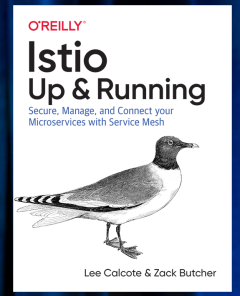
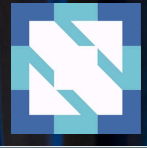
Introduction to Istio

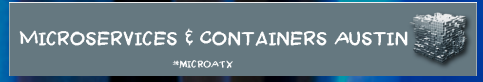
Decoupling at Layer 5

Lee Calcote

calcotestudios.com/talks







Lee Calcote

clouds, containers, functions, applications, and their management

@lcalcote

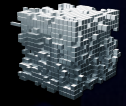
gingergeek.com

github.com/leecalcote

lee@calcotestudios.com

calcotestudios.com/talks

linkedin.com/in/leecalcote



Calcote Studios

Girish Ranganathan

Open Source Mantainer

You have questions.

He has answers.

What is Istio?

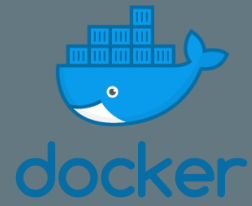
an open platform to connect, manage, and secure microservices

- Observability
- Resiliency
- Traffic Control
- Security
- Policy Enforcement

Prereq

Confirm Prerequisites

Docker and Kubernetes



1. Start Docker Desktop, Minikube or other.

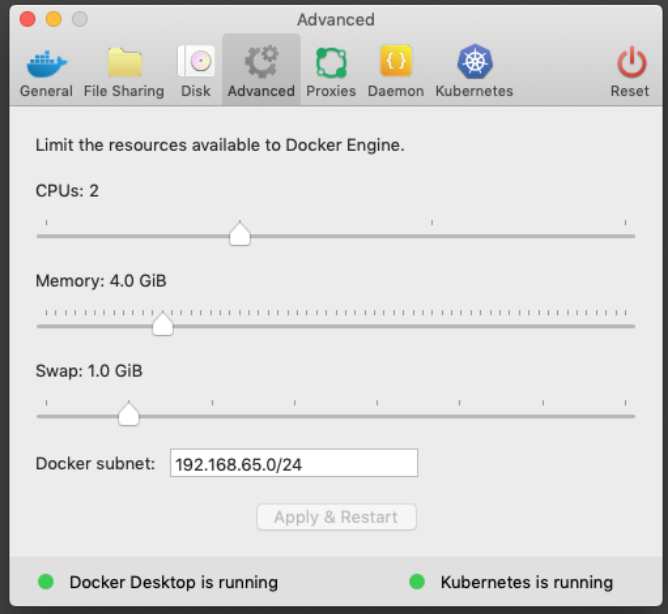
   (either single-node or multi-node clusters will work)

2. Verify that you have a functional Docker environment by running :

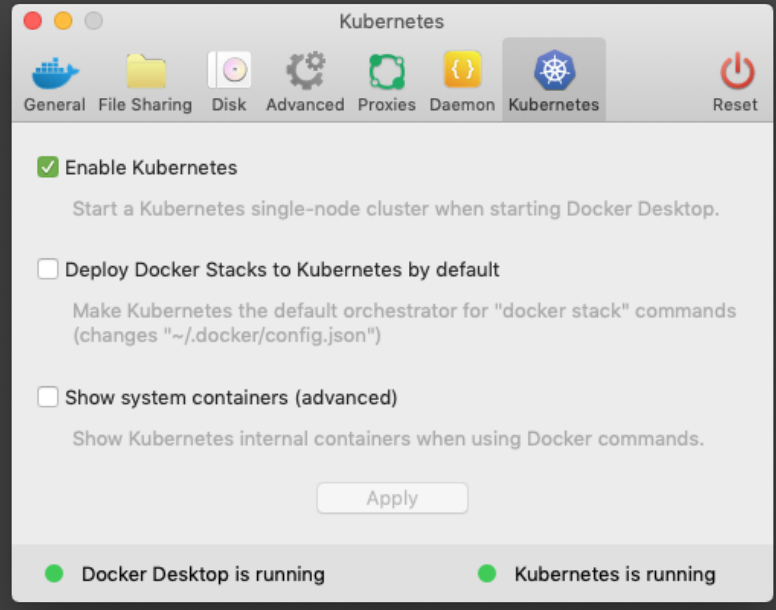
```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:0e11c388b664df8a27a901dce21eb89f11d8292f7fca1b3e3c4321bf7897bffe
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
```

Prereq

Prepare Docker Desktop



Ensure your Docker Desktop VM has 4GB of memory assigned.



Ensure Kubernetes is enabled.

Prereq

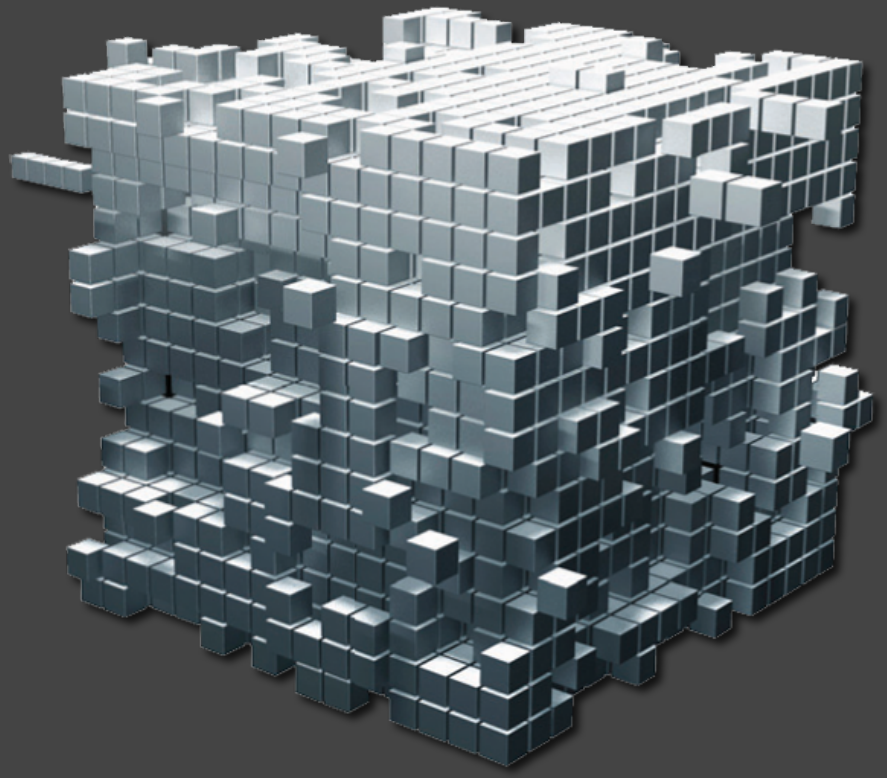
Deploy Kubernetes

1. Confirm access to your Kubernetes cluster.

```
$ kubectl version --short
Client Version: v1.14.1
Server Version: v1.14.1
```

```
$ kubectl get nodes
NAME                 STATUS   ROLES    AGE   VERSION
docker-desktop   Ready    master   10m   v1.14.1
```

v1.9 or higher



The Service Mesh Community

Join Slack
http://slack.layer5.io

Subscribe to newsletter
https://layer5.io/subscribe


Relating to
Service Meshes

Which is why...
I have a container orchestrator.

Service meshes generally rely on these underlying layers.

Core Capabilities

- Cluster Management
  - Host Discovery
  - Host Health Monitoring
- Scheduling
- Orchestrator Updates and Host Maintenance
- Service Discovery
- Networking and Load Balancing
- Stateful Services
- Multi-Tenant, Multi-Region

Additional Key Capabilities

- Application Health and Performance Monitoring
- Application Deployments
- Application Secrets

minimal capabilities required to qualify as a container orchestrator

Service meshes generally rely on these underlying layers.

Core Capabilities

- Cluster Management
  - Host Discovery
  - Host Health Monitoring
- Scheduling
- Orchestrator Updates and Host Maintenance
- Service Discovery
- Networking and Load Balancing
- Stateful Services
- Multi-Tenant, Multi-Region

