
cisco
You make possible

But... why do I need a Service Mesh?

Traffic management with Istio on Cisco Container Platform

Julio Gomez – CCIE 9302
Programmability Lead, EMEAR
@juliodevops

BRKCLD-2429

cisco Live!
Barcelona | January 27–31, 2020

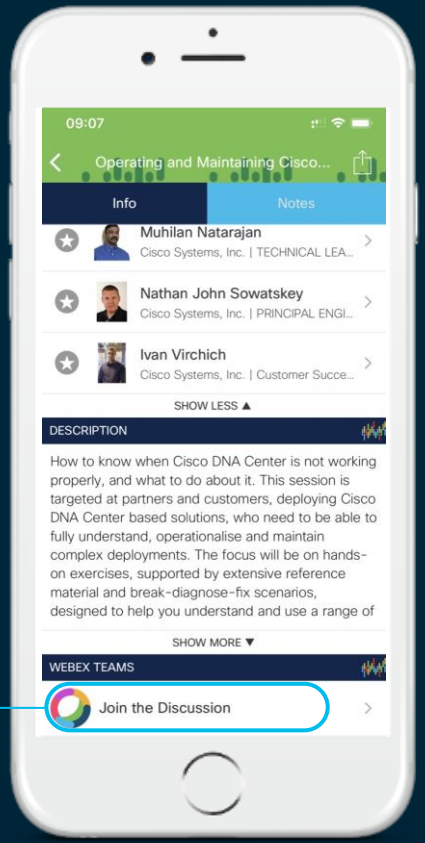
Cisco Webex Teams

Questions?

Use Cisco Webex Teams to chat with the speaker after the session

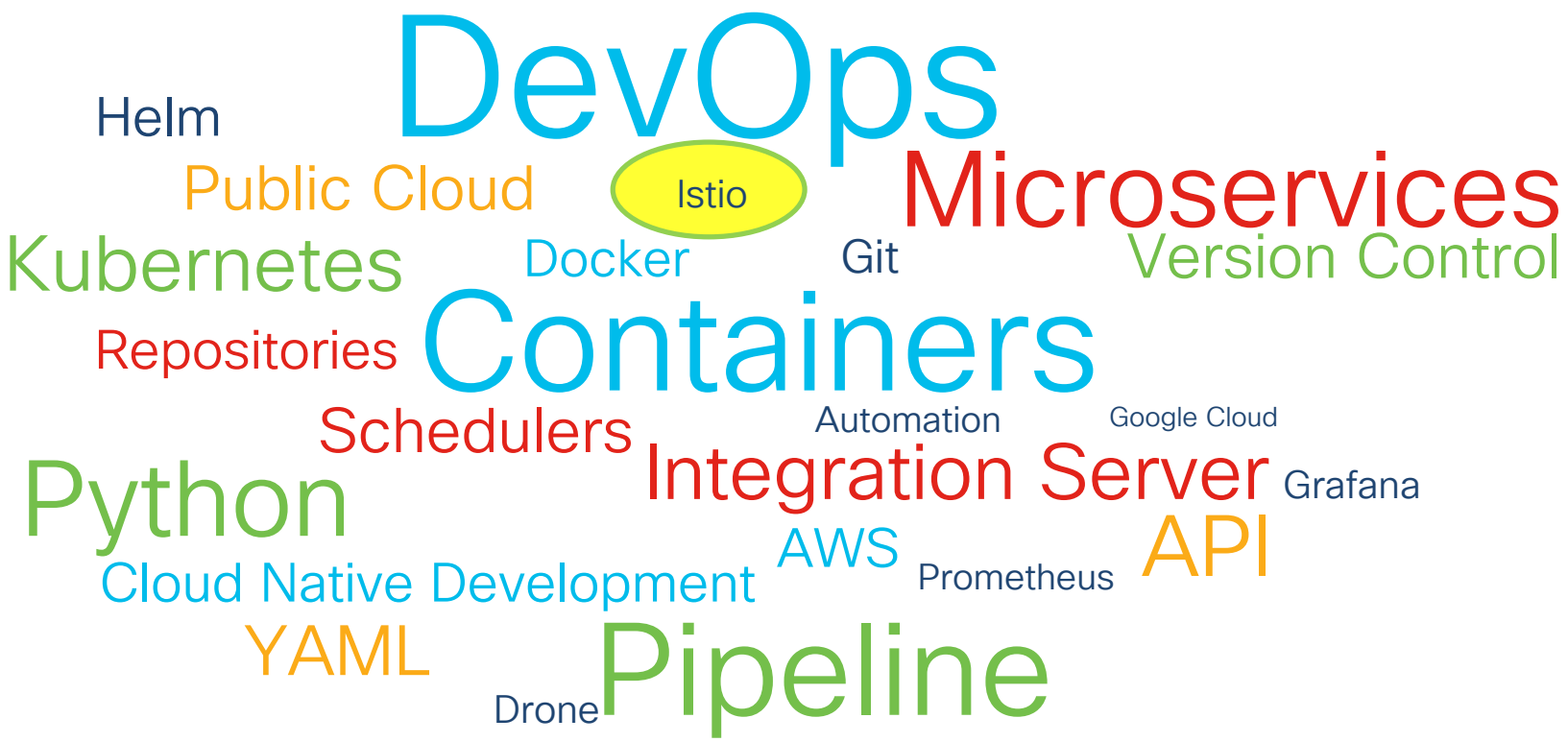
How

1. Find this session in the Cisco Events Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space

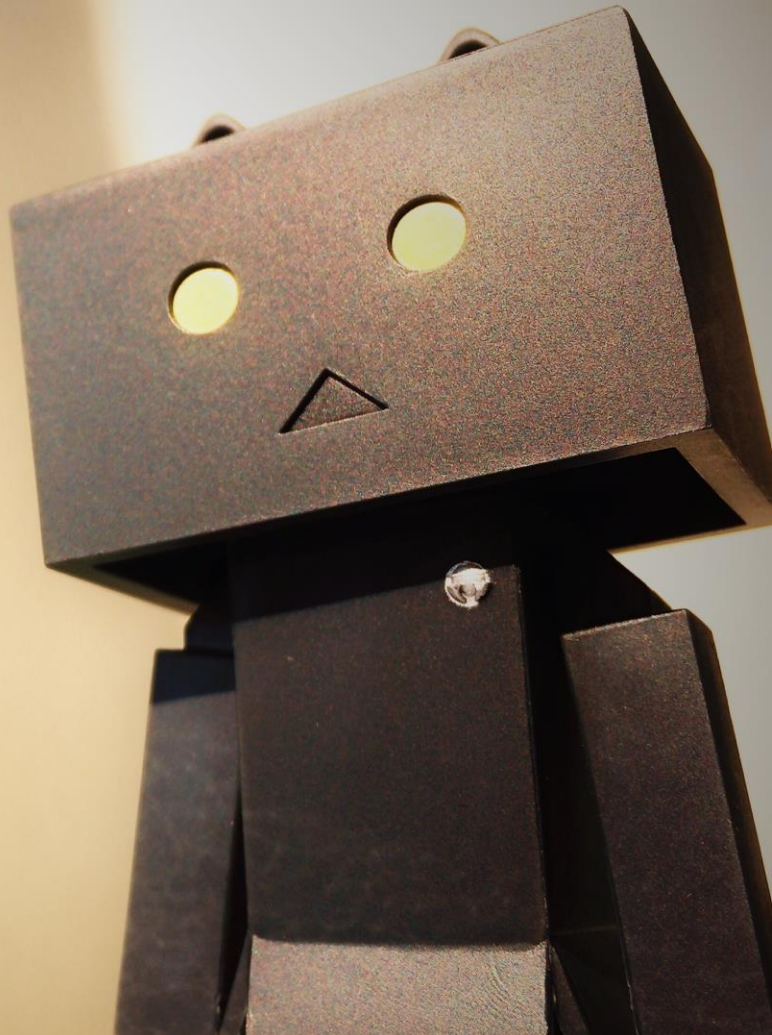


Agenda

- Introduction
- What is a Service Mesh?
- Istio Architecture
- How does it work?
- Capabilities
- Use cases demos
- Conclusion



“Service Meshes are the new black”

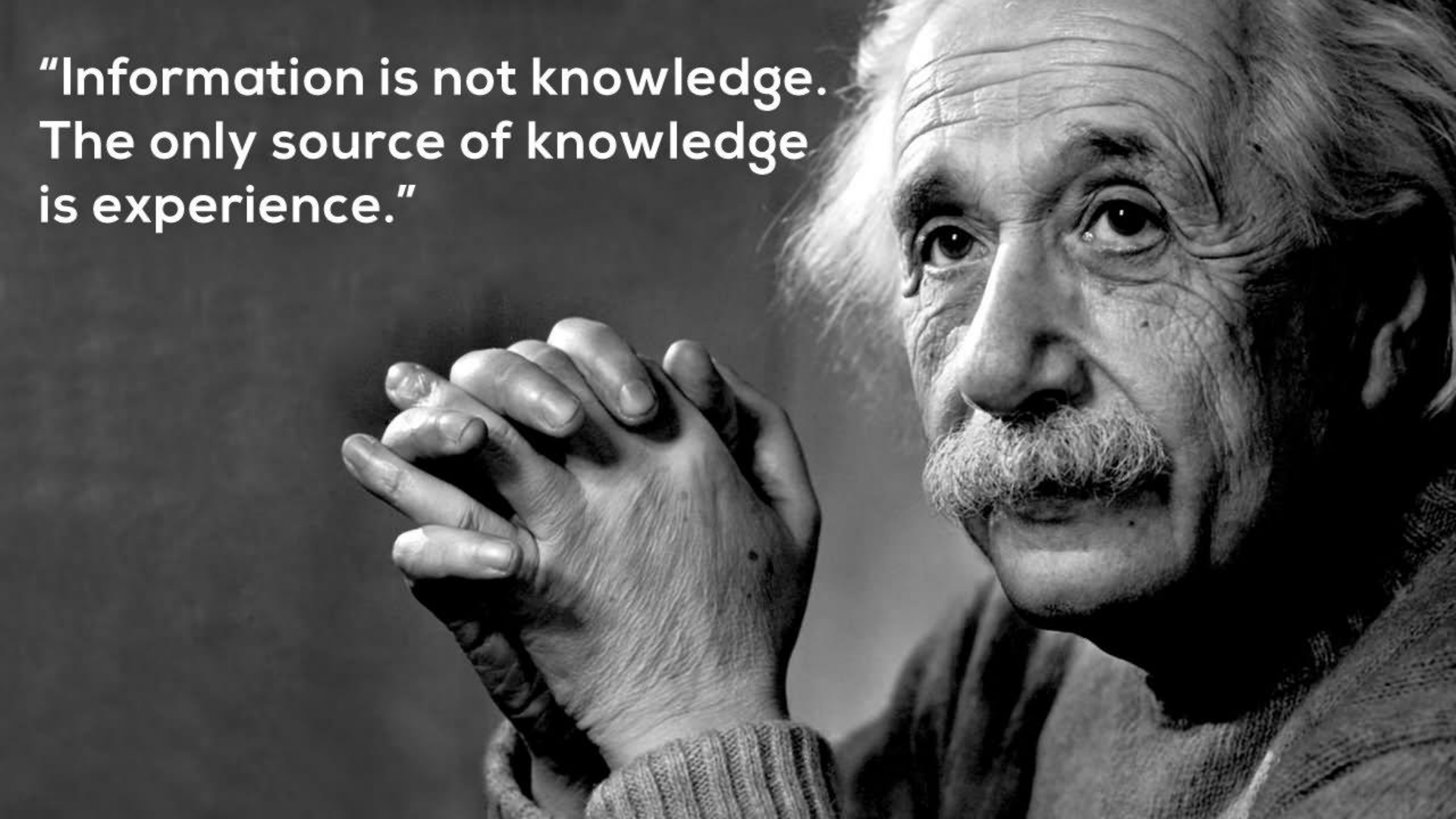


Jessie Frazelle

KubeCon ‘17

Dave Strebel @Hashiconf
@dave_strebel

"Finally got Istio into production"


“Information is not knowledge.
The only source of knowledge
is experience.”

