QCon London
March 2019

The Life of a Packet Through Istio

& the architecture along the way!

Matt Turner

@mt165
mt165.co.uk

Objectives

Learn how a packet traverses an Istio/Envoy/Kubernetes system

See what control plane calls are made in that process

Build a useful mental model for reasoning about, and debugging Istio

Prerequisites

Basic networking knowledge

Intermediate Kubernetes knowledge

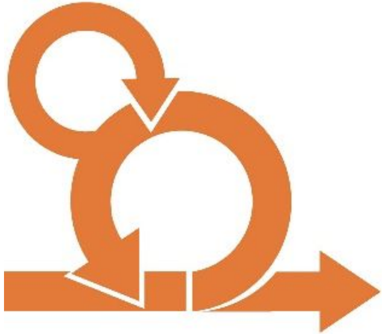
An understanding of what Istio is and does

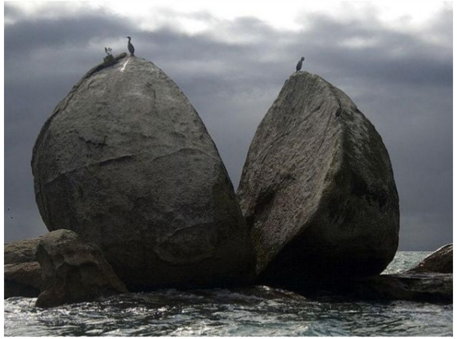
Outline

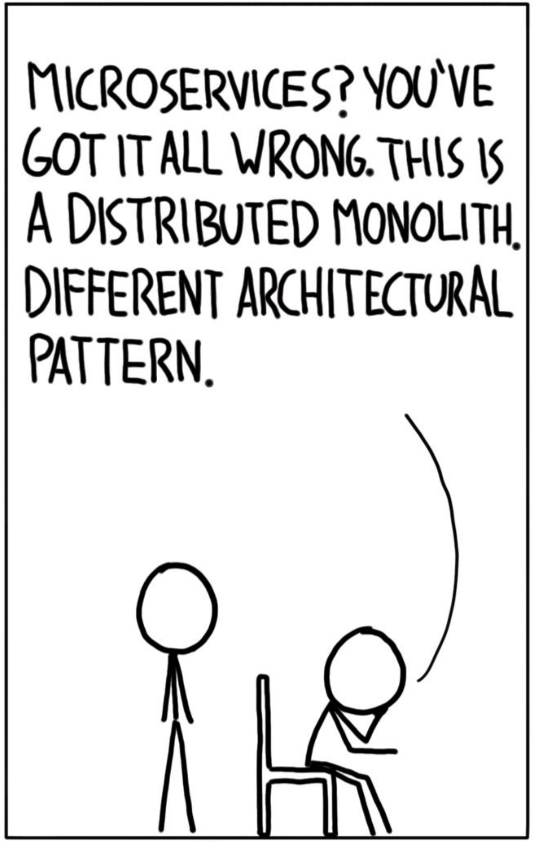
- Context and Introduction
- Networking and Containers
- Pilot and Routing
- Mixer and Policy
- Citadel and mTLS

Context and Introduction

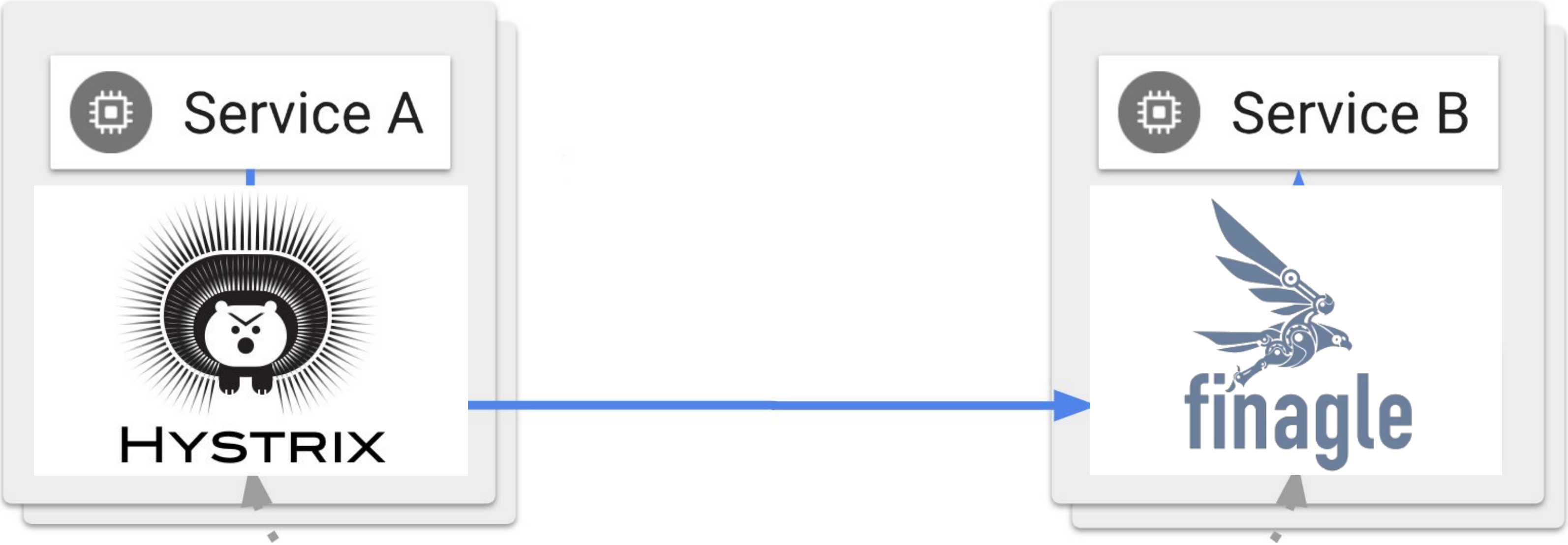
Why?

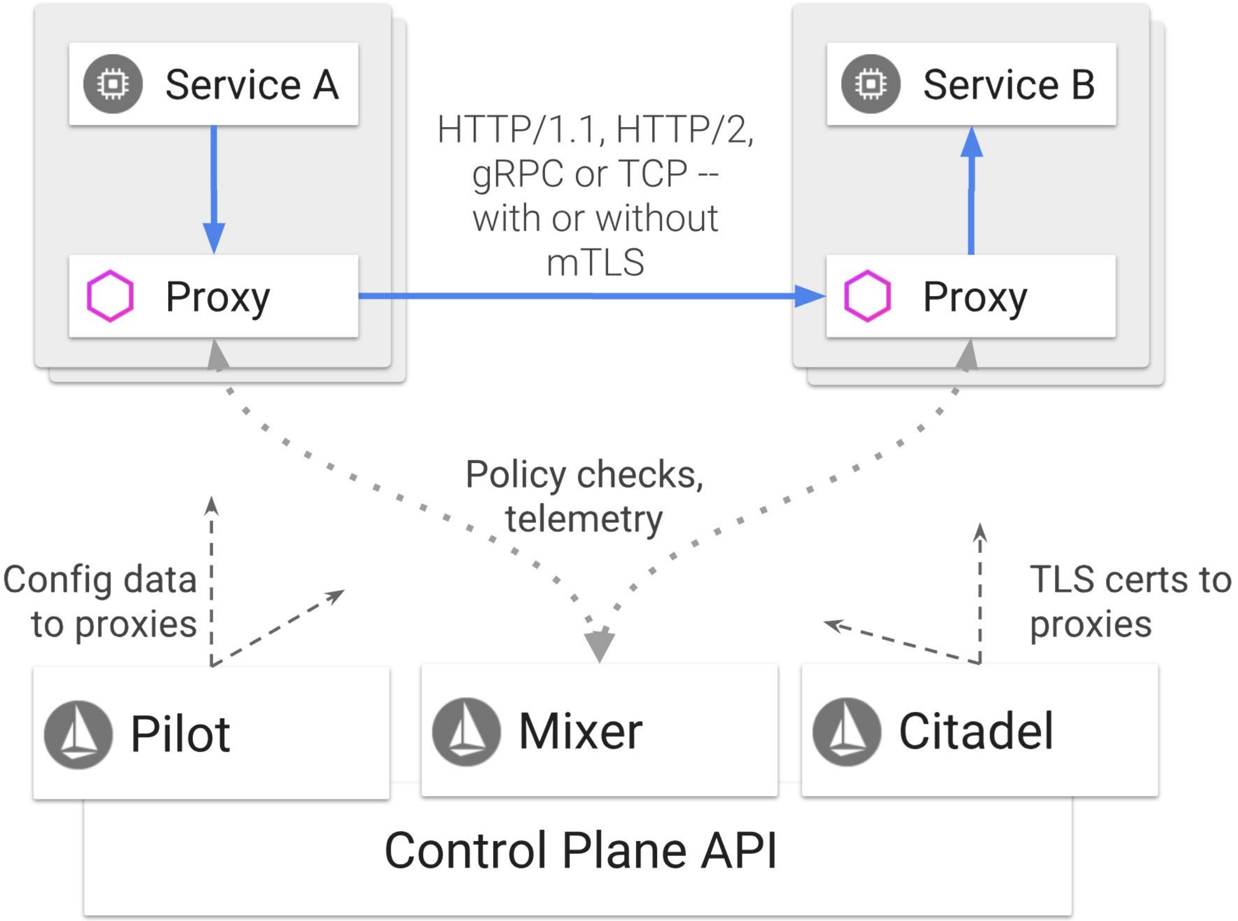





MICROSERVICES? YOU'VE GOT IT ALL WRONG. THIS IS A DISTRIBUTED MONOLITH. DIFFERENT ARCHITECTURAL PATTERN.

