Bare metal cluster with Kubernetes, Istio & MetalLB

Nguyen Phuong An <annp.cs51@gmail.com>
Nguyen Hai Truong <nguyenhaitruonghp@gmail.com>

FUJITSU
VietKubers

Who are we?

💼 Software Engineers **Fujitsu Vietnam**

📝 Organizers of **VietKubers** <https://vietkubers.github.io>

📝 Organizers of GDG Cloud Hanoi

@truongnh92

@annp87

Agenda

Kubernetes - Cluster

MetalLB - Load balancer

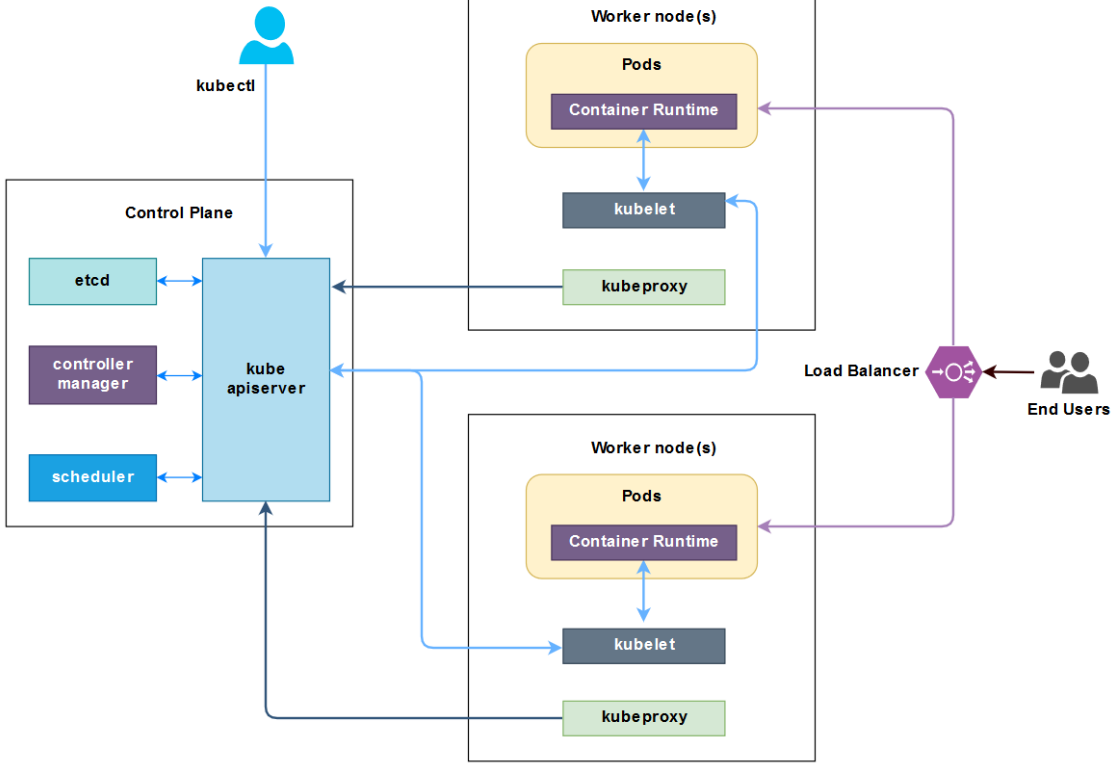
Istio - Service Mesh

Demo - Hands on

Kubernetes - Cluster

FUJITSU

Kubernetes architecture

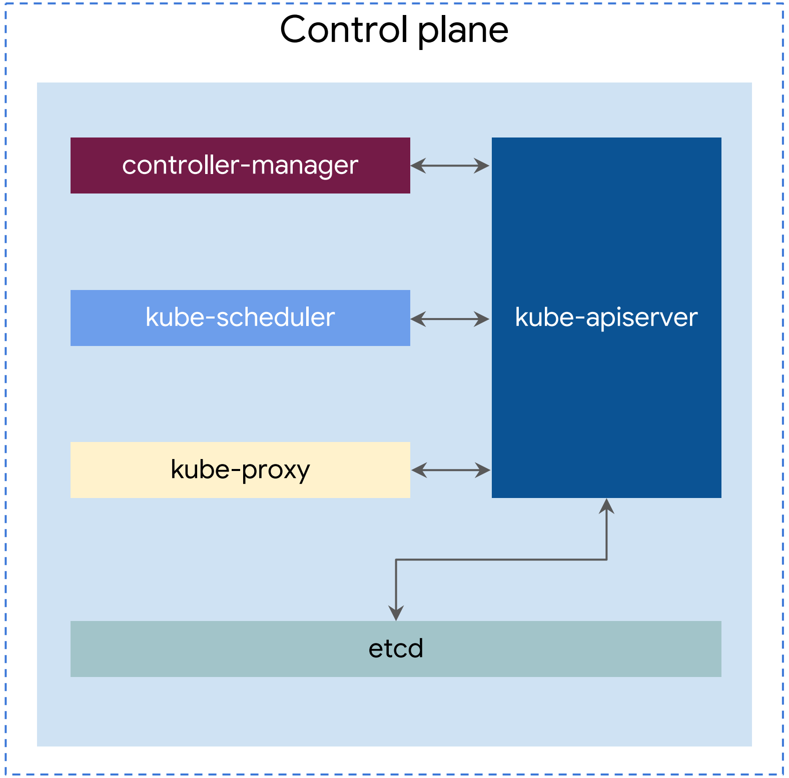


Control Plane (aka master)

- **etcd**: store cluster state
- **kube-scheduler**: Chooses hosts to run those containers on.
- **kube-apisever**: servers REST API request
- **kube-controller**:

✓ watch the desired state in the apiserver

✓ trigger reconciliation function to make actual state matching with desire state

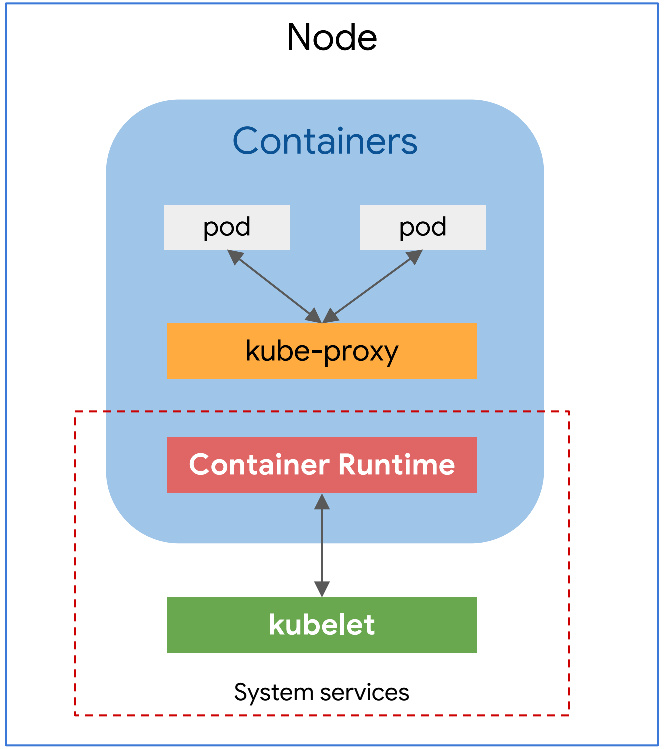


FUJITSU

Cluster Nodes (aka worker)

