


KubeCon | CloudNativeCon

Europe 2020

Virtual

Mutual TLS Adoption

Jianfei Hu/Google, Lizan Zhou/Tetrate

Intro

Lizan Zhou, Tetrate.io
Envoy maintainer, Istio Security WG

Jianfei Hu, Google
Istio Security WG

Agenda

- Why mTLS
- Why mTLS adoption is hard
- Our approach
- Summary and Lessons
- QA

Why mTLS

What is mTLS

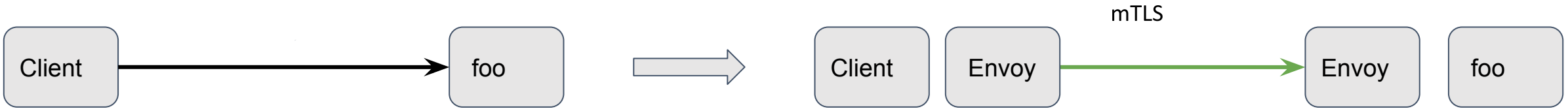
- extension of TLS that authenticate in both directions

Why mTLS

- strong identity with trust chain
- service-to-service authentication at transport layer
- encryption and integrity

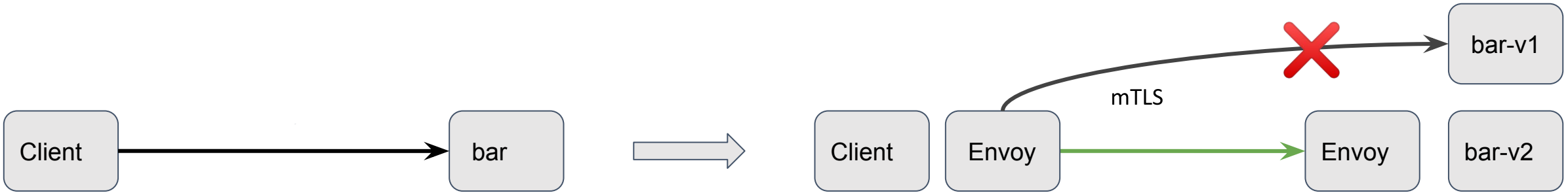
Why mTLS is hard?

In ideal world, just terminate mutual TLS between sidecar



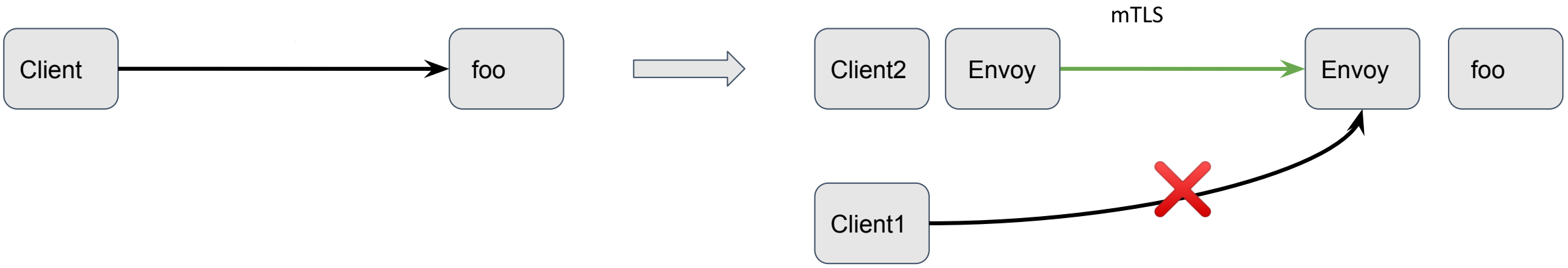
Why mTLS adoption is hard?