@SEBIWICB

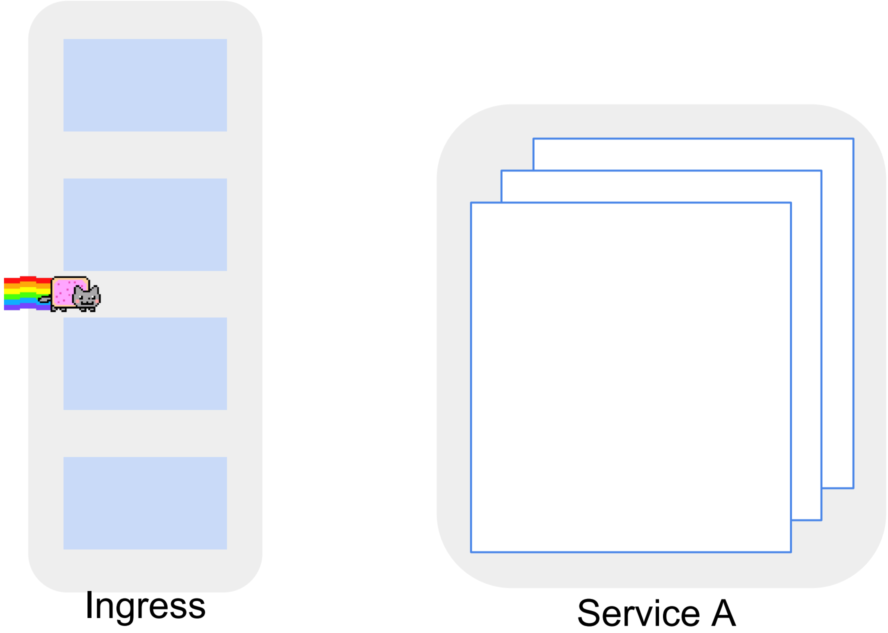

Service A
HYSTRIX
Service B
finagle

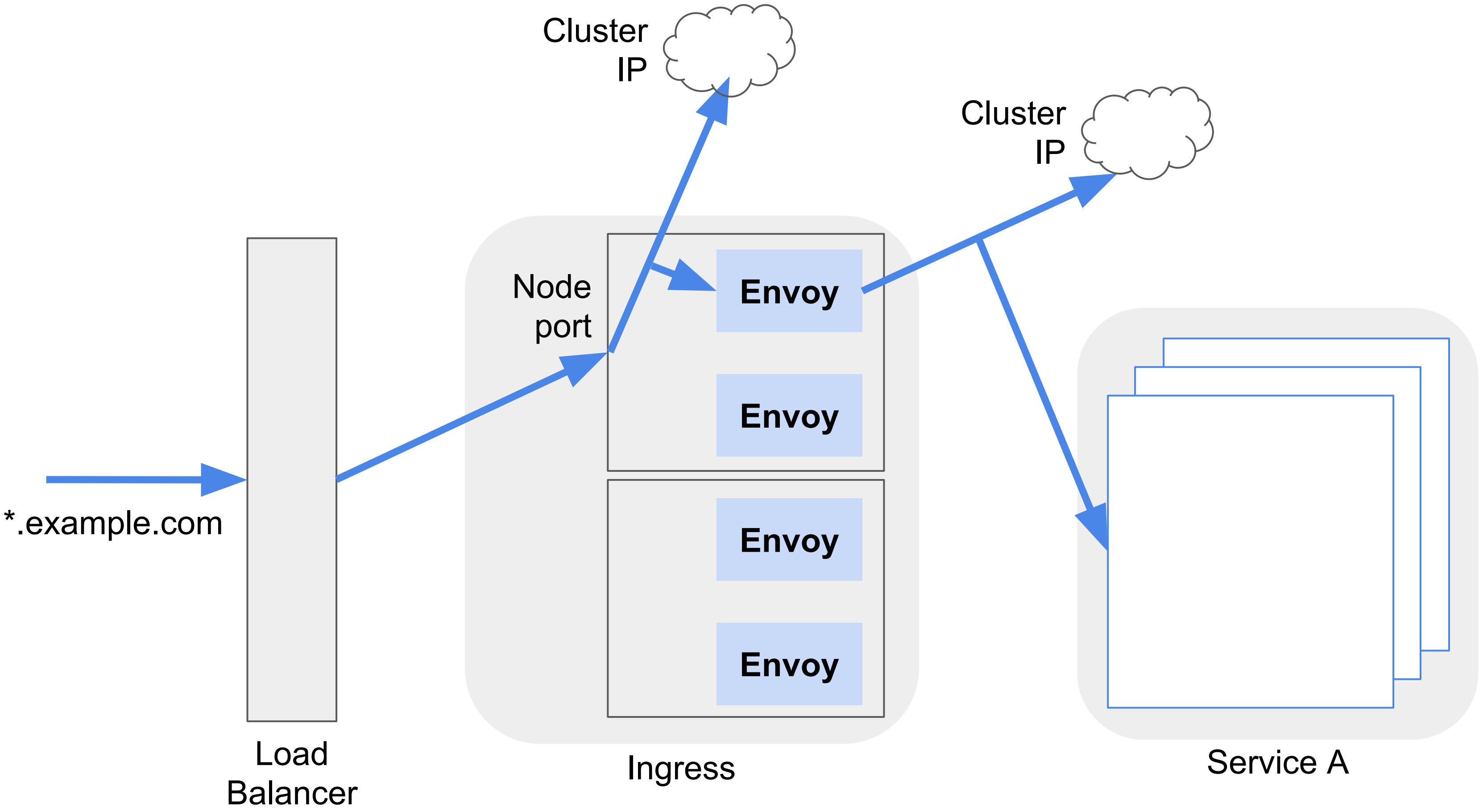

Service A
Proxy
HTTP/1.1, HTTP/2, gRPC or TCP -- with or without mTLS
Service B
Proxy
Policy checks, telemetry
Config data to proxies
TLS certs to proxies
Pilot
Mixer
Citadel
Control Plane API

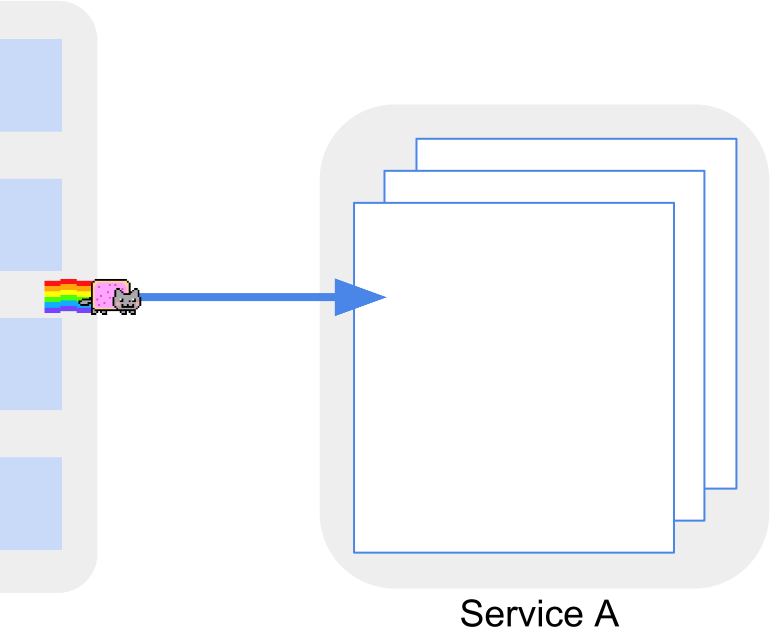
Istio

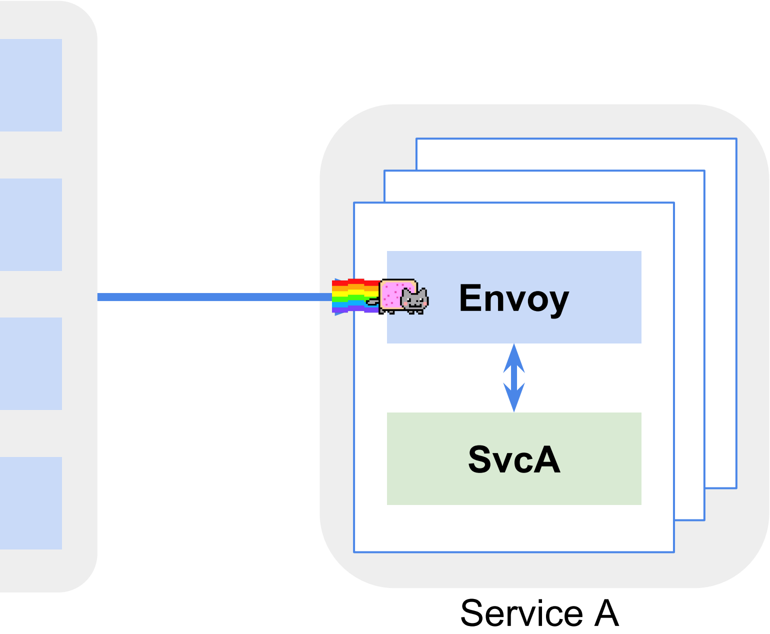
“An open platform to **connect**, **secure**, **control**, and **observe** services.”

Networking and Containers

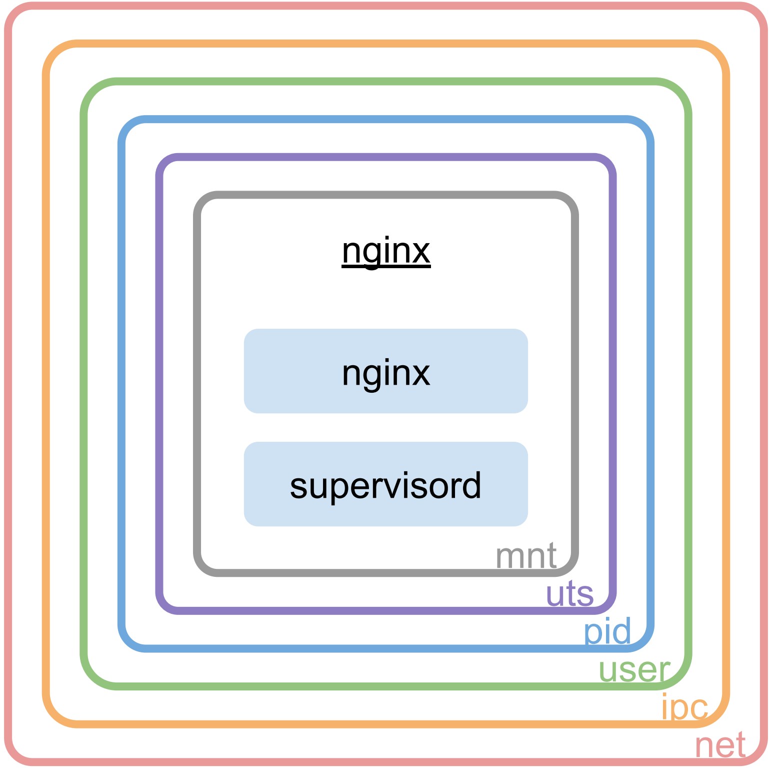

Ingress
Service A


Cluster
IP
Cluster
IP
Node
port
Envoy
Envoy
Envoy
Envoy
*.example.com
Load
Balancer
Ingress
Service A

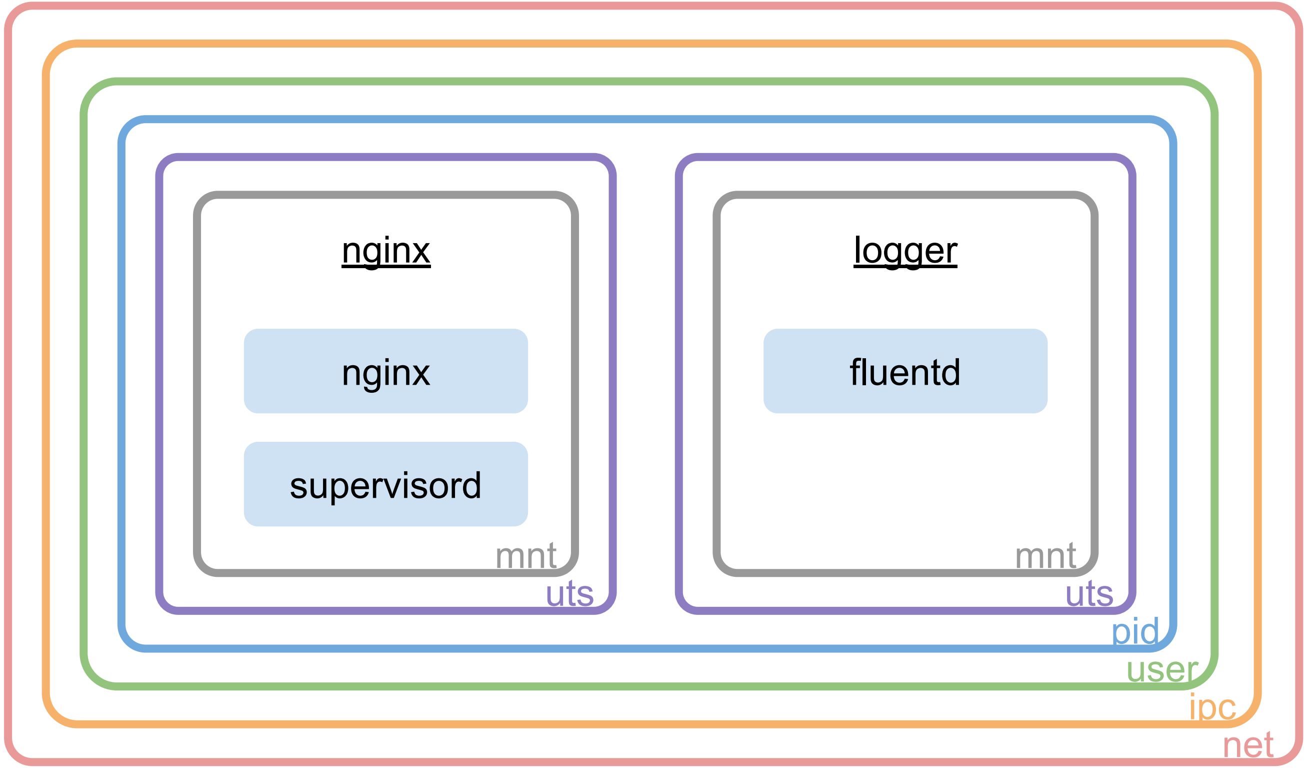

Service A


Envoy
SvcA
Service A

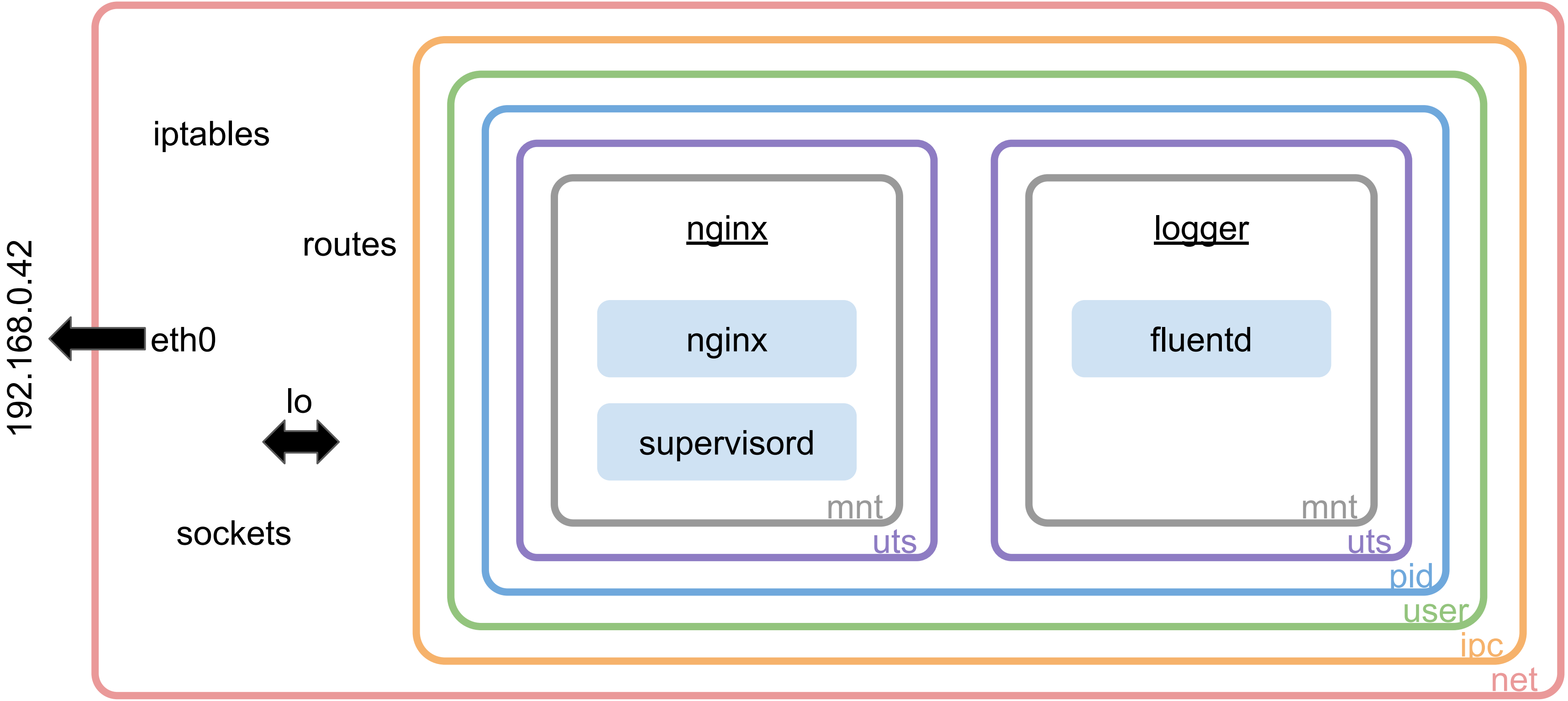
“Containers”