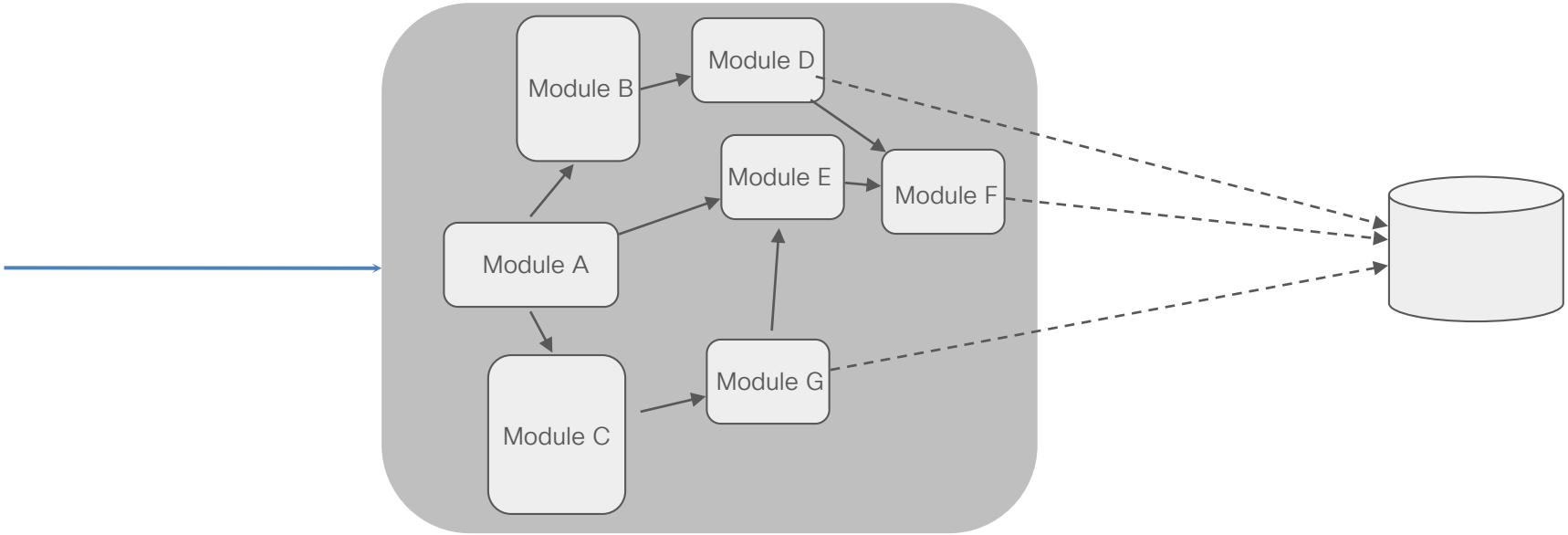
Introduction

Distributed world

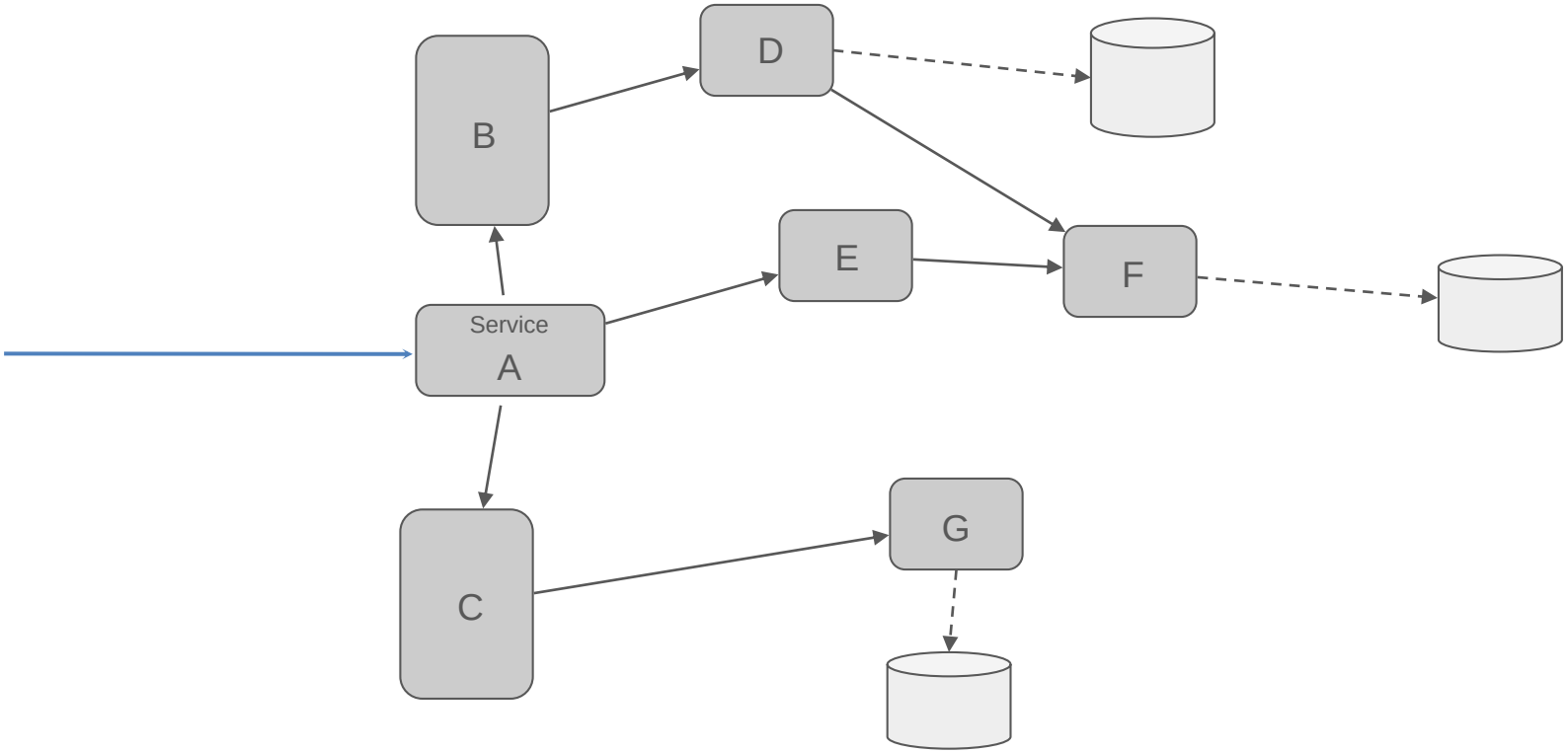
The trends of containerization, microservices and hybrid/multi-cloud deployments have created more distributed applications than ever.

Developers, DevOps and SecOps personnel need modern tools to **secure**, **manage** and **monitor** distributed applications.

Remember the Monolith?



Microservices



Microservices FTW

1. Gained development velocity
2. Easy testing because of abstractions
3. Scale services independently

But... what have we lost?

1. We replaced a reliable in-process call with an unreliable RPC
2. Trivial single stepping replaced by...?
3. Secure in-process communication is replaced by insecure network
4. Access control within process was a NOOP
5. Latency went up

wow, that abstraction was leaky...

Can developers fix it?

1. Add retry logic to the application code
2. Add entry-exit traces
3. Secure inter-service connections with strong authentication

And while you are adding code… choose the RPC endpoint intelligently

a) Endpoints with low latency
b) Endpoints with warm caches

```
try {
    HttpResponse response = httpClient.get(
      "http://secretsauce.internal/recipe");
    cook(response.body);
} catch (NetworkError ne) {
    fixmePleaseOMG(ne);
}
```

```
try {
   // Load balancing
   IP ip =
DNS.lookupSRV("secretsauce.internal").pickOne();
   HttpResponse response = httpClient.open(ip).get(
     "http://secretsauce.internal/recipe");
   cook(response.body);
} catch (NetworkError ne) {
   fixmePleaseOMG(ne);
}
```

Load-balance Regionality

```
for (int i = 0; i < 3; i++) {  // Retry
   try {
      IP ip =
   DNS.lookupSRV("secretsauce.internal").pickOne();
      HttpResponse response = httpClient.open(ip).get(
        "http://secretsauce.internal/recipe");
      cook(response.body);
   } catch (NetworkError ne) {
      if (i  == 2) fixmePleaseOMG(ne);
      else Thread.sleep(random(5) * 1000);
   }
}
```

Retries
Load-balance
Regionality

```
Secret key = new Secret(new File("/somewhere/safe/key");
for (int i = 0; i < 3; i++) {
   try {
      IP ip =
   DNS.lookupSRV("secretsauce.internal").pickOne();
      HttpResponse response = httpClient.open(ip)
        .setHeader("Authorization", key.toString())
        .get("http://secretsauce.internal/recipe");
      cook(response.body);
   } catch (NetworkError ne) {
      if (i  == 2) fixmePleaseOMG(ne);
      else Thread.sleep(random(5) * 1000);
   }
}
```

Authorization
Retries
Load-balance
Regionality

```
Secret key = new Secret(new File("/somewhere/safe/key");
for (int i = 0; i < 3; i++) {
   try {
      IP ip =
   DNS.lookupSRV("secretsauce.internal").pickOne();
      HttpResponse response = httpClient.open(ip)
        .setHeader("Authorization", key.toString())
        .get("http://secretsauce.internal/recipe");
      log("Success");
      cook(response.body);
   } catch (NetworkError ne) {
      log("Failed");
      if (i  == 2) fixmePleaseOMG(ne);
      else Thread.sleep(random(5) * 1000);
   }
}
```

Monitoring?
Health checks?
Latency?
Circuit breaking?
Alerting?

Logging
Authorization
Retries
Load-balance
Regionality

Problem: Too Much Infra Code in Services

- Too much work to make services production-ready
  - Load balancing, auto scaling, rate limiting, traffic routing...
- Done in different ways in every service
  - Retry, TLS, failover, deadlines, cancellation, etc, for each language, framework
- Service management across services is responsibility of each service
- Custom siloed implementations lead to fragmented implementation, no uniform policy application, difficult debugging
- Code bloat for each service

Solution: Systemic Service Management

Core Idea: Deploy lightweight sidecar proxies to take over all ingress and egress traffic

- Transparently attach the proxies to service backends
- Mediate all inbound and outbound traffic between the application and the services it consumes
- Provides an array of infrastructure features

Outbound features:

- Service authentication
- Load balancing
- Retry and failover
- Fine-grained routing
- Telemetry
- Tracing

Inbound features:

- Service authentication
- Authorization
- Rate limits
- Load shedding
- Telemetry
- Tracing

Service Mesh to democratize the solution

- Address service level concerns
- Unlock the full power of microservices
- Elevate the network to the needs of applications
- Uniform observability regardless of platform
- Security by default, everywhere

What is a service mesh?

An open platform to manage service interactions among microservices in a consistent and secure way

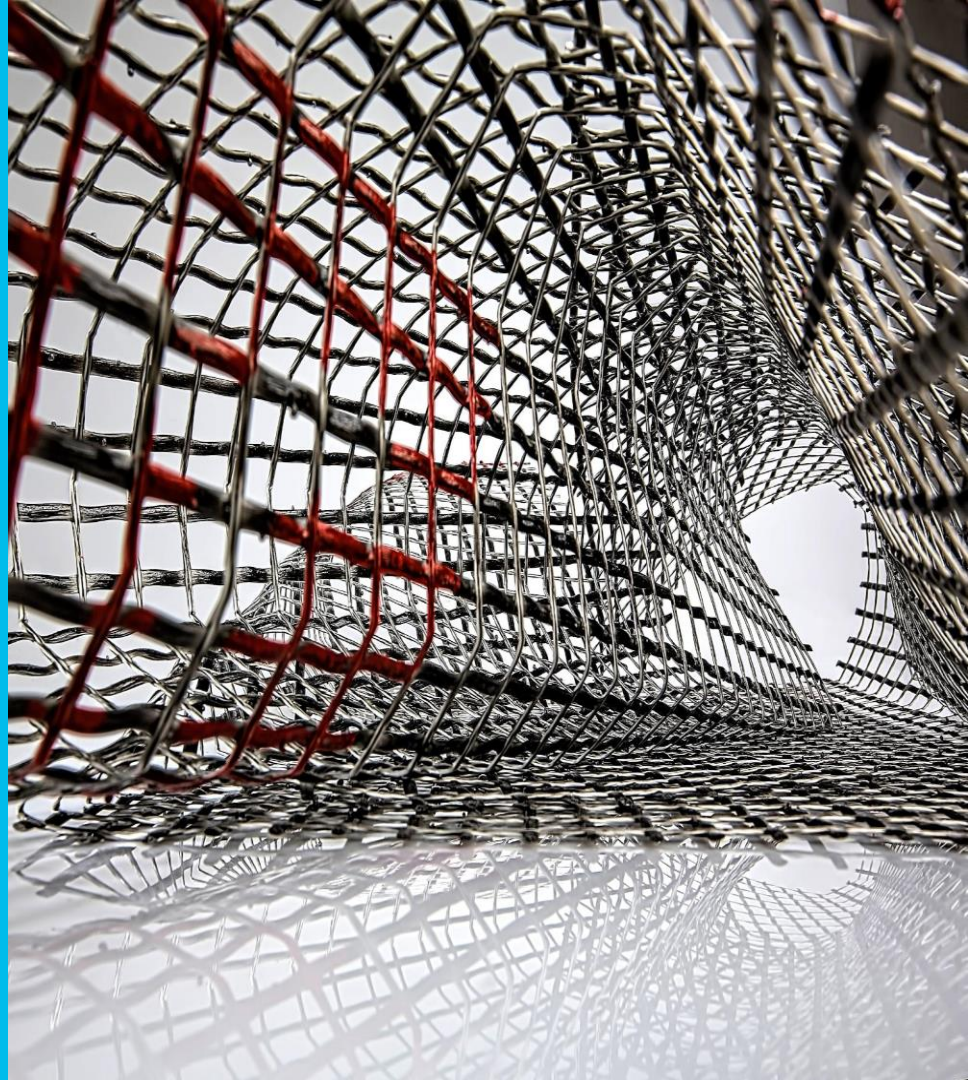
Decouple operations from development with Istio anywhere

Enable customers to secure, monitor and manage services everywhere

Kubernetes first, but not Kubernetes only

Why a service mesh?