Incomplete server sidecar rollout



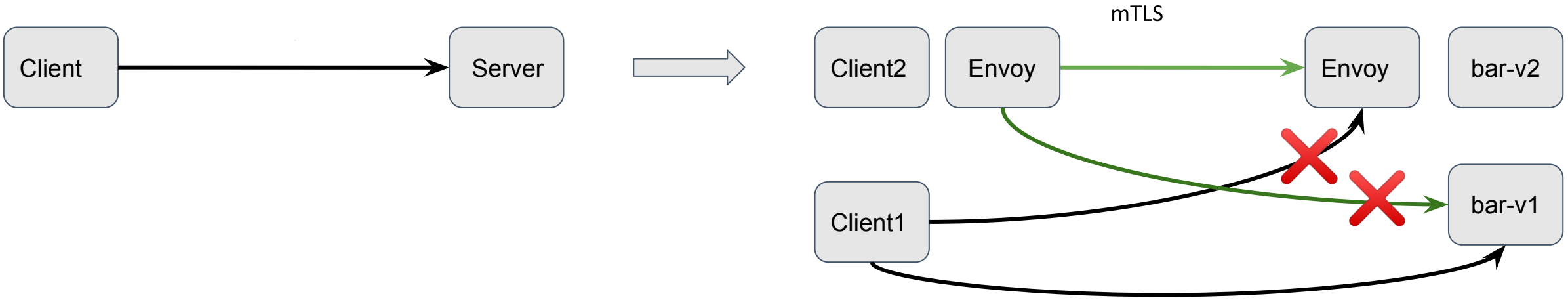
Why mTLS is hard?

Incomplete client sidecar rollout



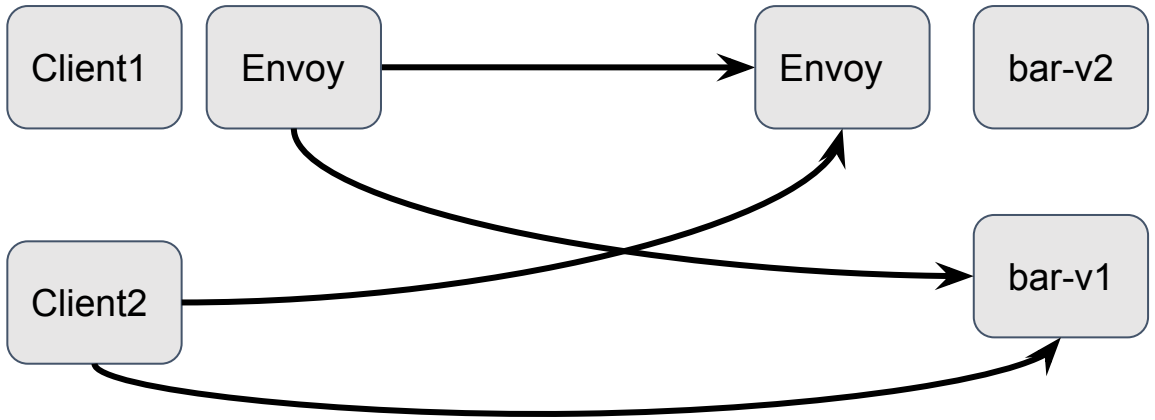
Why mTLS is hard?

Both can happen

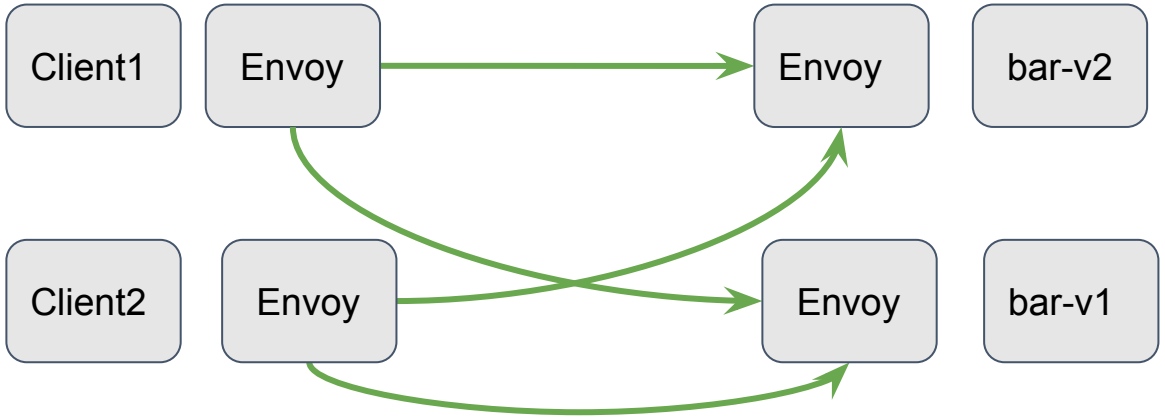


Manual approach

Start with plain text everywhere.