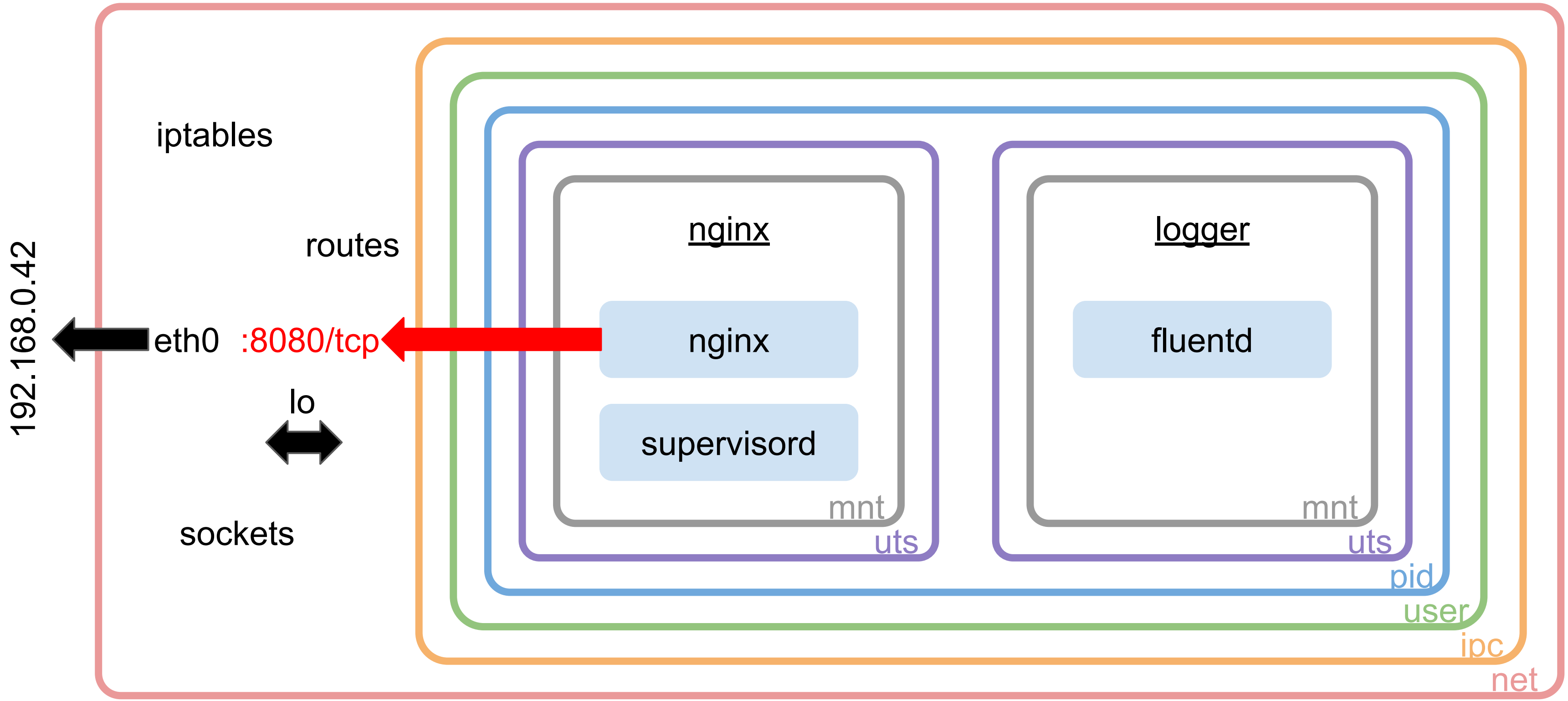
Kubernetes Pods



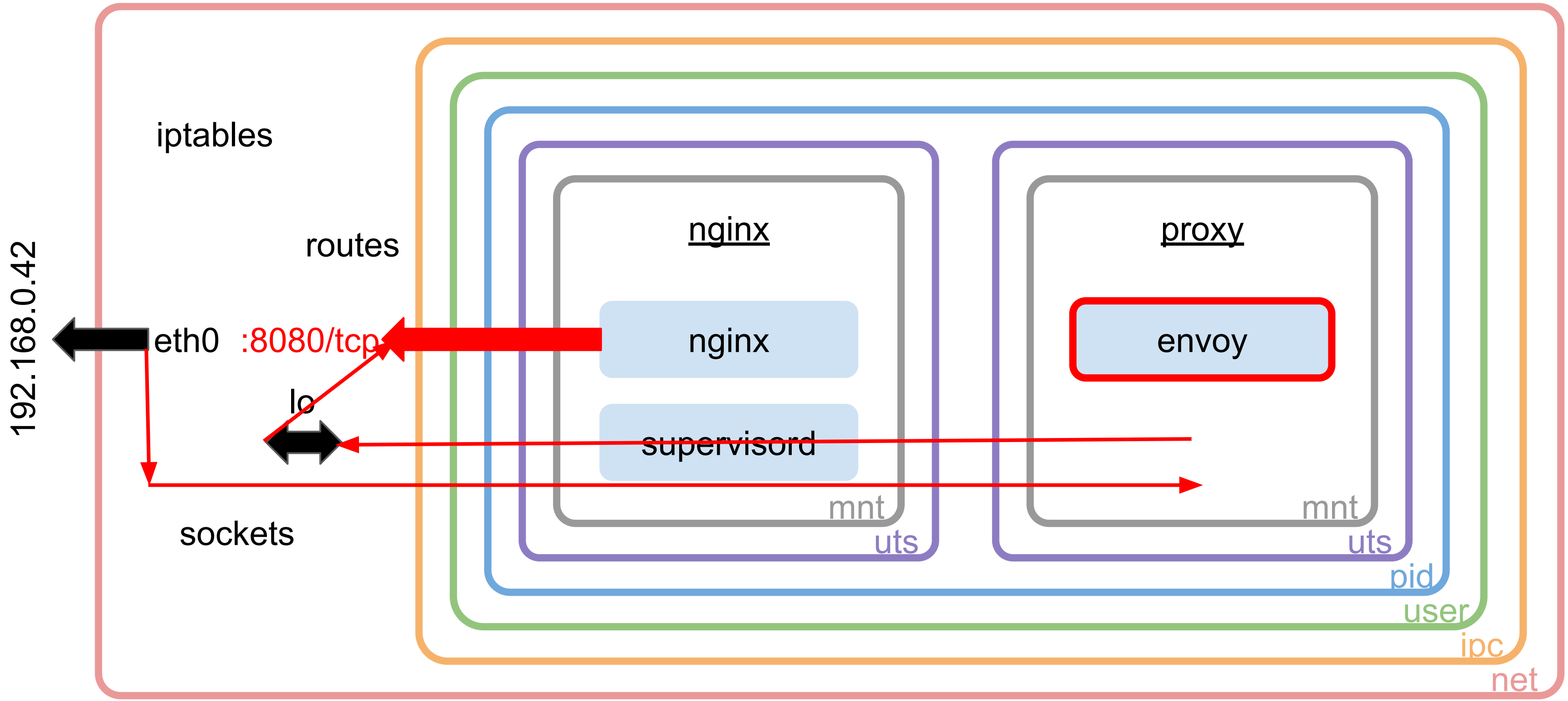
Kubernetes Pods



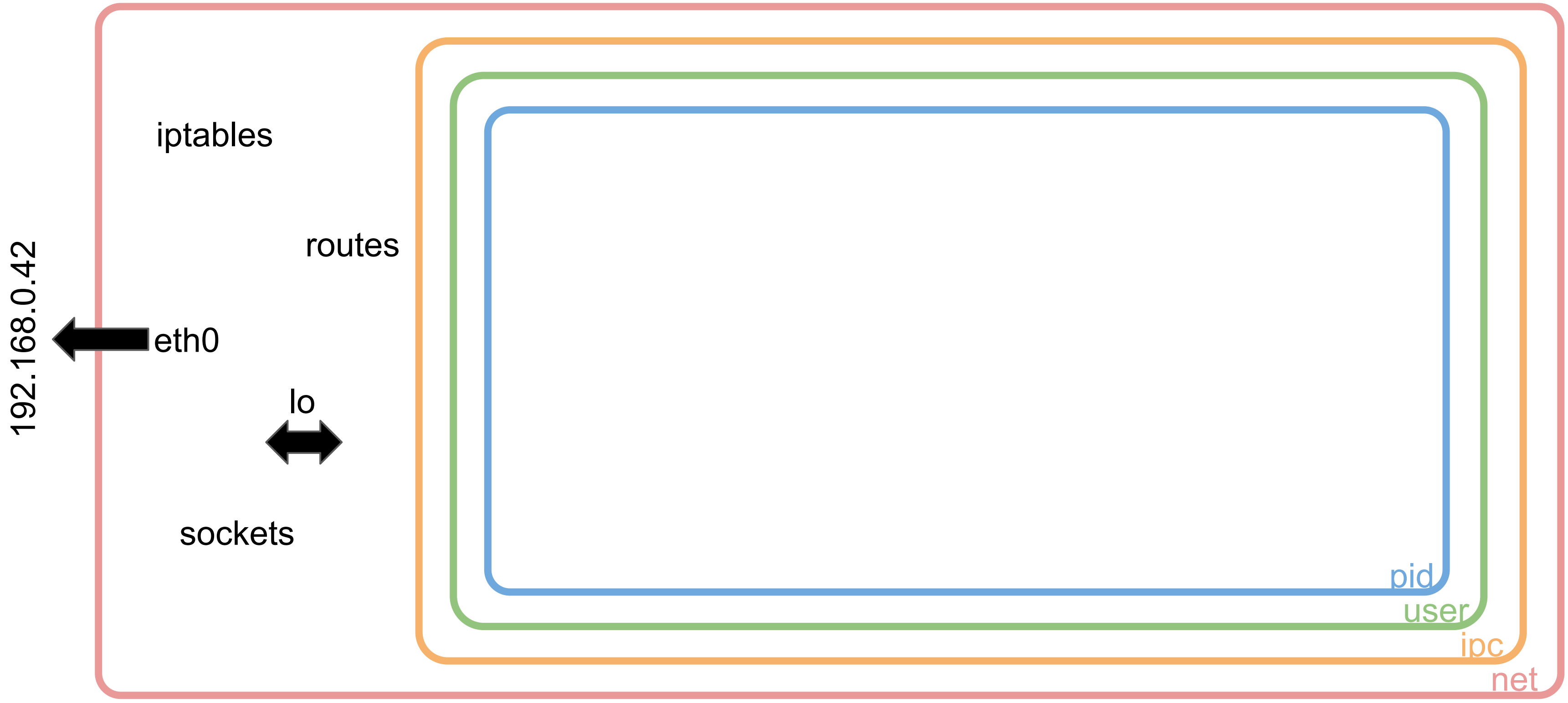
Kubernetes Pods



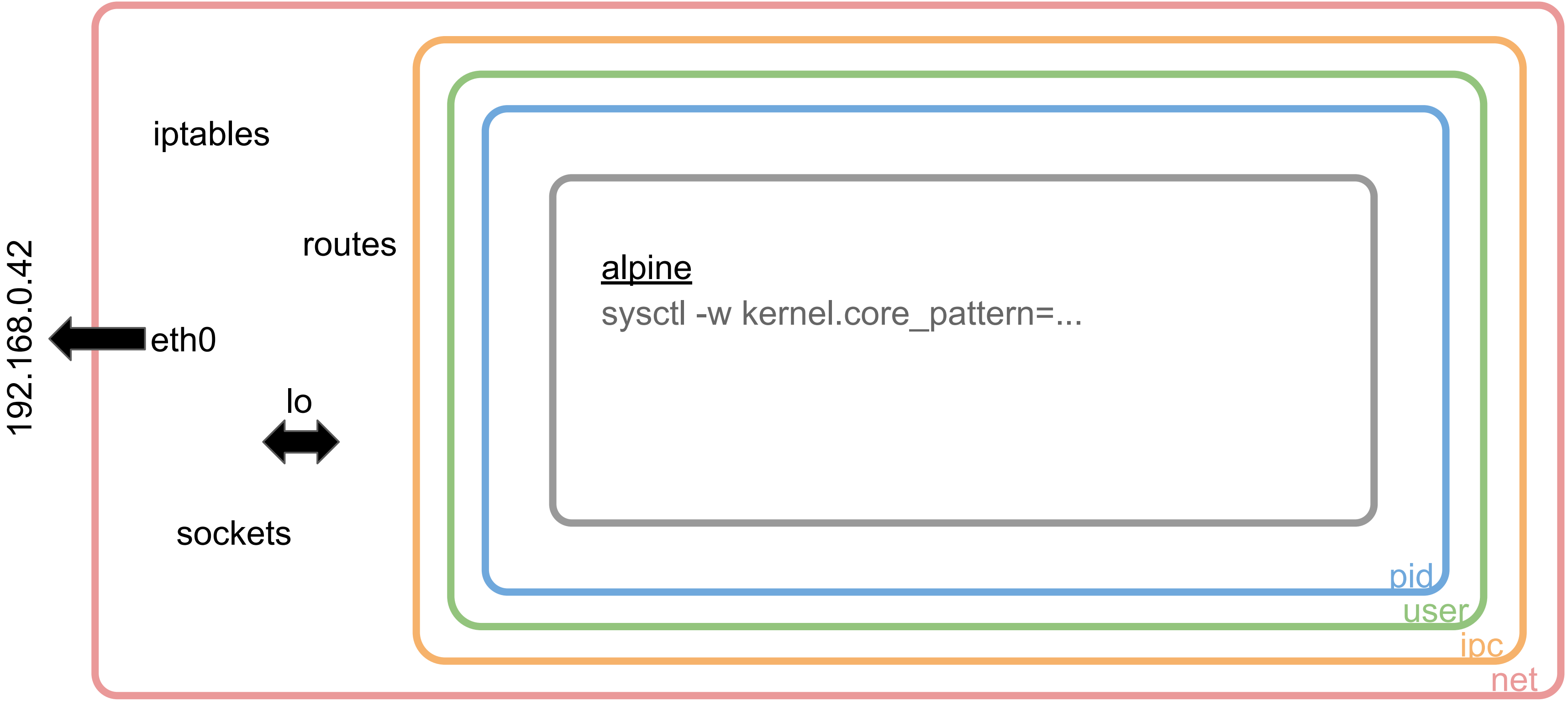
Kubernetes Pods



Sidecar Injection

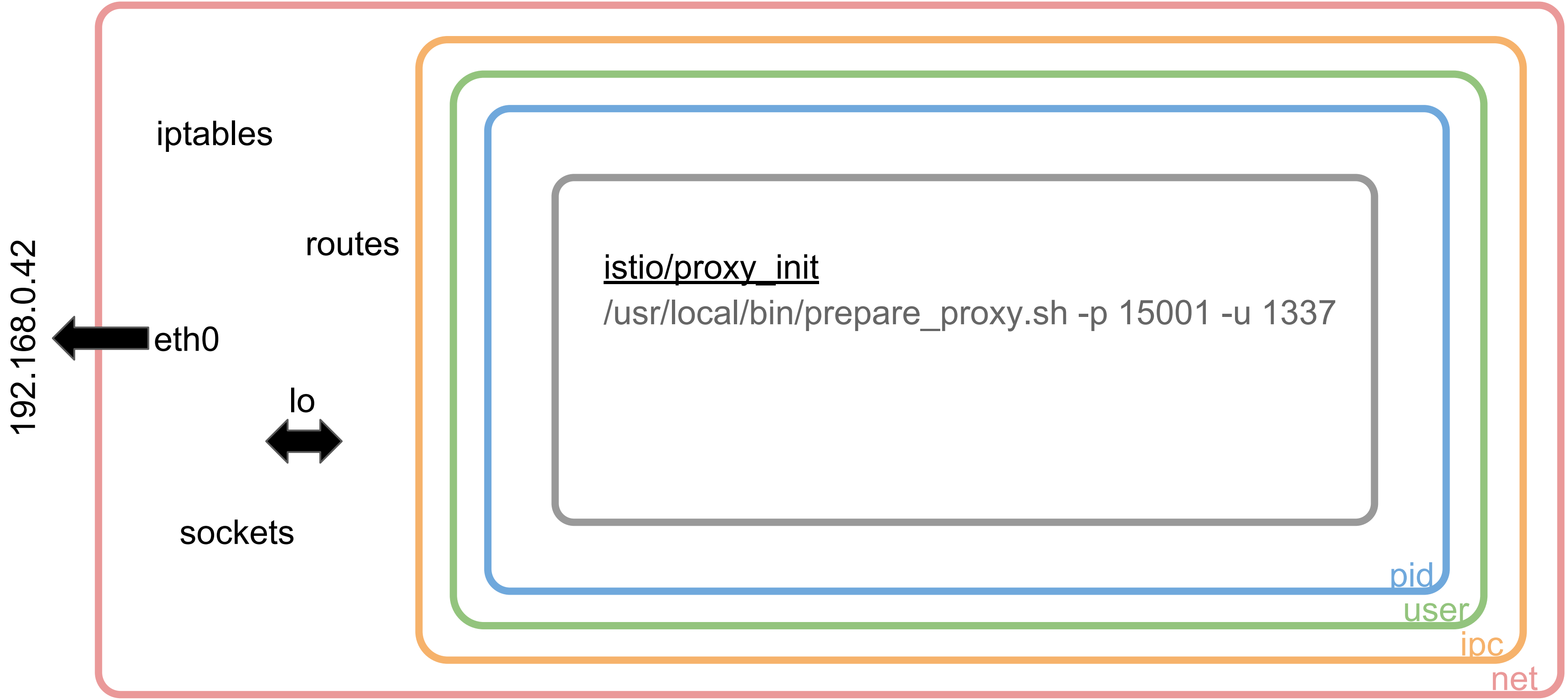


Sidecar Injection



```