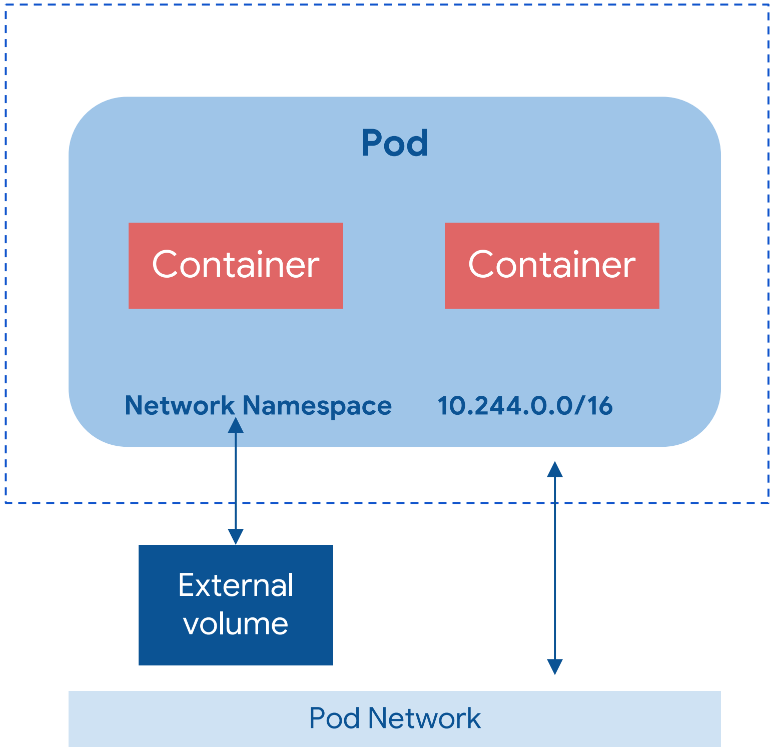
- **container runtime**: which is responsible for downloading images and running containers such as Docker
- **kube-proxy**: a network proxy/loadbalancer to route traffic in/out pods. It is implemented by iptables.
- **kubelet**: Responsible for communicate with master nodes and keep tracks of a pod to ensure that all container are running.

Refers to:
https://github.com/kubernetes/community/blob/master/contributors/design-proposals/architecture/architecture.md



Pod

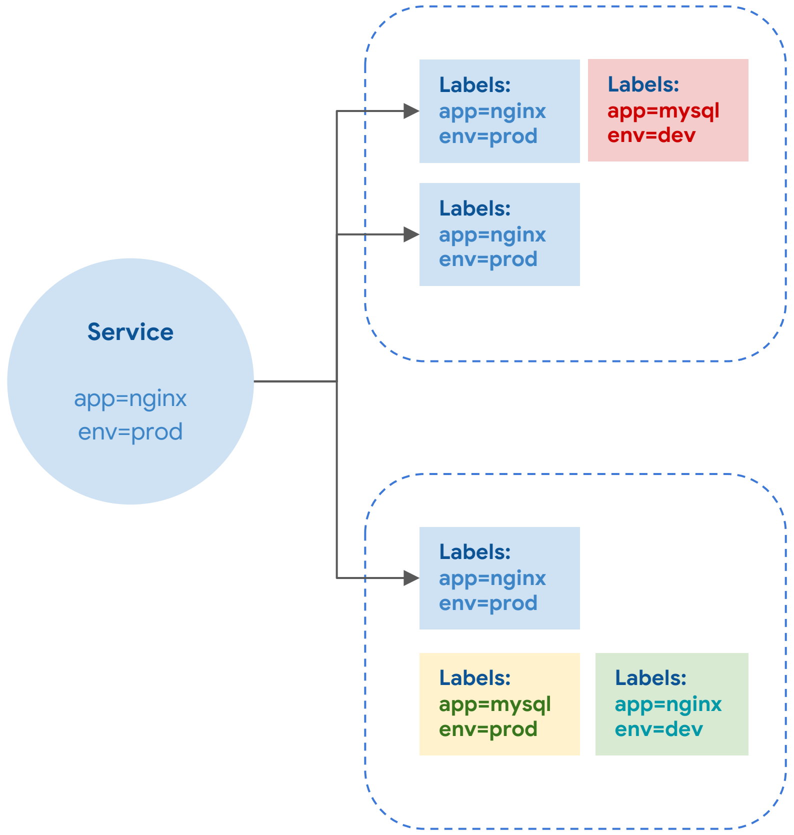
- **ReplicaSet:** the default, is a relatively simple type. It ensures the specified number of pods are running
- **Deployment:** is a declarative way of managing pods via ReplicaSets. Includes rollback and rolling update mechanisms
- **Daemonset:** is a way of ensuring each node will run an instance of a pod
- **StatefulSet:** is tailored to managing pods that must persist or maintain state



Service discovery

- Kubernetes heavily relies on its **integrated DNS service** (either **Kube-DNS** or **CoreDNS**, depending on the cluster version) to do discovery
  - CoreDNS create, update and delete DNS records for services and associated pods.
  - An example of a DNS record for a Kubernetes service: **service.namespace.svc.cluster.local** →A pod would have a DNS record such as: **10.32.0.125.namespace.pod.cluster.local**

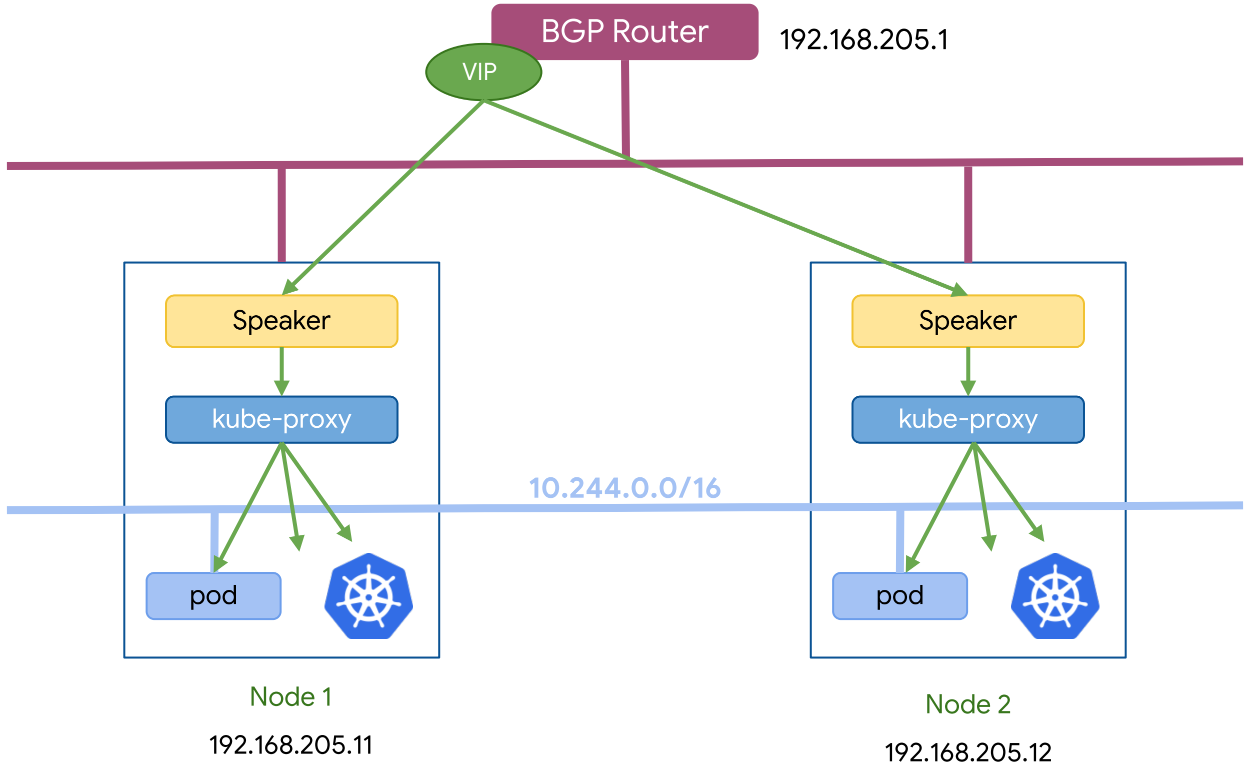
- **ClusterIP:** exposes the service on an internal IP only → access in cluster
- **NodePort:** exposes the service on each node's IP at a specific port
- **LoadBalancer:** exposes the service with external IP


