Istio - A service mesh on Kubernetes

Chandresh Pancholi

About Me

- Senior Developer at Arvind Internet
- Committer at Apache software foundation
- Ex-Flipkart Engineer
- Occasional technical book reviewer

Email: chandresh.pancholi@arvindinternet.com
Linkedin: https://www.linkedin.com/in/chandresh-pancholi-467a8015

Microservices Architecture

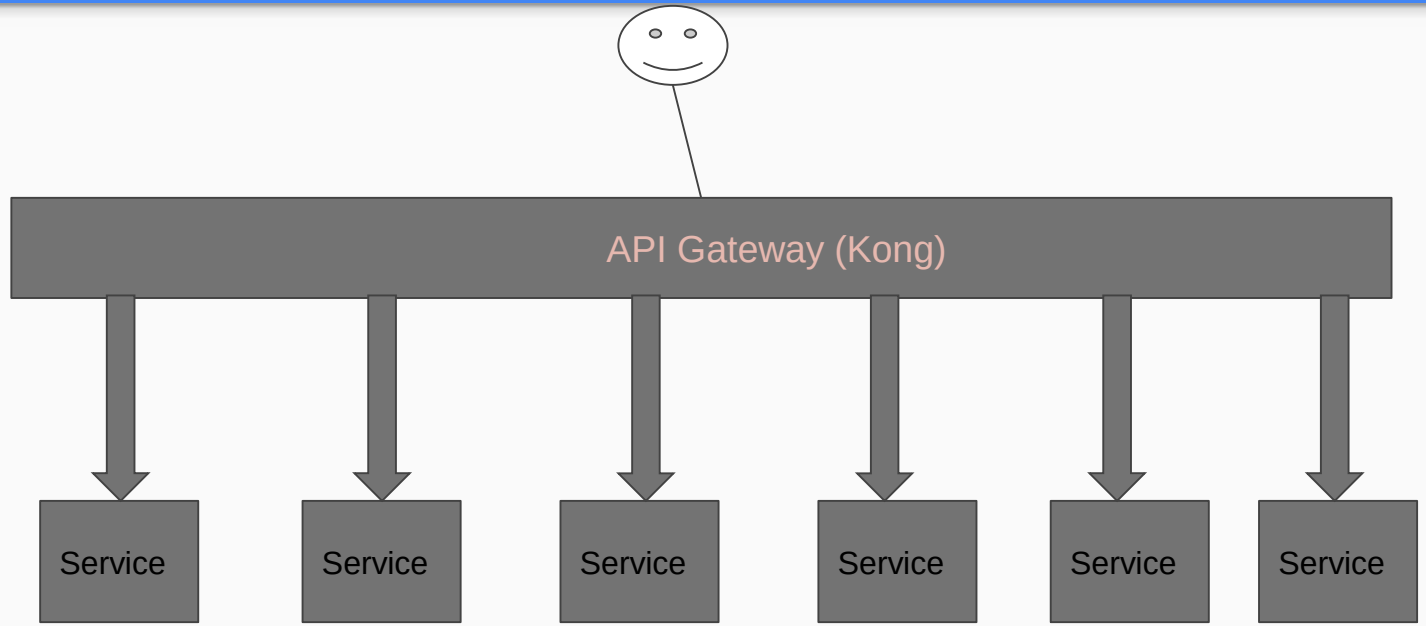
Microservice Architecture - is an architectural style that structures an application as a collection of loosely coupled services, which implement business capabilities. The microservice architecture enables the continuous delivery/deployment of large, complex applications. It also enables an organization to evolve its technology stack.

Source: http://microservices.io/

What do we need in Microservices?

- Service discovery → Netflix Eureka
- Monitoring → Netflix Hystrix dashboard or Turbine
- Tracing → Zipkin
- Routing → Netflix Ribbon
- Security → OAuth 2.0 with OpenId connect
- Traffic management → Load Balancer
- Rate Limiting
- Circuit Breaking → Netflix Hystrix

Earlier Architect



Issues

- Single point of failures
- Need to manage routing roles
- Kong clustering on Kubernetes
- Cascading affects when one Kong pod goes down
- A/B Testing, Canary deployment, Tracing
- Difficult to upgrade Kong

Istio - A Service Mesh

Service Mesh - "A decentralized application-networking infrastructure between your services that provides security, resiliency, observability, and routing control".

Istio - "An open platform to connect, manage, and secure microservices."

