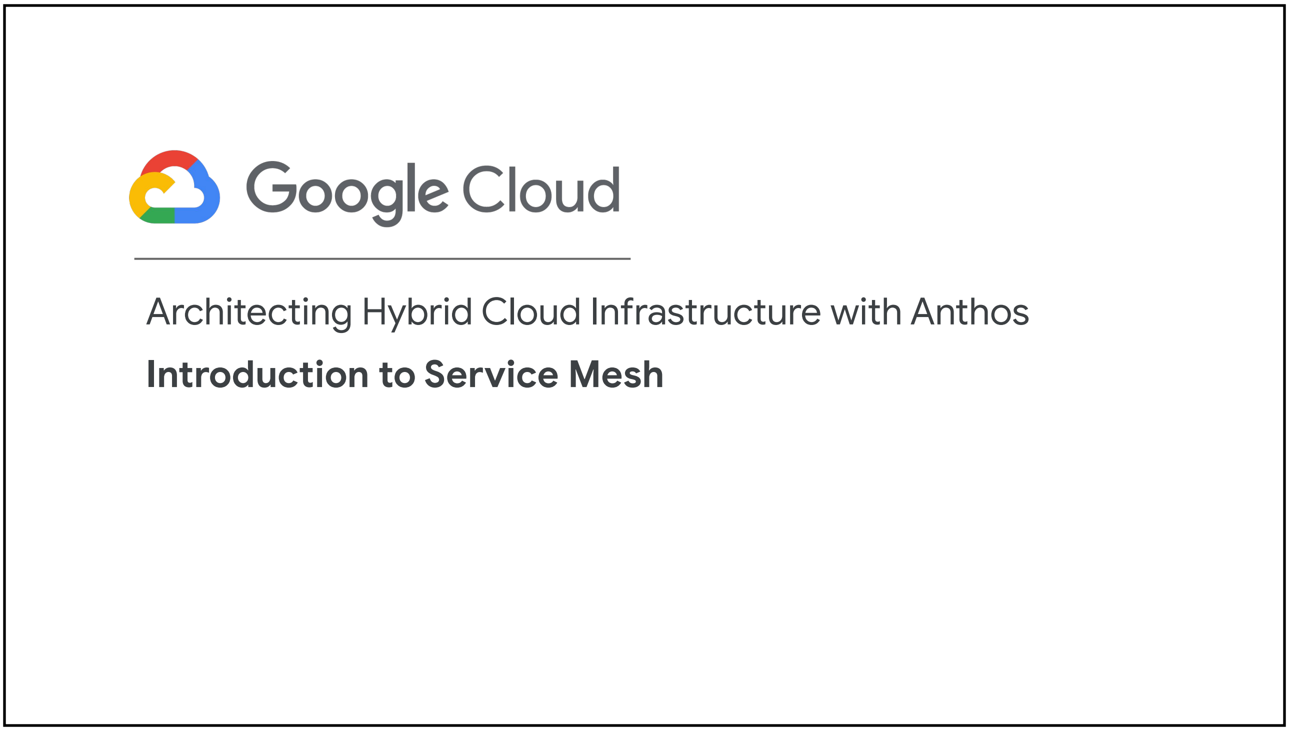
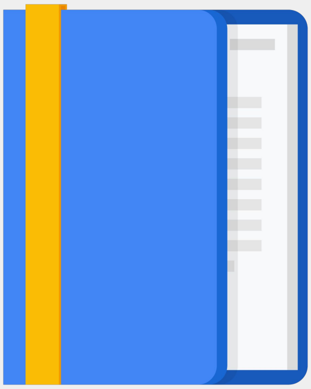


Google Cloud

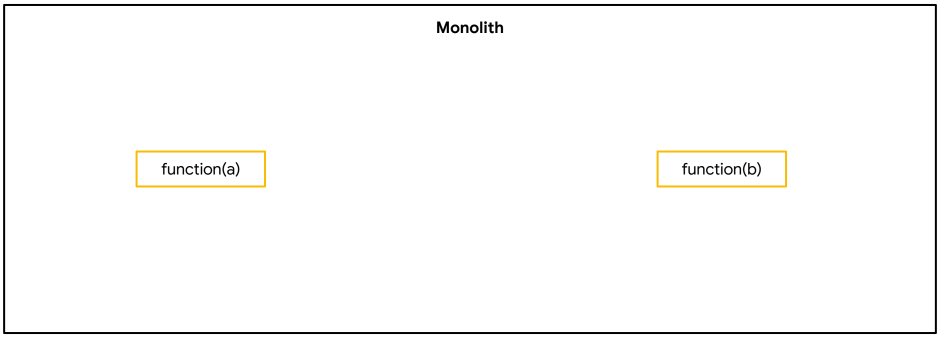
Architecting Hybrid Cloud Infrastructure with Anthos

Introduction to Service Mesh

Agenda



- **Service Mesh**
- Istio overview
- Traffic in an Istio Mesh
- Traffic Shaping Intro
- Anthos Service Mesh



Monolith

function(a)

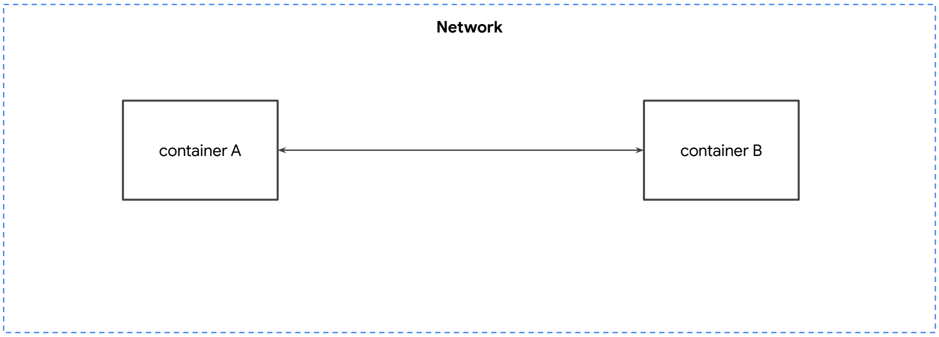
function(b)

container

function(a)

container

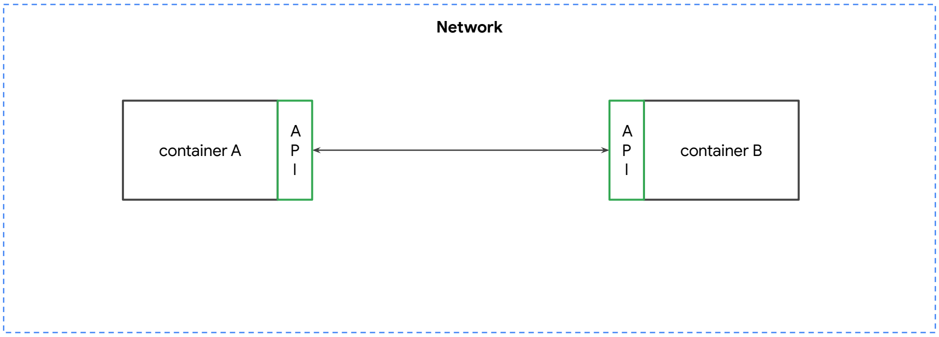
function(b)