Additional Key Capabilities

- Application Health and Performance Monitoring
- Application Deployments
- Application Secrets

minimal capabilities required to qualify as a container orchestrator

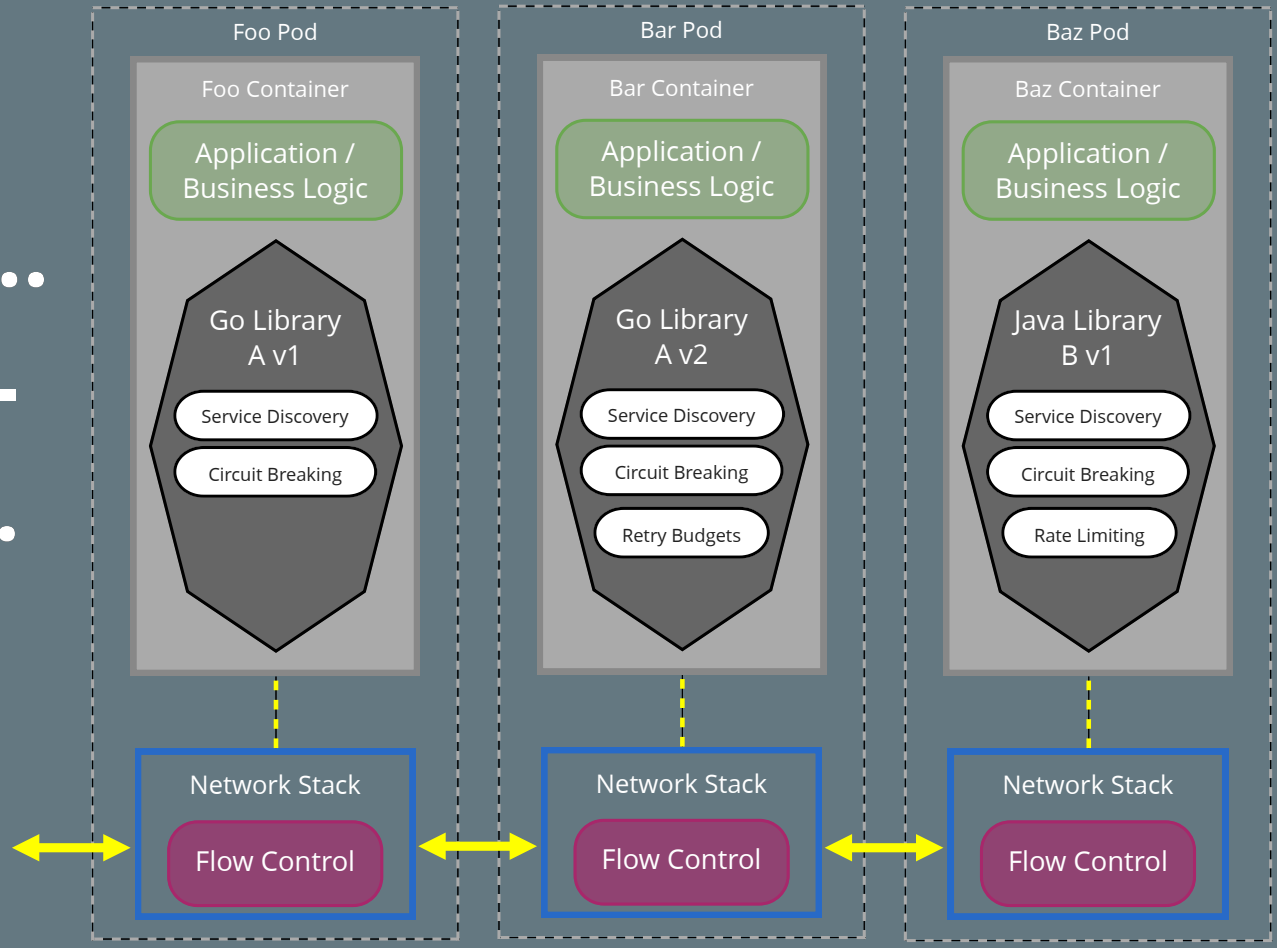
Which is why... I have an API gateway.

Which is why... I have an API gateway.

Microservices API Gateways

- Ambassador uses Envoy
- Kong uses Nginx
- OpenResty uses Nginx
- Traefik

Which is why... I have client-side libraries.



Enforcing consistency is challenging.

@lcalcote

Why use a Service Mesh?

to avoid...

- Bloated service code
- Duplicating work to make services production-ready
  - Load balancing, auto scaling, rate limiting, traffic routing...
- Inconsistency across services
  - Retry, tls, failover, deadlines, cancellation, etc., for each language, framework
  - Siloed implementations lead to fragmented, non-uniform policy application and difficult debugging
- Diffusing responsibility of service management

What is a Service Mesh?

a dedicated layer for managing service-to-service communication

What is a Service Mesh?

a dedicated layer for managing service-to-service communication

So, a microservices platform?

What is a Service Mesh?

a dedicated layer for managing service-to-service communication

So, a microservices platform?

partially.

What is a Service Mesh?

a dedicated layer for managing service-to-service communication

So, a microservices platform?

a services-first network

What is a Service Mesh?

a dedicated layer for managing service-to-service communication

So, a microservices platform?

a services-first network

Orchestrators don't bring all that you need
and neither do service meshes,
but they do get you closer.

What do we need?

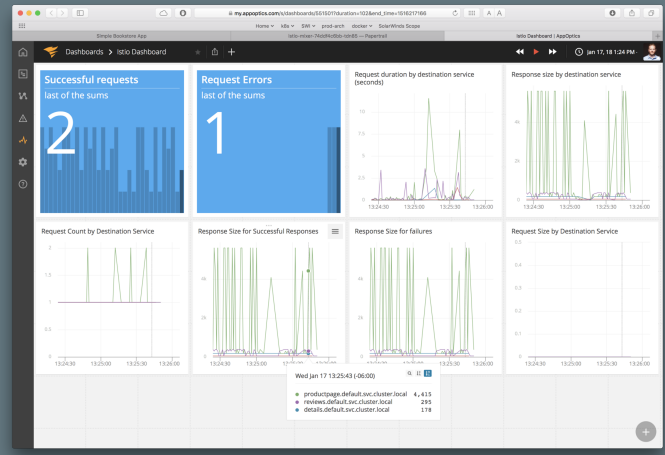
- Observability
  - Logging
  - Metrics
  - Tracing
- Traffic Control
- Resiliency
- Efficiency
- Security
- Policy

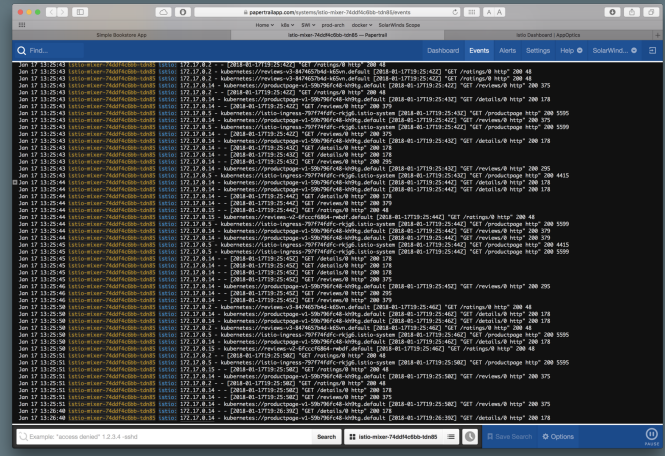
Observability

what gets people hooked on service metrics

Goals

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





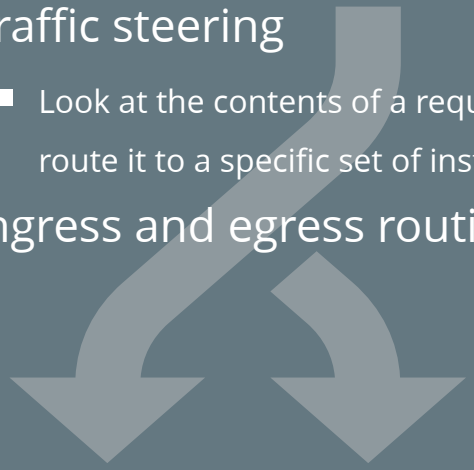
You get a metric! You get a metric! Everyone gets a metric!

@lcalcote

Traffic Control

content-based traffic steering

- Traffic splitting
  - L7 tag based routing?
- Traffic steering
  - Look at the contents of a request and route it to a specific set of instances.
- Ingress and egress routing



Resiliency

control over chaos

- Systematic fault injection
- Timeouts and Retries with timeout budget
- Control connection pool size and request load
- Circuit breakers and Health checks

Missing: distributed debugging; provide nascent visibility (topology)
Missing: application lifecycle management, but not by much