Service
app=nginx
env=prod
Labels: app=nginx env=prod
Labels: app=mysql env=dev
Labels: app=nginx env=prod
Labels: app=nginx env=prod
Labels: app=mysql env=prod
Labels: app=nginx env=dev

FUJITSU

MetalLB - Load balancer

MetalLB (BGP mode)



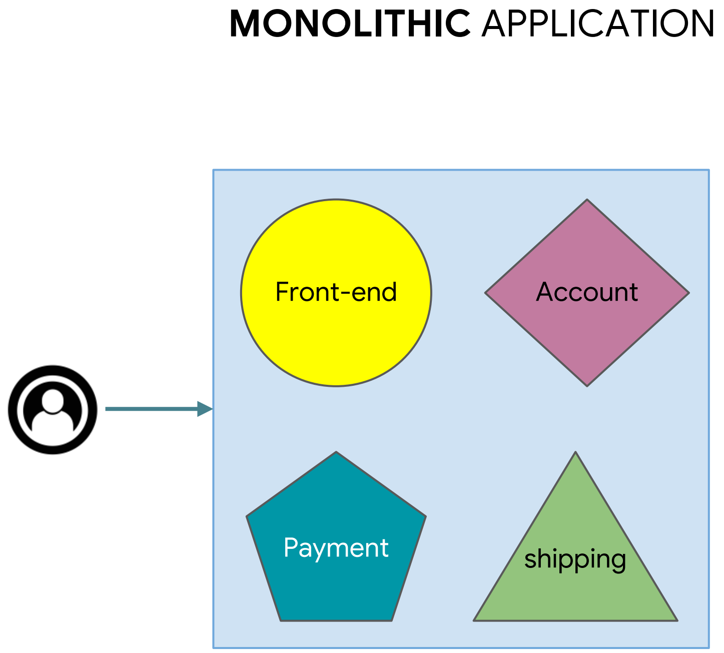
MetalLB overview

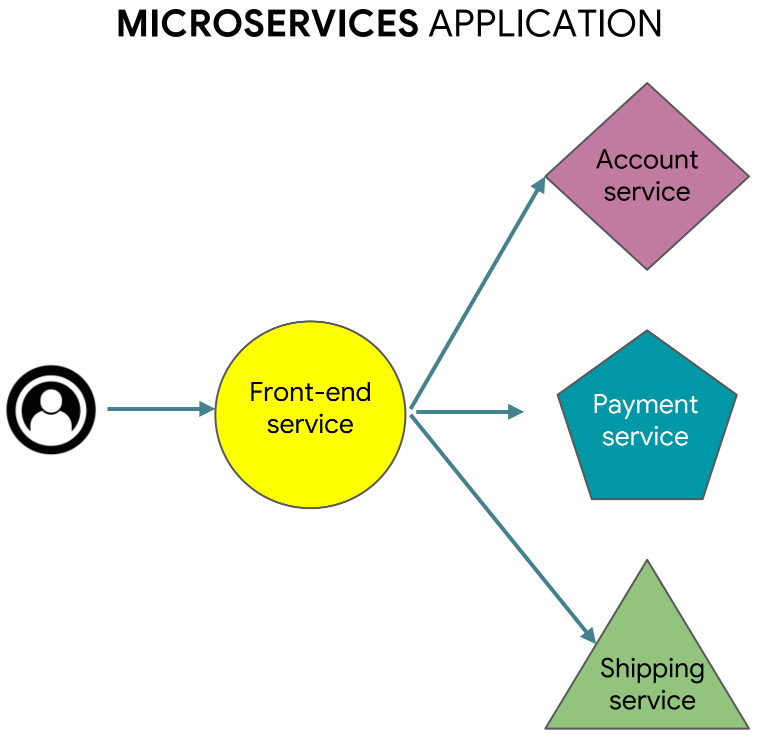
MetalLB uses standard routing protocols:

- **ARP** (IPv4), **NDP** (IPv6) (Layer 2 mode)
  - MetalLB responds to ARP requests for IPv4 services and NDP requests for IPv6
  - It will work on any ethernet network with no special hardware required, not event fancy routers
  - Limitations: single-node bottle-necking and potentially slow failover
- **BGP**
  - Each node in your cluster establishes a BGP peering session with network routers, and uses that peering session to advertise the IPs of external cluster services

Istio - Service Mesh

Microservices





Advantages and Drawbacks of Microservices

Advantages

- Smaller codebase
- Without depend on language programing
- CD will be easier
- Scalability
- Decentralized data
- Isolate failures

Drawbacks

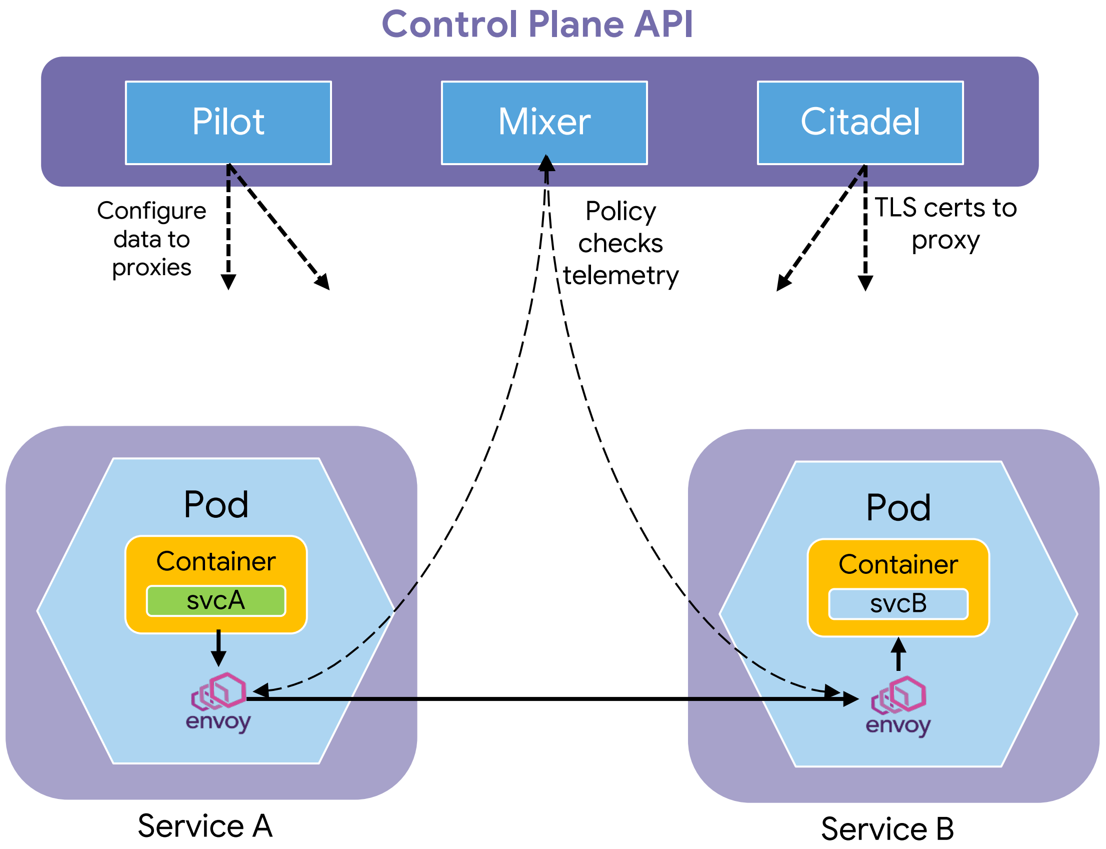
- Hard to keep track of microservices
- Complexity
- Routing microservices will need more work
- Consume more resources