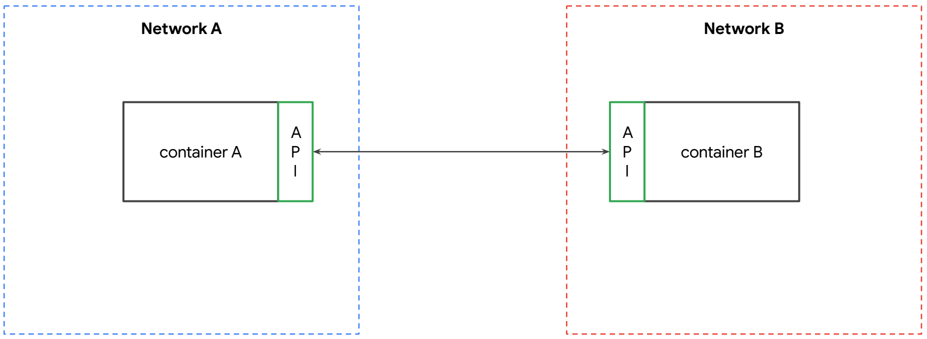
Network

container A

container B


Network
container A
A
P
I
A
P
I
container B

Zero trust network



Network A

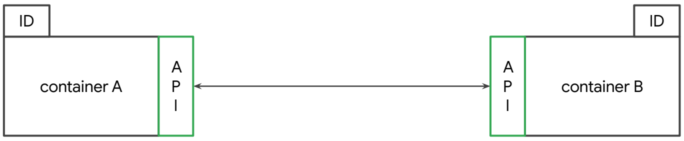
container A

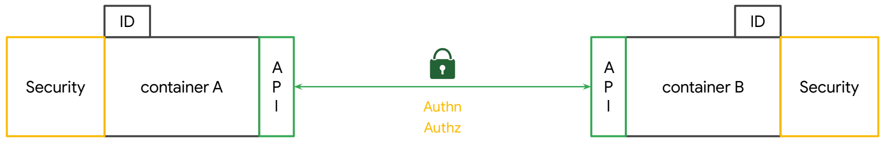
A
P
I

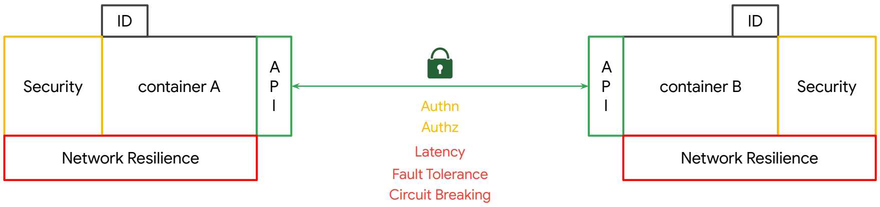
Network B

A
P
I

container B


ID
container A
A
P
I
A
P
I
container B
ID


ID
Security
container A
A
P
I
Authn
Authz
ID
A
P
I
container B
Security



ID

Security

container A

A
P
I

Network Resilience

Authn
Authz

Latency
Fault Tolerance
Circuit Breaking

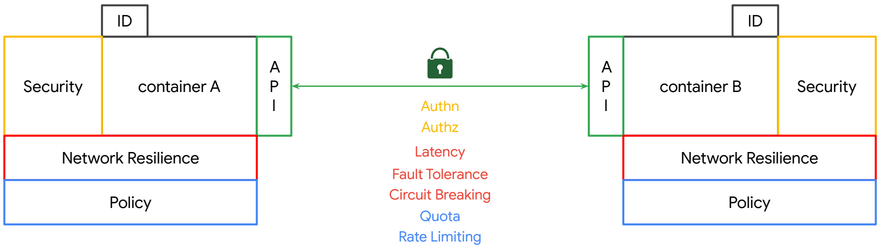
ID

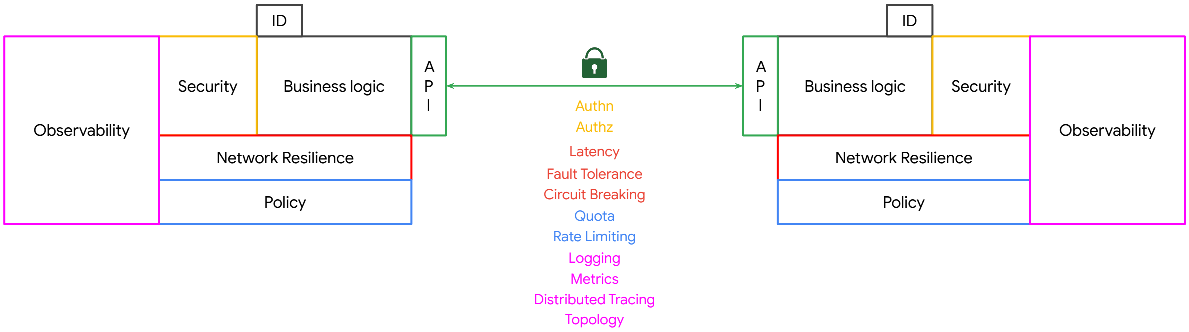
A
P
I

container B

Security

Network Resilience


ID
Security
container A
A
P
I
Network Resilience
Policy
Authn
Authz
Latency
Fault Tolerance
Circuit Breaking
Quota
Rate Limiting
ID
A
P
I
container B
Security
Network Resilience
Policy



ID

Observability

Security

Business logic

A
P
I

Network Resilience

Policy

Authn
Authz
Latency
Fault Tolerance
Circuit Breaking
Quota
Rate Limiting
Logging
Metrics
Distributed Tracing
Topology

ID

A
P
I

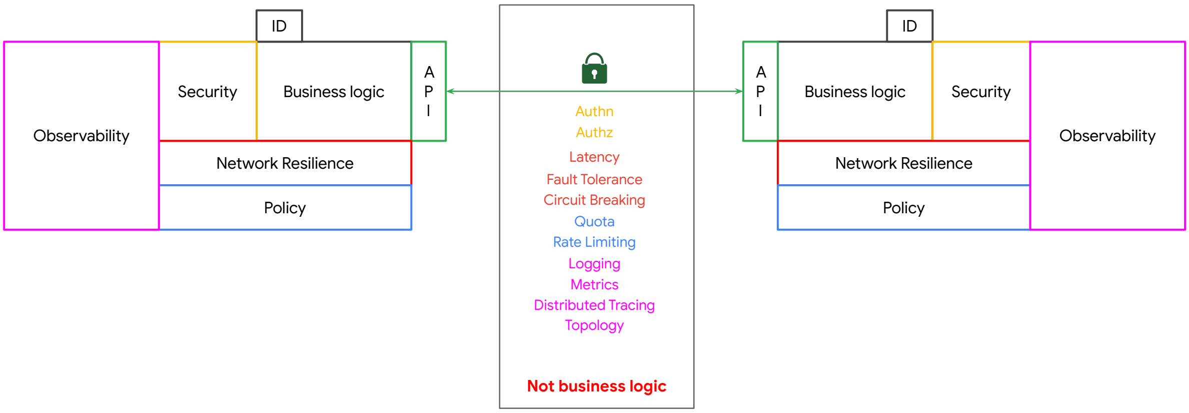
Business logic

Security

Observability

Network Resilience

Policy



ID

Observability

Security

Business logic

API

Network Resilience

Policy

Authn
Authz
Latency
Fault Tolerance
Circuit Breaking
Quota
Rate Limiting
Logging
Metrics
Distributed Tracing
Topology

