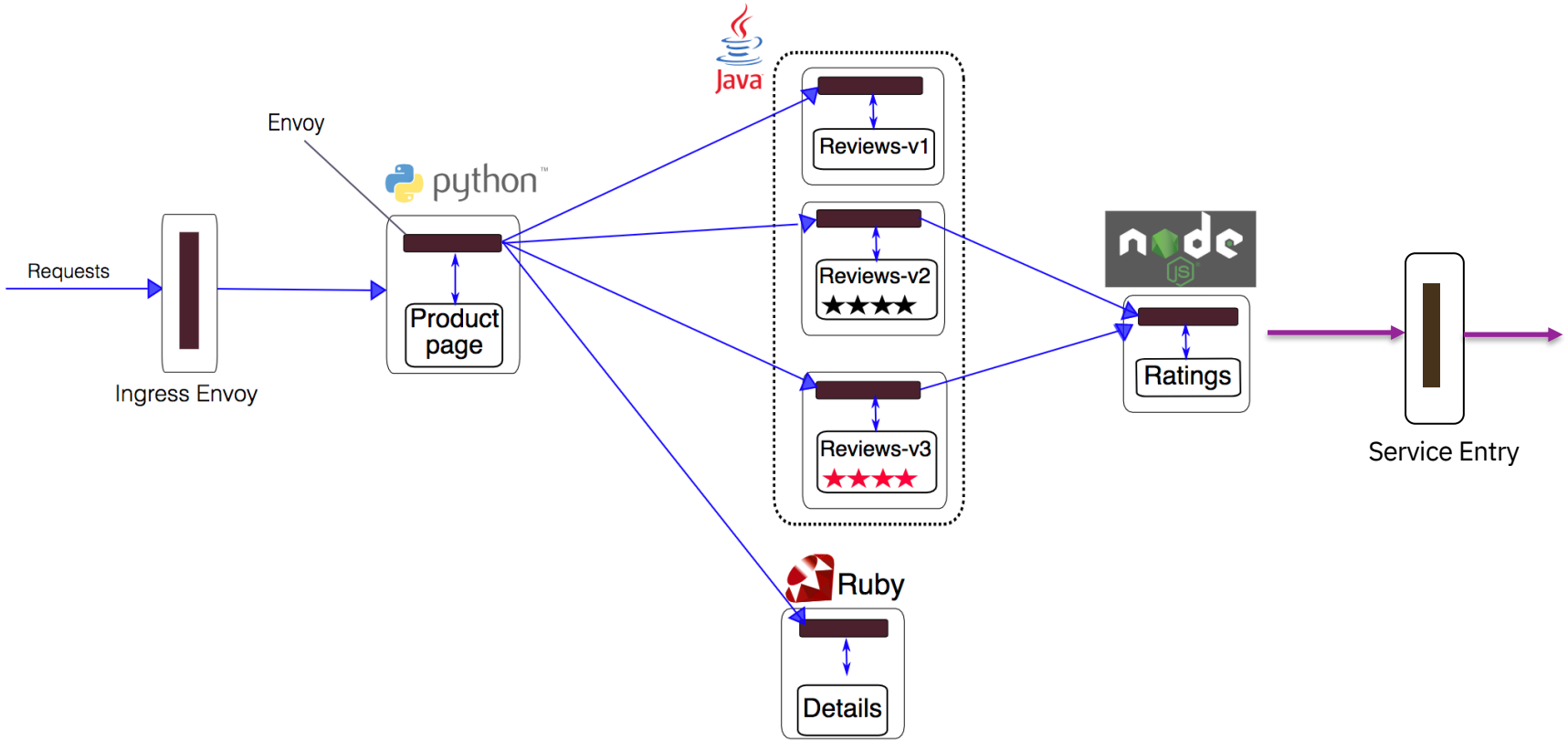
- Istio modifies the individual pods and not deployments

Bookinfo Sample Application (without Istio)



Bookinfo Application without Istio

Bookinfo Sample Application (with Istio)