→ Istio addresses some of the drawbacks in microservices

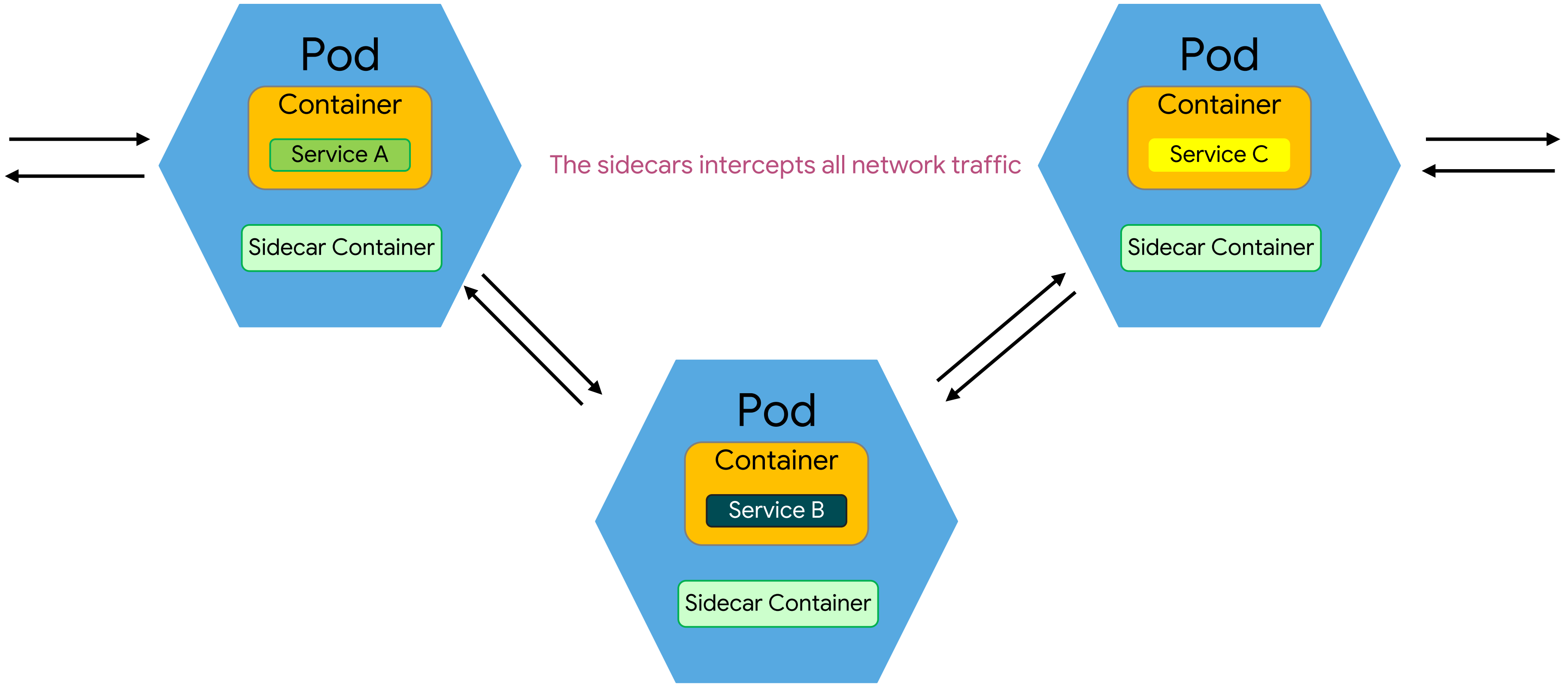
Istio

- An open platform to connect, manage and secure microservices
- Installed on top of K8s cluster, keep track of statuses, bugs of application
- Manages the traffic of Microservice
- Provides security within Microservices like mutual TLS

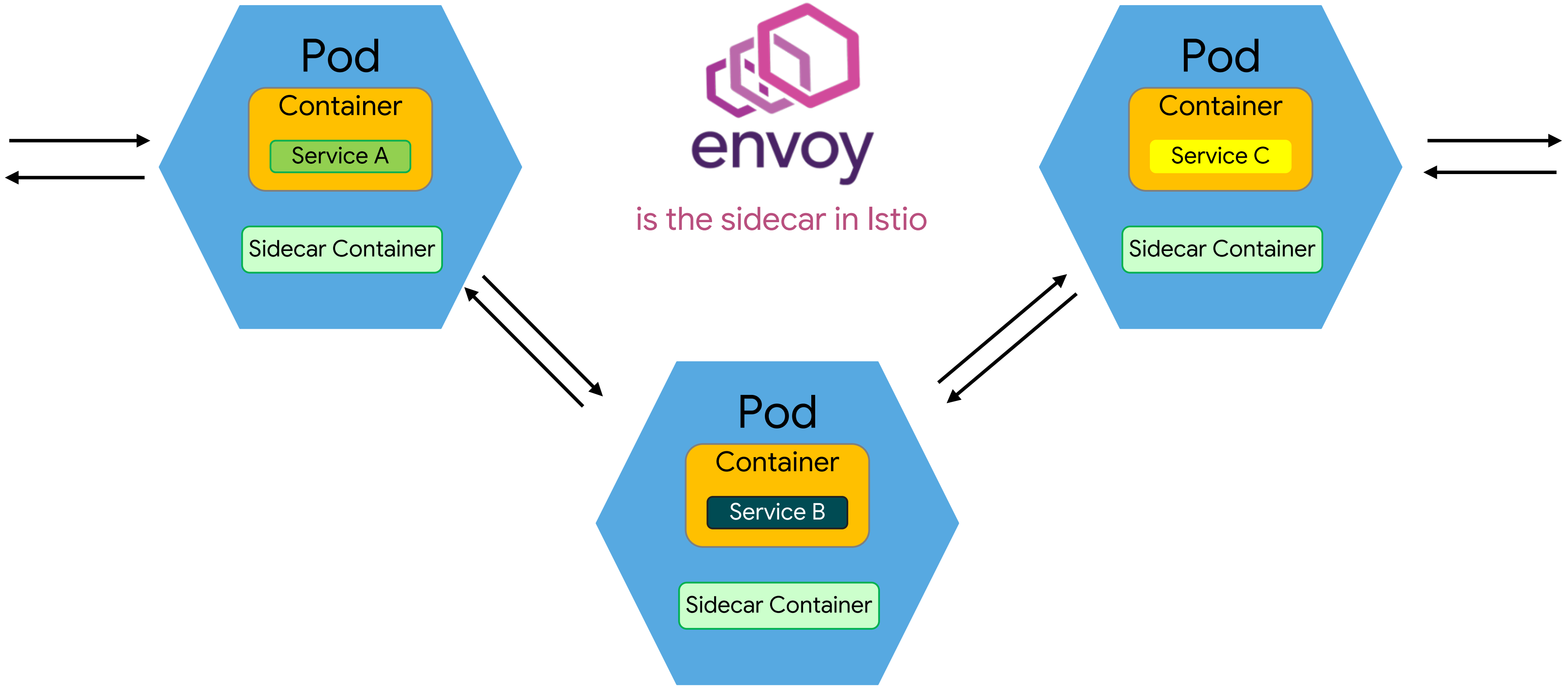
Istio Service Mesh architecture



Sidecar on Kubernetes



Envoy


Pod
Container
Service A
Sidecar Container
envoy
is the sidecar in Istio
Pod
Container
Service C
Sidecar Container
Pod
Container
Service B
Sidecar Container