Not business logic

ID

API

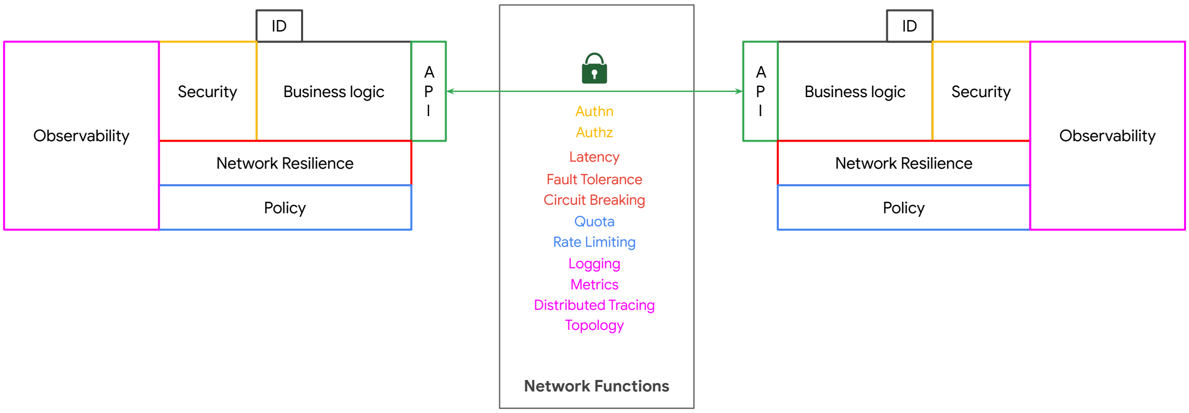
Business logic

Security

Observability

Network Resilience

Policy



ID

Observability

Security

Business logic

API

Network Resilience

Policy

Authn

Authz

Latency

Fault Tolerance

Circuit Breaking

Quota

Rate Limiting

Logging

Metrics

Distributed Tracing

Topology

Network Functions

ID

API

Business logic

Security

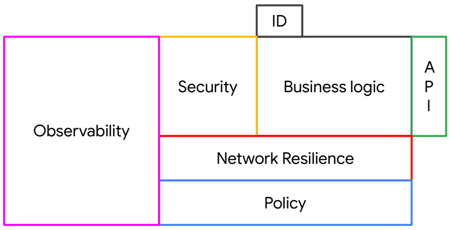
Observability

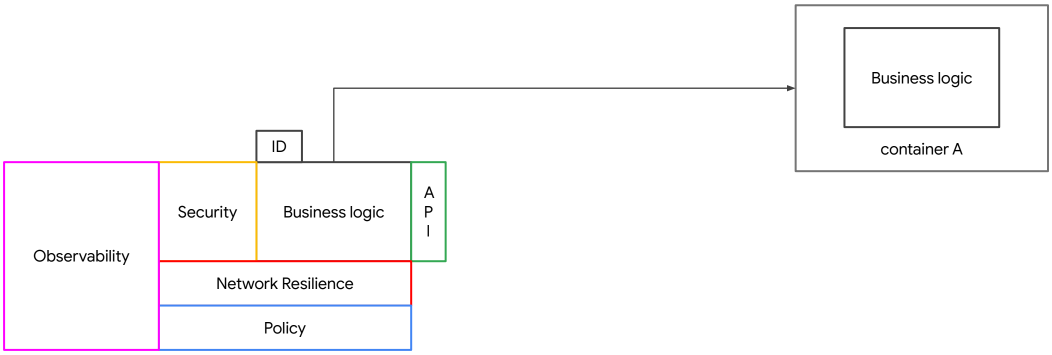
Network Resilience

Policy

Service Mesh

Separating **applications**
from **network functions**


ID
Observability
Security
Business logic
A
P
I
Network Resilience
Policy



Business logic

container A

ID

Observability

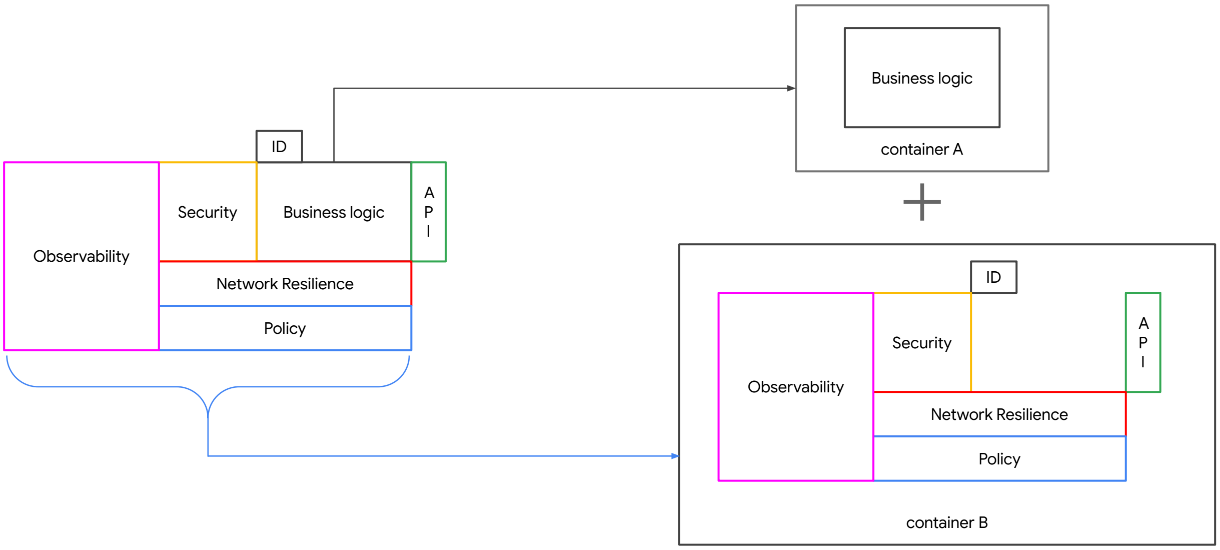
Security

Business logic

A
P
I

Network Resilience

Policy



ID

Observability

Security

Business logic

A
P
I

Network Resilience

Policy

Business logic

container A

ID

Observability

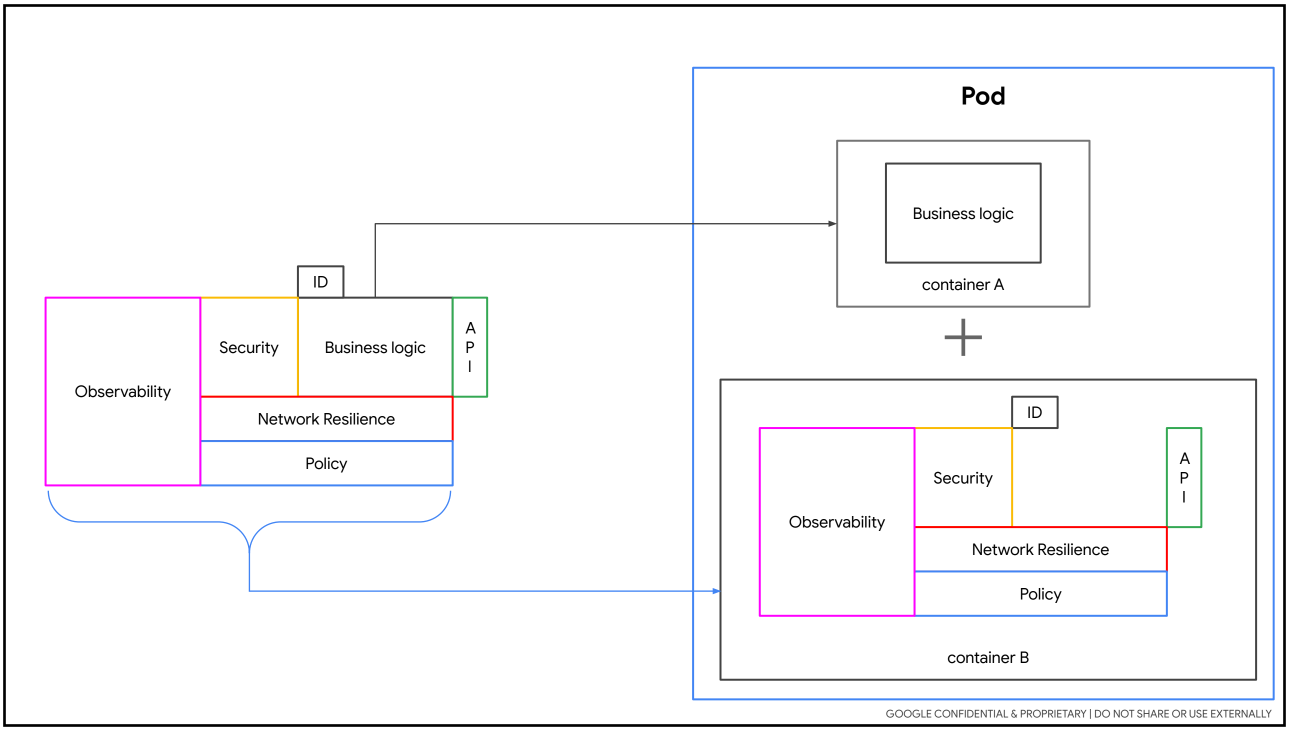
Security

A
P
I

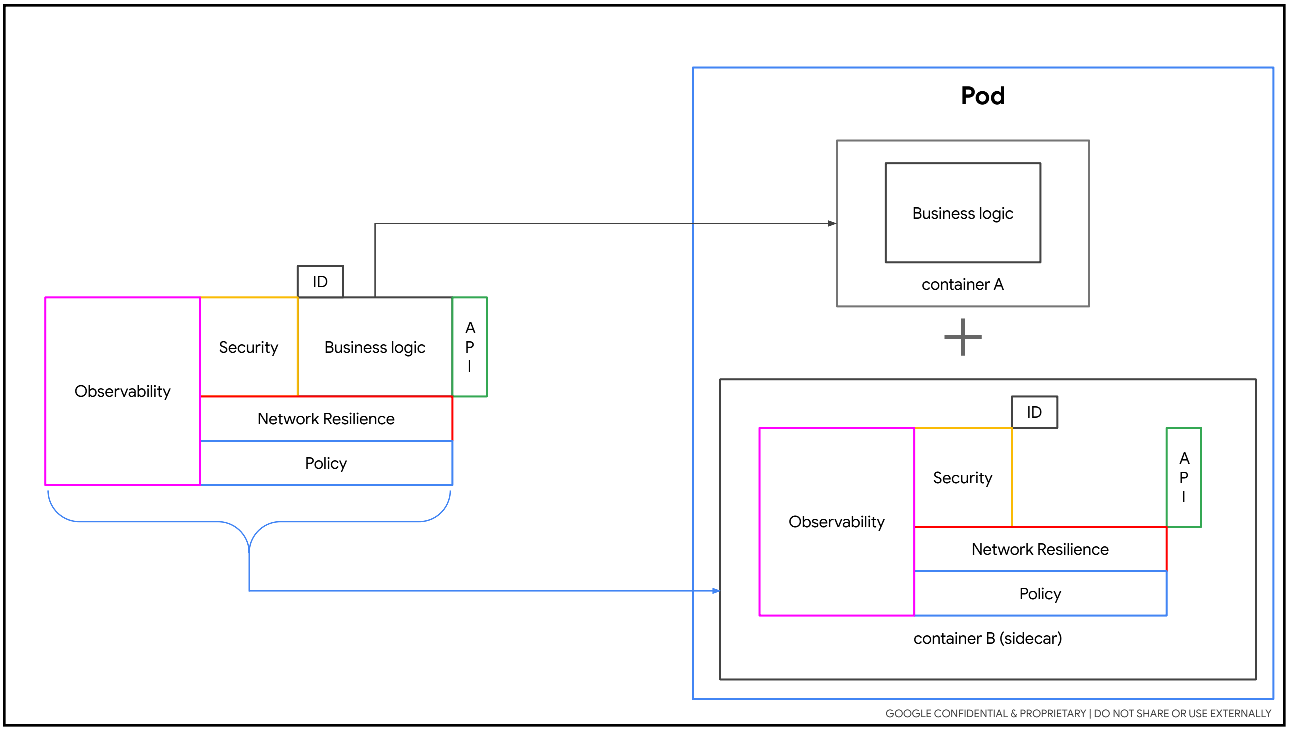
Network Resilience

Policy

container B



GOOGLE CONFIDENTIAL & PROPRIETARY | DO NOT SHARE OR USE EXTERNALLY



Observability

Security

ID

Business logic

A
P
I

Network Resilience

Policy

Pod

Business logic

container A

ID

Observability

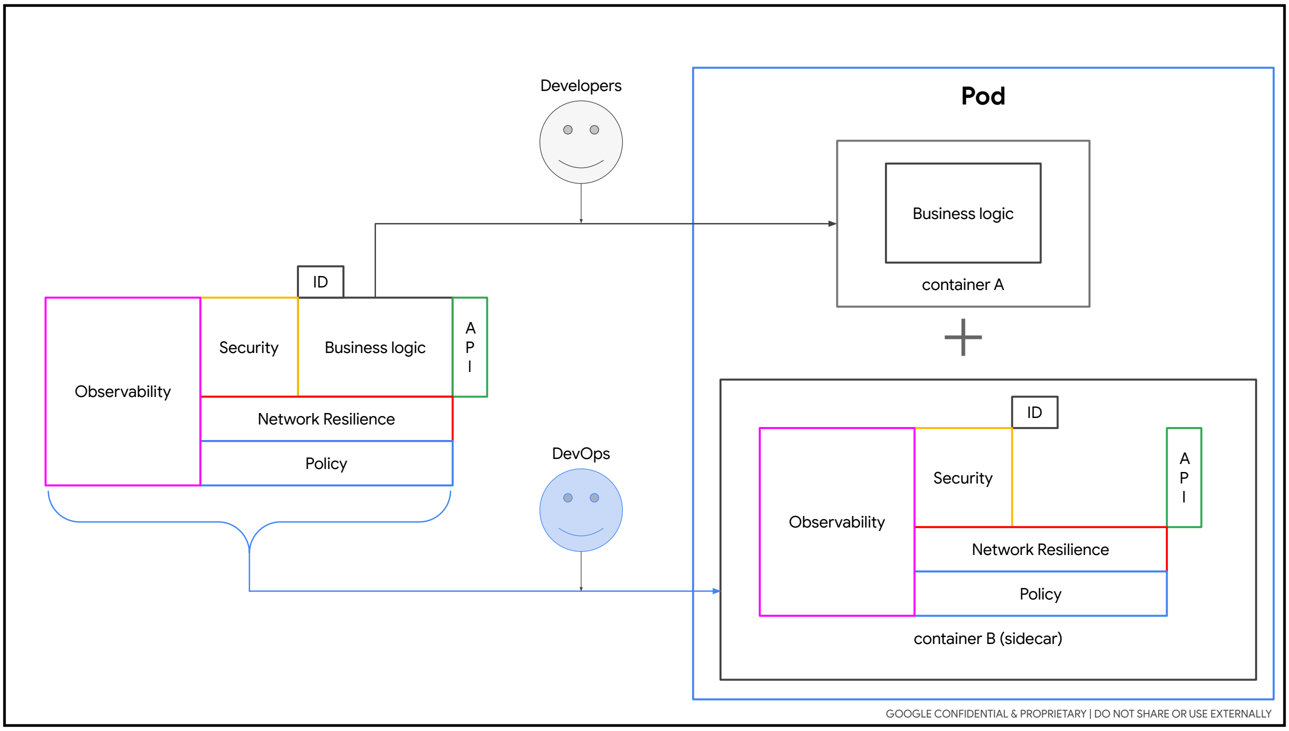
Security

A
P
I

Network Resilience

Policy

container B (sidecar)



Developers

ID

Observability

Security

Business logic

A
P
I

Network Resilience

Policy

DevOps

Pod

Business logic