A service mesh provides a transparent and language-independent way to flexibly and easily automate application network functions



Value Proposition

(everybody needs something...
but not everybody needs everything)

À La Carte

Observability

Control

Security

cisco Live!

Why not Istio everything?

Why not Istio everything?

Observability

Understanding services and their dependencies

Monitor uniform Service Level Indicators for every service

Collect telemetry, logs and traces

Improved understanding of applications at the *service* (not network) level

Control

Scale by directing traffic to multiple versions

Roll out new versions without worrying about ops challenge

Apply access control, rate limiting policies to protect services from bad behavior

Security

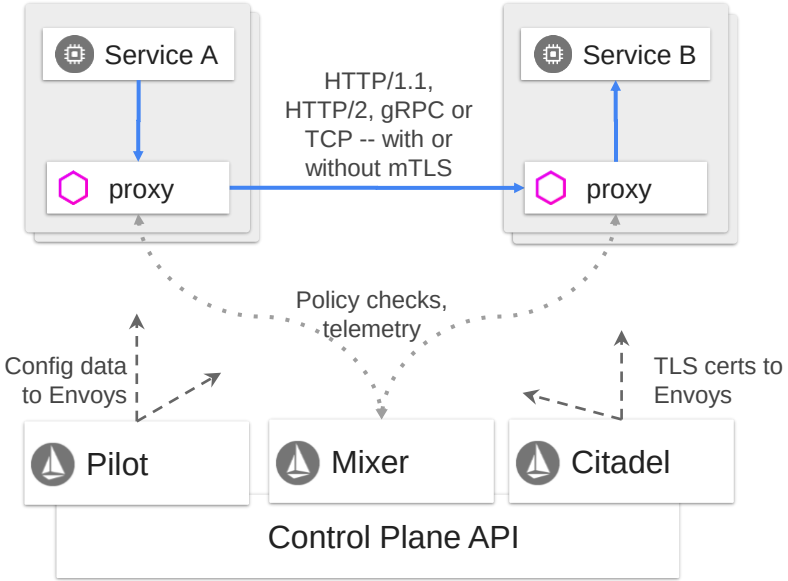
Secure by default – new and existing applications

Meet compliance obligations by **encrypting data in transit**

mTLS assures a secure, proven **service-based identity** for every call

With strong identity, **authorization** can be explicitly required

Istio architecture



Pilot: Control plane to configure and push service communication policies

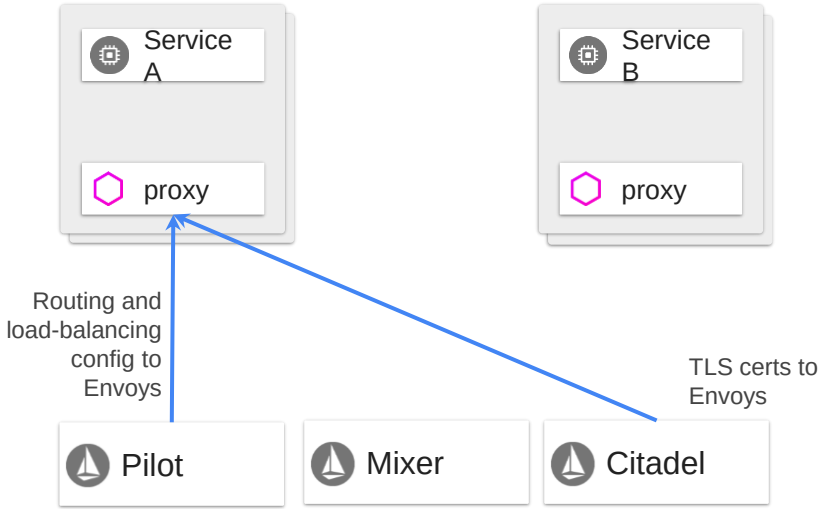
Envoy: Network proxy to intercept communication and apply policies

Mixer: Policy enforcement with a flexible plugin model for providers for a policy

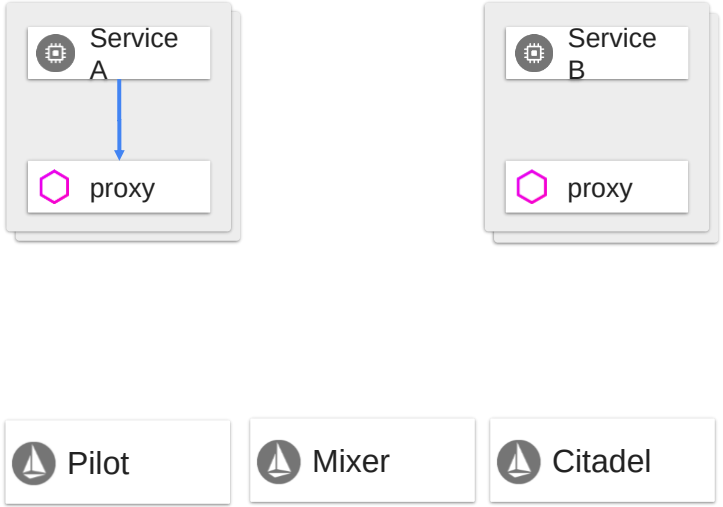
Citadel: Service-to-service auth[n,z] using mutual TLS, with built-in identity and credential management

How does it work?

Life of a request in the service mesh



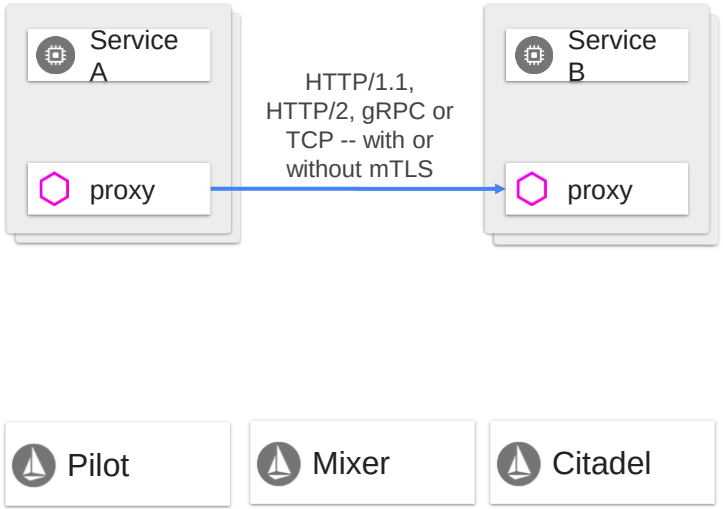
Service A comes up. Envoy is deployed with it and fetches service information, routing and configuration policy from Pilot. If Citadel is being used, TLS certs are securely distributed as well.



Service A places a call to service B

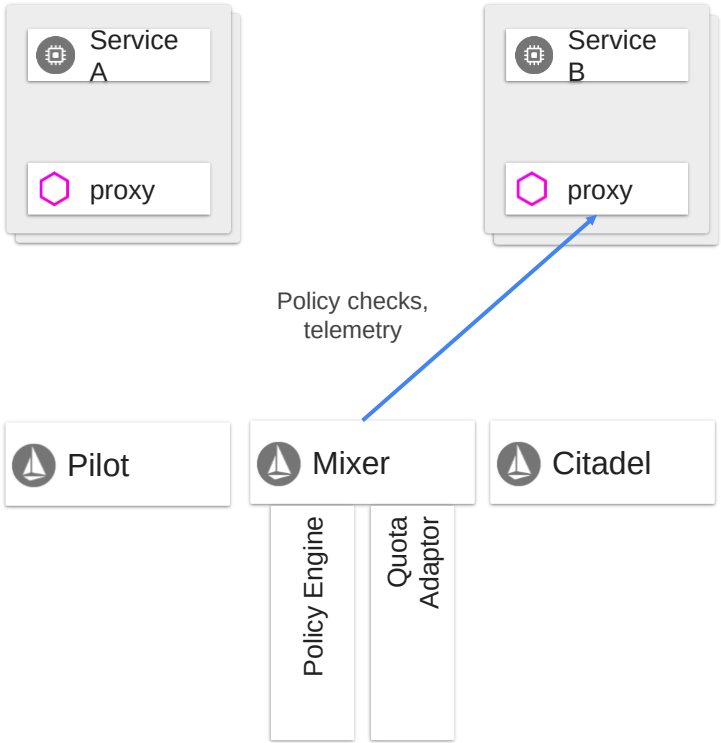
Client-side Envoy intercepts the call

Envoy consults config to know how/where to route call to service B

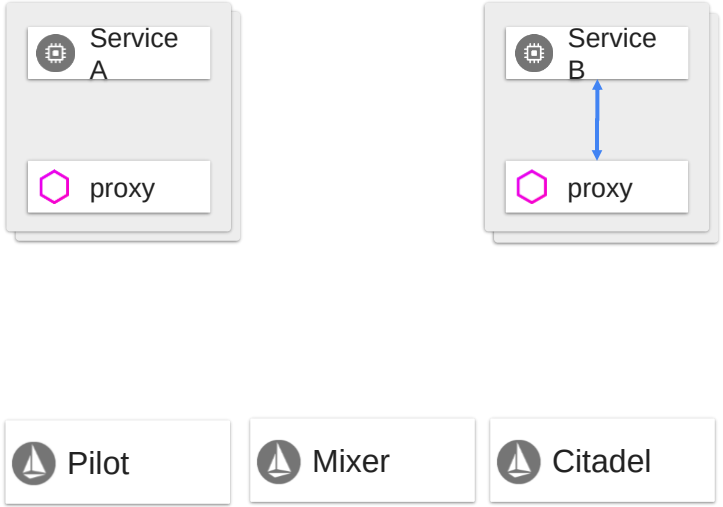


Envoy forwards request to appropriate instance of service B

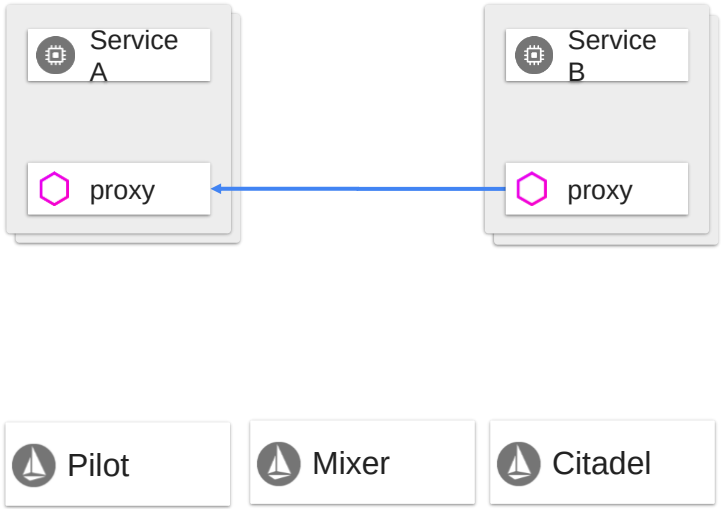
There, the Envoy proxy deployed with the service intercepts the call



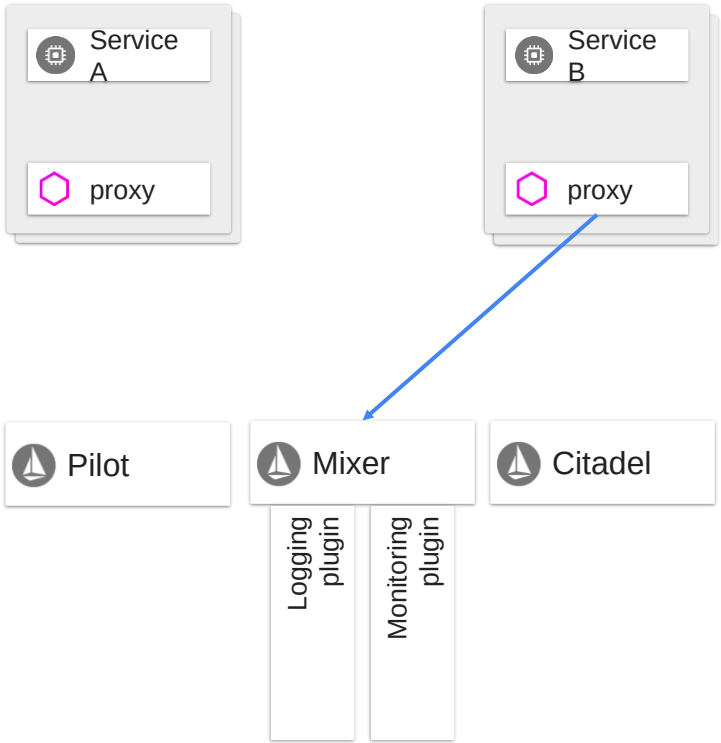
Mixer checks with appropriate adaptors (policy engine, quota adaptor) to verify that the call can proceed and returns true/false to Envoy