Envoy

envoy

An open source edge and service proxy, designed for cloud-native applications

- L3/4 network filter

- Advanced load balancing

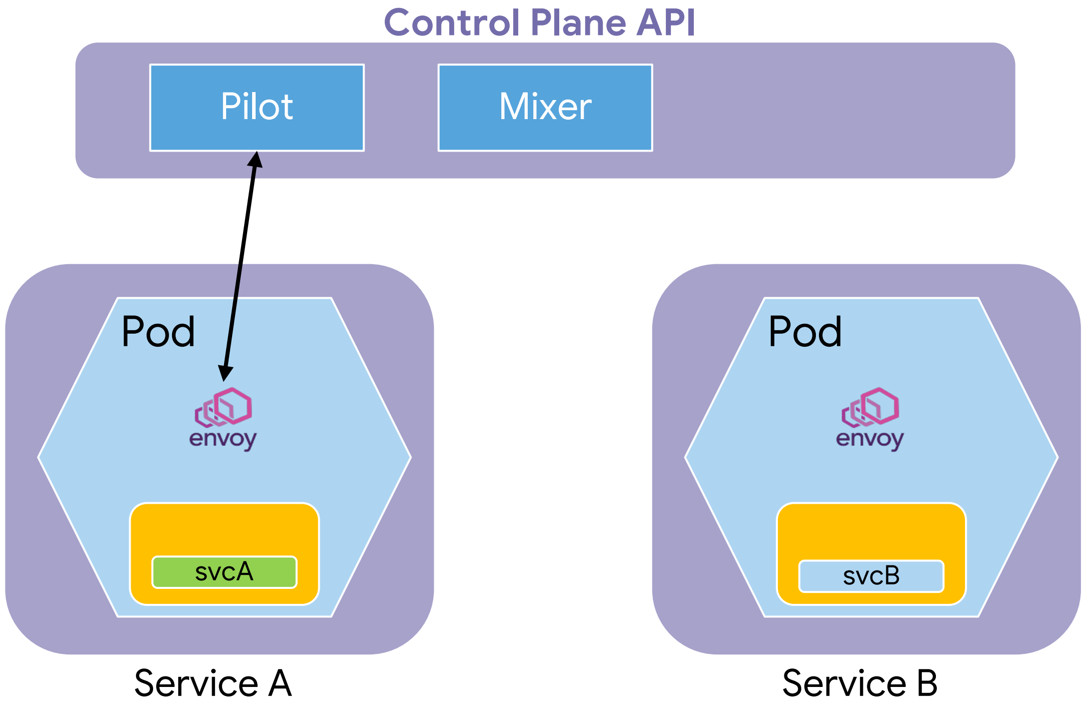
- Stats, metrics, tracing

How Istio works

Service A comes up

Envoy is deployed alongside it

Routing and configuration policy from Pilot



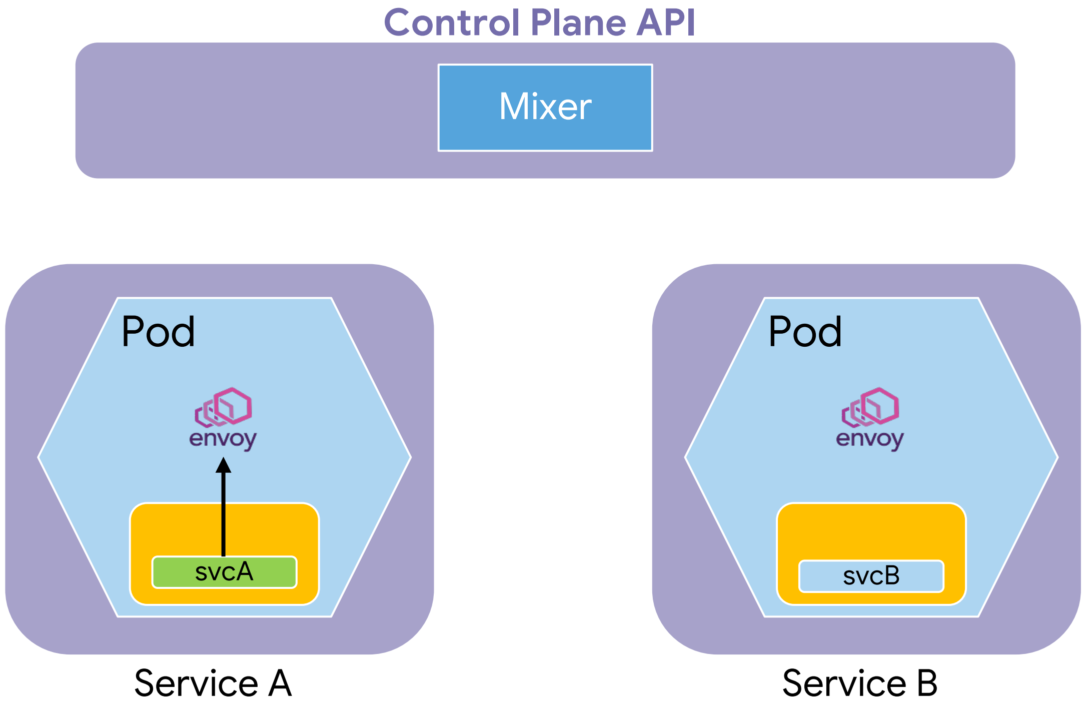
FUJITSU

How Istio works

Service A calls service B