container A

+

ID

Observability

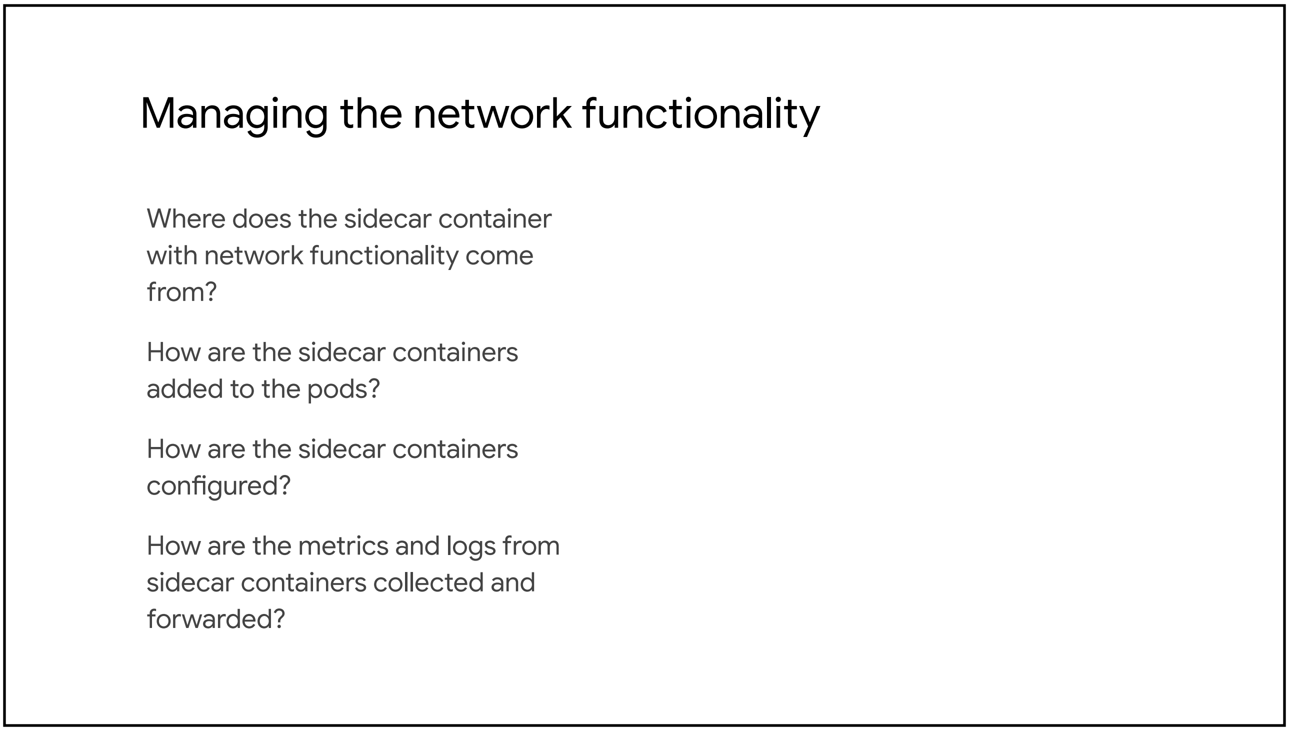
Security

A
P
I

Network Resilience

Policy

container B (sidecar)



Managing the network functionality

Where does the sidecar container with network functionality come from?

How are the sidecar containers added to the pods?

How are the sidecar containers configured?

How are the metrics and logs from sidecar containers collected and forwarded?

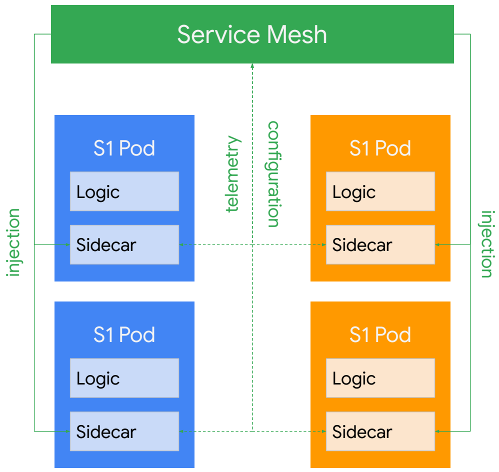
Managing the network functionality

Where does the sidecar container with network functionality come from?

How are the sidecar container added to the pods?

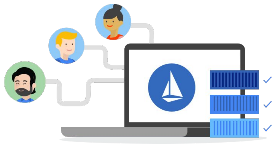
How are the sidecar containers configured?

How are the metrics and logs from sidecar containers collected and forwarded?

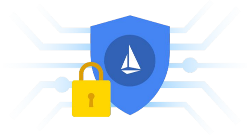


Service Mesh in details

Traffic Control

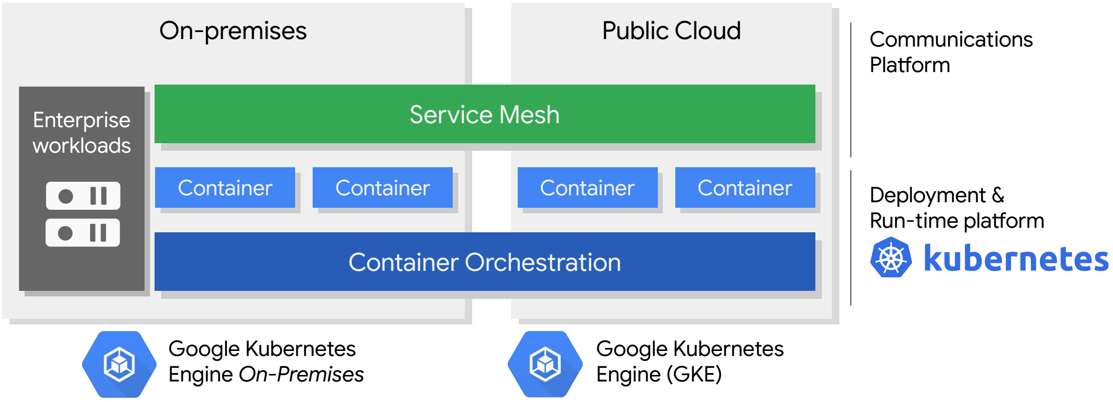


Observability



Security

Service Mesh in details



On-premises

Public Cloud

Enterprise workloads

Service Mesh

Container

Container

Container

Container

Container Orchestration

Communications Platform

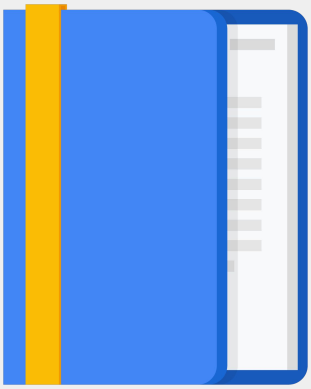
Deployment & Run-time platform

kubernetes

Google Kubernetes Engine *On-Premises*

Google Kubernetes Engine (GKE)

Agenda



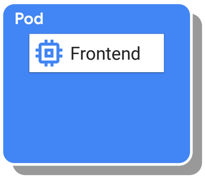
- Service Mesh
- **Istio overview**
- Traffic in an Istio Mesh
- Traffic Shaping Intro
- Anthos Service Mesh

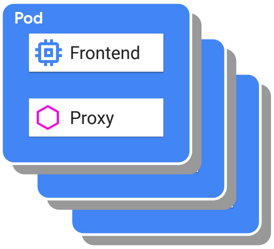
Google Cloud

What is Istio?