Bookinfo Application

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: ratings
      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
```

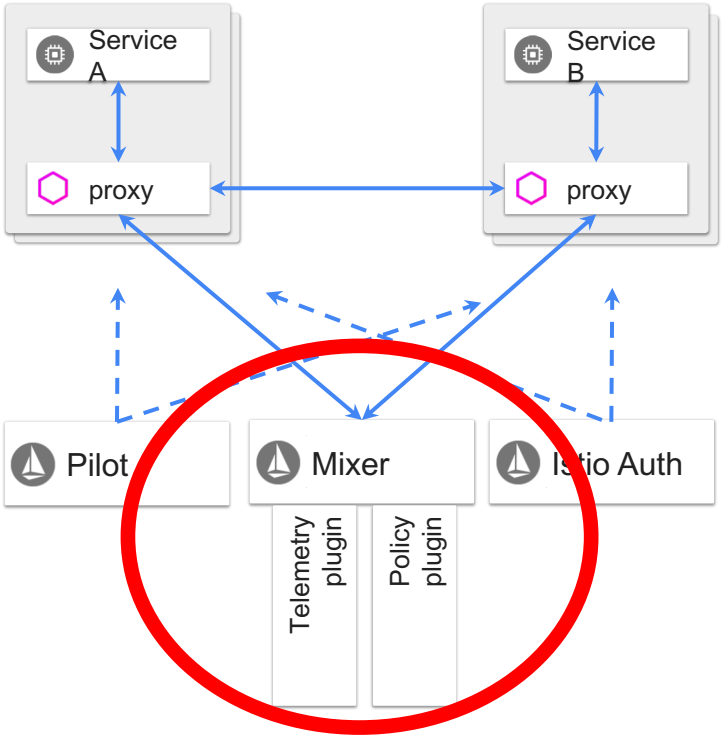
jason: Return with Error 500

Istio Gateway

Traffic Management

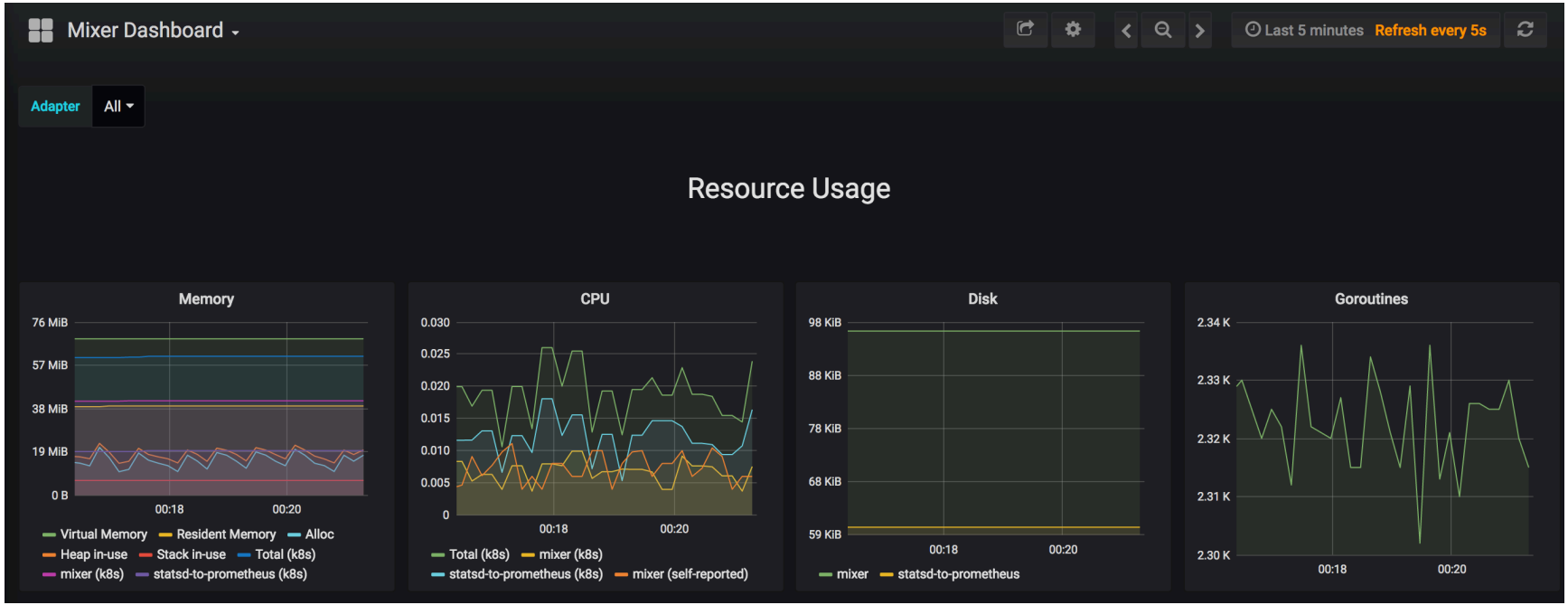
Telemetry

Telemetry



TLS certs to Envoys

Telemetry


Mixer Dashboard
Last 5 minutes
Refresh every 5s
Adapter
All
Resource Usage
Memory
76 MiB
57 MiB
38 MiB
19 MiB
0 B
00:18
00:20
Virtual Memory
Resident Memory
Alloc
Heap in-use