sysctl -w kernel.core_pattern=...
```

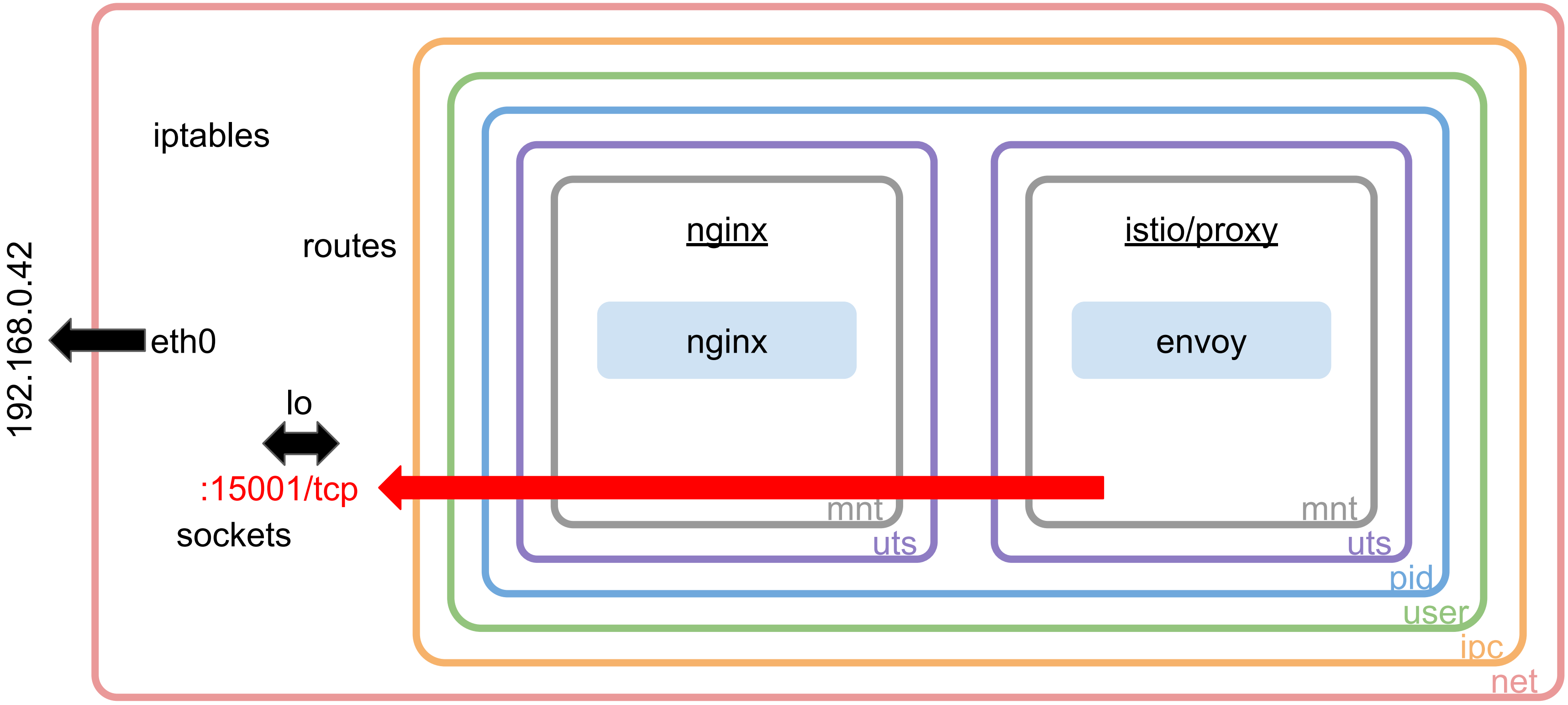
Sidecar Injection

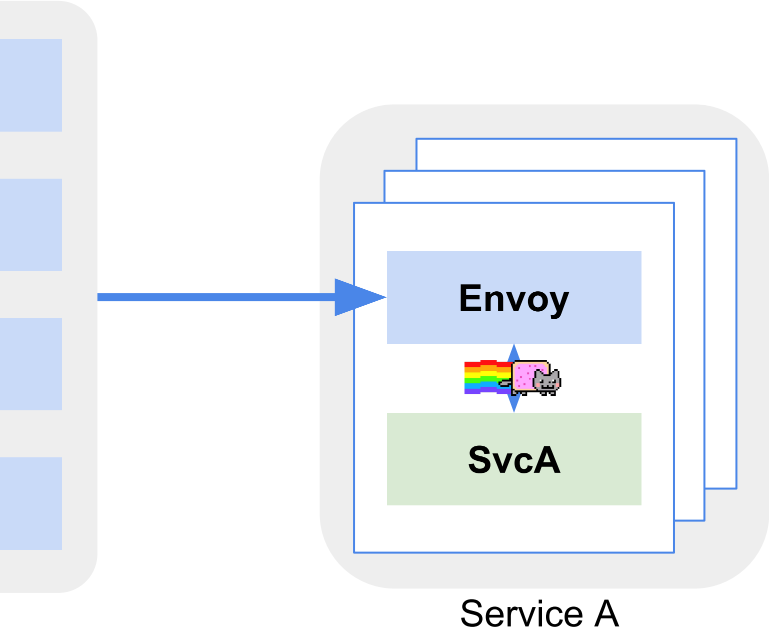


istio/proxy_init

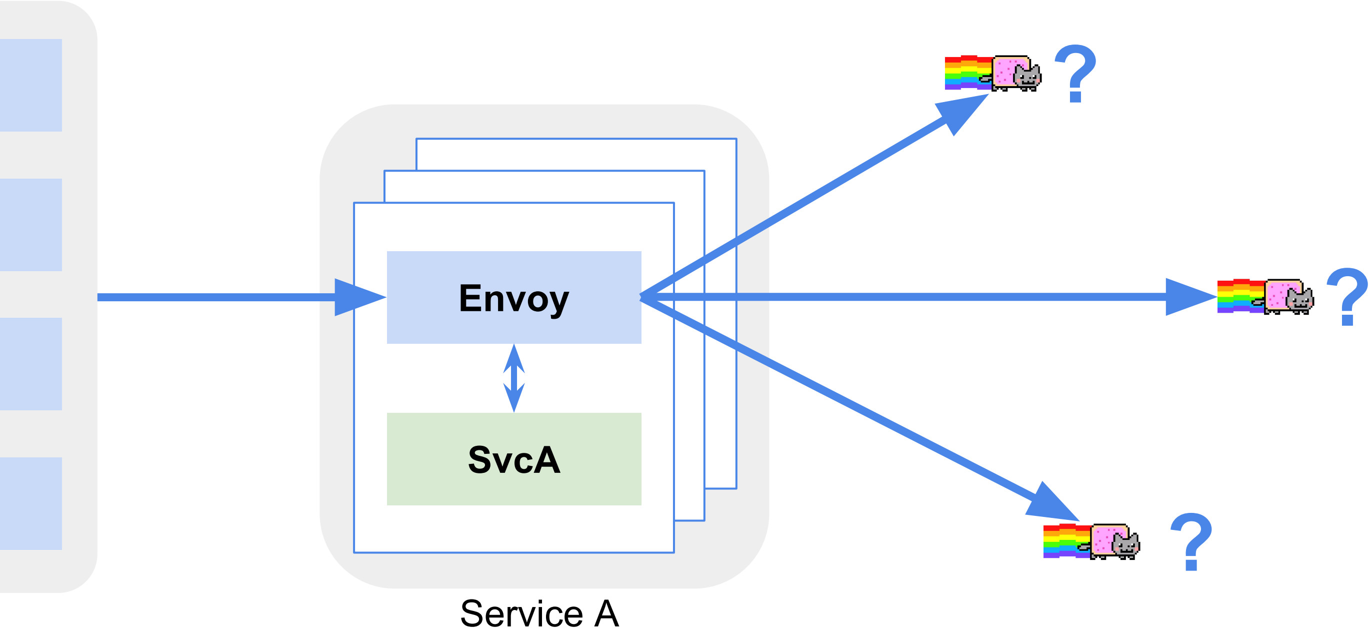
/usr/local/bin/prepare_proxy.sh -p 15001 -u 1337

Sidecar Injection




Envoy
SvcA
Service A

Pilot and Routing


Envoy
SvcA
Service A
?
?
?