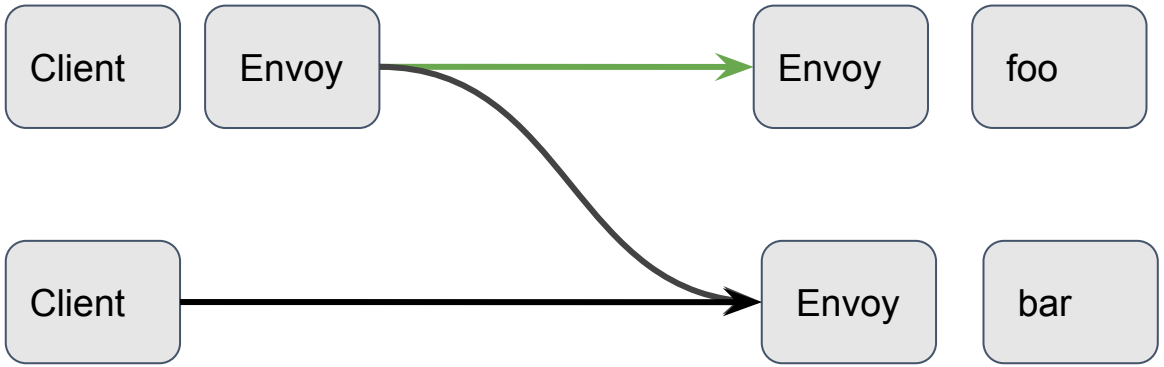
mTLS only after sidecar everywhere



- User tracking sidecar rollout progress.
- Configure server & client.
- Different teams.
- Rollout mTLS all at once.

Manual approach, config

Service by Service



Istio Configuration

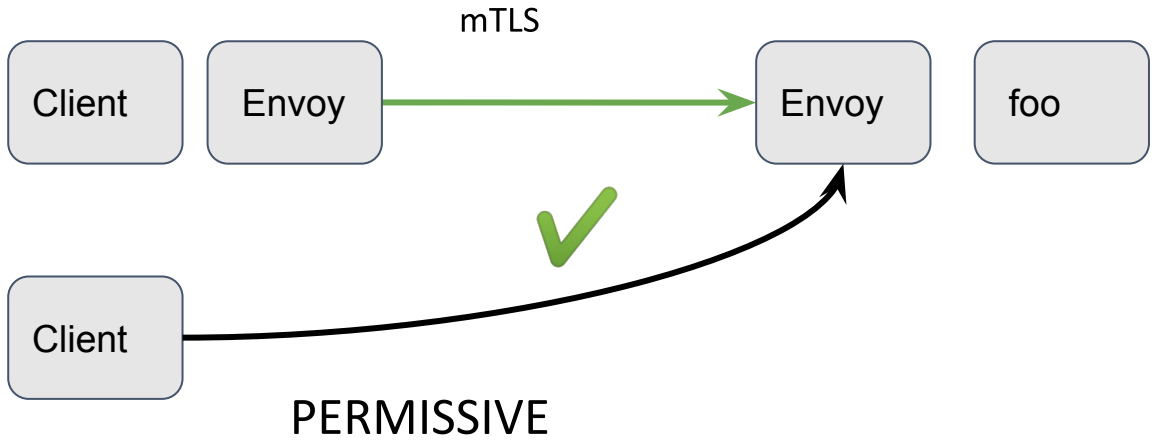
```
DestinationRule:
 -  host: "foo"
    tls:
      mode: ISTIO_MUTUAL


AuthenticationPolicy:
  - host: foo
     tls: enabled
```

Server Improvement, sniffing

- Envoy server TLS sniffing, detecting TLS and plain text traffic
- **Server able to serve both**
- Istio API: PERMISSIVE & STRICT

```
server_listener:
  name: x
  listener_filters:
    - name: "envoy.listener.tls_inspector"
  filter_chain:
  - filter_chain_match:
      transport_protocol: "tls"
    transport_socket:
      name: envoy.transport_socket.tls
  - transport_socket:
      name: envoy.transport_socket.raw_buffer
```