Istio is an open framework for connecting, securing, managing and monitoring services, even across environments

The sidecar model





kubernetes

```
spec:
  containers:
  - image: frontend:v2.0.17
```

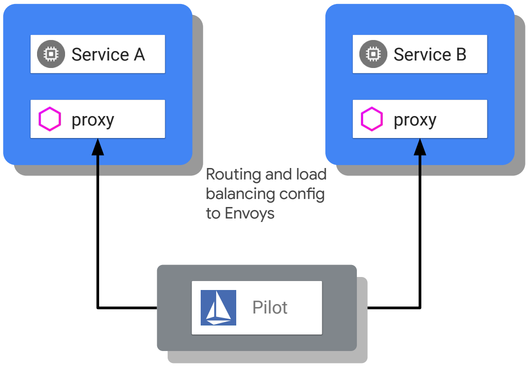
kubernetes

```
spec:
  containers:
  - image: frontend:v2.0.17
  - image: istio/proxy:v1.0
```

Pilot

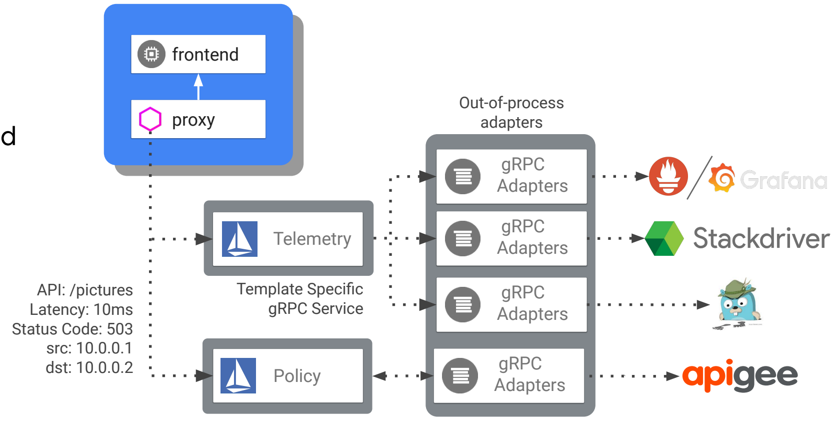
Manages the distributed proxies across the either environments, providing

- Service Discovery
- Traffic Management
- Intelligent Routing
- Resiliency



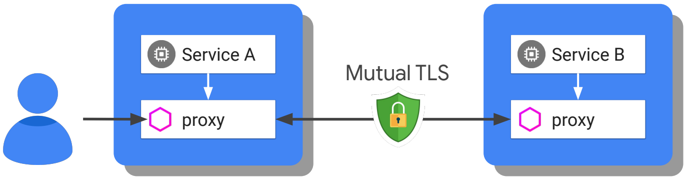
Mixer

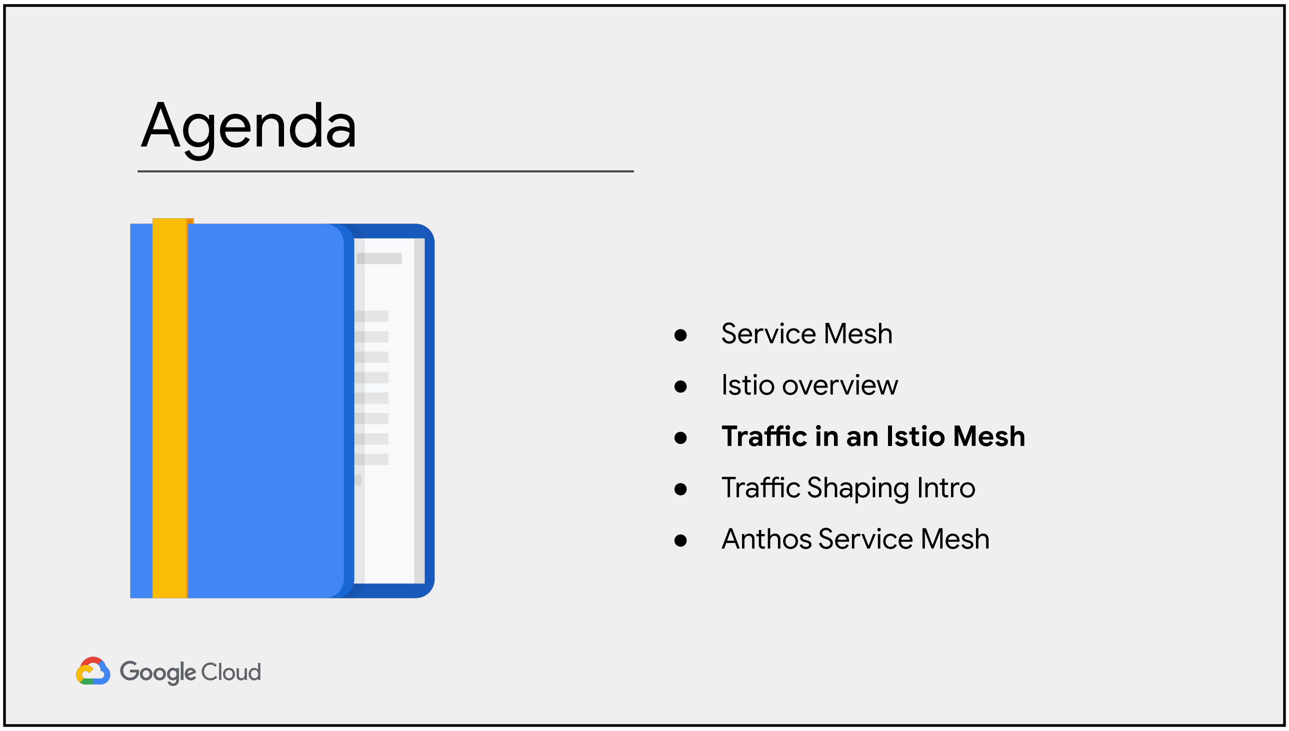
Mixer is a platform independent component. Send telemetry, logs, and traces to your system of choice



Citadel: Certificate Management

Built-in identity and certificate management which enables Strong service-to-service and end-user authentication and encryption



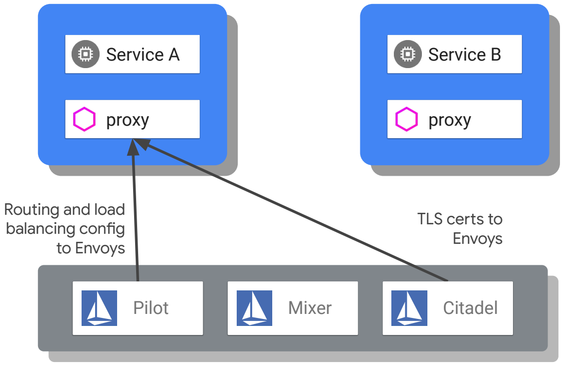


Agenda

- Service Mesh
- Istio overview
- **Traffic in an Istio Mesh**
- Traffic Shaping Intro
- Anthos Service Mesh

Google Cloud

Life of a request in the mesh

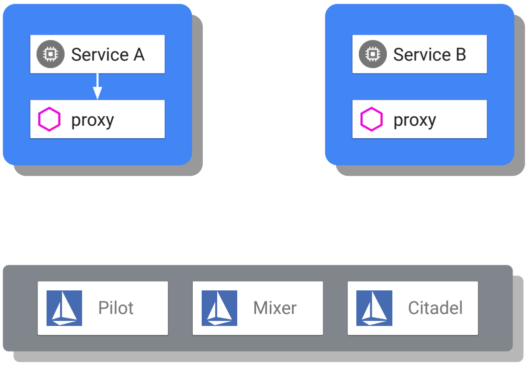


Service A comes up.

Envoy is deployed with it and fetches service information, routing, and configuration policy from Pilot.