Services

```
$ kubectl get service -o wide service-b
NAME        TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE   SELECTOR
service-b   ClusterIP   10.98.84.169   <none>        80/TCP    90s   app=service-b
```

Service DNS exposure

```
$ dig service-b.default.svc.cluster.local.
;; ANSWER SECTION:
service-b.default.svc.cluster.local. 5 IN A 10.98.84.169
```

Pods

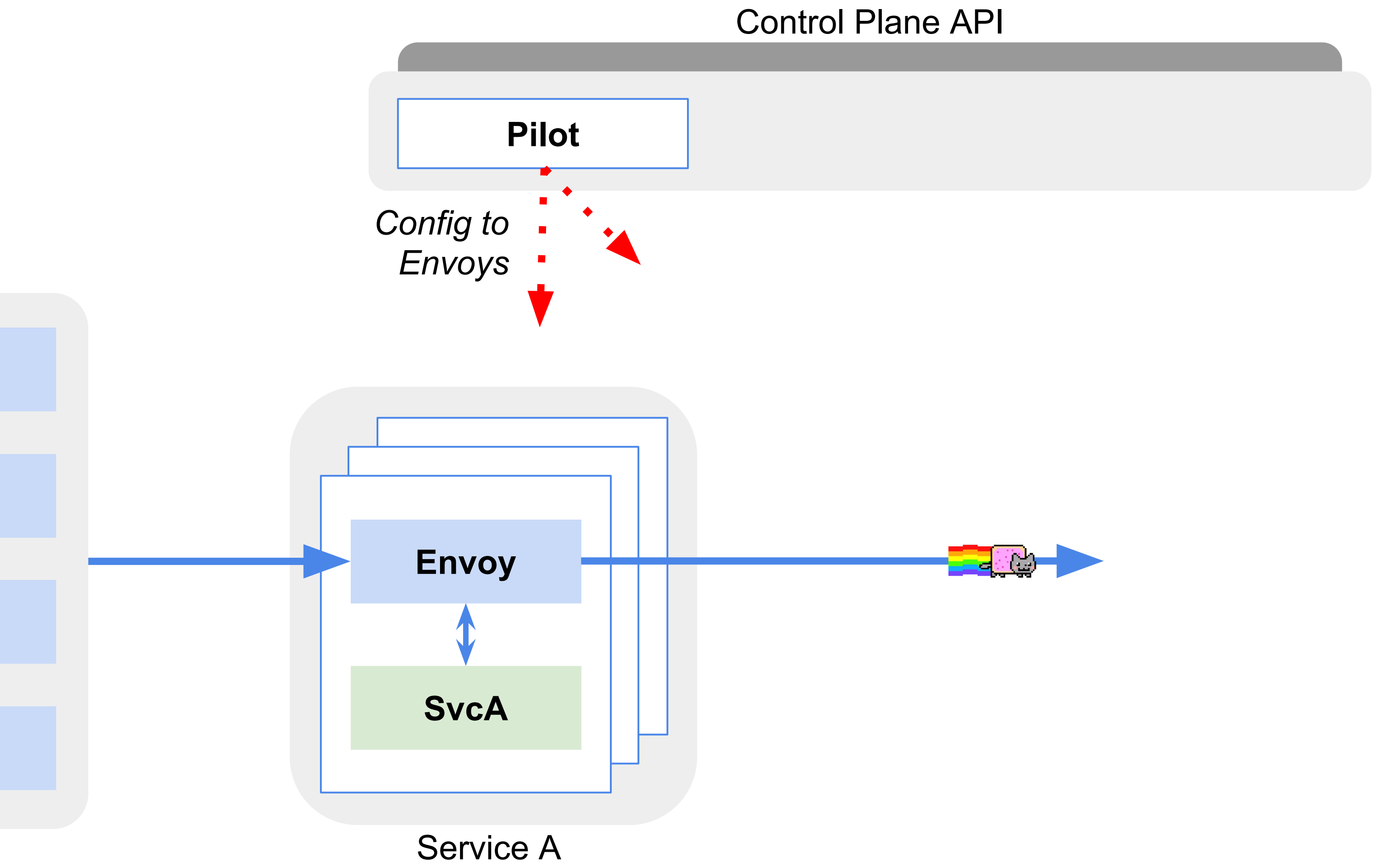
```
$ kubectl get pods -o wide | grep service-b
service-b-644856485c-4rk88   1/1     Running   0          7m46s   10.32.0.4   kind-1-control-plane   <none>
service-b-644856485c-dc2zv   1/1     Running   0          7m46s   10.32.0.6   kind-1-control-plane   <none>
service-b-644856485c-gr75k   1/1     Running   0          7m46s   10.32.0.5   kind-1-control-plane   <none>
```

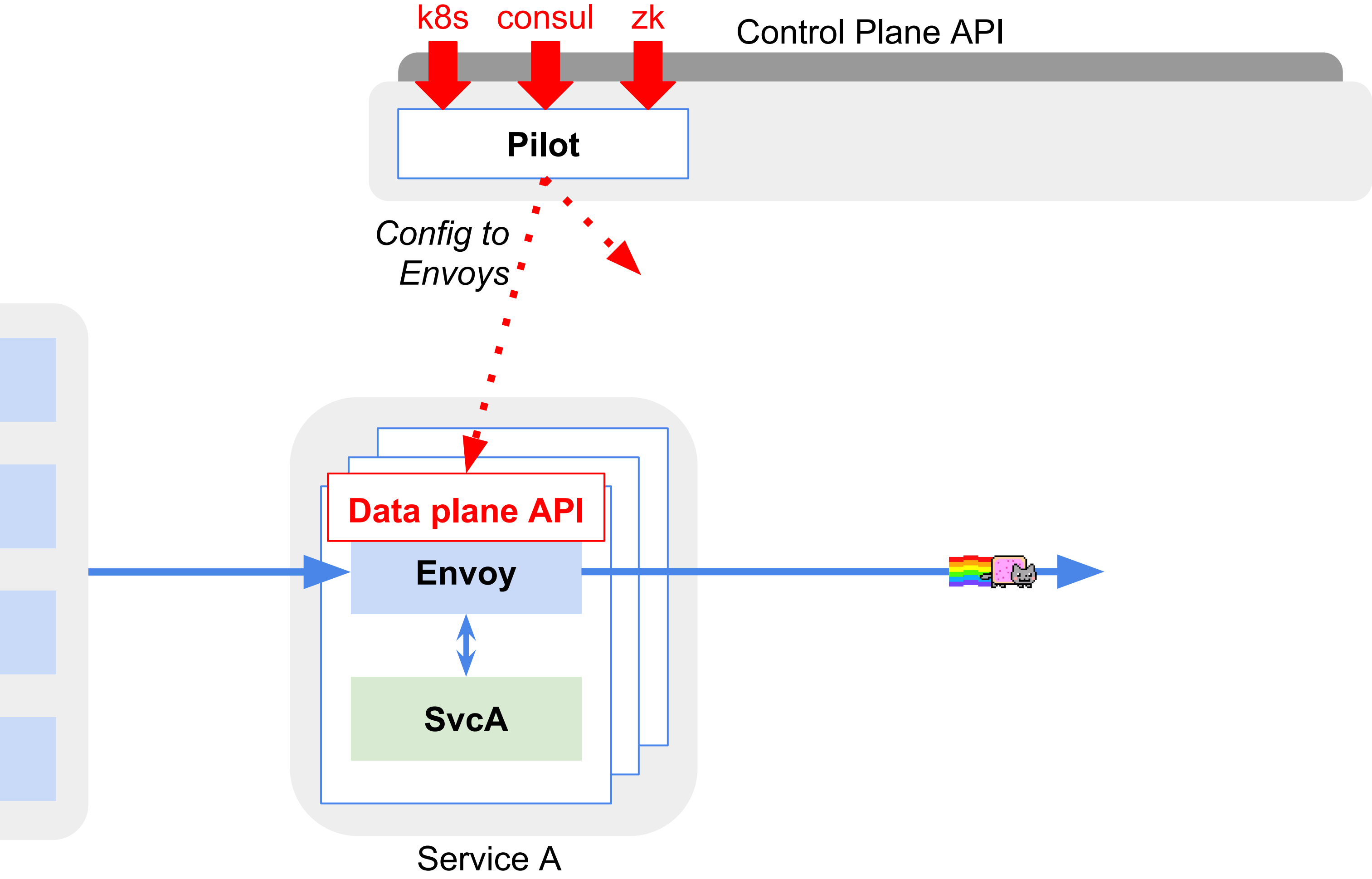
Endpoints

```
$ kubectl get endpoints service-b
NAME        ENDPOINTS                                      AGE
service-b   10.32.0.4:8080,10.32.0.5:8080,10.32.0.6:8080   8m55s
```

Endpoints

```
$ kubectl get endpoints service-b -o yaml
...
subsets:
- addresses:
  - ip: 10.32.0.4
    nodeName: kind-1-control-plane
    targetRef:
      kind: Pod
      …
  ports:
  - name: http
    port: 8080
    protocol: TCP
```