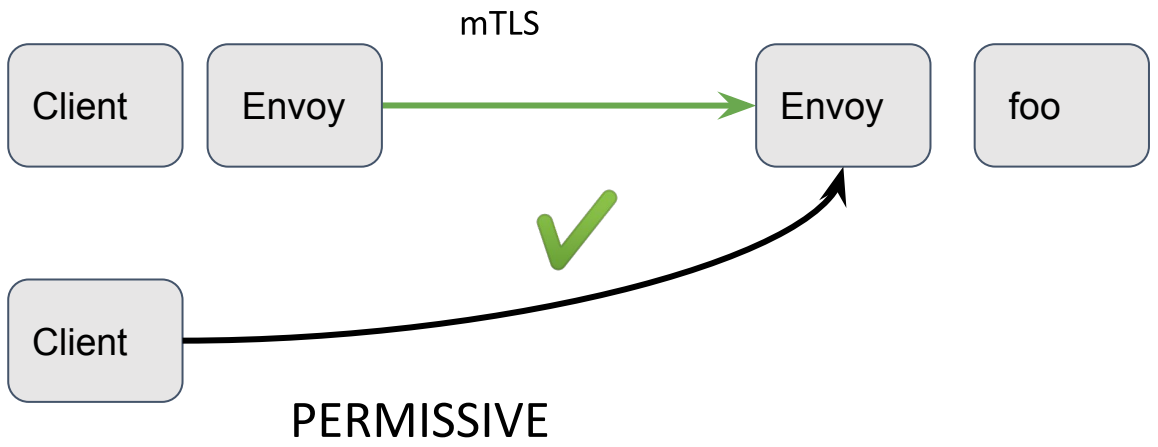
Server Improvement, sniffing

How it works

- TLS sniffing inspect first packet from client, i.e. ClientHello in TLS
- TLS, ALPN, SNI are extracted for filter chain match

Cons

- Add latency to server first protocol (e.g. MySQL) when they are used without TLS



Server Improvement, new config

Istio Configuration

```
DestinationRule:
-  host: foo
   tls:
      mode: ISTIO_MUTUAL
- host: payroll
   tls:
      mode: ISTIO_MUTUAL


AuthenticationPolicy:
  - service: foo
     tls:
       mode: PERMISSIVE
  - service: payroll
       mode: STRICT
```

Customer Journey

1. Server sidecar rollout finish.
2. Client DestinationRule ISTIO_MUTUAL
3. PERMISSIVE -> STRICT

Config complexity

DestinationRule UX issues

- Service addressing, *.cluster.local, foo.default.svc.cluster.local
- TrafficPolicy includes more than TLS
  - Circuit breaking.
  - Connection pool size.

```
- host: "*.cluster.local"
  trafficPolicy:
     tls: { mode: ISTIO_MUTUAL }
```

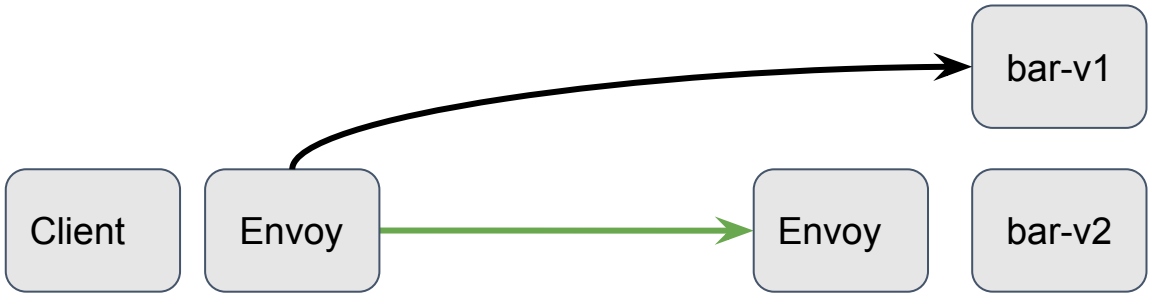
```
- host: "foo.default.svc.cluster.local"
  trafficPolicy:
     connection_time_out: 4s
```

```
- host: "bar.default.svc.cluster.local"
  trafficPolicy:
    tls: { mode: ISTIO_MUTUAL }
  subsets:
  - trafficPolicy:
      loadBalancer: ROUND_ROBIN
```