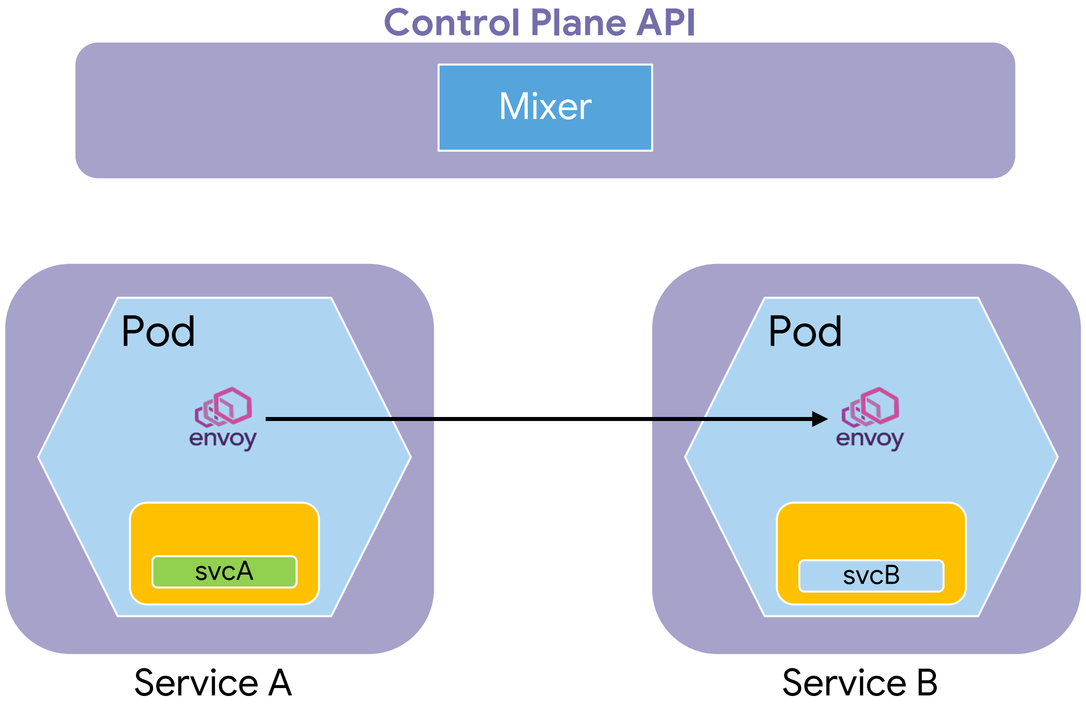
Envoy intercepts the call

Envoy consults Pilot to know **How/Where** to route call to service B



How Istio works

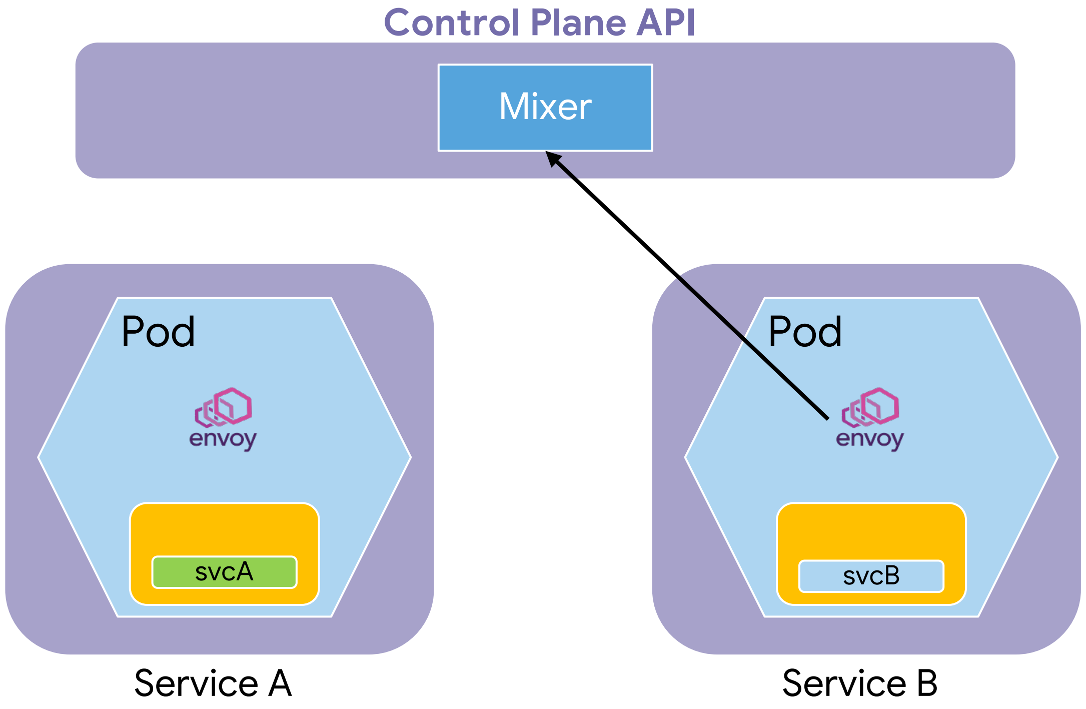
Envoy forwards request to appropriate instance of service B



FUJITSU

How Istio works

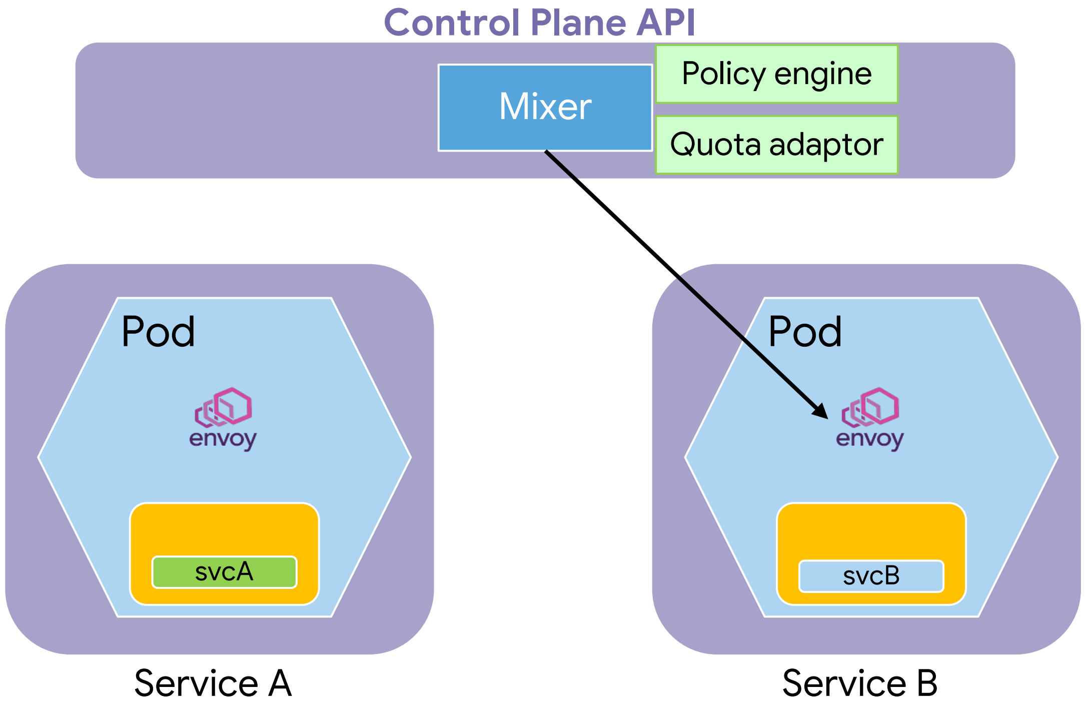
Server-side Envoy checks with Mixer to validate that call should be allowed