@lcalcote

Decoupling at Layer 5

where Dev and Ops meet

Problem: too much infrastructure code in services

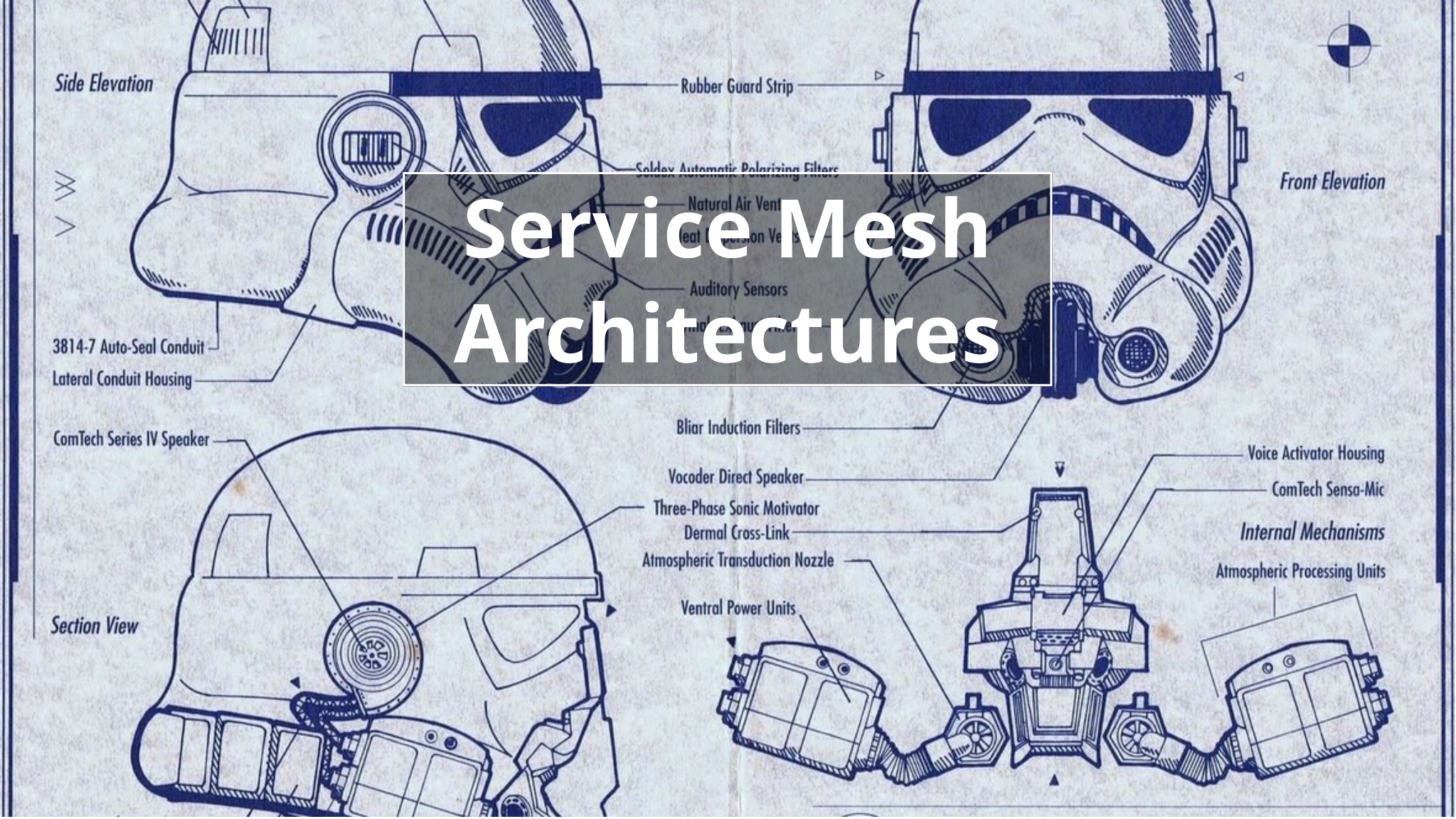
@lcalcote

Help with Modernization

address the long-tail of IT services

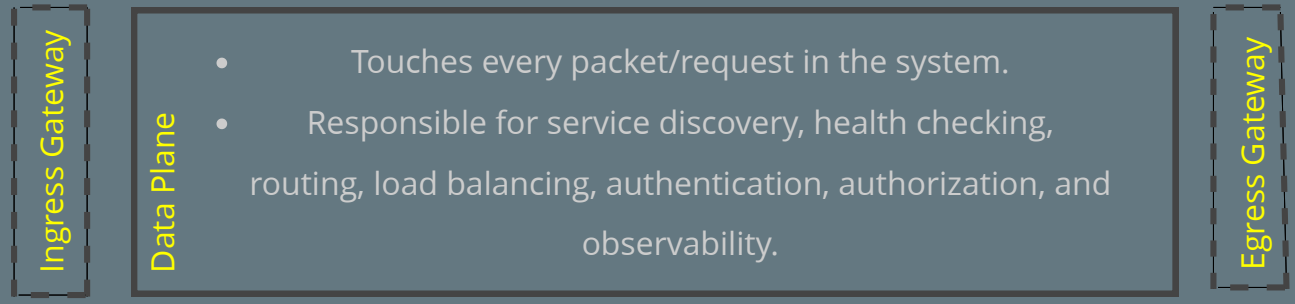
- Can modernize your IT inventory without:
  - Rewriting your applications
  - Adopting microservices, regular services are fine
  - Adopting new frameworks
  - Moving to the cloud

Q&A

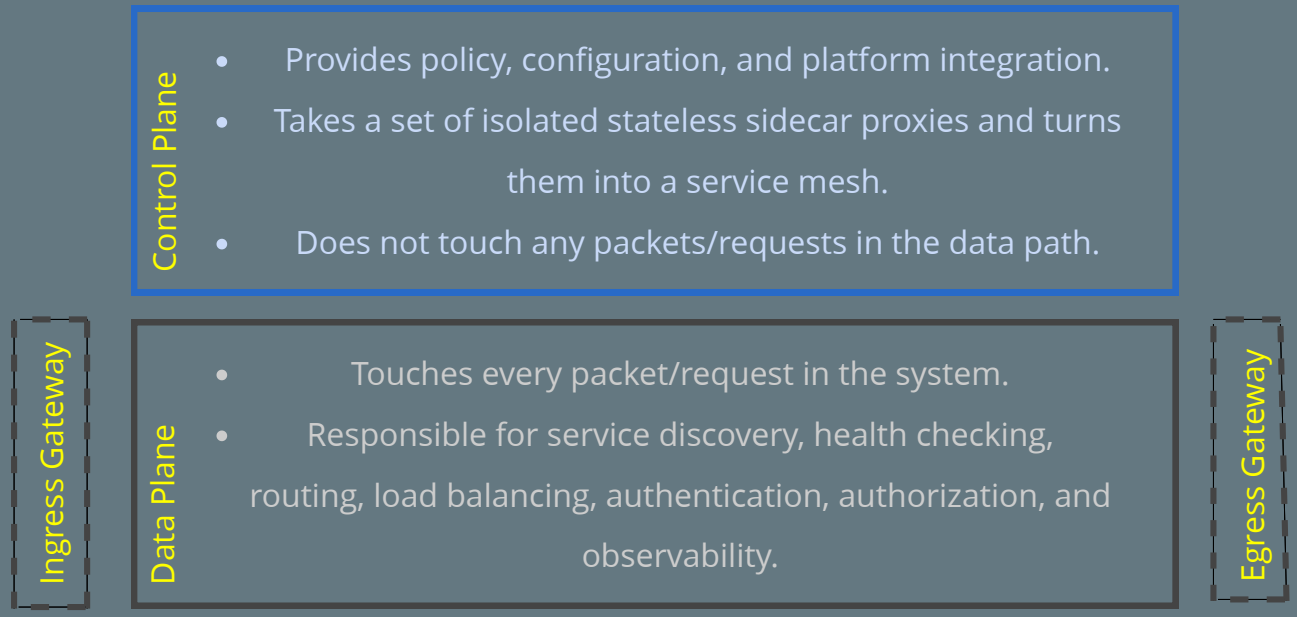


Service Mesh Architectures

Service Mesh Architecture



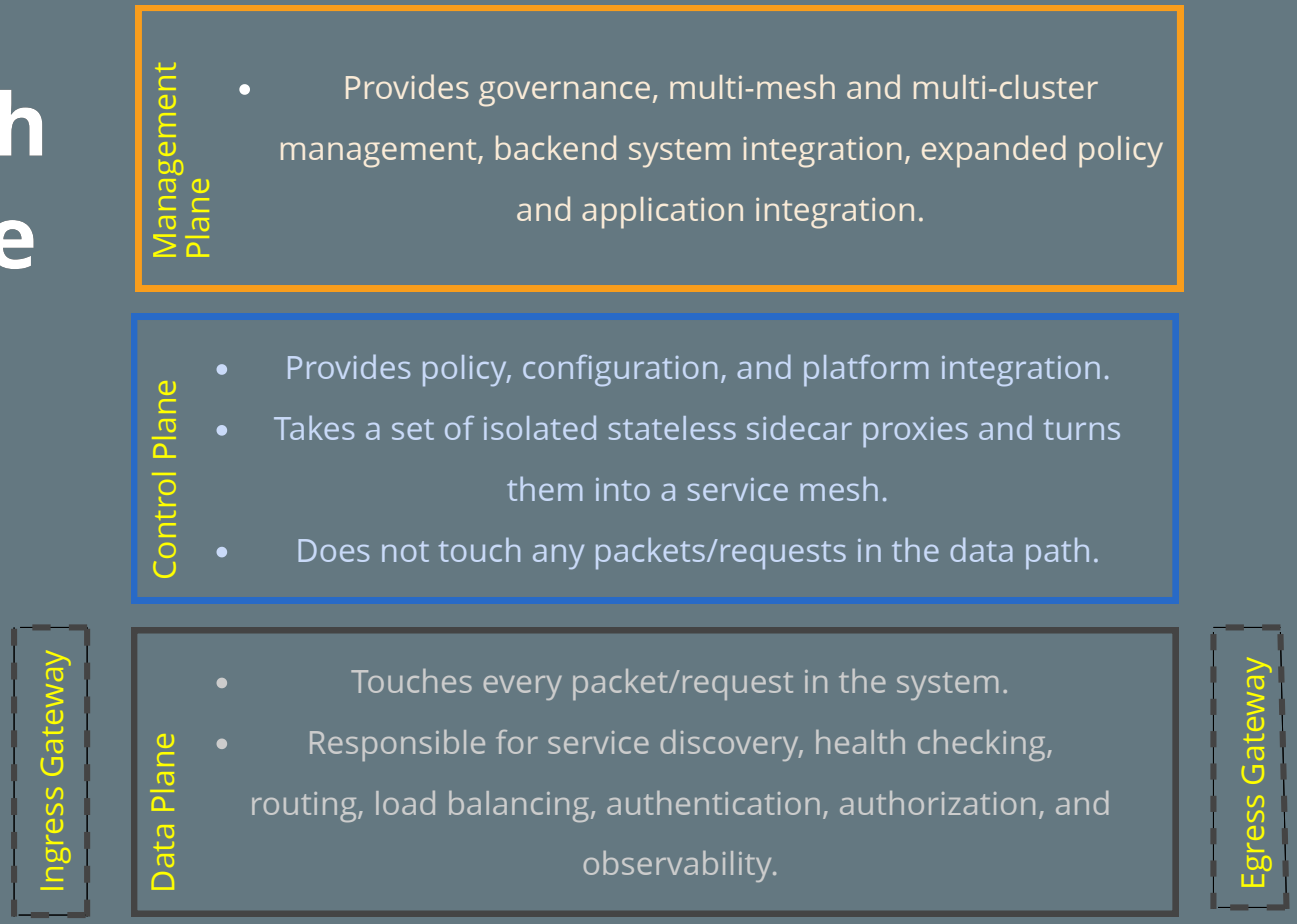
Service Mesh Architecture



No control plane? Not a service mesh.

