Istio → Service Mesh

Service Mesh → Network of Microservices

* Monitoring
* Load Balancing → A/B, Canary
* Metrics → Rate Limiting
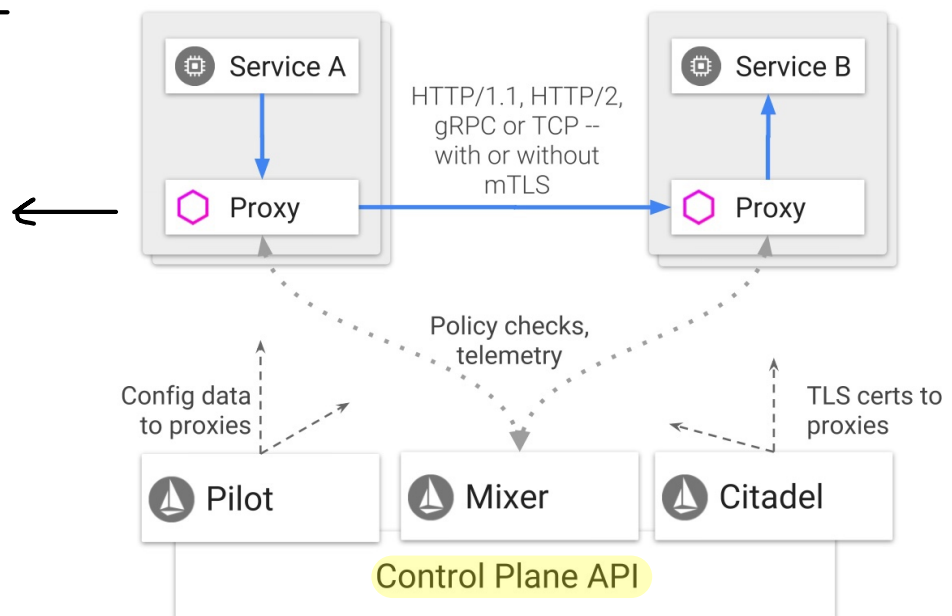* Failure Recovery
* Security → Access control, encription

Architecture

Data Plane Envoy → Sidecar Container

Service A → Proxy → Proxy → Service B

HTTP/1.1, HTTP/2, gRPC or TCP -- with or without mTLS

Policy checks, telemetry

Config data to proxies

TLS certs to proxies

Pilot   Mixer   Citadel

Control Plane API

<u>Pilot</u> == Traffic Management

Configures the Envoy's

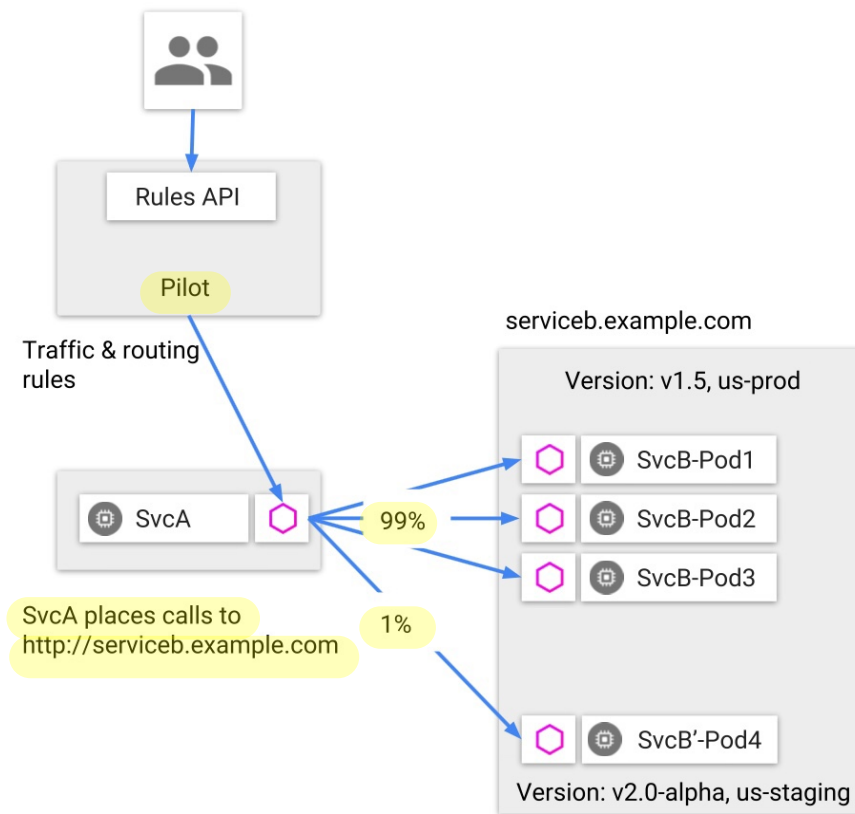Failure Recovery :
  - Timeouts
  - Retries
  - Circuit Breakers