If Citadel is being used, TLS certs are securely distributed as well

Life of a request in the mesh

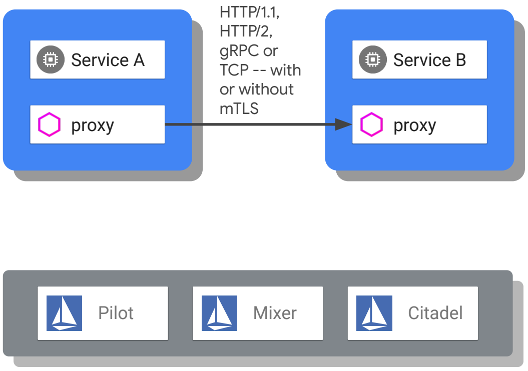


Service A places a call to service B

Client-side Envoy intercepts the call

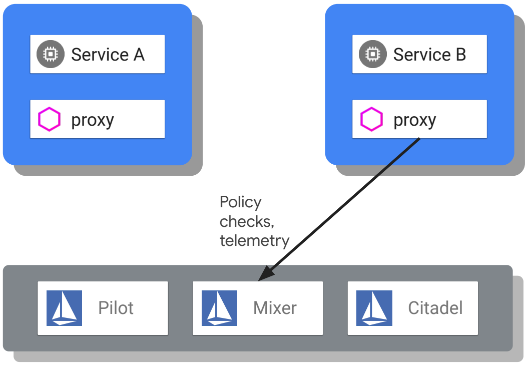
Envoy consults config to know how/where to route call to service B

Life of a request in the mesh



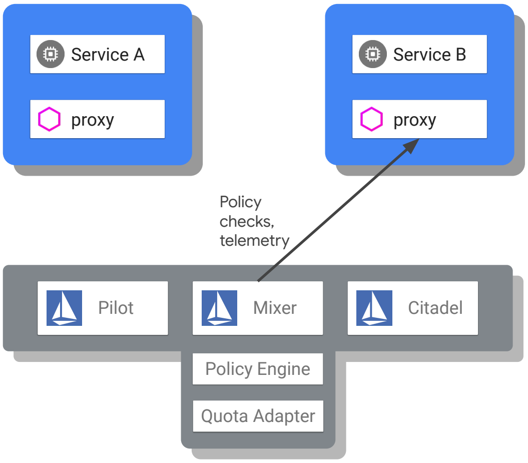
Envoy forwards request to appropriate instance of service B. There, the Envoy proxy deployed with the service intercepts the call

Life of a request in the mesh



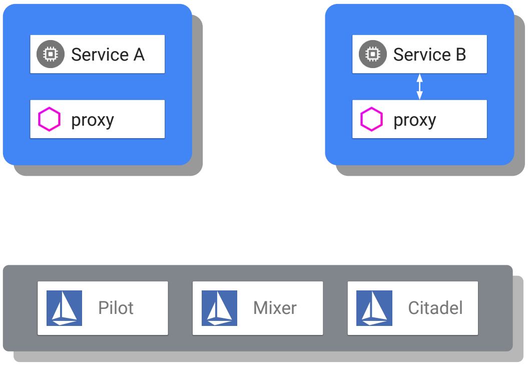
Server-side Envoy checks with Mixer to validate that call should be allowed (ACL check, quota check, etc).

Life of a request in the mesh



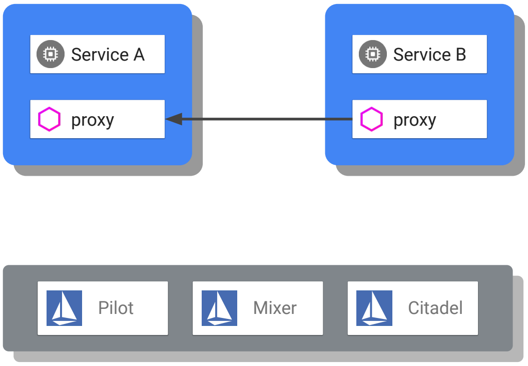
Mixer checks with appropriate adapters (policy engine, quota adapter) to verify that the call can proceed and returns true/false to Envoy

Life of a request in the mesh



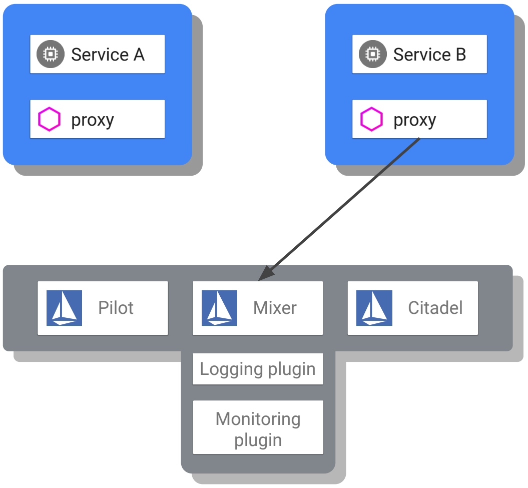
Server-side Envoy forwards request to service B, which process request and returns response

Life of a request in the mesh



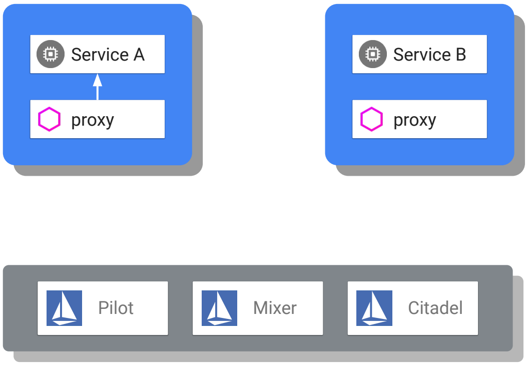
Envoy forwards response to the original caller, where response is intercepted by Envoy on the caller side

Life of a request in the mesh



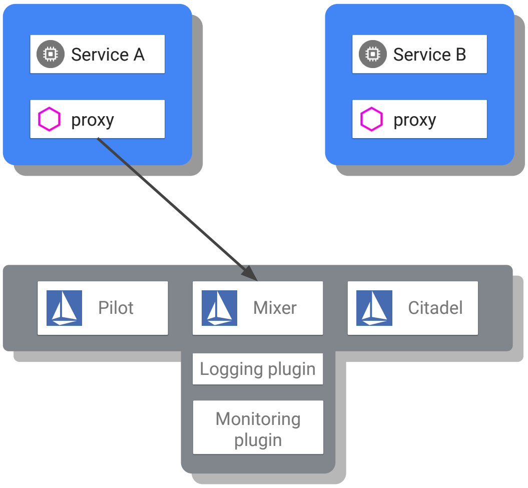
Envoy reports telemetry to Mixer, which in turn notifies appropriate plugins

Life of a request in the mesh



Client-side Envoy forwards response to original caller

Life of a request in the mesh



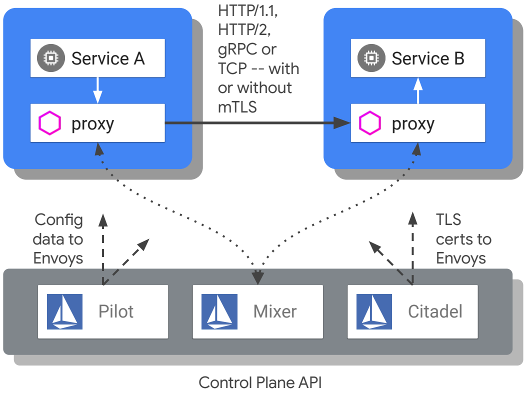
Client-side Envoy reports telemetry to Mixer (including client-perceived latency), which in turn notifies appropriate plugins

Istio Architectural Components

Pilot: Control plane to configure and push service communication policies

Mixer: Policy enforcement with a flexible plugin model for providers for a policy

Citadel: Service-to-service auth[n,z] using mutual TLS, with built-in identity and credential management