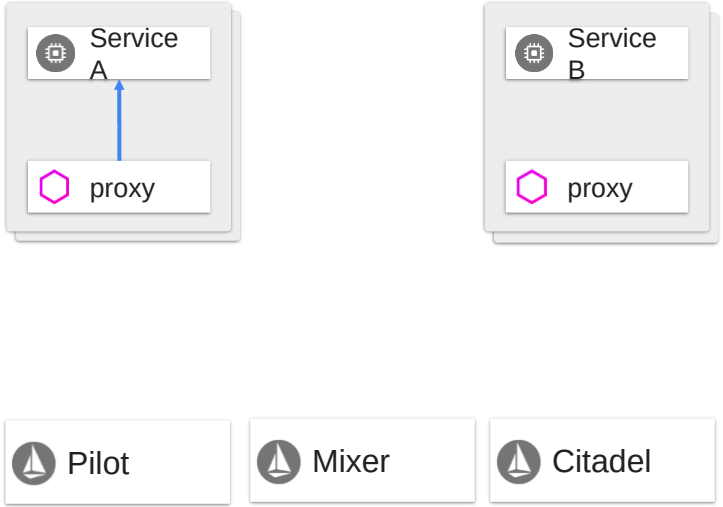
Server-side Envoy forwards request to service B, which processes request and returns response



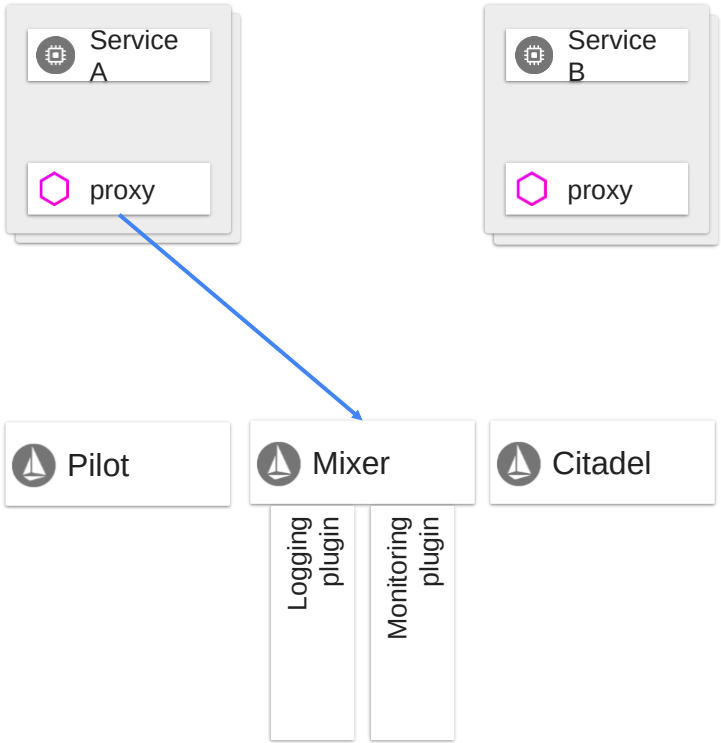
Envoy forwards response to the original caller, where response is intercepted by Envoy on the caller side



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



Client-side Envoy forwards response to original caller

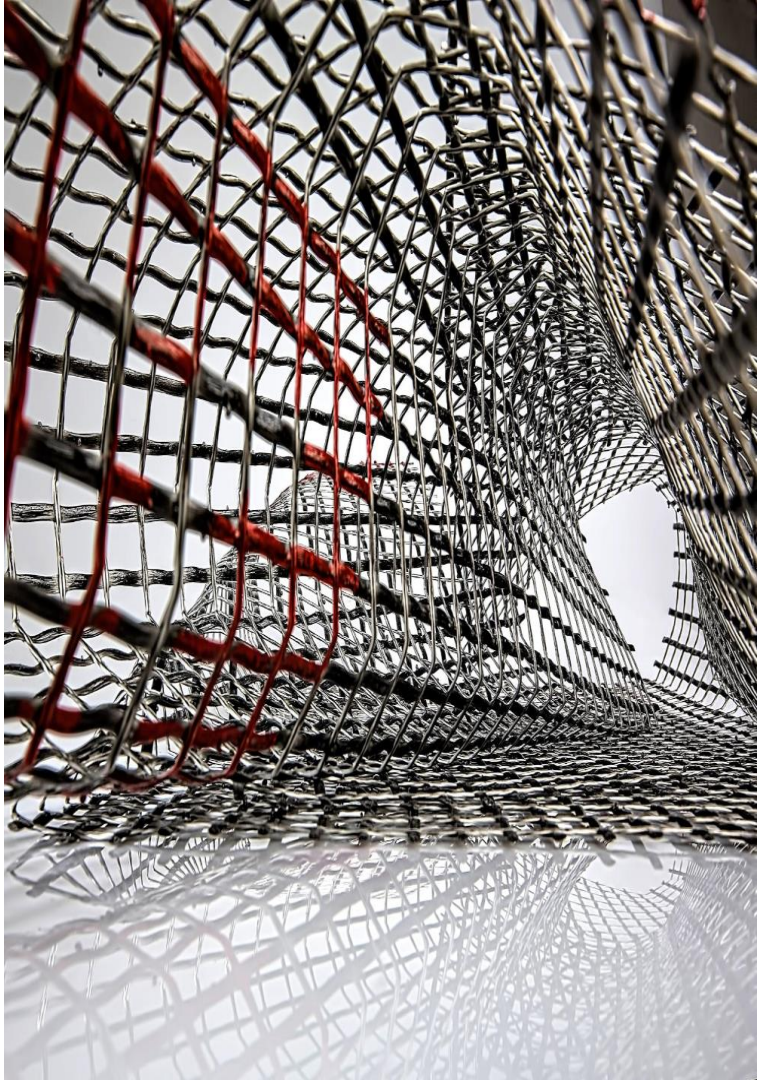


Client-side Envoy reports telemetry to Mixer (including client-perceived latency), which in turn notifies appropriate plugins

Capabilities

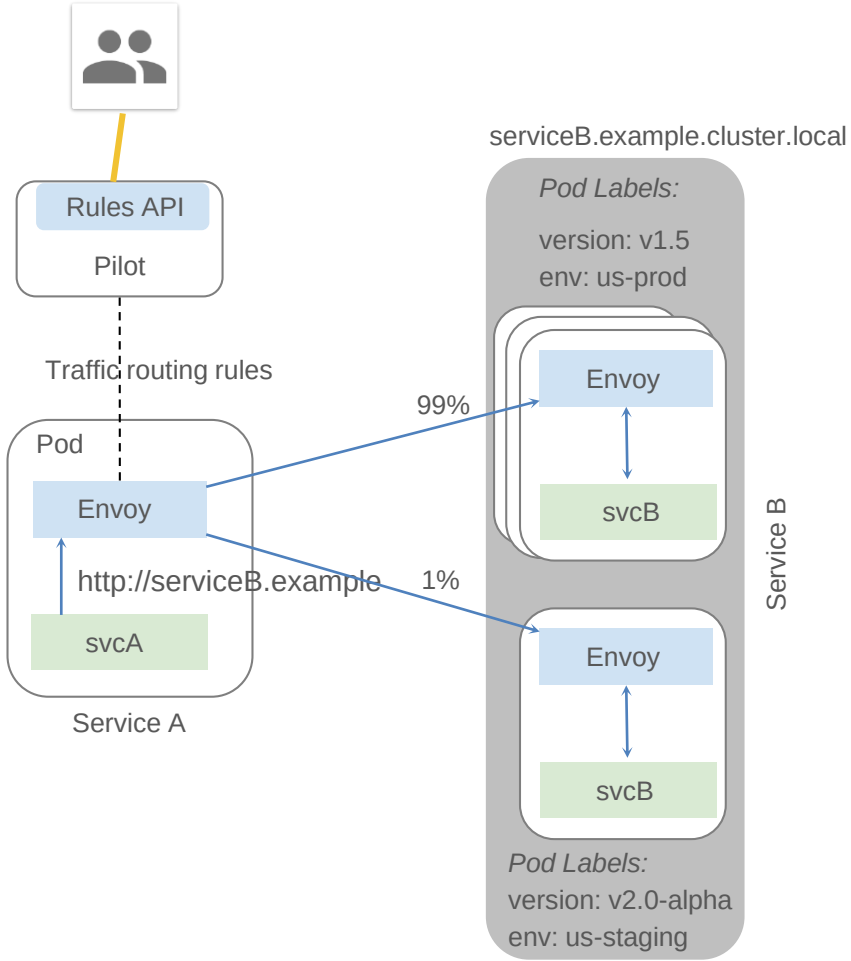
A network for services

- **Traffic Control**
- Visibility
- Resiliency & Efficiency
- Security



Application Rollout

```
// A simple traffic splitting rule

destination: serviceB.example.cluster.local
match:
  source: serviceA.example.cluster.local
route:
- tags:
    version: v1.5
    env: us-prod
  weight: 99
- tags:
    version: v2.0-alpha
    env: us-staging
  weight: 1
```

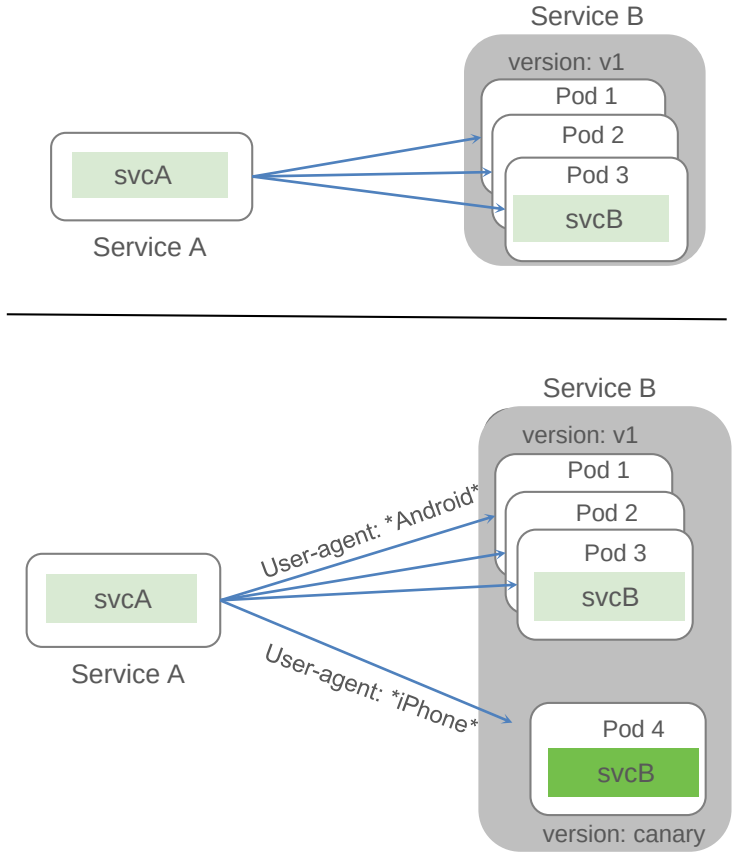


Traffic control is decoupled from infrastructure scaling

cisco Live!