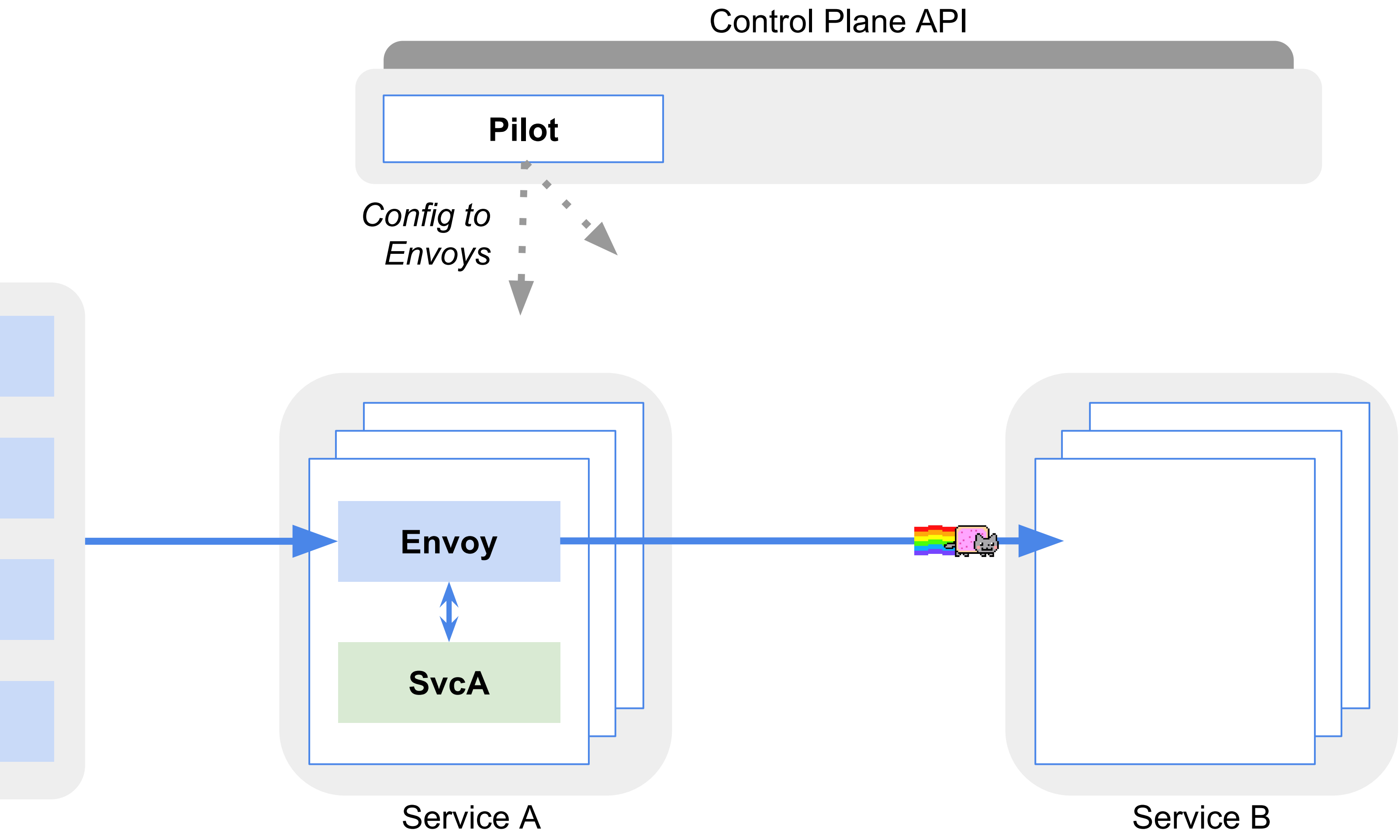

Control Plane API
Pilot
Config to Envoys
Envoy
SvcA
Service A

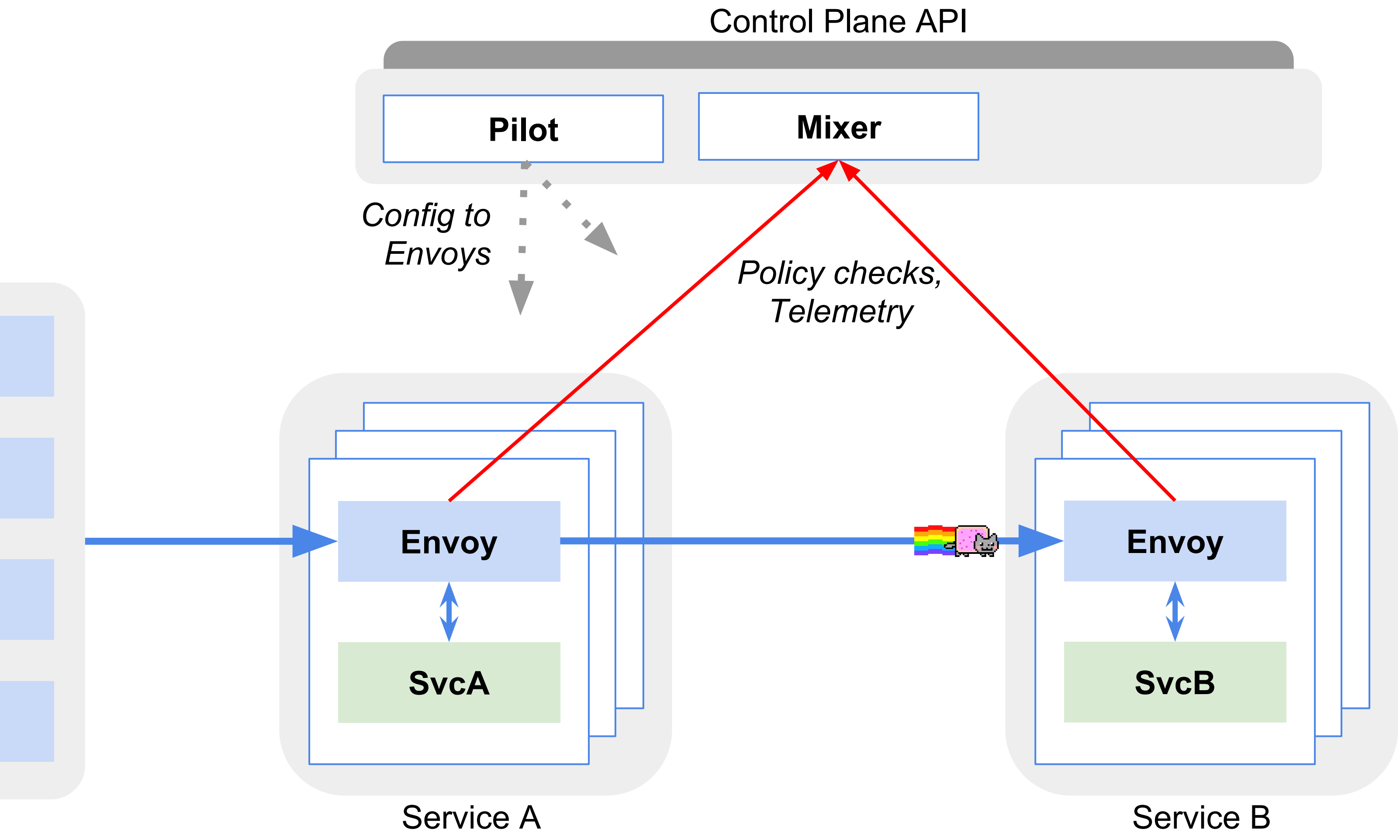

k8s
consul
zk
Control Plane API
Pilot
Config to Envoys
Data plane API
Envoy
SvcA
Service A

Pilot

- Ingress Routing
- Traffic Mirroring
- Traffic Shifting
- Canary Deployments
- Circuit Breaking
- Fault Injection

Mixer and Policy

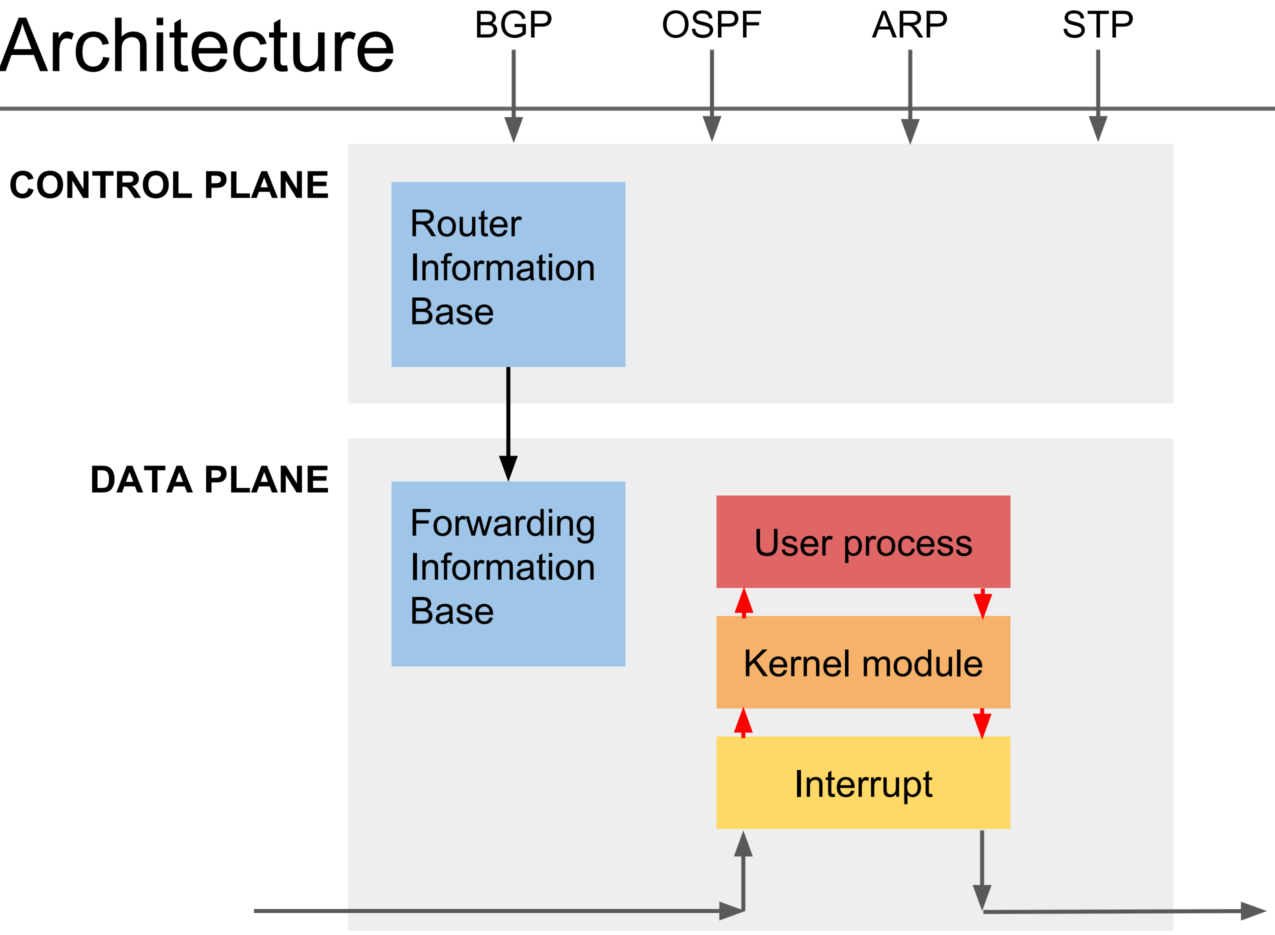

Control Plane API
Pilot
Config to Envoys
Envoy
SvcA
Service A
Service B


Control Plane API
Pilot
Mixer
Config to Envoys
Policy checks, Telemetry
Envoy
SvcA
Envoy
SvcB
Service A
Service B