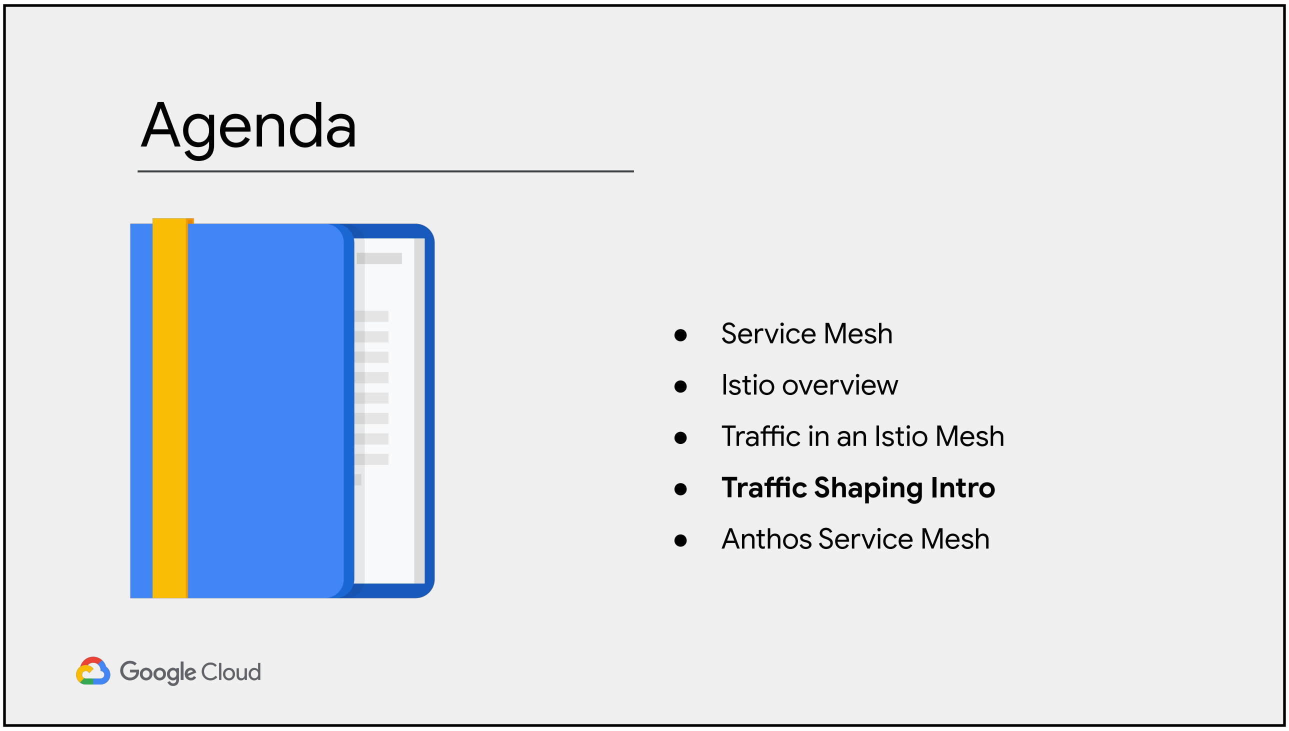
Service Mesh Features

Traffic Splitting independent from infrastructure instances

Content-based traffic steering

Fault injection

Circuit breaking



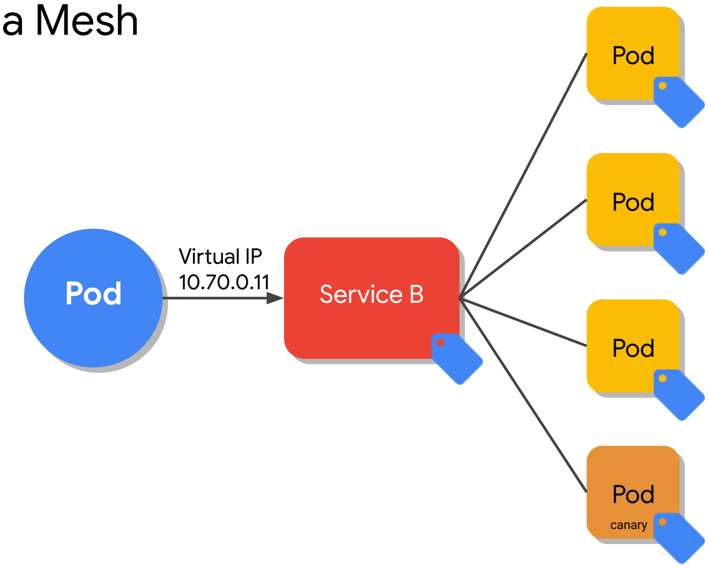
Agenda

- Service Mesh
- Istio overview
- Traffic in an Istio Mesh
- **Traffic Shaping Intro**
- Anthos Service Mesh

Network Traffic in a Mesh

Kubernetes traffic is shaped by

- Services as network endpoint abstraction
- Labels as compute endpoint abstraction



Traffic Shaping via Kubernetes Services

```
kind: Service
apiVersion: v1
metadata:
  name: frontend
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: myapp
    role: frontend
```

```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: frontend-prod
spec:
  replicas: 3
  template:
    metadata:
      name: frontend
      labels:
        app: myapp
        role: frontend
    spec:
      containers:
      - name: frontend
        image: my-img:v1
        ports:
        - name: ui
          containerPort: 80
```

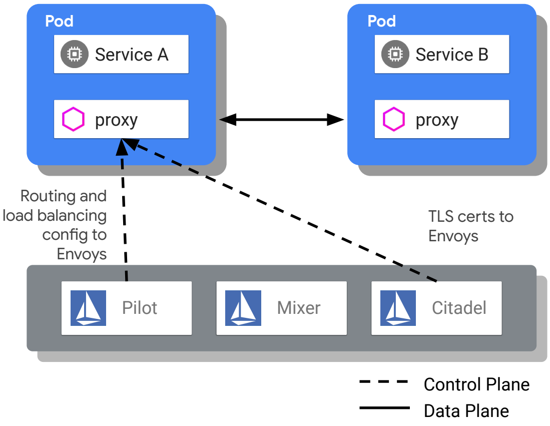
```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: frontend-staging
spec:
  replicas: 1
  template:
    metadata:
      name: frontend
      labels:
        app: myapp
        role: frontend
    spec:
      containers:
      - name: frontend
        image:my-img:v2
        ports:
        - name: ui
          containerPort: 80
```

Network Traffic in a Mesh

Service Mesh Traffic does not rely on or communicates with Kubernetes Services VIPs

A direct connection is created between the proxies, bypassing existing Kubernetes Virtual IPs

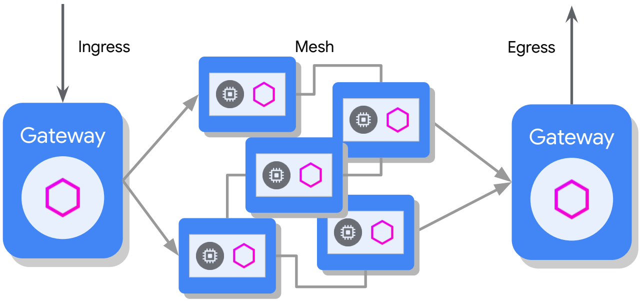
The Proxies' routing is configured via Pilot

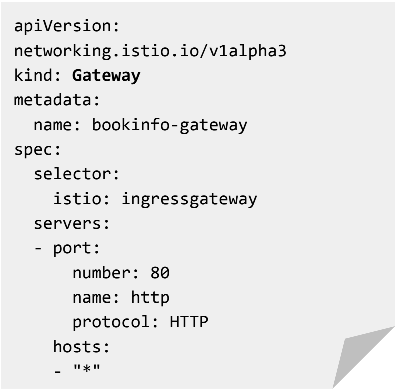


Istio Gateways

Allows traffic from outside the cluster into the Mesh

Contains ports, protocols, and certificates



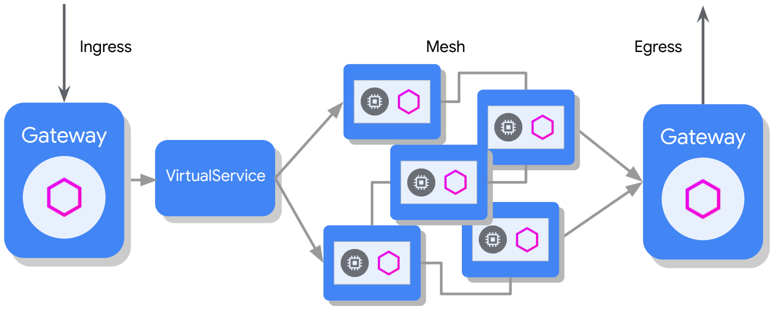


```
apiVersion:
networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: bookinfo-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
```

Istio VirtualServices

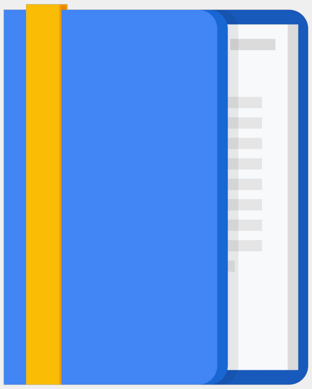
Configures how Envoy proxies route requests to a service within an Istio service mesh

Works similarly to Kubernetes Services, but allows richer traffic configuration



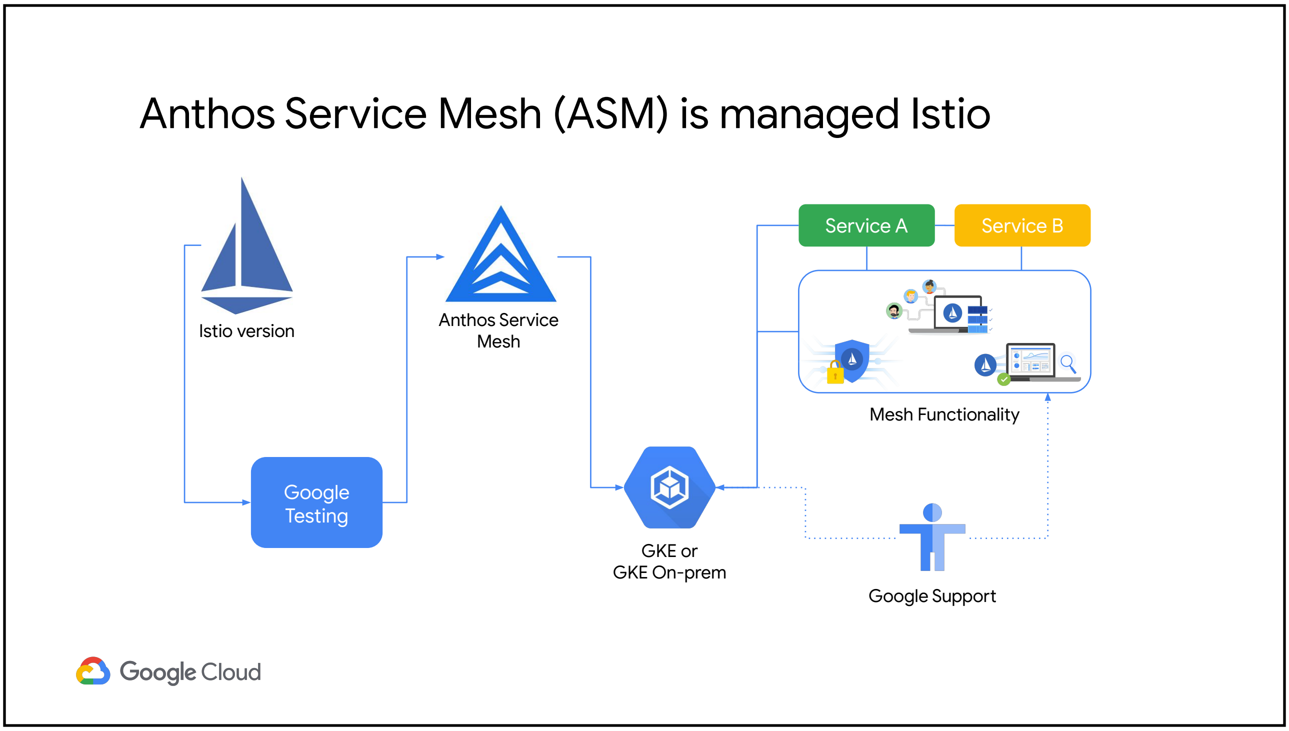
```
apiVersion:
networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
  - "*"
  gateways:
  - bookinfo-gateway
  http:
  - match:
    - uri:
        exact: /productpage
        - destination:
        host: productpage
        port:
          number: 9080
```