@lcalcote

Service Mesh Architecture

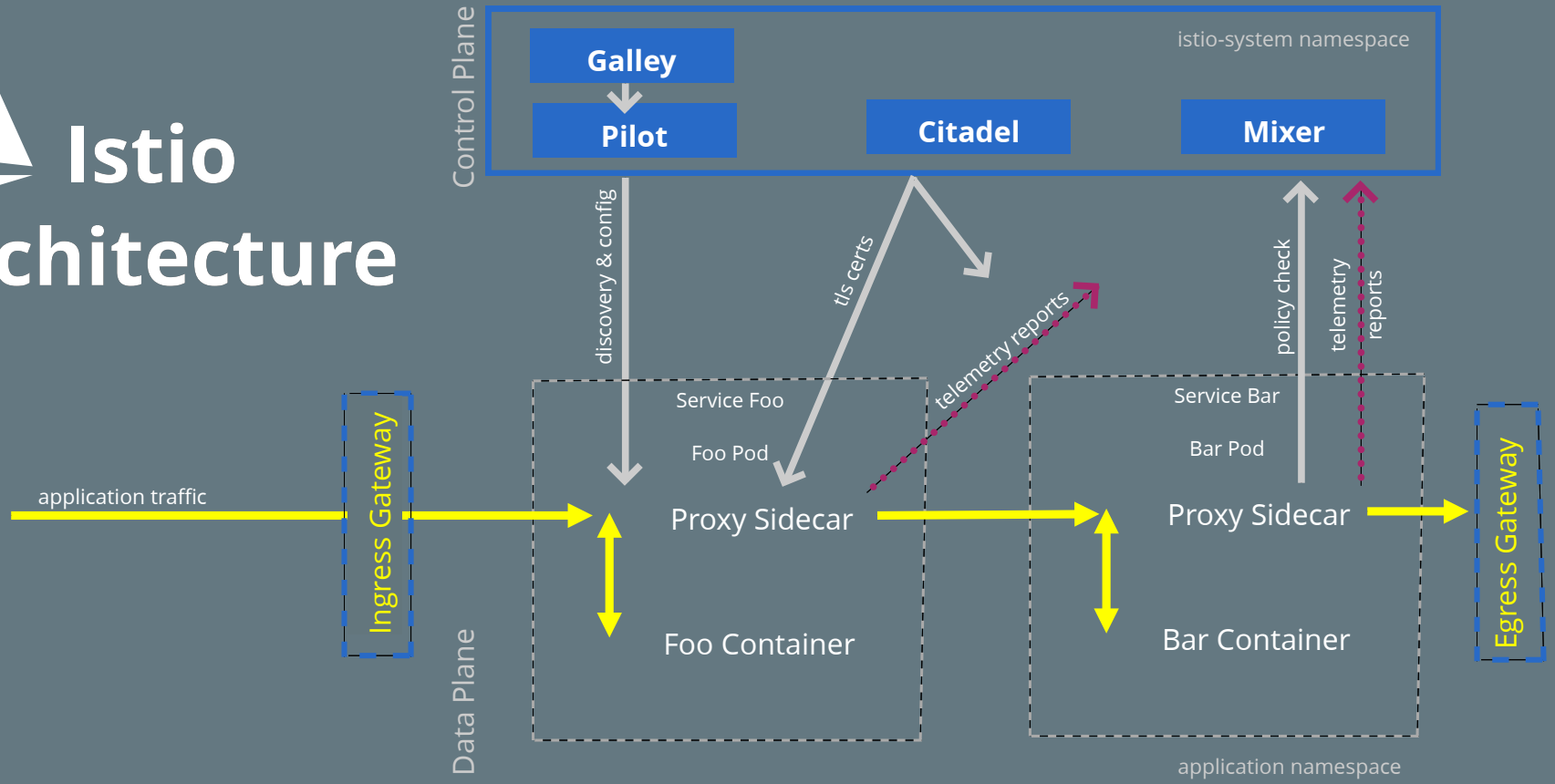


You need a management plane.

@lcalcote

Istio Architecture

Istio Architecture



Out-of-band telemetry propagation

Control flow

Application traffic

@lcalcote

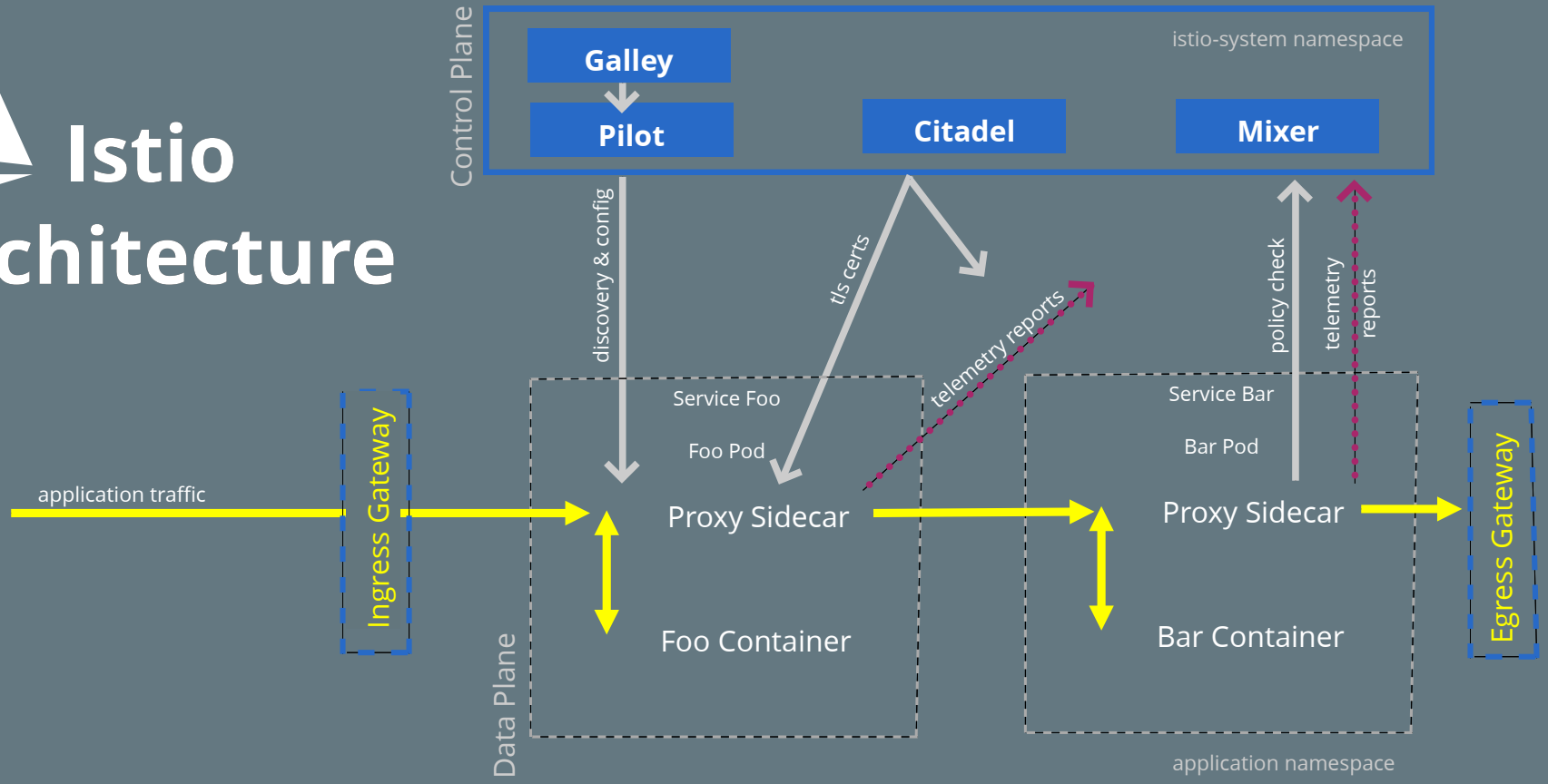
layer5.io/landscape

@layer5

Our service mesh of study: Istio

Istio Architecture

Istio Architecture



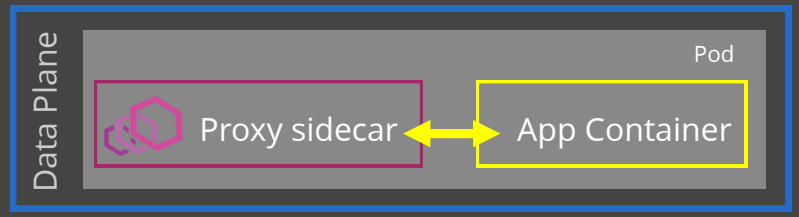
Out-of-band telemetry propagation

Control flow during request processing

Application traffic

@lcalcote

Service Proxy Sidecar



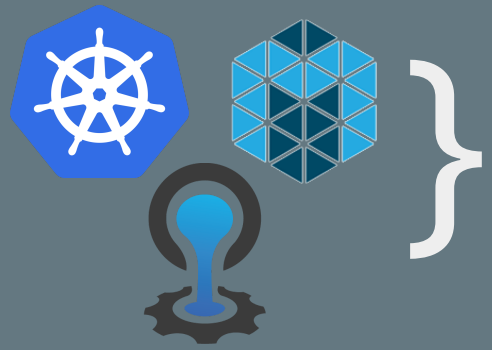
- A C++ based L4/L7 proxy
- Low memory footprint
- In production at Lyft™