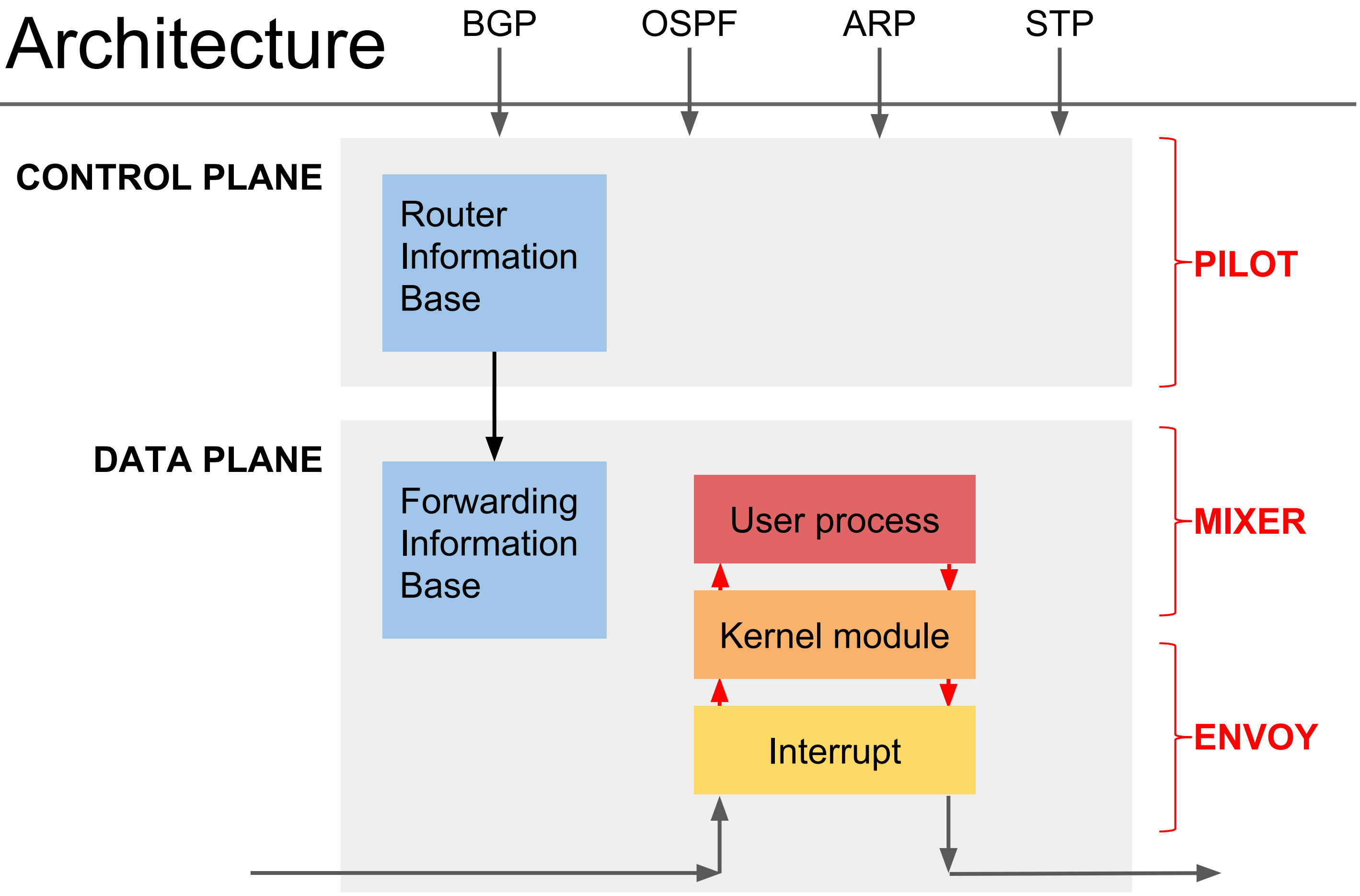
IP 5-tuple

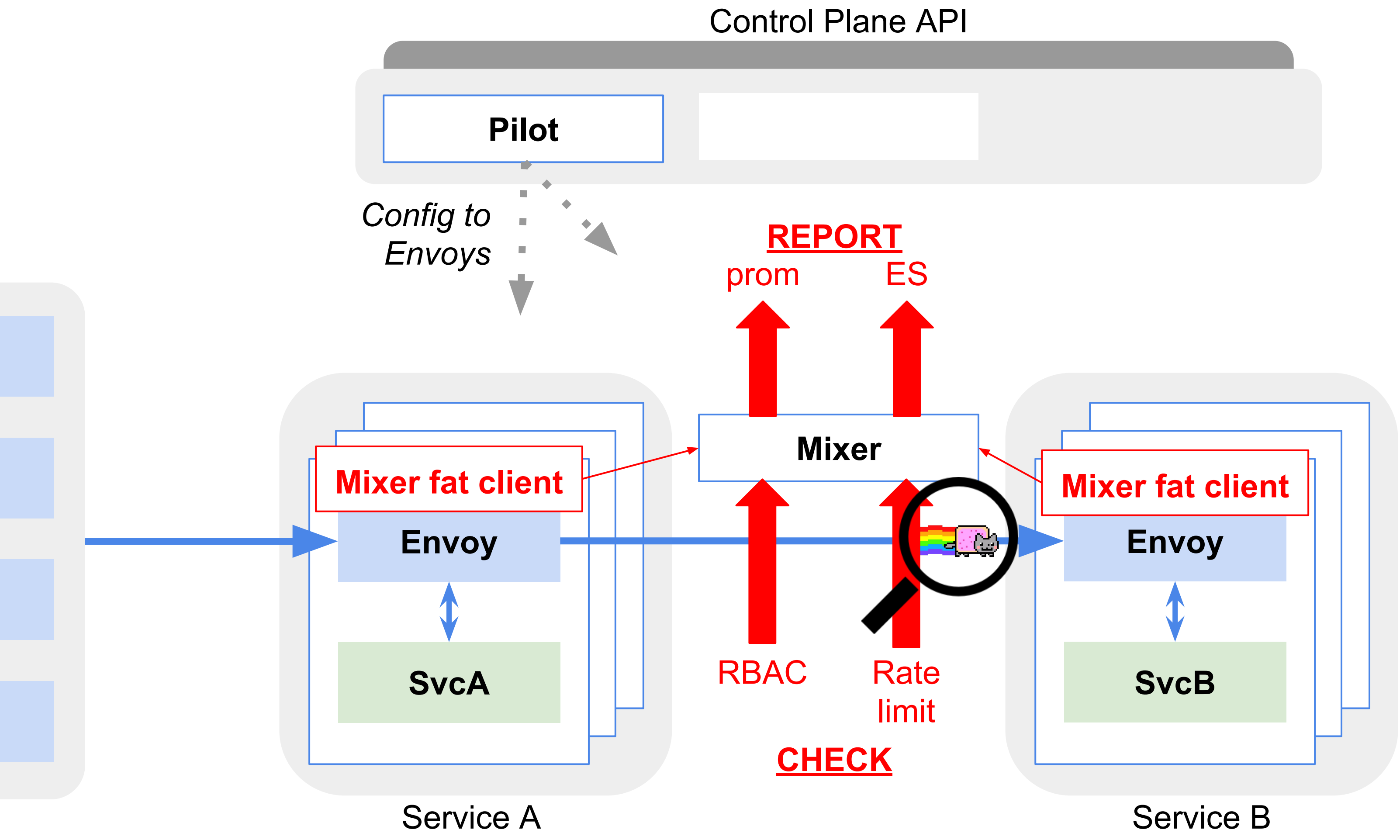
(src_addr, src_port, dst_addr, dst_port, proto)

IP Router Architecture



IP Router Architecture



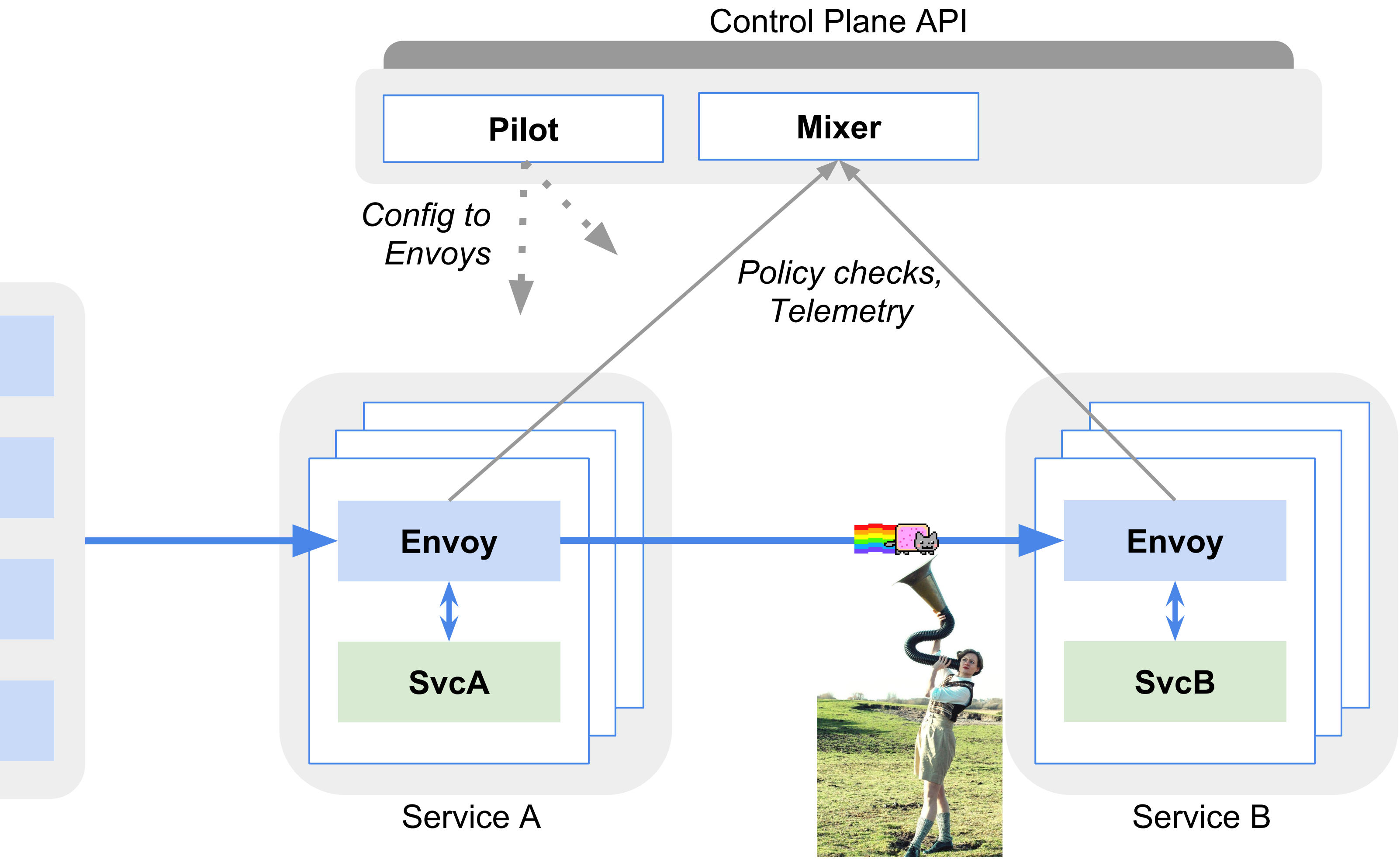

Control Plane API
Pilot
Config to Envoys
REPORT
prom
ES
Mixer
Mixer fat client
Envoy
SvcA
Mixer fat client
Envoy
SvcB
RBAC
Rate limit
CHECK
Service A
Service B

Mixer

- Check
  - ACLs / Authorization
  - Rate Limiting
- Report
  - Logs
  - Metrics
  - Tracing