Client, improvement

- Client mTLS socket based on server having sidecar or not.
- Envoy endpoint labeling: transport_socket_matches
  - Delivered by CDS
- Webhook (sidecar injector) automatically label pods
  - Labels delivered by EDS

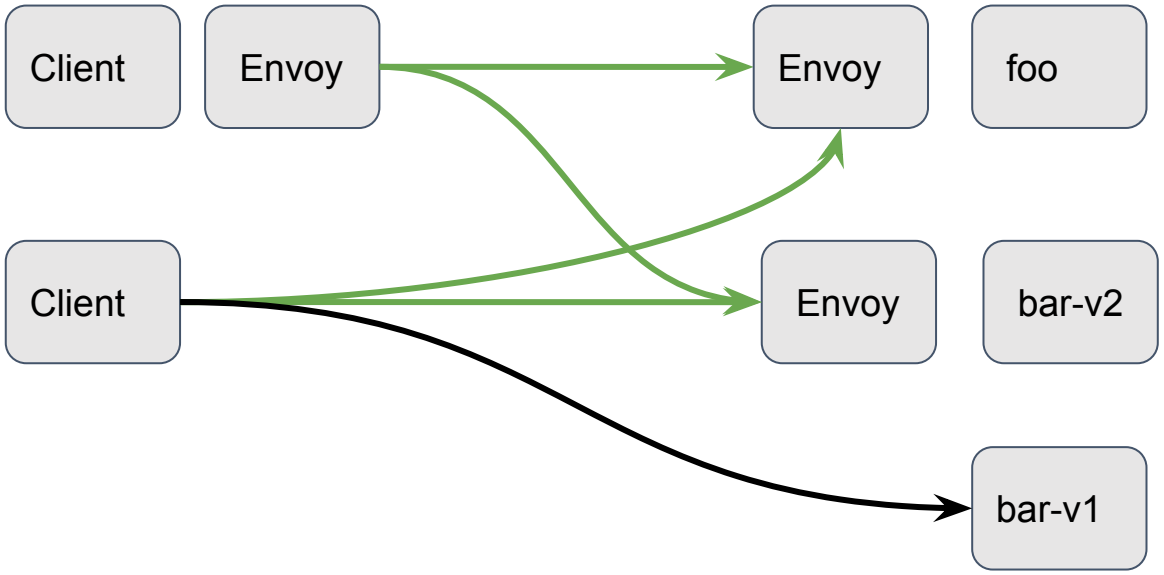
```
server_cluster:
  name: x
  transport_socket_matches:
  - socket: envoy.tls
        { key, cert, root }
    labels:
    - mtlsReady: true
    name: mtls
  - socket: envoy.raw_buffer
    name: default
```



`mtlsReady: true`

Our Approach, combined

Service by Service



Istio Configuration

```
AuthenticationPolicy:
  - service: foo
     tls: { mode: PERMISSIVE }
  - service: payroll
     tls: { mode: STRICT }
```

1. PERMISSIVE by default.
2. Client is automatically detecting server.
3. Sidecar rollout => mTLS ramp up automatically
4. Switch to STRICT once finishes (only step).

Lessons

- Service mesh: sidecar rollout takes time.
- Amount Config and UX
  - Bad alternative: manage subset to rollout mTLS; separate port.
  - DestinationRule its own UX issue.
- Don't forget security
  - STRICT for finishing.
- Real world traffic is complicated
  - Server: application TLS, conflicting.
  - Cient: ORIGIN_DNS typed cluster.

Summary

- Server Envoy supports both mTLS and plaintext
  - TLS Sniffing
  - Config: PERMISSIVE & STRICT
- Client Envoy auto decides mTLS or plaintext
  - Envoy: endpoint labeling
  - Remove DestinationRule entirely for mTLS.
- Only one step for user.

Thanks!
Q & A


KubeCon
CloudNativeCon
Europe 2020
Virtual
HELM
KEEP CLOUD NATIVE
CONNECTED