Service Discovery

Concept of
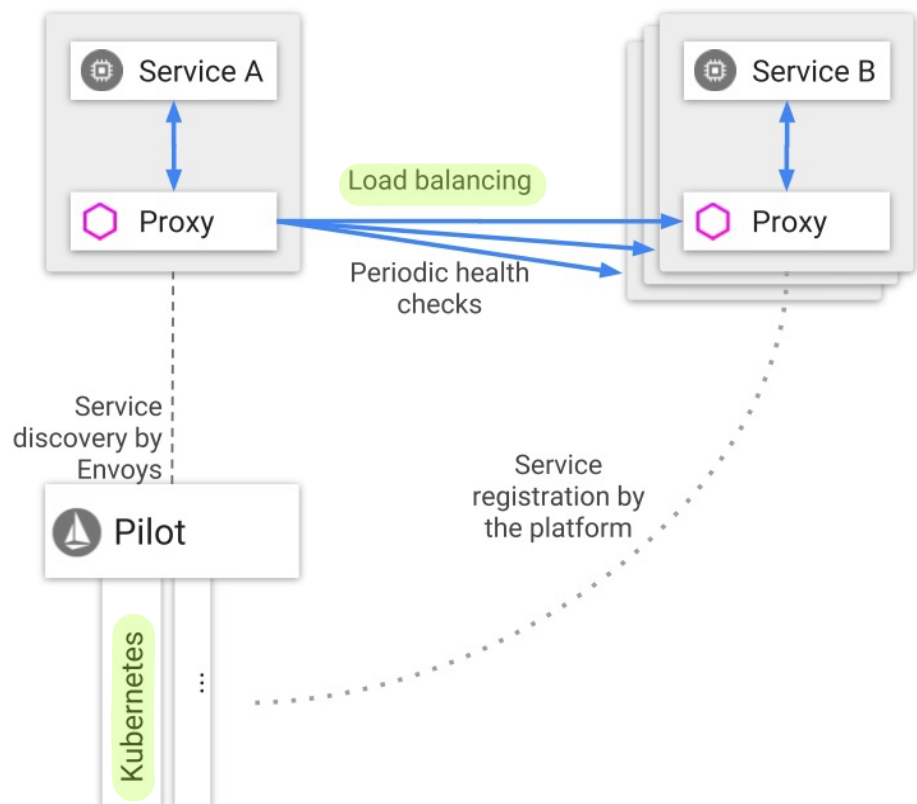Service Version
$(V_1, V_2, V_3, ...)$

Each Envoy
maintains load balancing
information injected
by Pilot

* A/B
* Canaries
* Dark Launch

Communication between Services

Rules API

Pilot

Traffic & routing rules

serviceb.example.com

Version: v1.5, us-prod

SvcA

99%

SvcB-Pod1

SvcB-Pod2

SvcB-Pod3

SvcA places calls to
http://serviceb.example.com

1%

SvcB'-Pod4

Version: v2.0-alpha, us-staging

Load Balancing

Service A

Proxy

Load balancing

Service B

Proxy

Periodic health checks

Service discovery by Envoys

Pilot

Service registration by the platform

Kubernetes

Security

* Citadel : Certificate Management

* Pilot : Auth Policies (Distribution)

* Mixer : Auth & Audit

Telemetry == Mixer