Capabilities:

- API driven config updates → no reloads
- Zone-aware load balancing w/ failover
- Traffic routing and splitting
- Health checks, circuit breakers, timeouts, retry budgets, fault injection...
- HTTP/2 & gRPC
- Transparent proxying
- Designed for observability

What's Pilot for?

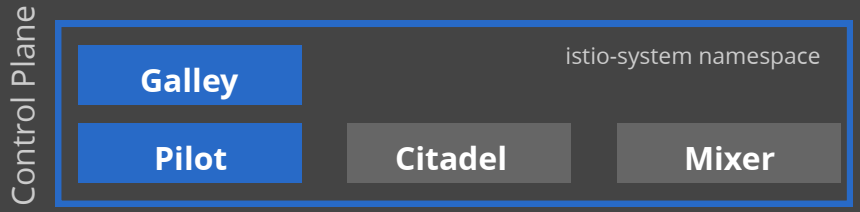
the head of the ship



provides abstraction from underlying platforms

provides service discovery to sidecars

manages sidecar configuration

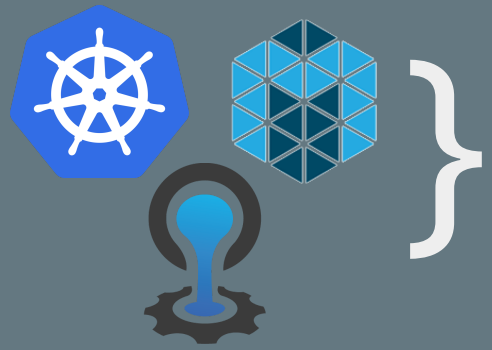


system of record for service mesh

@lcalcote

What's Pilot for?

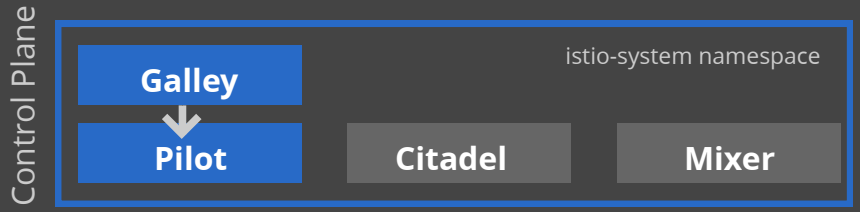
the head of the ship



provides abstraction from underlying platforms

provides service discovery to sidecars

manages sidecar configuration



system of record for service mesh

@lcalcote

What's Mixer for?

operator-focused

1. Precondition checking
2. Quota management
3. Telemetry reporting

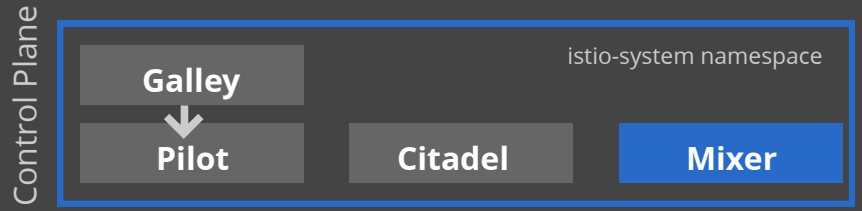
- Point of integration with infrastructure back ends
  - Intermediates between Istio and back ends, under operator control
  - Enables platform and environment mobility
- Responsible for policy evaluation and telemetry reporting
  - Provides granular control over operational policies and telemetry
- Has a rich configuration model
  - Intent-based config abstracts most infrastructure concerns

an attribute-processing and routing machine

@lcalcote

What's Mixer for?

operator-focused



1. Precondition checking
2. Quota management
3. Telemetry reporting

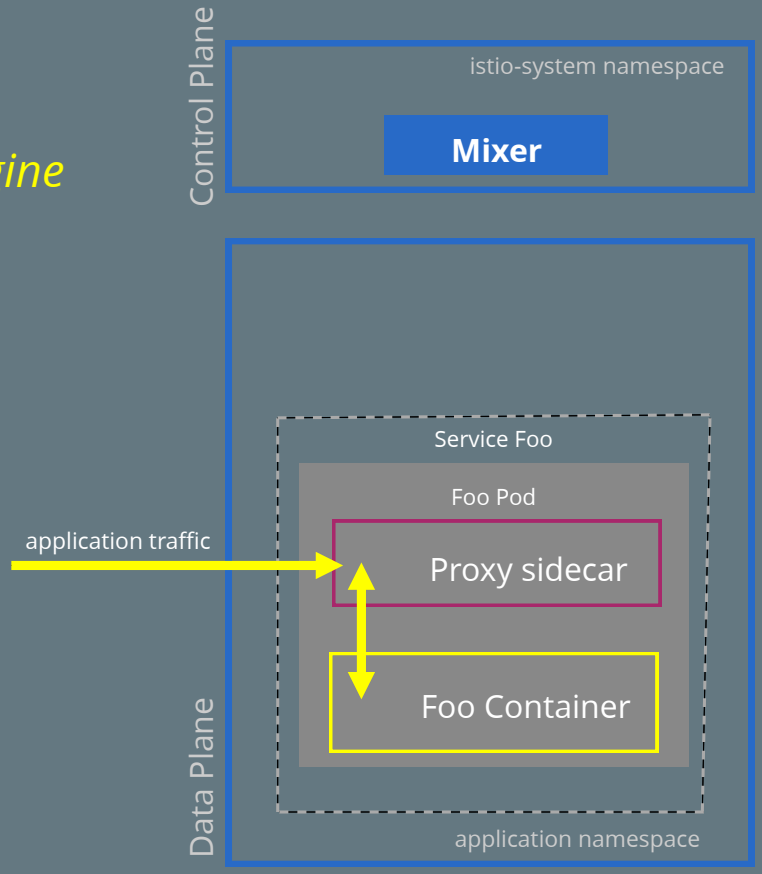
- Point of integration with infrastructure back ends
  - Intermediates between Istio and back ends, under operator control
  - Enables platform and environment mobility
- Responsible for policy evaluation and telemetry reporting
  - Provides granular control over operational policies and telemetry
- Has a rich configuration model
  - Intent-based config abstracts most infrastructure concerns