FUJITSU

How Istio works

Mixer checks with appropriate adaptors to verify that the call can proceed

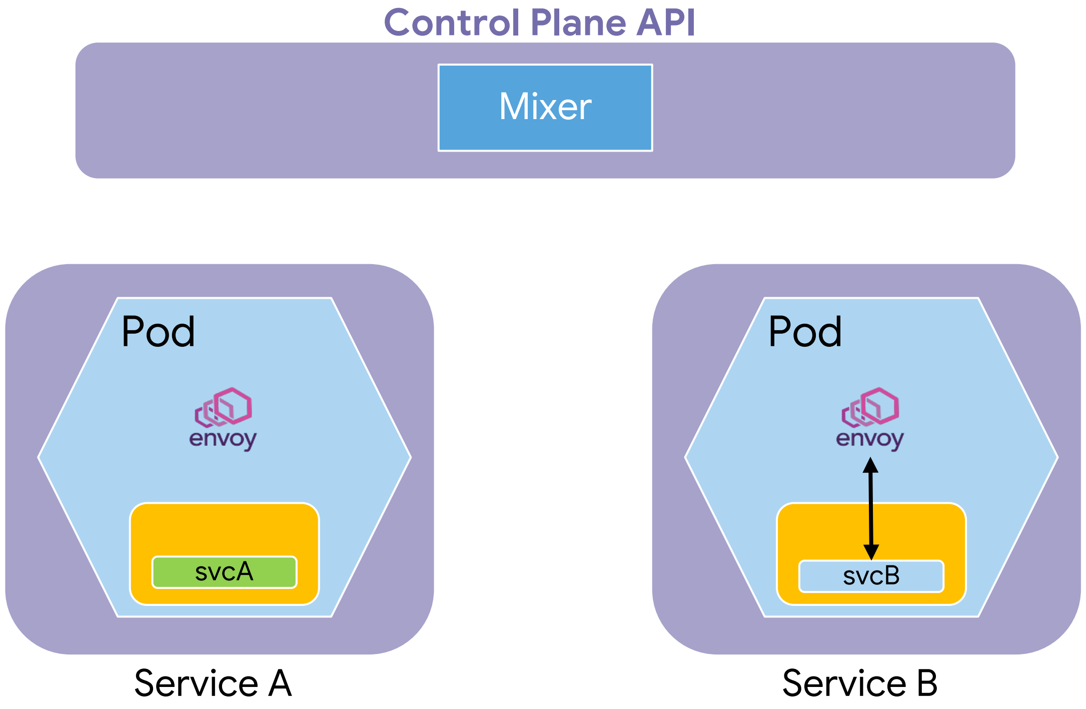


FUJITSU

How Istio works

Server-side Envoy forwards requests to service B

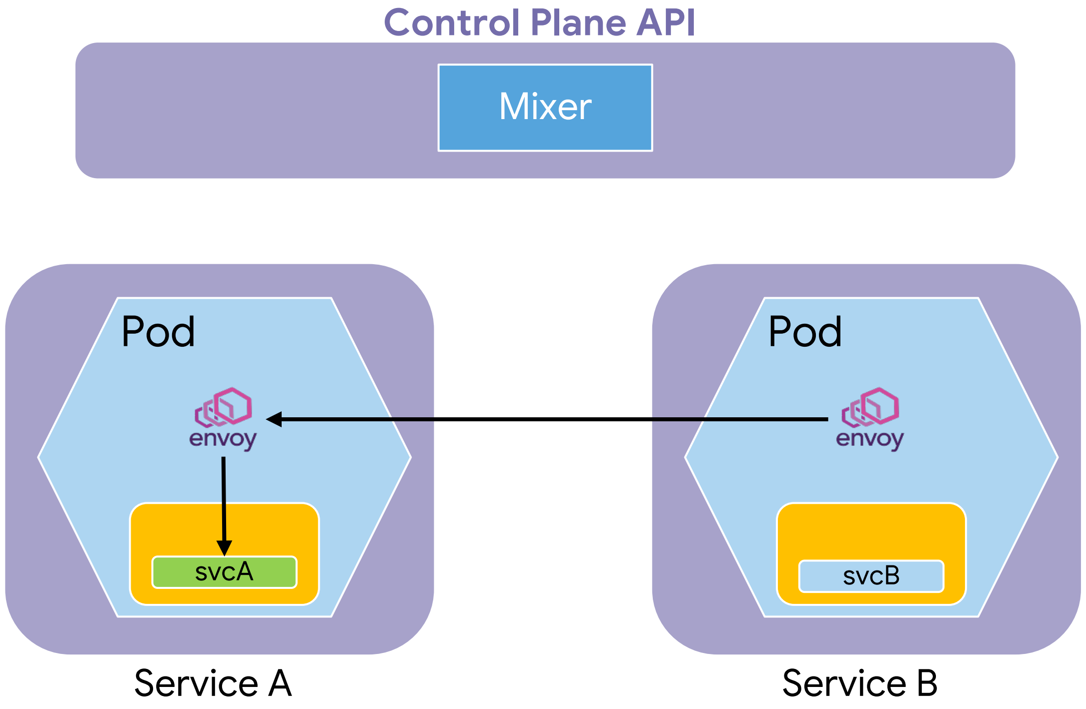
Service B processes the request and returns response



How Istio works

Envoy forwards response to the caller

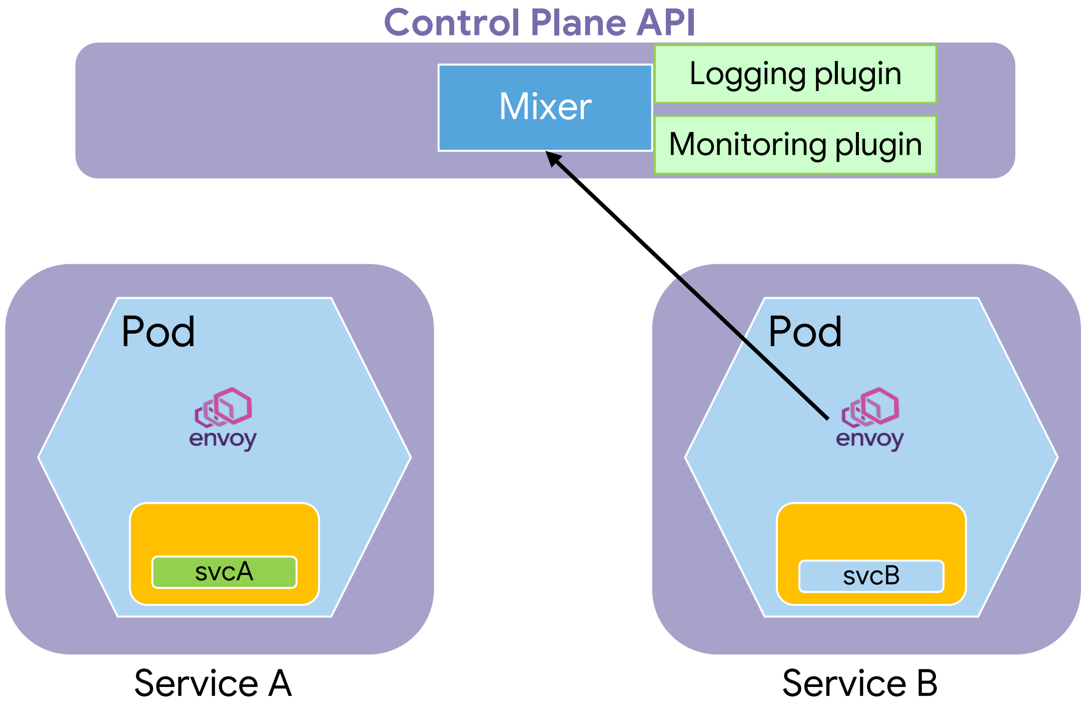
Client-side Envoy forwards response to the original caller