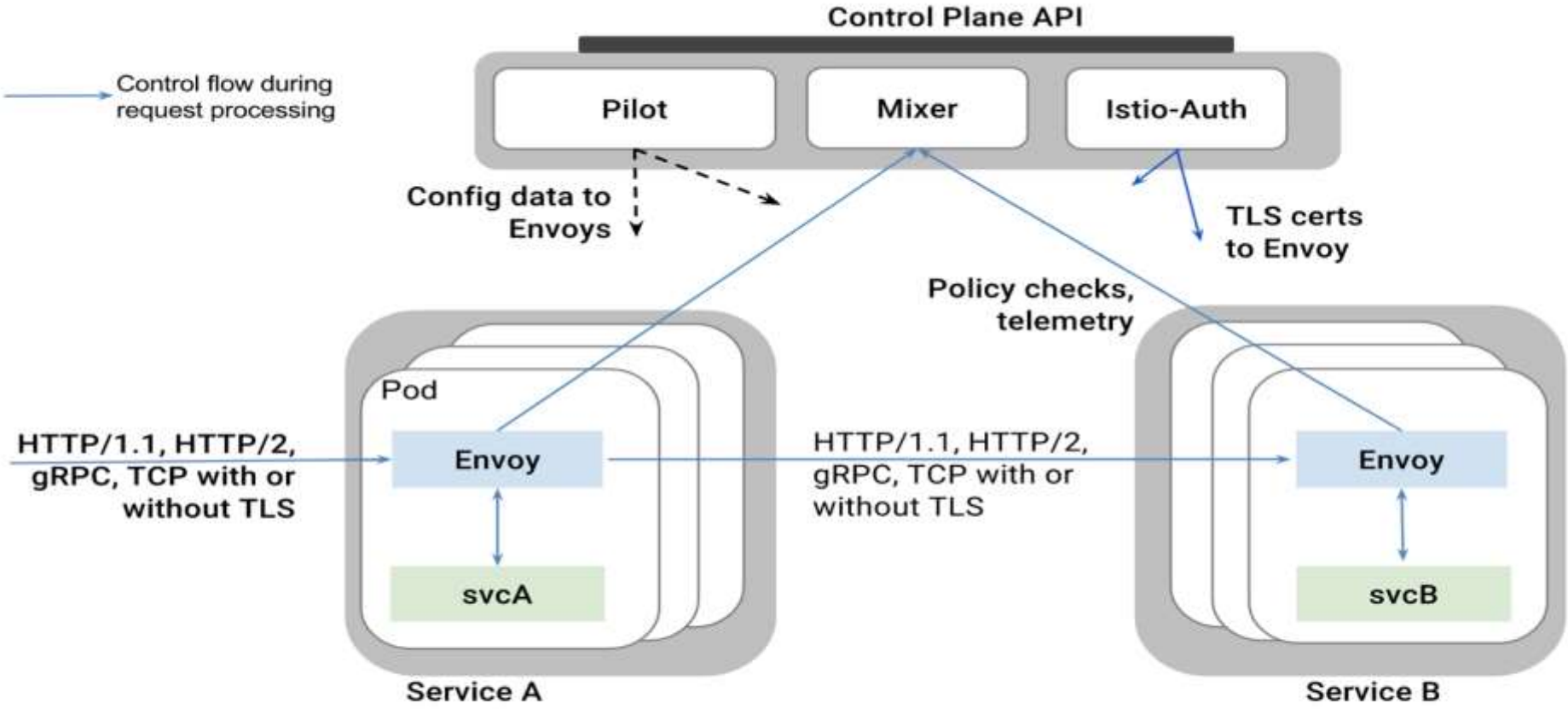
Istio is build and managed by Google, IBM, Lyft

Istio Architecture

An Istio service mesh is logically split into a data plane and a control plane.

Data plane is composed of a set of intelligent proxies (Envoy) deployed as sidecars that mediate and control all network communication between microservices.

Control plane is responsible for managing and configuring proxies to route traffic, as well as enforcing policies at runtime. Example Pilot, Mixer, Istio-Auth