Agenda



- Service Mesh
- Istio overview
- Traffic in an Istio Mesh
- Traffic Shaping Intro
- **Anthos Service Mesh**

Google Cloud



Anthos Service Mesh (ASM) is managed Istio

Istio version

Google Testing

Anthos Service Mesh

GKE or GKE On-prem

Service A

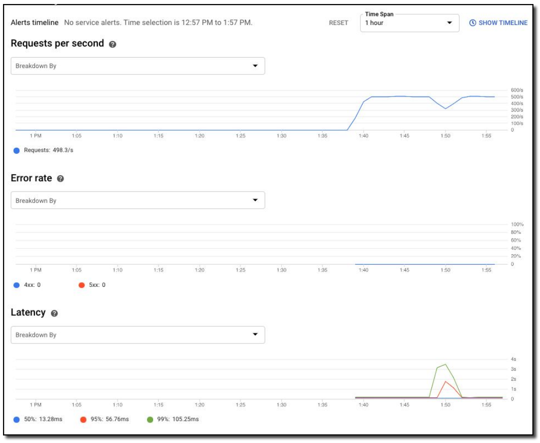
Service B

Mesh Functionality

Google Support

Google Cloud

ASM provides observability into service performance



Alerts timeline No service alerts. Time selection is 12:57 PM to 1:57 PM.

Time Span: 1 hour

Requests: 498.3/s

4xx: 0

5xx: 0

50%: 13.28ms

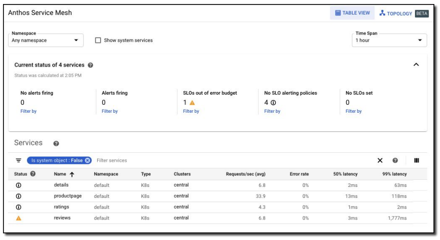
95%: 56.76ms

99%: 105.25ms

- All service requests and responses are logged and measured
- Metrics and logs are automatically ingested into Google Cloud
- Summary metrics are reported for 3/4 golden signals of monitoring
  - Latency
  - Errors
  - Traffic
- SLOs can be defined and tracked

ASM delivers predefined service dashboards

- Displays service health vs. SLOs
- Displays golden signals for each service
- Allows drill down by service
- Identifies how traffic flows between services



Anthos Service Mesh

Namespace: Any namespace

□ Show system services

Time Span: 1 hour

Current status of 4 services

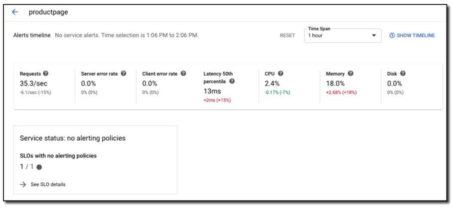
Status was calculated at 2:05 PM

No alerts firing	Alerts firing	SLOs out of error budget	No SLO alerting policies	No SLOs set
0	0	1 ⚠	4 ⓘ	0

Services

Is system object : False

Status	Name	Namespace	Type	Clusters	Requests/sec (avg)	Error rate	50% latency	99% latency
ⓘ	details	default	K8s	central	6.8	0%	2ms	63ms
ⓘ	productpage	default	K8s	central	33.9	0%	13ms	118ms
ⓘ	ratings	default	K8s	central	4.3	0%	1ms	2ms
⚠	reviews	default	K8s	central	6.8	0%	3ms	1,777ms



productpage

Alerts timeline No service alerts. Time selection is 1:06 PM to 2:06 PM.

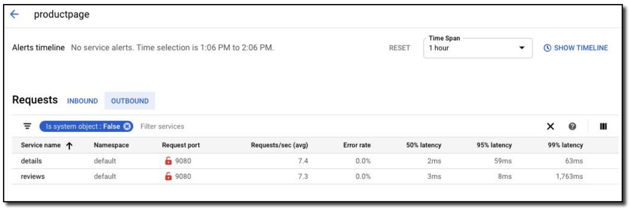
Time Span: 1 hour

Requests	Server error rate	Client error rate	Latency 50th percentile	CPU	Memory	Disk
35.3/sec	0.0%	0.0%	13ms	2.4%	18.0%	0.0%
-6.1/sec (-15%)	0% (0%)	0% (0%)	+2ms (+15%)	-0.17% (-7%)	+2.68% (+18%)	0% (0%)

Service status: no alerting policies

SLOs with no alerting policies

1 / 1 ●



productpage

Alerts timeline No service alerts. Time selection is 1:06 PM to 2:06 PM.

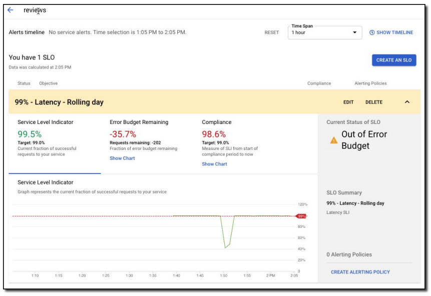
Time Span: 1 hour

Requests

Is system object : False

Service name	Namespace	Request port	Requests/sec (avg)	Error rate	50% latency	95% latency	99% latency
details	default	9080	7.4	0.0%	2ms	59ms	63ms
reviews	default	9080	7.3	0.0%	3ms	8ms	1,763ms

ASM enables SLO reporting and alerting



reviews

Alerts timeline No service alerts. Time selection is 1:05 PM to 2:05 PM.

Time Span: 1 hour

You have 1 SLO

Data was calculated at 2:05 PM

Status	Objective	Compliance	Alerting Policies
	99% - Latency - Rolling day		

Service Level Indicator

99.5%

Target: 99.0%

Current fraction of successful requests to your service

Error Budget Remaining

-35.7%

Requests remaining: -202

Fraction of error budget remaining

Compliance

98.6%

Target: 99.0%

Measure of SLI from start of compliance period to now

Current Status of SLO

Out of Error Budget

Service Level Indicator

Graph represents the current fraction of successful requests to your service

SLO Summary

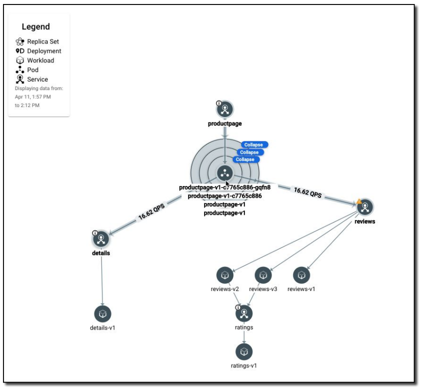
99% - Latency - Rolling day

Latency SLI

0 Alerting Policies

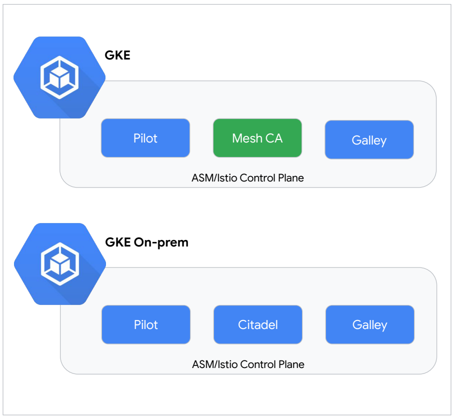
- Allows operators to define SLOs
- Shows SLO performance, compliance, and error budget for every service
- Allows alerting based on SLO performance

ASM visualizes your mesh topology



- Creates a chart to represent relationships and traffic flow between services
- Allows drill down to see the workloads and pods behind services
- Displays QPS rates between services

ASM replaces Citadel as a certificate authority



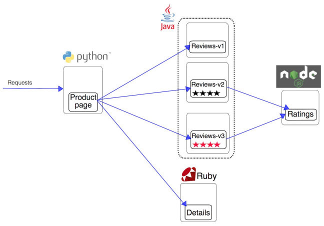
- Mesh CA only used for cloud-based GKE clusters (not on-prem)
- Highly reliable, Google managed service

Google Cloud

Lab

Installing Anthos Service Mesh on Kubernetes Engine

30 min



BookInfo Application without Istio

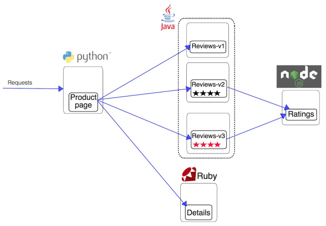
Objectives

- Provision a cluster on Google Kubernetes Engine (GKE)
- Install and configure Anthos Service Mesh
- Deploy Bookinfo, an Istio-enabled multi-service application
- Enable external access using an Istio Ingress Gateway
- Use the Bookinfo application
- Monitor service performance with the Anthos Service Mesh Dashboard

Optional Lab

Installing the Istio on GKE Add-On with Kubernetes Engine

30 min



BookInfo Application without Istio

Objectives

- Provision a cluster on Google Kubernetes Engine (GKE)
- Install and configure the Istio on GKE Add-On, which includes the Istio control-plane and a method to deploy Envoy proxies as sidecars
- Deploy Bookinfo, an Istio-enabled multi-service application
- Enable external access using an Istio Ingress Gateway
- Use the Bookinfo application