FUJITSU

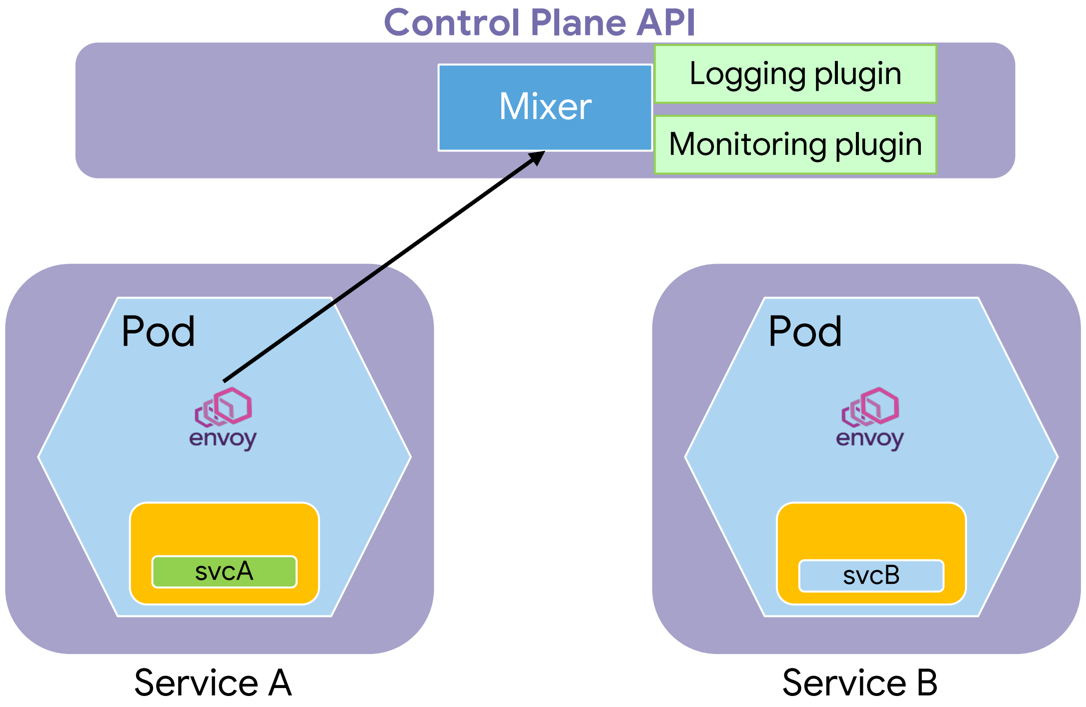
How Istio works

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



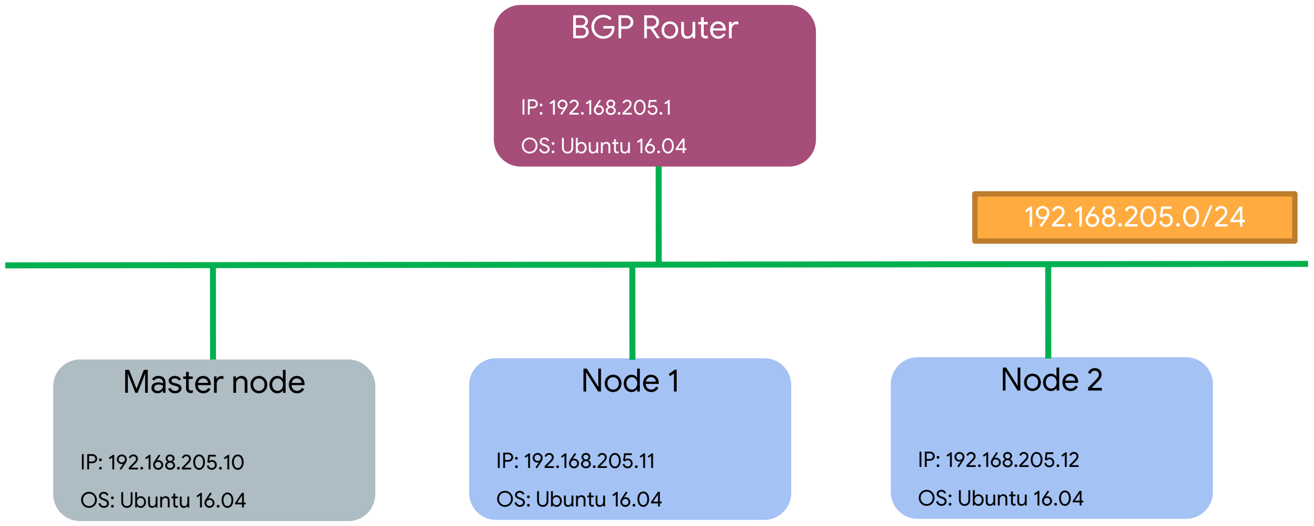
How Istio works

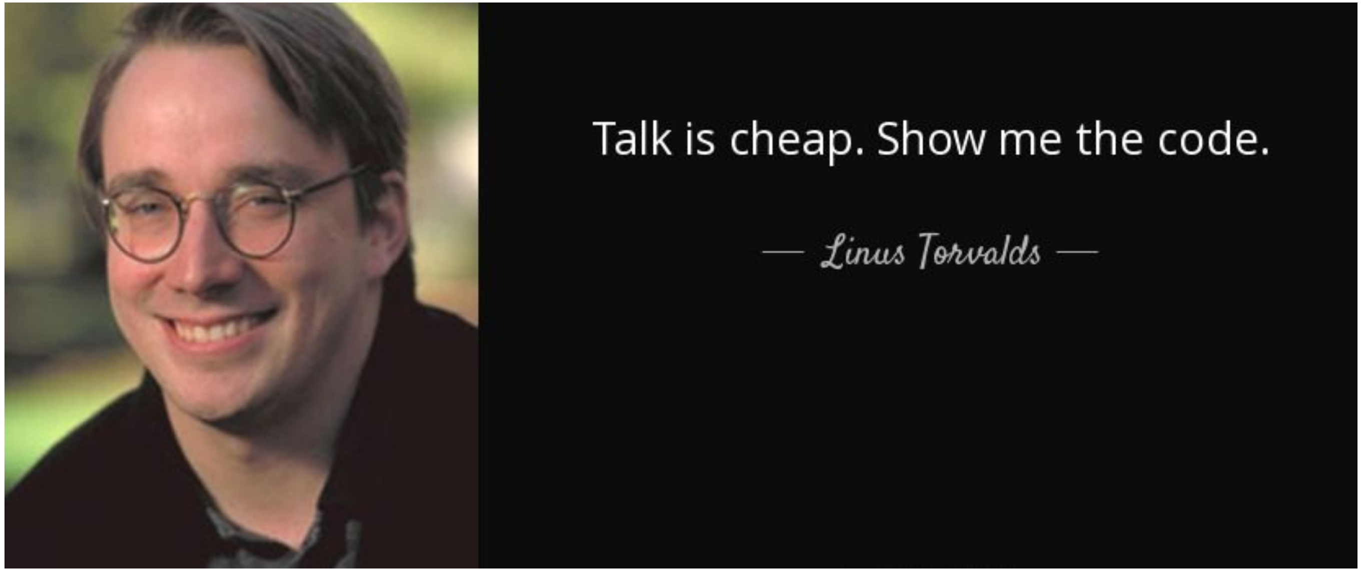
Client-side Envoy reports telemetry to Mixer (including client-perceived latency)



Demo - Hands on

Topology


BGP Router
IP: 192.168.205.1
OS: Ubuntu 16.04
192.168.205.0/24
Master node
IP: 192.168.205.10
OS: Ubuntu 16.04
Node 1
IP: 192.168.205.11
OS: Ubuntu 16.04
Node 2
IP: 192.168.205.12
OS: Ubuntu 16.04



https://github.com/vietkubers/k8s-istio-metallb-hands-on-lab

FUJITSU

shaping tomorrow with you