Stack in-use
Total (k8s)
mixer (k8s)
statsd-to-prometheus (k8s)
CPU
0.030
0.025
0.020
0.015
0.010
0.005
0
00:18
00:20
Total (k8s)
mixer (k8s)
statsd-to-prometheus (k8s)
mixer (self-reported)
Disk
98 KiB
88 KiB
78 KiB
68 KiB
59 KiB
00:18
00:20
mixer
statsd-to-prometheus
Goroutines
2.34 K
2.33 K
2.32 K
2.31 K
2.30 K
00:18
00:20


Prometheus


kiali

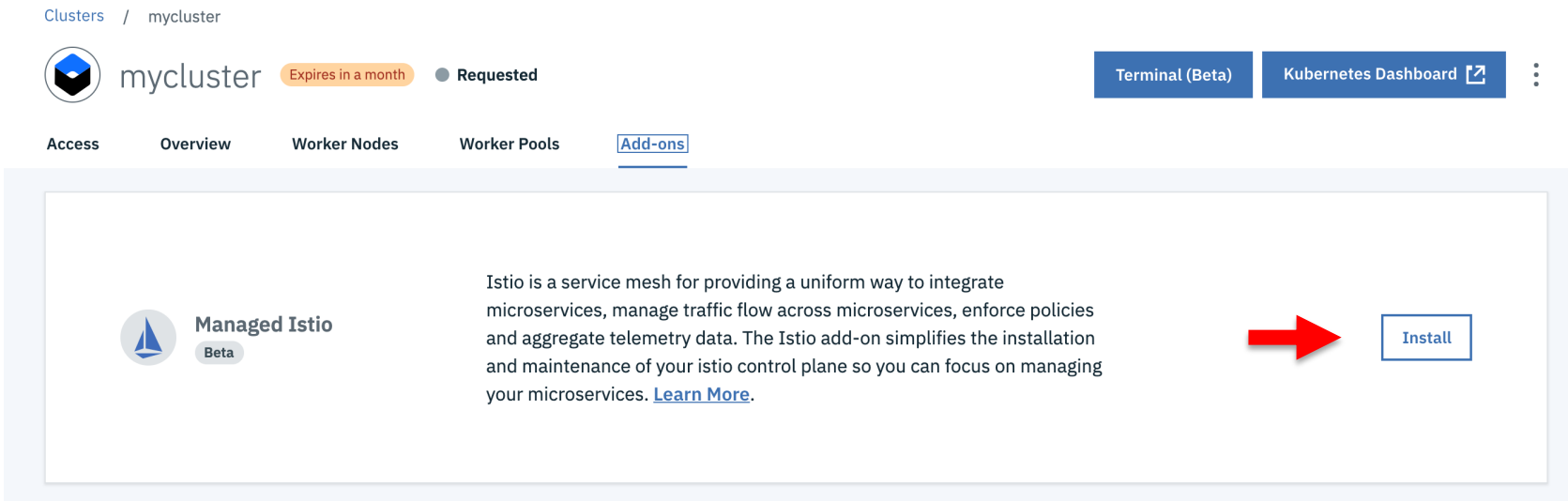

Grafana


JAEGER

Security

https://istio.io/docs/

https://www.ibm.com/cloud/info/istio



Labs

http://ibm.biz/SSTIstioLab


IBM