Istio Architecture

Istio Components

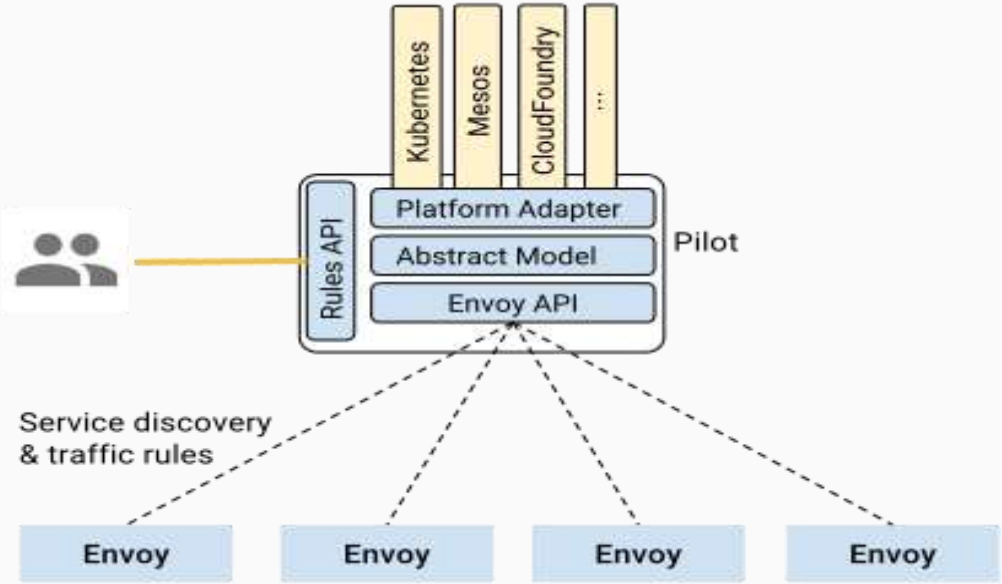
- Envoy
- Istio Pilot
- Istio Mixer

Envoy

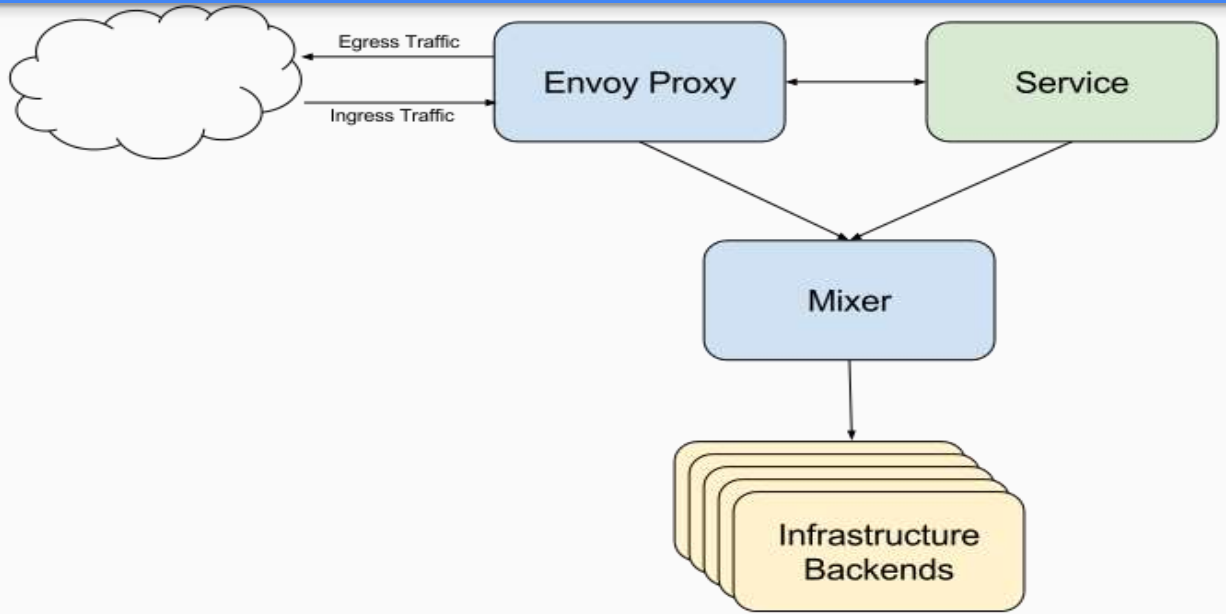
Envoy proxy, a high-performance proxy developed in C++, to mediate all inbound and outbound traffic for all services in the service mesh.

Istio leverages Envoy's many built-in features such as dynamic service discovery, load balancing, TLS termination, HTTP/2 & gRPC proxying, circuit breakers, health checks, staged rollouts with %-based traffic split, fault injection, and rich metrics.

Istio pilot


Kubernetes
Mesos
CloudFoundry
...
Rules API
Platform Adapter
Abstract Model
Envoy API
Pilot
Service discovery
& traffic rules
Envoy
Envoy
Envoy
Envoy

Istio mixer


Egress Traffic
Ingress Traffic
Envoy Proxy
Service
Mixer
Infrastructure
Backends

Inject sidecar container

```
kubectl apply -f <(istioctl kube-inject -f application.yaml )
```

Route rule

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: optim-default
  namespace: default
spec:
  destination:
    name: optim
  precedence: 1
  route:
  - labels:
      version: v1
    weight: 50
  - labels:
      version: v2
    weight: 50
~
~
```

Circuit breaker

```
apiVersion: config.istio.io/v1beta1
kind: DestinationPolicy
metadata:
  name: optim-cb
spec:
  destination:
    name: optim
    labels:
      version: v1
  circuitBreaker:
    simpleCb:
      maxConnections: 1
      httpMaxPendingRequests: 1
      sleepWindow: 3m
      httpDetectionInterval: 1s
      httpMaxEjectionPercent: 100
      httpConsecutiveErrors: 1
      httpMaxRequestsPerConnection: 1
```

Demo

Questions ?