Traffic Steering

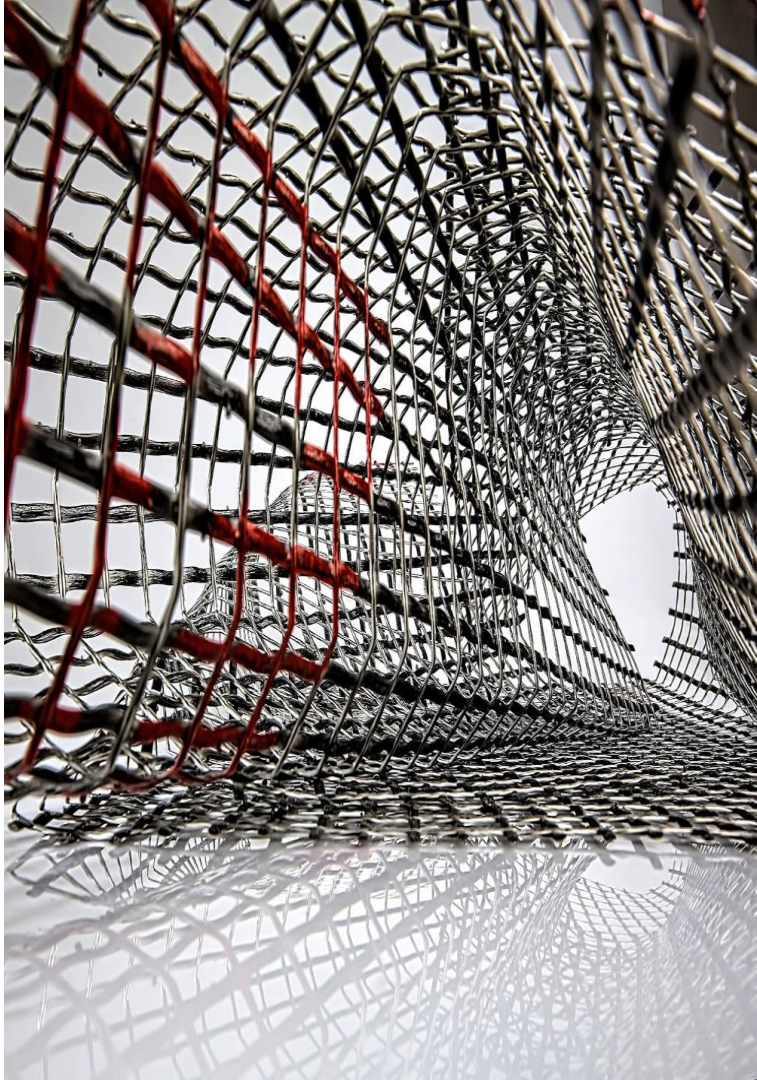
```
// Content-based traffic steering rule

destination: serviceB.example.cluster.local
match:
  httpHeaders:
    user-agent:
      regex: ^(.*?;)?(iPhone)(;.*)?$
precedence: 2
route:
- tags:
    version: canary
```



Content-based traffic steering

A network for services

- Traffic Control
- **Visibility**
- Resiliency & Efficiency
- Security



Admin
Namespaces
Nodes
Persistent Volumes

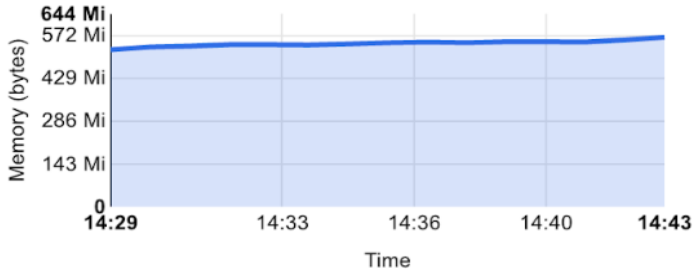
Namespace
kube-system

Workloads
Deployments
Replica Sets
Replication Controllers
Daemon Sets
Stateful Sets
Jobs
Pods

Services and discovery
Services

CPU usage

Memory usage



Pods

1 - 10 of 18

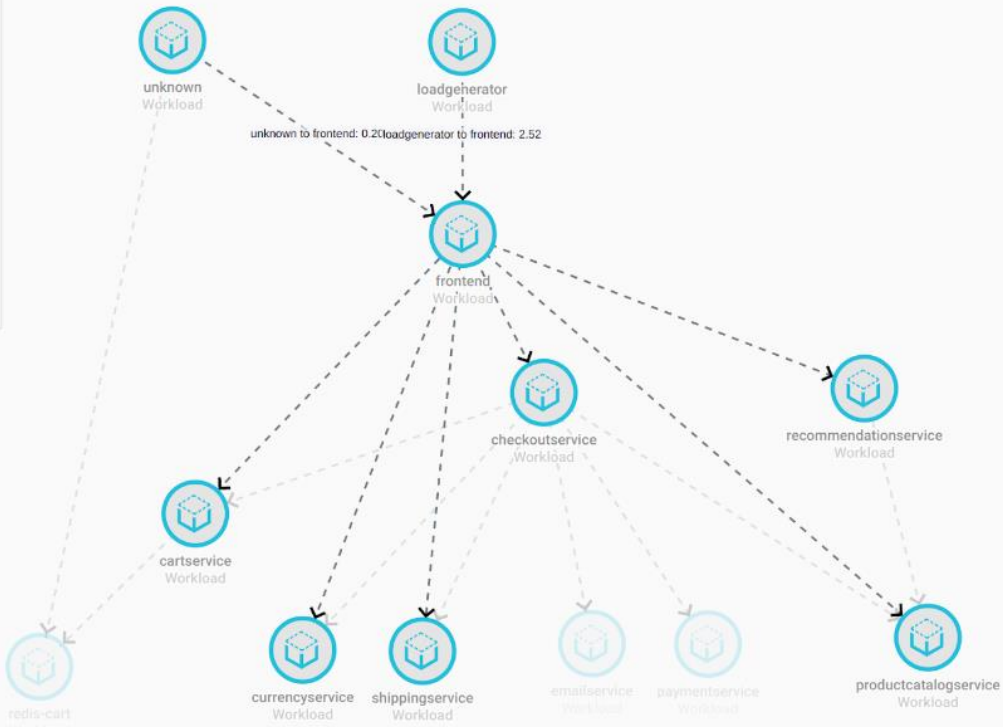
Name	Status	Restarts	Age	CPU (cores)	Memory (bytes)
dashboard-same-i...	Running	0	14 minutes	0	5.4 Mi
fluentd-cloud-logg...	Running	0	2 months	0.01	120.8 Mi
fluentd-cloud-logg...	Running	0	2 months	0.009	94.3 Mi
fluentd-cloud-logg...	Running	0	2 months	0.014	174.3 Mi
heapster-v1.2.0-4...	Running	0	10 days	0.002	44.8 Mi

Topology Istio View

Filters

- Daemon Set
- Workload
- Pod
- Container
- Replica Set
- Deployment

Workload: exclude istio Filter



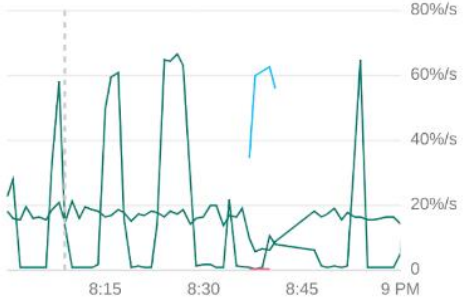
Jul 17, 9:00 pm

Jul - 8 Jul - 9 Jul - 10 Jul - 11 Jul - 12 Jul - 13 Jul - 14

Percent non 200 response codes

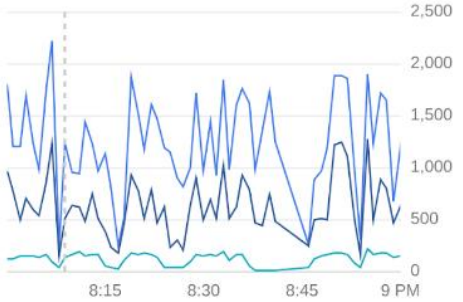
3 by response code (sum)

1 min interval (rate)



Response Latencies

2 1 min interval (delta)

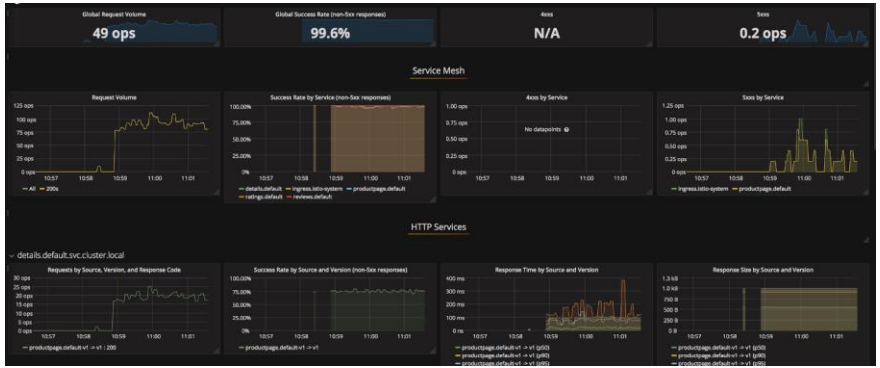


Visibility

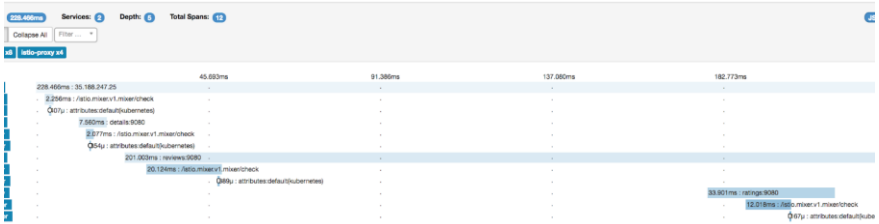
Monitoring & tracing should not be an afterthought in the infrastructure

Goals

- Metrics without instrumenting apps
- Consistent metrics across fleet
- Trace flow of requests across services
- Portable across metric backend providers



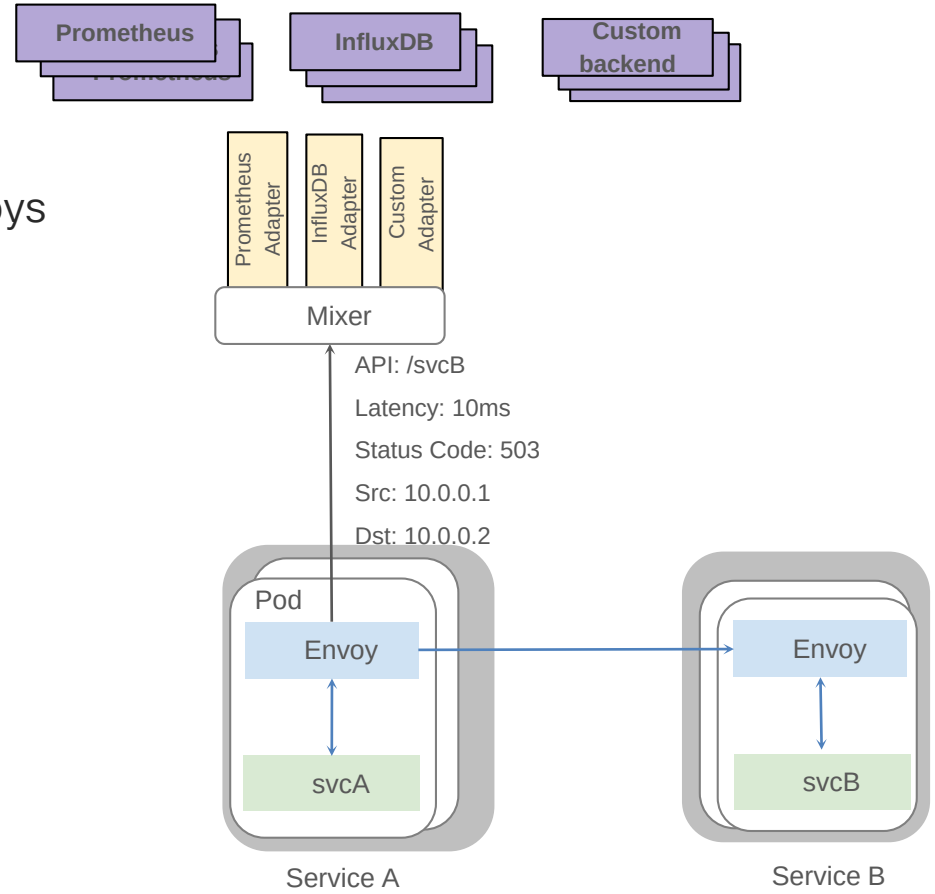
Istio - Grafana dashboard w/ Prometheus backend