an attribute-processing and routing machine

@lcalcote

Mixer

an attribute processing engine



Out-of-band telemetry propagation

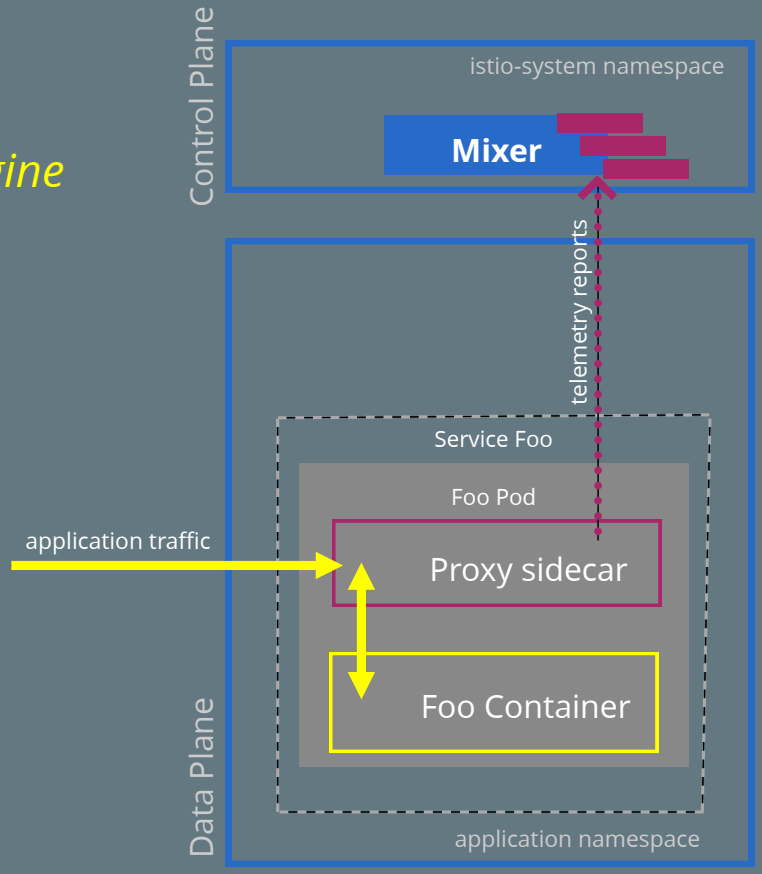
Control flow during request processing

application traffic

@lcalcote

Mixer

an attribute processing engine



Out-of-band telemetry propagation

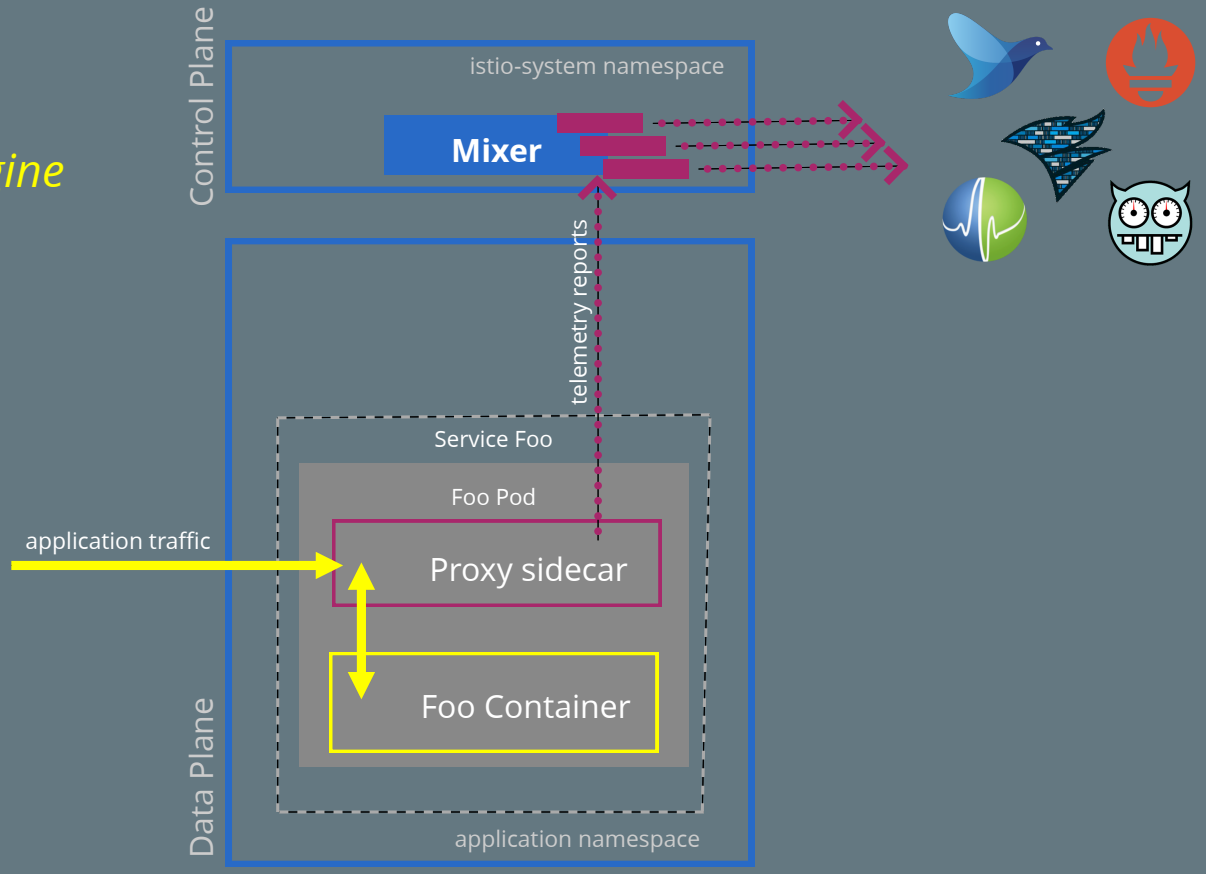
Control flow during request processing

application traffic

@lcalcote

Mixer

an attribute processing engine



Out-of-band telemetry propagation

Control flow during request processing

application traffic

@lcalcote

What's Citadel for?

security at scale

- Verifiable identity
  - Issues certs
  - Certs distributed to service proxies
  - Mounted as a Kubernetes secret
- Secure naming / addressing
- Traffic encryption

Orchestrate Key & Certificate:

- Generation
- Deployment
- Rotation
- Revocation

security by default

@lcalcote

What's Citadel for?

security at scale

- Verifiable identity
  - Issues certs
  - Certs distributed to service proxies
  - Mounted as a Kubernetes secret
- Secure naming / addressing
- Traffic encryption

Orchestrate Key & Certificate:

- Generation
- Deployment
- Rotation
- Revocation

security by default

@lcalcote

Lab 1

Download and Deploy Istio Resources

1. Download Istio

```
curl -L https://git.io/getLatestIstio | ISTIO_VERSION=1.1.7 sh -
```

2. Familiarize with `istioctl`

```
istioctl version
```

3. Install Istio custom resources

```
for i in install/kubernetes/helm/istio-init/files/crd*yaml;
do kubectl apply -f $i; done
```

4. Verify custom resource installation

```
kubectl get crd | grep istio
```

Lab 1

Deploy Istio

1. Install Istio

```
kubectl apply -f install/kubernetes/istio-demo.yaml
```

2. Find control plane namespace

```
kubectl get namespaces
```

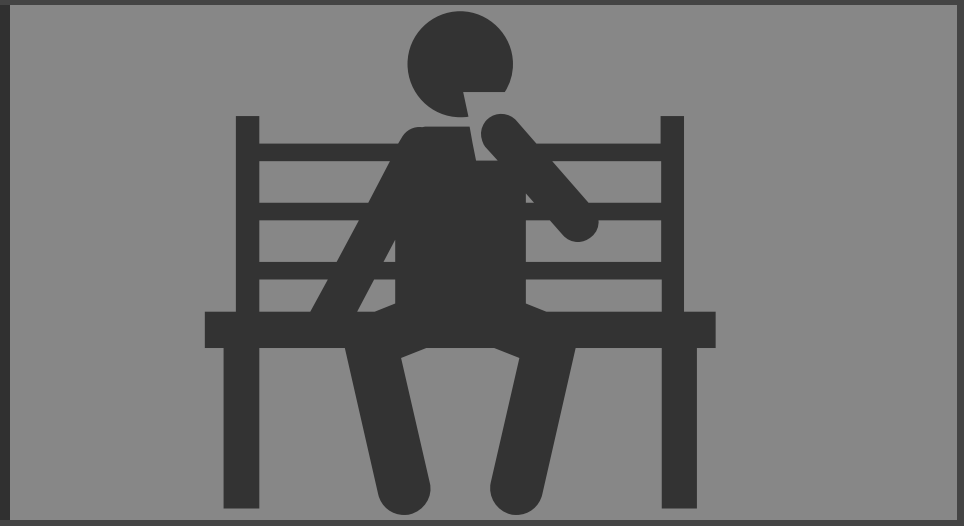
3. Inspect control plane services

```
kubectl get svc -n istio-system
```