The life of a packet through Istio

@mt165



Control Plane API

Pilot

Mixer

Config to Envoys

Policy checks, Telemetry

Envoy

SvcA

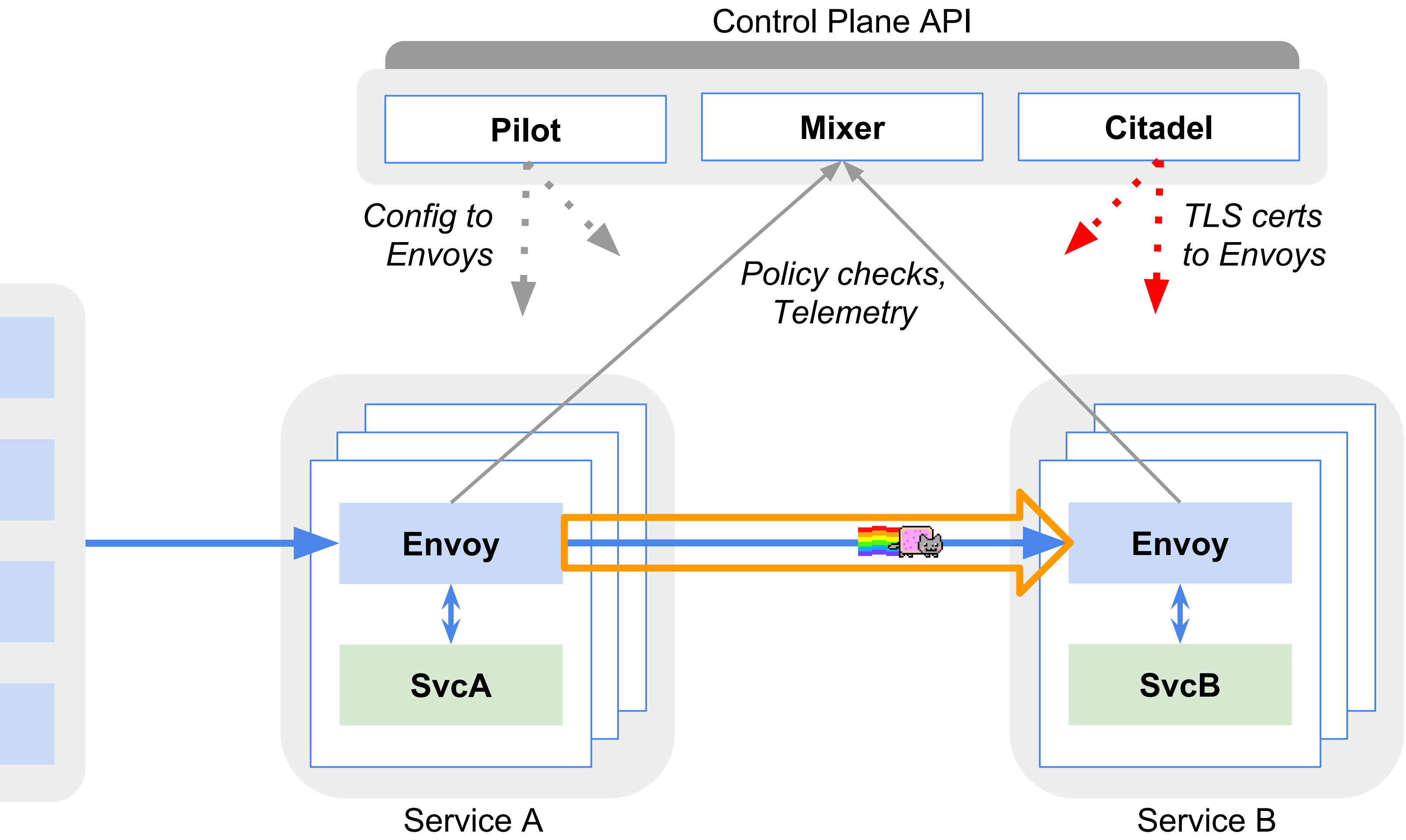
Service A

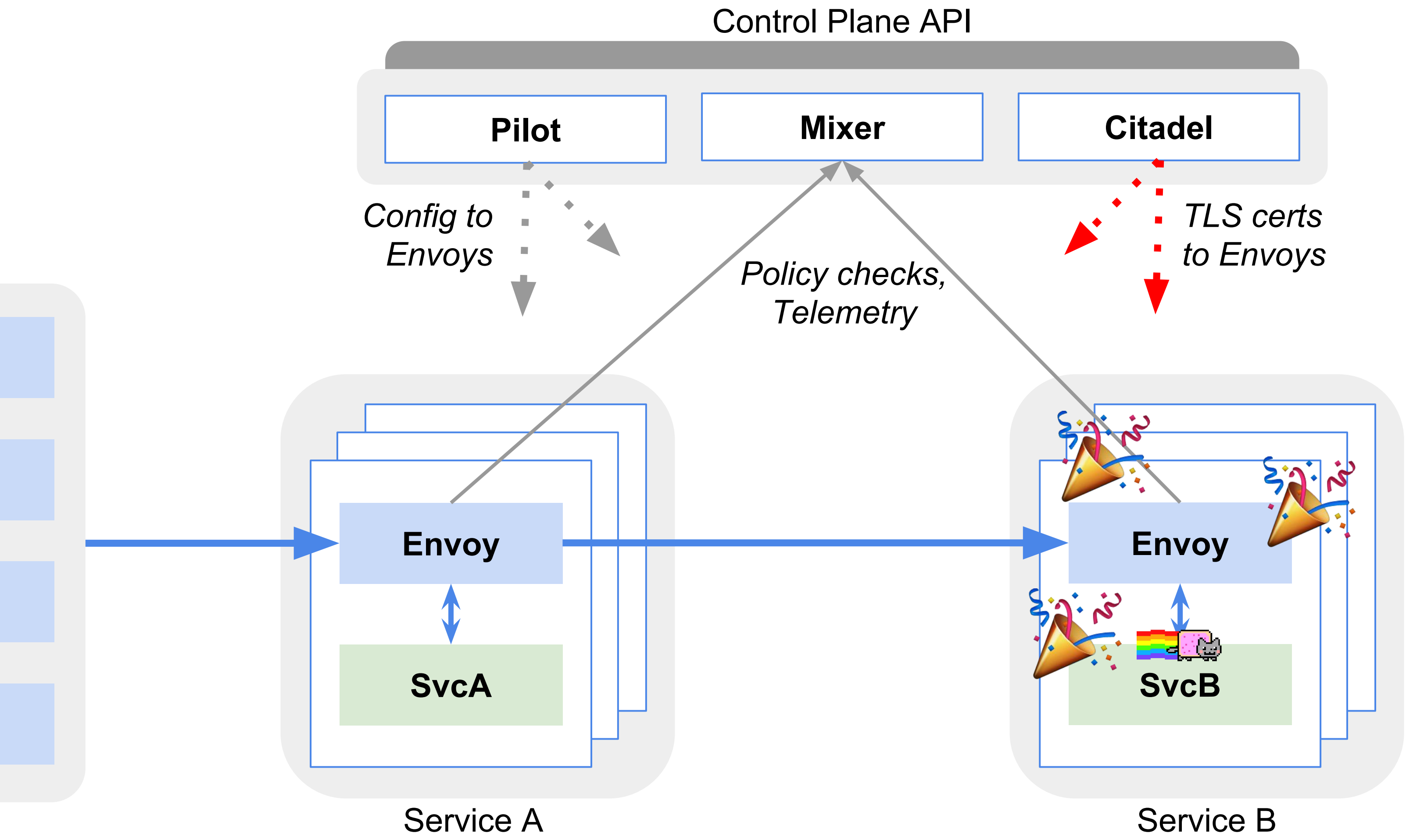
Envoy

SvcB

Service B

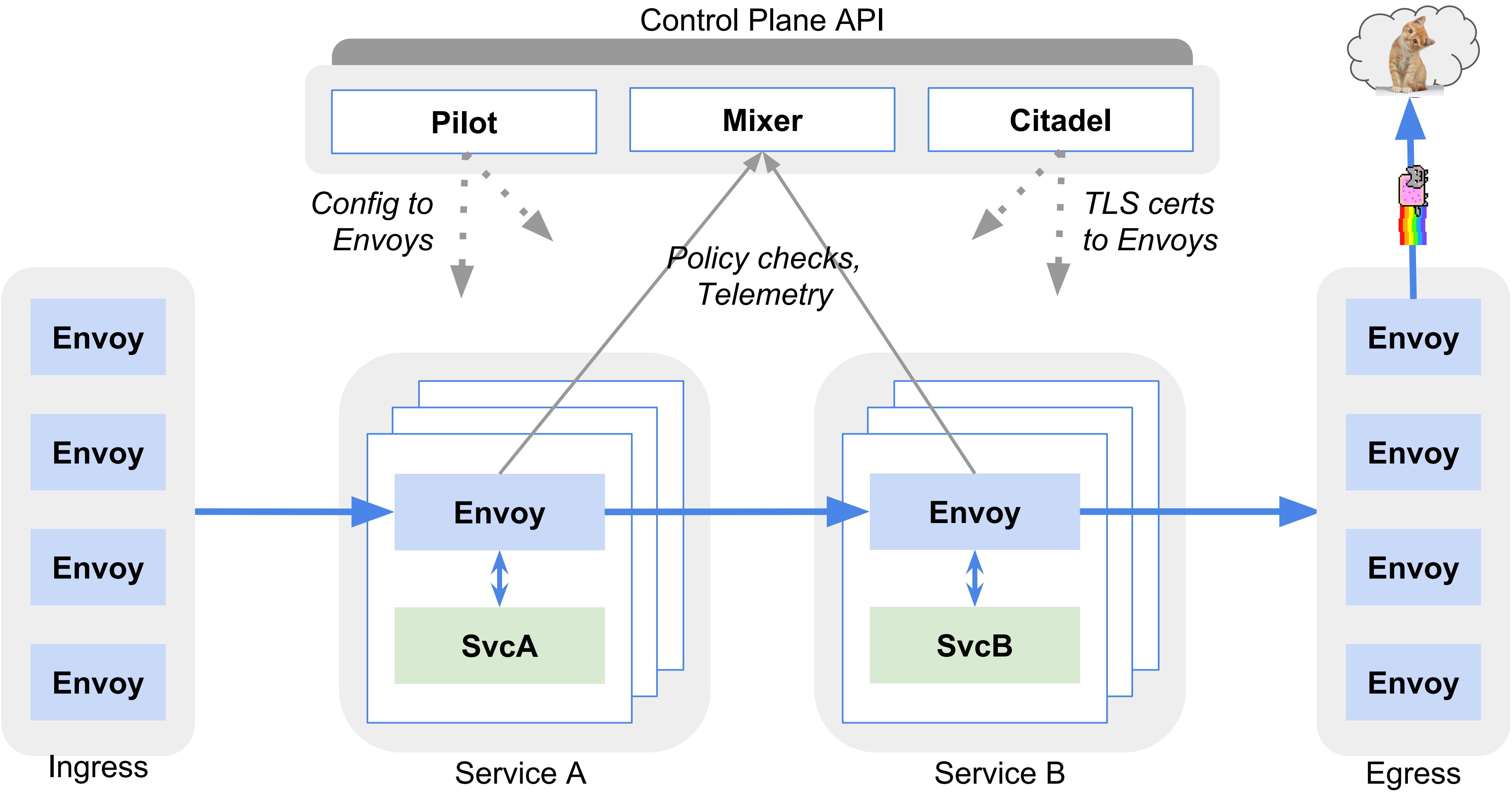
The life of a packet through Istio

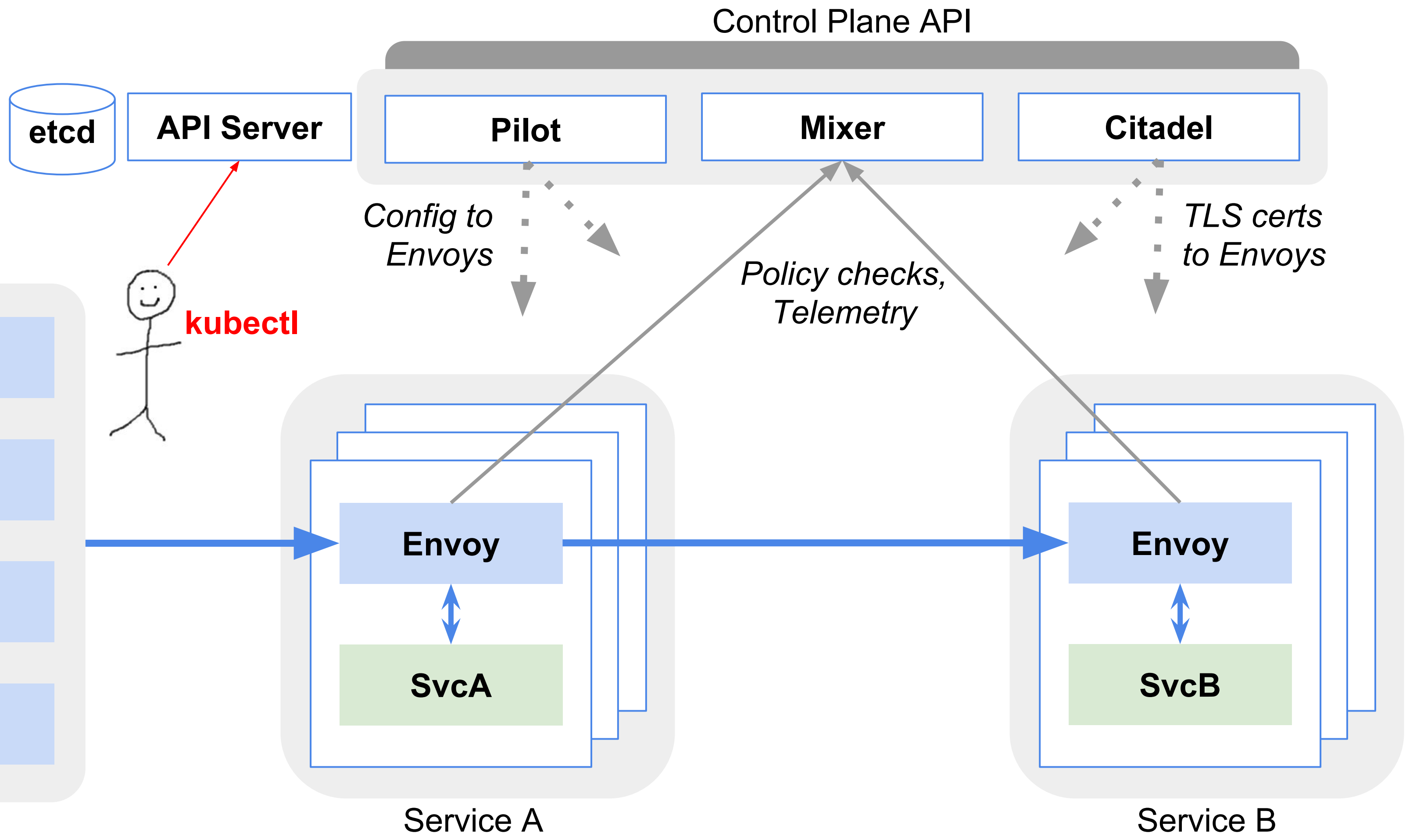


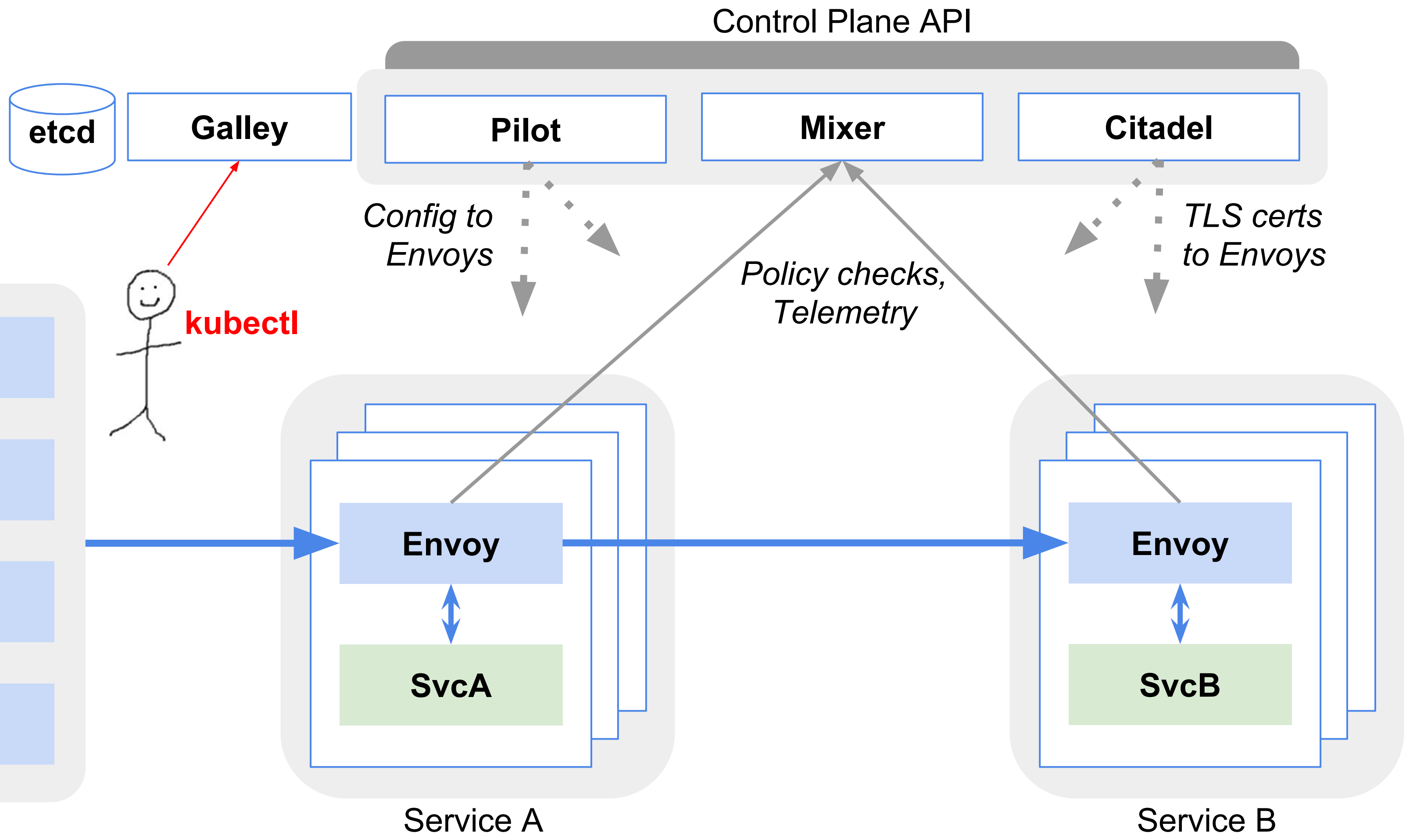

Control Plane API
Pilot
Mixer
Citadel
Config to Envoys
Policy checks, Telemetry
TLS certs to Envoys
Envoy
SvcA
Service A
Envoy
SvcB
Service B

The life of a packet through Istio

@mt165




Control Plane API
etcd
API Server
Pilot
Mixer
Citadel
Config to Envoys
TLS certs to Envoys
Policy checks, Telemetry
kubectl
Envoy
SvcA
Envoy
SvcB
Service A
Service B


Control Plane API
etcd
Galley
Pilot
Mixer
Citadel
Config to Envoys
Policy checks, Telemetry
TLS certs to Envoys
kubectl
Envoy
SvcA
Envoy
SvcB
Service A
Service B

Outline

- Context and Introduction
- Networking and Containers
- Pilot and Routing
- Mixer and Policy
- Citadel and mTLS

Recap

We learned:

- How a packet traverses an Istio/Envoy/Kubernetes system
- What control plane calls are made in that process
- A useful mental model for reasoning about, and debugging Istio

Thanks!

@mt165