Istio Zipkin tracing dashboard

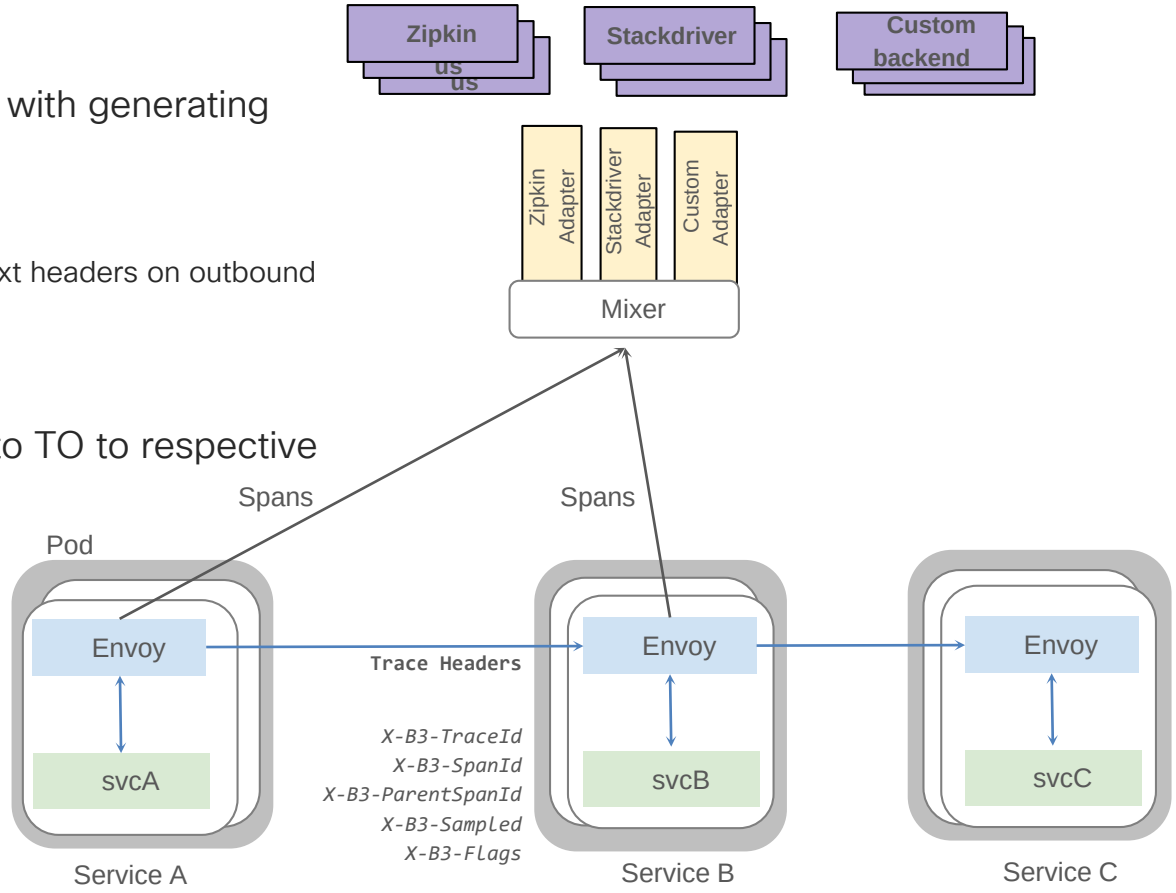
Metrics flow

- Mixer collects metrics emitted by Envoys
- Adapters in the Mixer normalize and forward to monitoring backends
- Metrics backend can be swapped at runtime



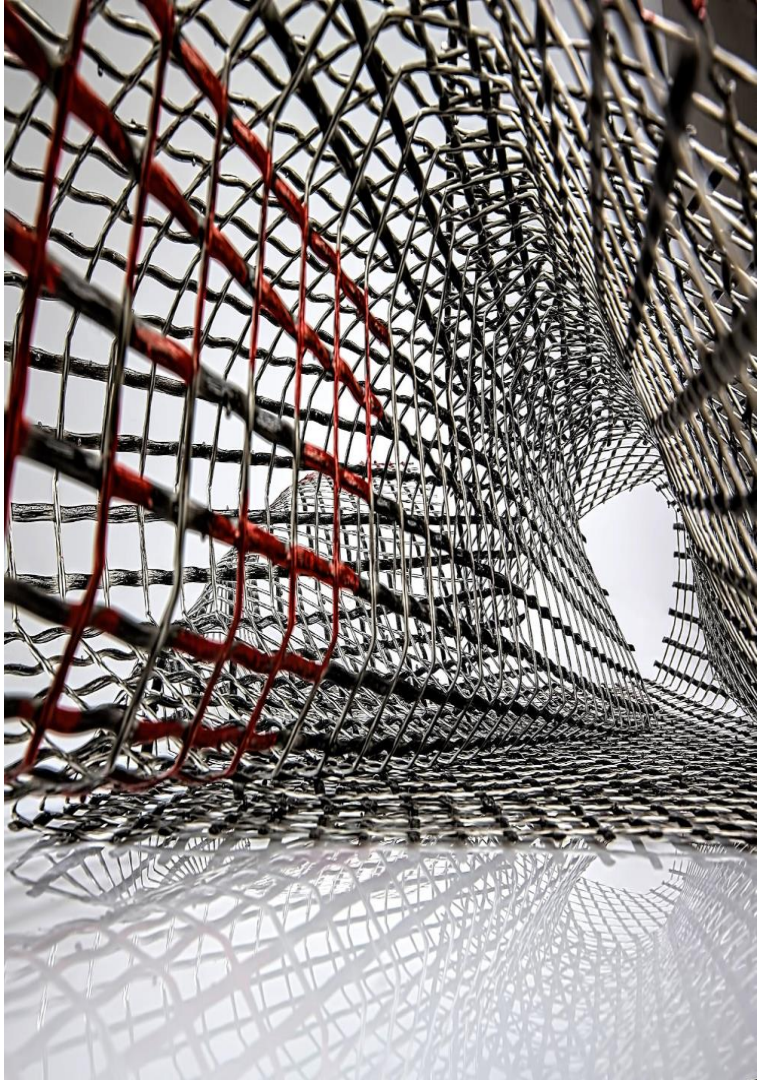
Tracing flow

- Application do not have to deal with generating spans or correlating causality
- Envoys generate spans
  - Applications need to *forward* context headers on outbound calls
- Envoys send traces to Mixer
- Adapters at Mixer send traces to TO to respective backends



A network for services

- Traffic Control
- Visibility
- **Resiliency & Efficiency**
- Security



Resiliency

Istio adds fault tolerance to your application without any changes to code

```
// Circuit breakers

destination: serviceB.example.cluster.local
policy:
- tags:
    version: v1
  circuitBreaker:
    simpleCb:
      httpConsecutiveErrors: 7
      sleepWindow: 5m
      httpDetectionInterval: 1m
```

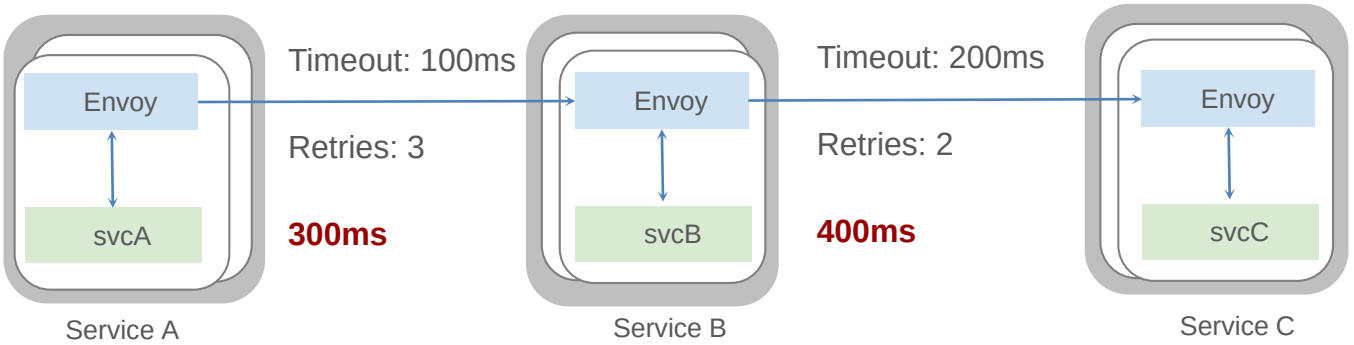
Resilience features

- Timeouts
- Retries with timeout budget
- Circuit breakers
- Health checks
- AZ-aware load balancing w/ automatic failover
- Control connection pool size and request load
- Systematic fault injection

Resiliency Testing

Systematic fault injection to identify weaknesses in failure recovery policies

- HTTP/gRPC error codes
- Delay injection

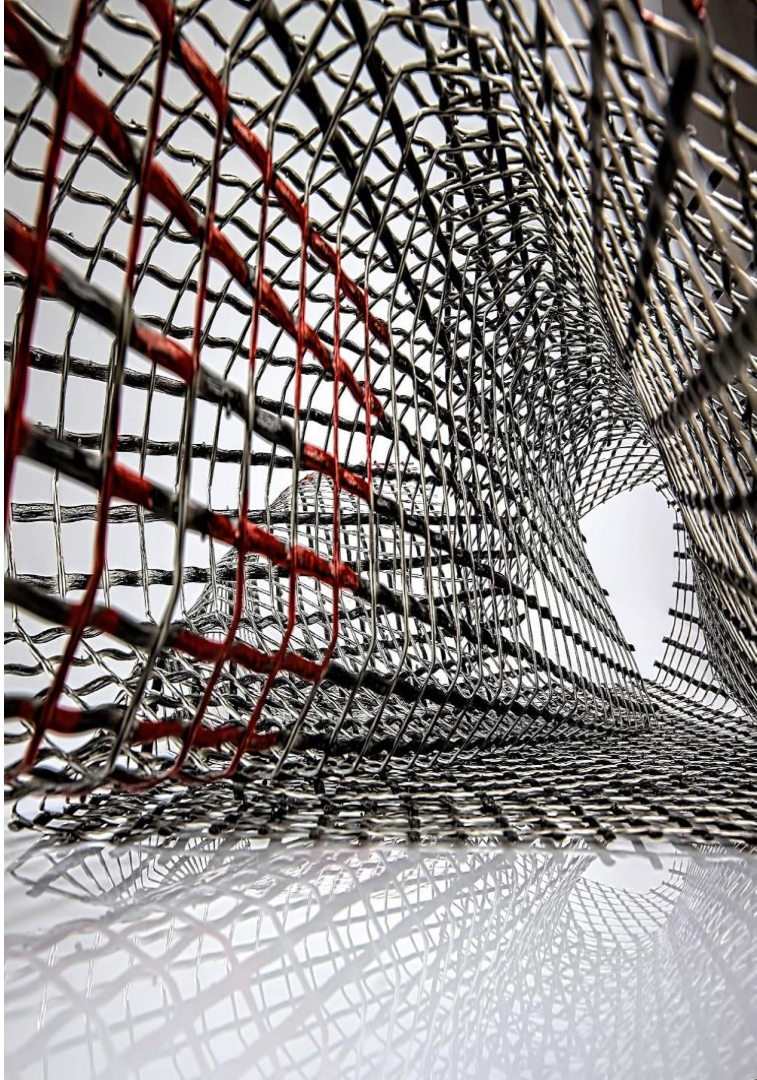


Efficiency

- L7 load balancing
  - Passive/Active health checks, circuit breaks
  - Backend subsets
  - Affinity
- TLS offload
  - No more JSSE or stale SSL versions.
- HTTP/2 and gRPC proxying

A network for services

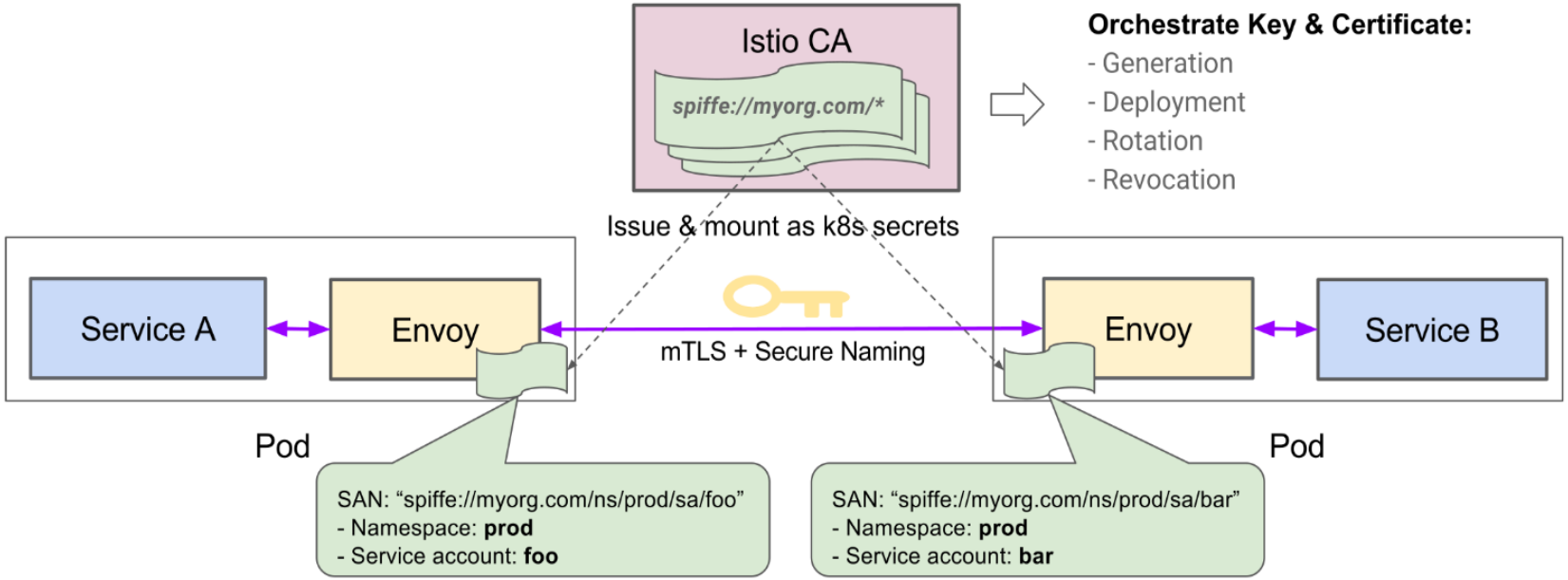
- Traffic Control
- Visibility
- Resiliency & Efficiency
- **Security**