4. Inspect control plane components

```
kubectl get pod -n istio-system
```

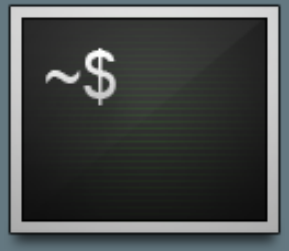
Q&A



Break

Lab 2

Deploy Sample App

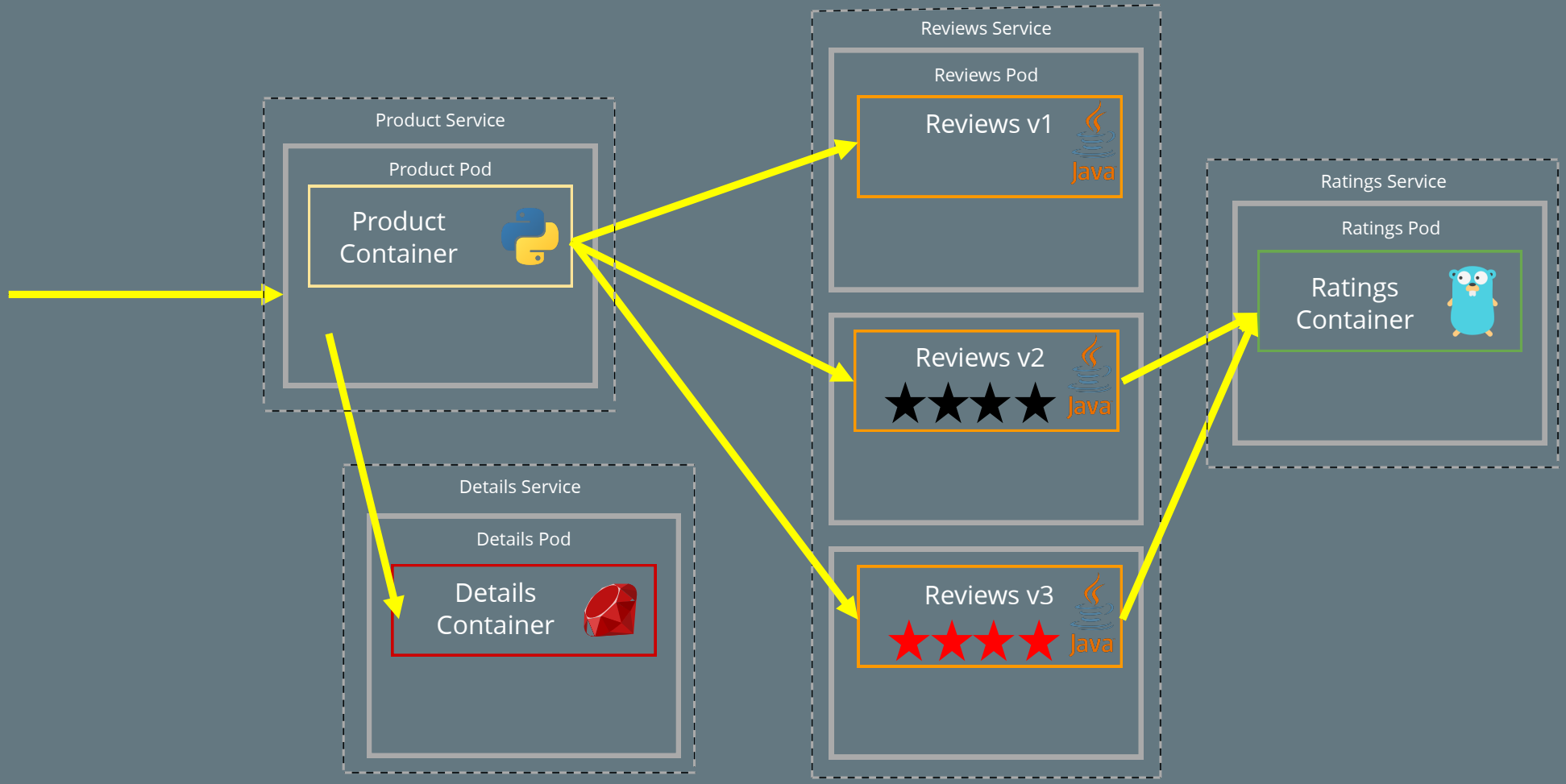


1. Multi-language, multi-service application
2. Automatic vs manual sidecar injection
3. Verify install

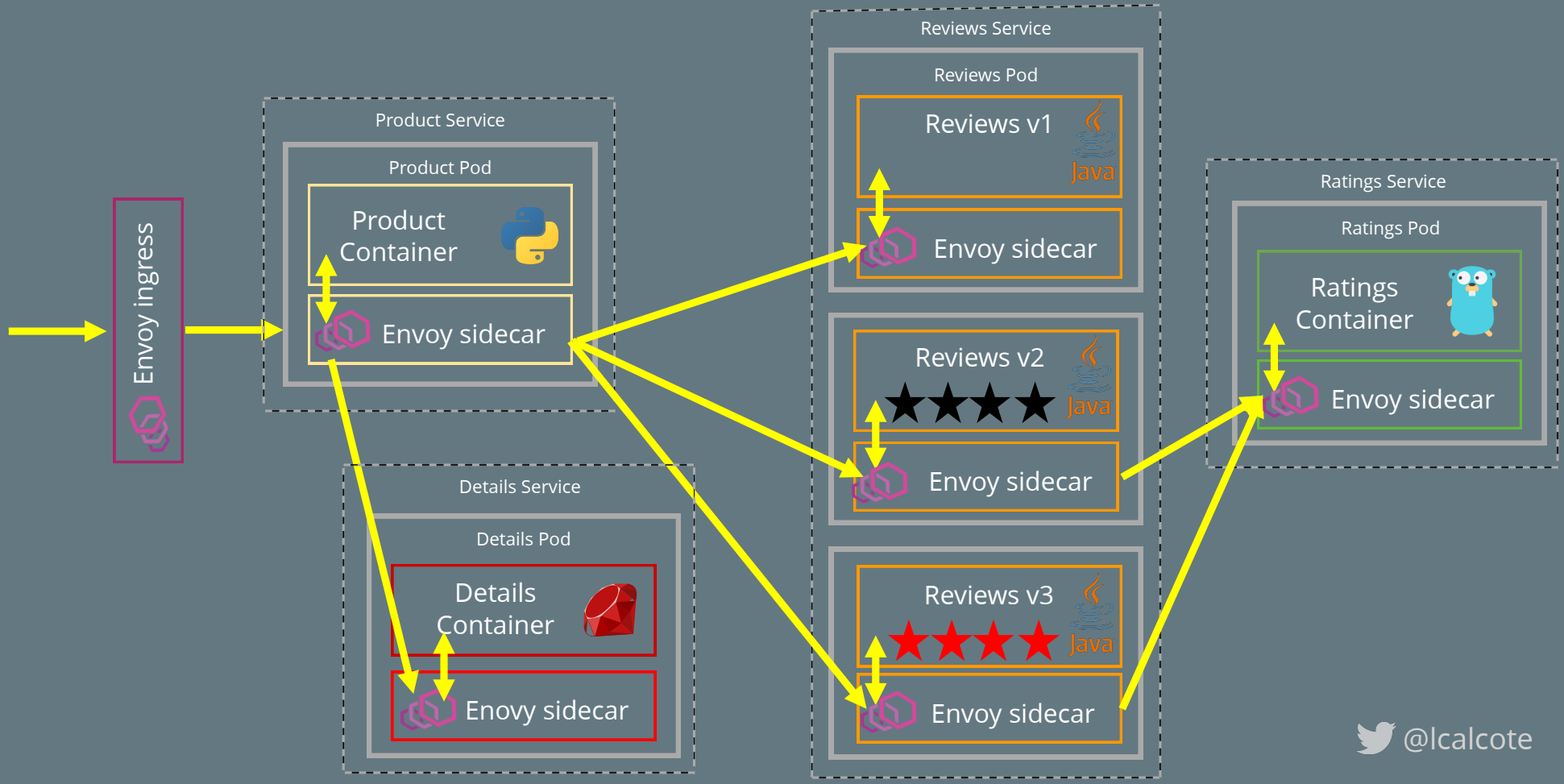
Sample app is in your release distribution folder at samples/bookinfo

@lcalcote

BookInfo Sample App



BookInfo Sample App on Service Mesh



@lcalcote

Lab 2

Sidecar Injection

Sidecars proxy can be either manually or automatically injected into your pods.

Automatic sidecar injection leverages Kubernetes' Mutating Webhook Admission Controller.

1. Verify whether your Kubernetes deployment supports these APIs by executing:

```
kubectl api-versions | grep admissionregistration
```

2. Inspect the istio-sidecar-injector webhook:

```
kubectl get mutatingwebhookconfigurations
kubectl get mutatingwebhookconfigurations istio-sidecar-injector -o yaml
```

If your environment does NOT this API, then you may manually inject the istio sidecar.

Lab 2

Deploy Sample App with Automatic Sidecar Injection

1. Verify presence of the sidecar injector

```
kubectl -n istio-system get deployment -l istio=sidecar-injector
```

2. Confirm namespace label

```
kubectl get ns -L istio-injection
```

Lab 2

Deploy Sample App with Manual Sidecar Injection

In namespaces without the istio-injection label, you can use istioctl kube-inject to manually inject Envoy containers in your application pods before deploying them:

```
istioctl kube-inject -f <your-app-spec>.yaml | kubectl apply -f -
```

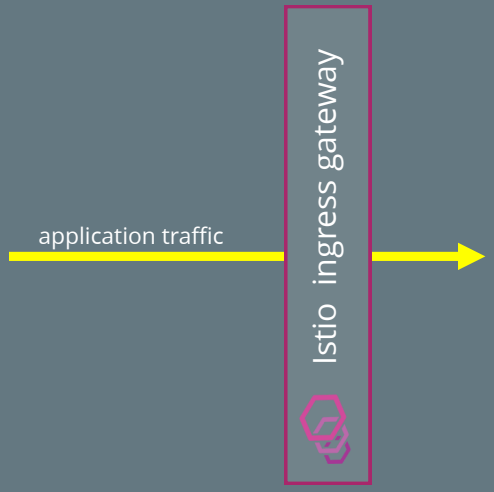
Lab 3

Expose BookInfo via Istio Ingress Gateway

1. Inspecting the Istio Ingress Gateway
2. Configure Istio Ingress Gateway for Bookinfo
3. Inspect the Istio proxy of the productpage pod

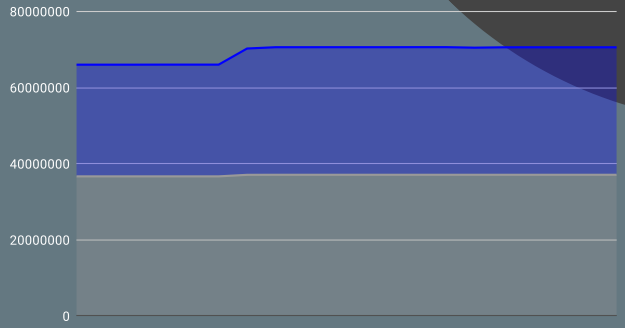
Lab 3

Expose BookInfo via Istio Ingress Gateway

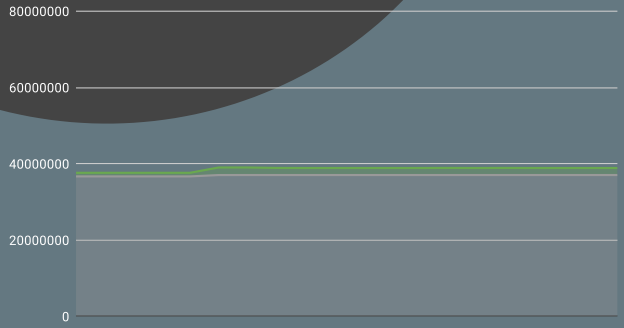


1. Inspecting the Istio Ingress Gateway
2. Configure Istio Ingress Gateway for Bookinfo
3. Inspect the Istio proxy of the productpage pod