Security

- Verifiable identity
- Secure naming / addressing
- Traffic encryption
- Revocation

Security at Scale



Istio vs Linkerd

	Istio	Linkerd
Source	Google + IBM + Lyft	CNCF
Support for VMs	Via Consul	No
Sidecar proxy	Envoy (mature and more features – ingress/egress and LB)	Own implementation (lightweight and fast)
Service relations	Included	Not included
Management console	Not included	Included
Policy Mgmt & Auth	Included	Not included
Ease of use	Harder	Easier

Use cases demos

Cisco Container Platform

Turnkey Solution
For Production-Grade Container Environments

Native Kubernetes (100% Upstream)
Direct updates and best practices from open source community

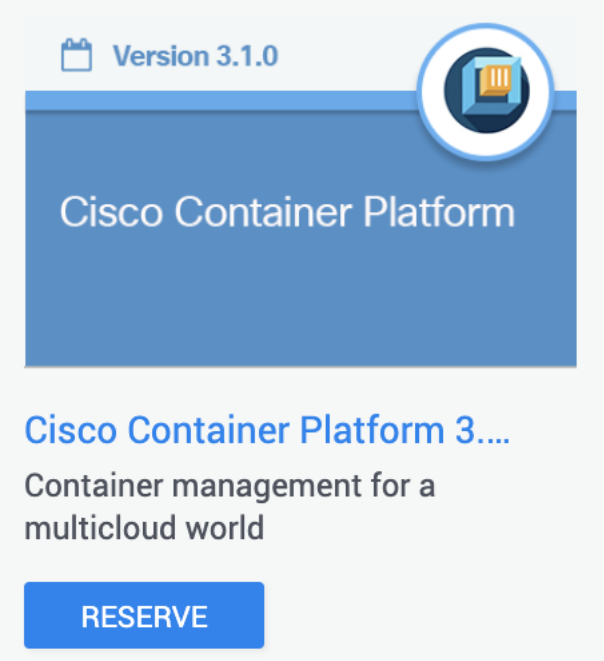
Hybrid Cloud Optimized
Deploy on-premise and natively on public Cloud providers

Integrated
Networking | Storage | Management | Security | Analytics

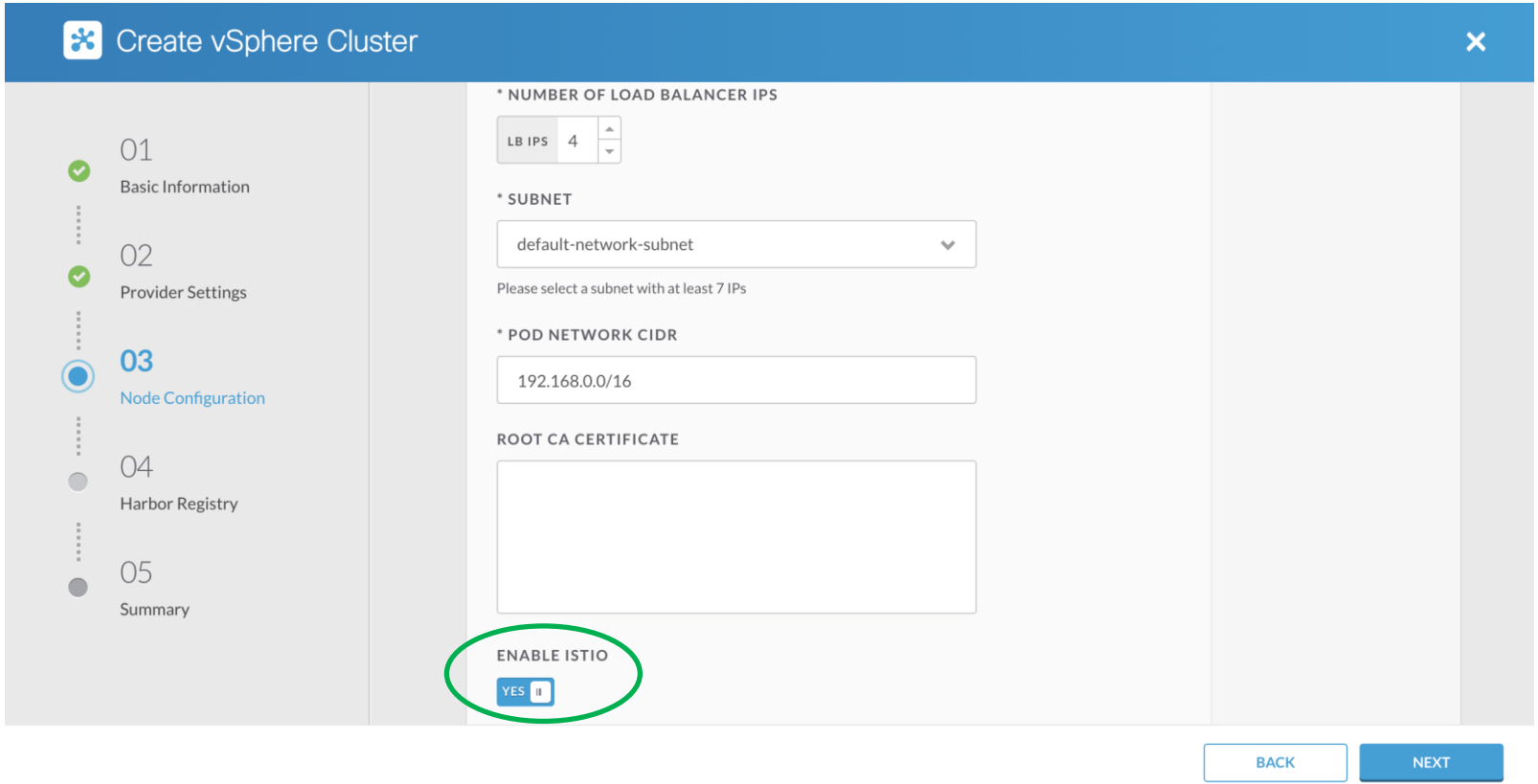
Flexible Deployment Model
VM | Bare metal ←→ HX, UCS, ACI | Public cloud

Easy to acquire, deploy and manage | Open and consistent | Extensible platform | World-class advisory and support

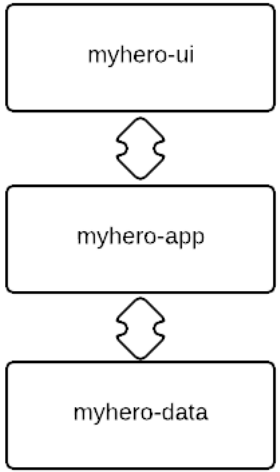
Cisco Container Platform Sandbox



How to enable Istio on Cisco Container Platform

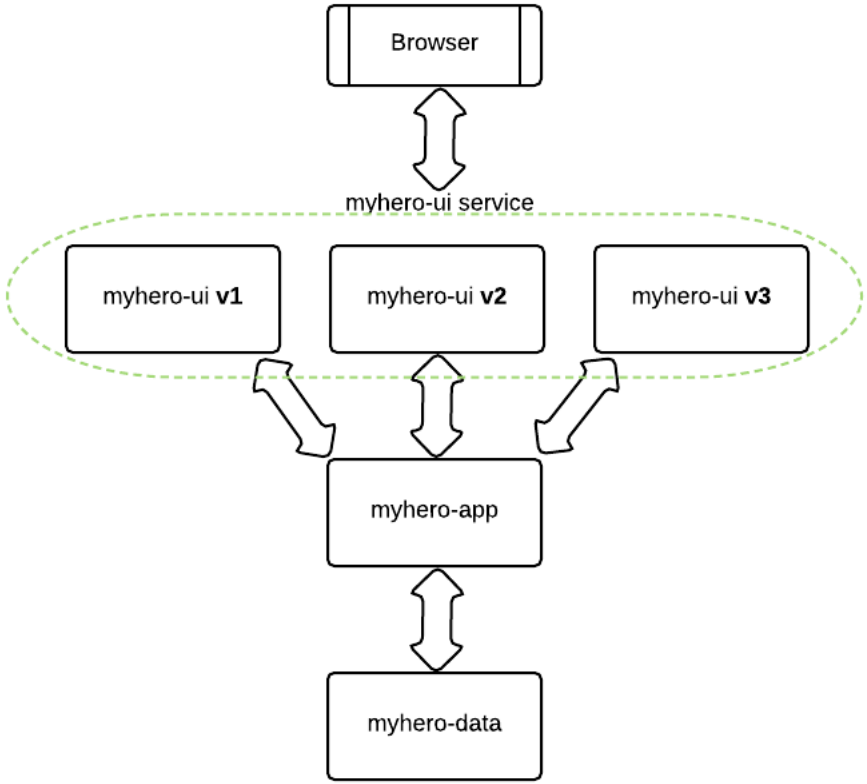


Example microservices-based application



3 different HTTP front-end versions

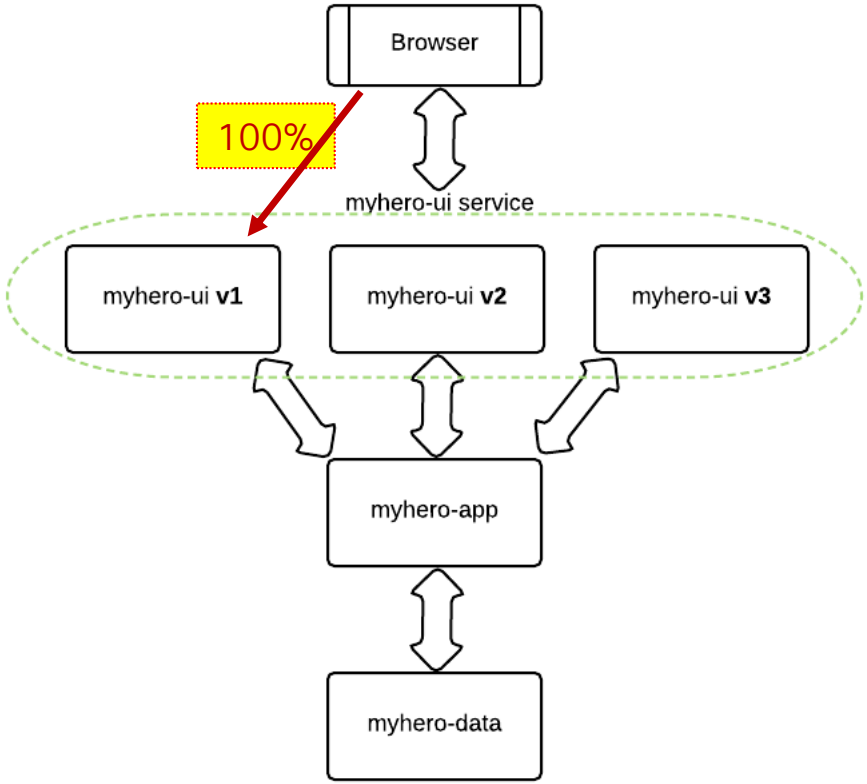
```
kind: DestinationRule
spec:
  host: myhero-ui.myhero.svc.cluster.local
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  - name: v3
    labels:
      version: v3
```



Use case 1

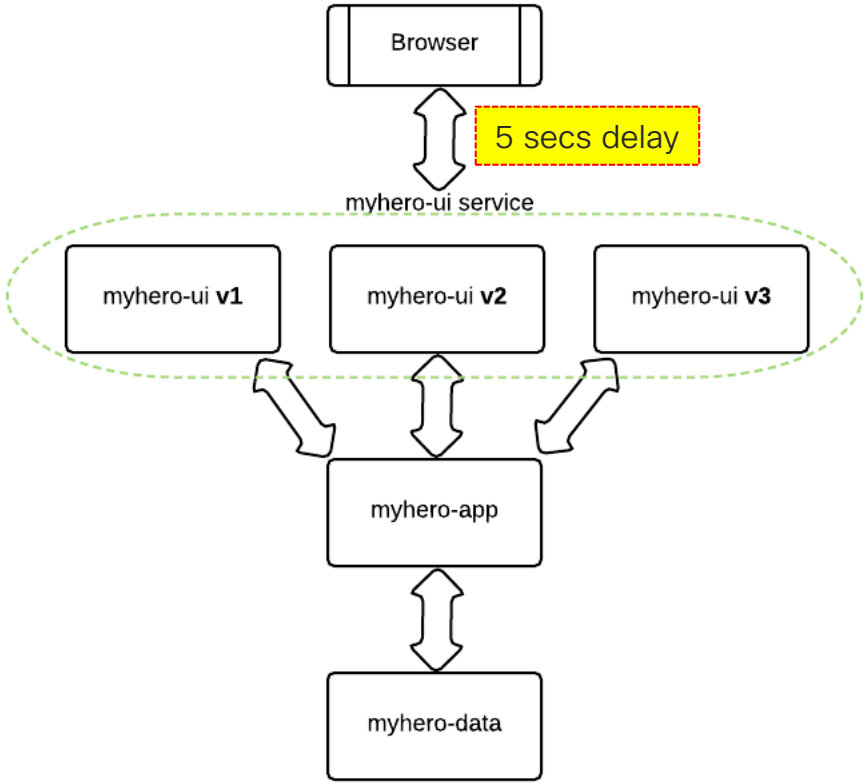
Routing to specific service version

```
kind: VirtualService
spec:
  hosts:
  - "ui.xxx.com"
  gateways:
  - myhero-gateway
  http:
  - route:
    - destination:
        host: myhero-ui.myhero.svc.cluster.local
        subset: v1
```



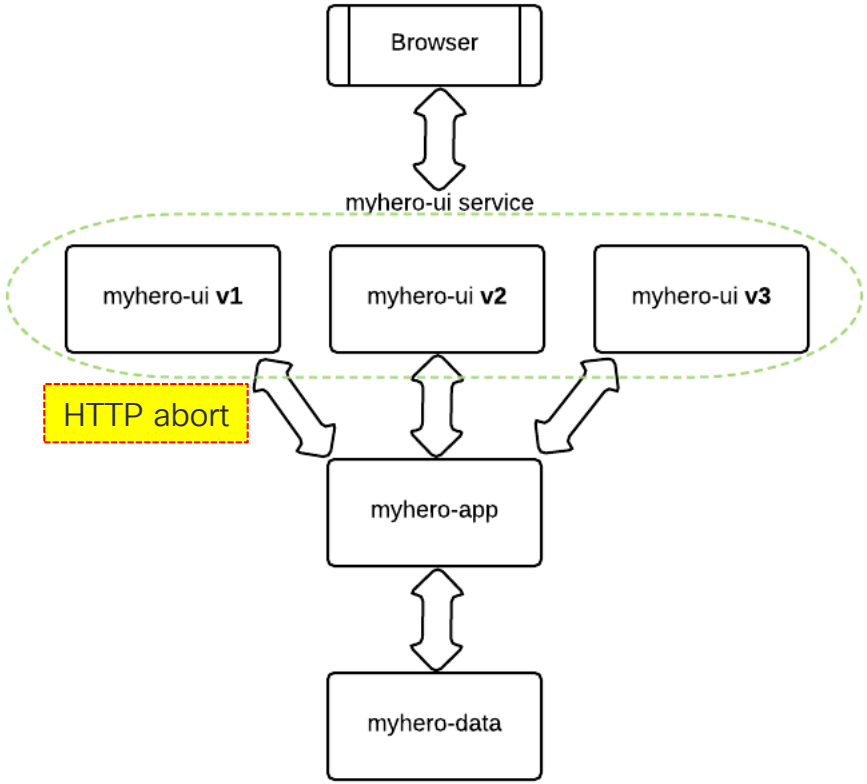
Use case 2
Delay injection

```
kind: VirtualService
spec:
  hosts:
  - "ui.xxx.com"
  gateways:
  - myhero-gateway
  http:
  - fault:
      delay:
        percent: 100
        fixedDelay: 5s
    route:
    - destination:
        host: myhero-ui.myhero.svc.cluster.local
        subset: v1
```



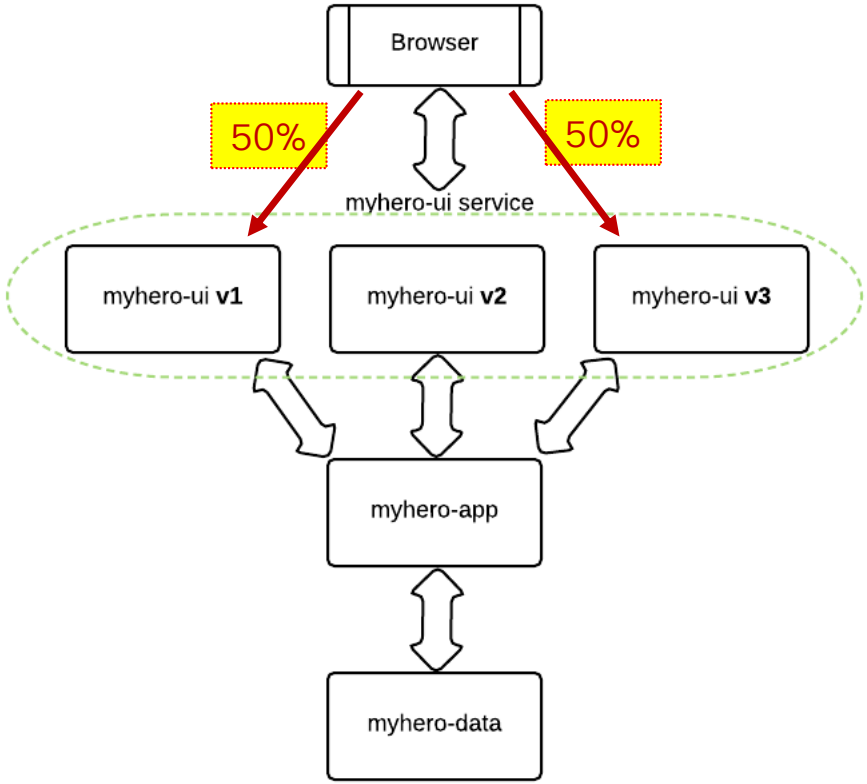
Use case 3
HTTP abort injection

```
kind: VirtualService
spec:
  hosts:
  - "ui.xxx.com"
  gateways:
  - myhero-gateway
  - mesh
  http:
  - fault:
      abort:
        percent: 100
        httpStatus: 500
    route:
    - destination:
        host: myhero-app.myhero.svc.cluster.local
```



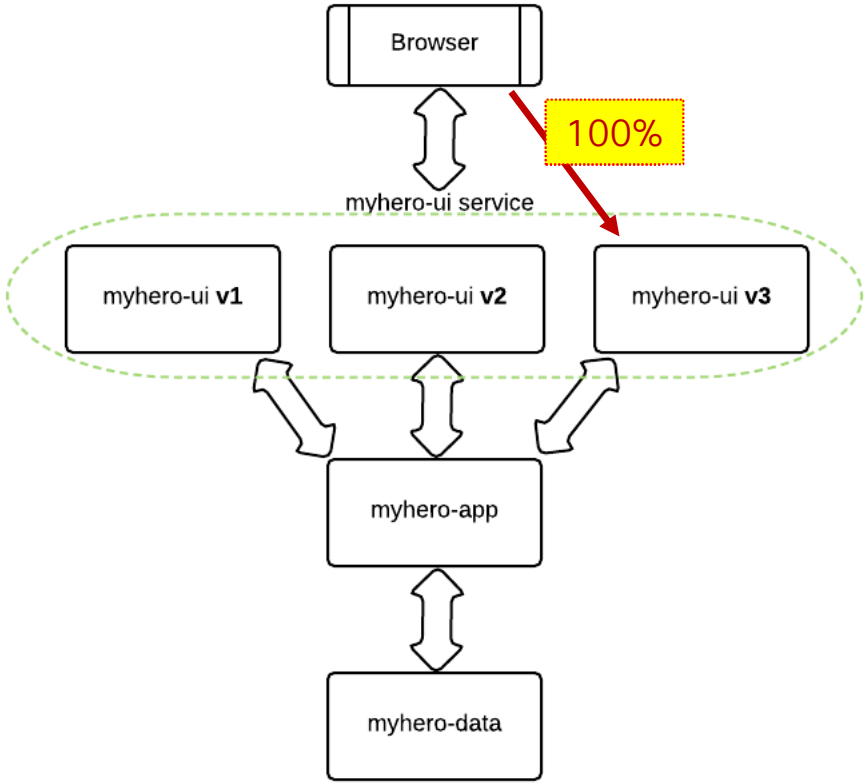
Use case 4
Gradual migration

```
kind: VirtualService
spec:
  hosts:
  - "ui.xxx.com"
  gateways:
  - myhero-gateway
  http:
  - route:
    - destination:
        host: myhero-ui.myhero.svc.cluster.local
        subset: v1
      weight: 50
    - destination:
        host: myhero-ui.myhero.svc.cluster.local
        subset: v3
      weight: 50
```



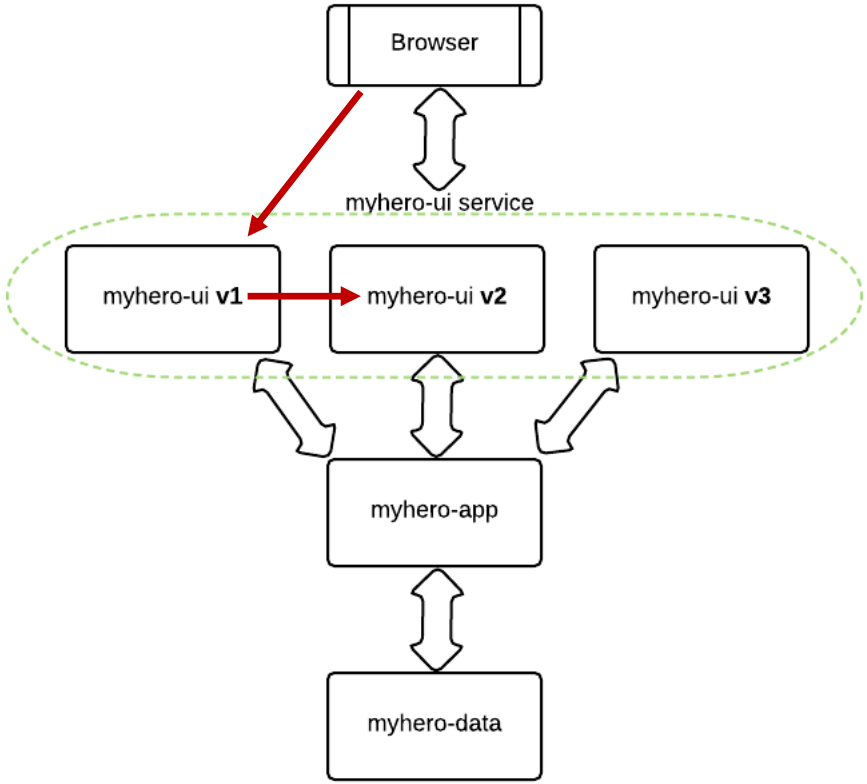
Use case 4
Gradual migration

```
kind: VirtualService
spec:
  hosts:
  - "ui.xxx.com"
  gateways:
  - myhero-gateway
  http:
  - route:
    - destination:
        host: myhero-ui.myhero.svc.cluster.local
        subset: v3
      weight: 100
```



Use case 5
Traffic mirroring

```
kind: VirtualService
spec:
  hosts:
  - "ui.xxx.com"
  gateways:
  - myhero-gateway
  http:
  - route:
    - destination:
        host: myhero-ui.myhero.svc.cluster.local
        subset: v1
    mirror:
      host: myhero-ui.myhero.svc.cluster.local
      subset: v2
```



Conclusion

What did we learn?

- What is a service mesh
- Why you might need one
- Istio's architecture
- How does it work
- What are its capabilities
- How you can use Istio for Traffic Management on Cisco Container Platform

“Service Meshes actually are the new black.”

Questions?

cs.co/juliodevops

cs.co/julioblog

Got more questions? Stay in touch!

Julio Gómez

jgomez2@cisco.com
@juliodevops
github.com/juliogomez

CISCO DEVNET
LEARN CODE INSPIRE CONNECT

developer.cisco.com

@CiscoDevNet
facebook.com/ciscodevnet/
github.com/CiscoDevNet

Complete your online session survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on ciscolive.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.com.

cisco Live!

Continue your education

Demos in the Cisco Showcase

Walk-In Labs

Meet the Engineer 1:1 meetings

Related sessions

Thank you

cisco Live!

You make **possible**