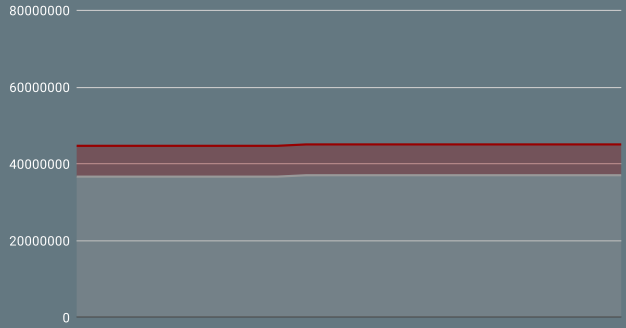
Istio sidecar + app memory usage



Linkerd sidecar + app memory usage



Consul sidecar + app memory usage



Lab 4

Load Generation

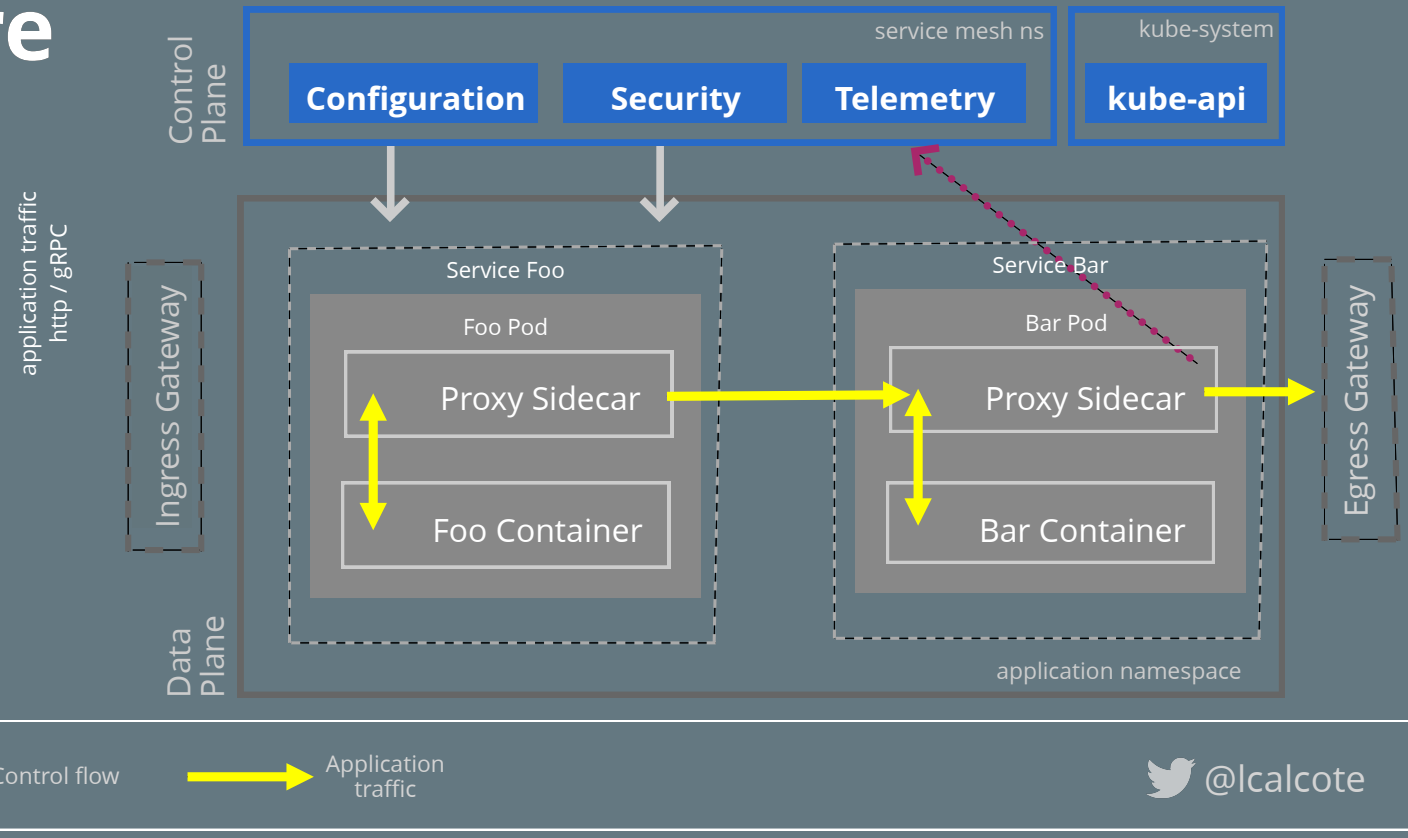
Meshery

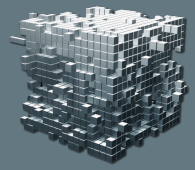
a multi-service mesh management plane

Google
Summer of Code

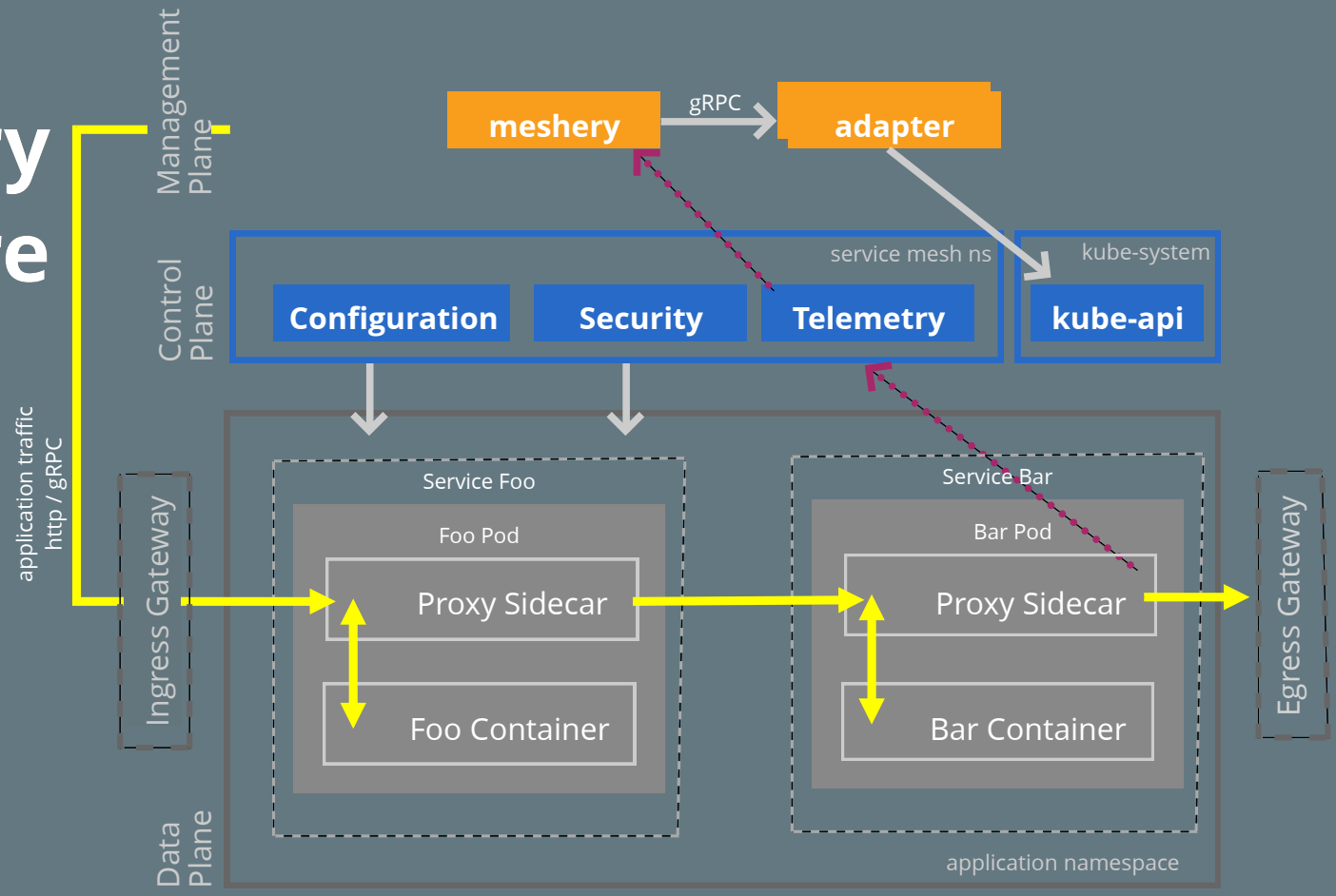
Service Mesh Interface
(SMI)

Meshery Architecture





Meshery Architecture

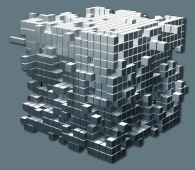


Out-of-band telemetry propagation

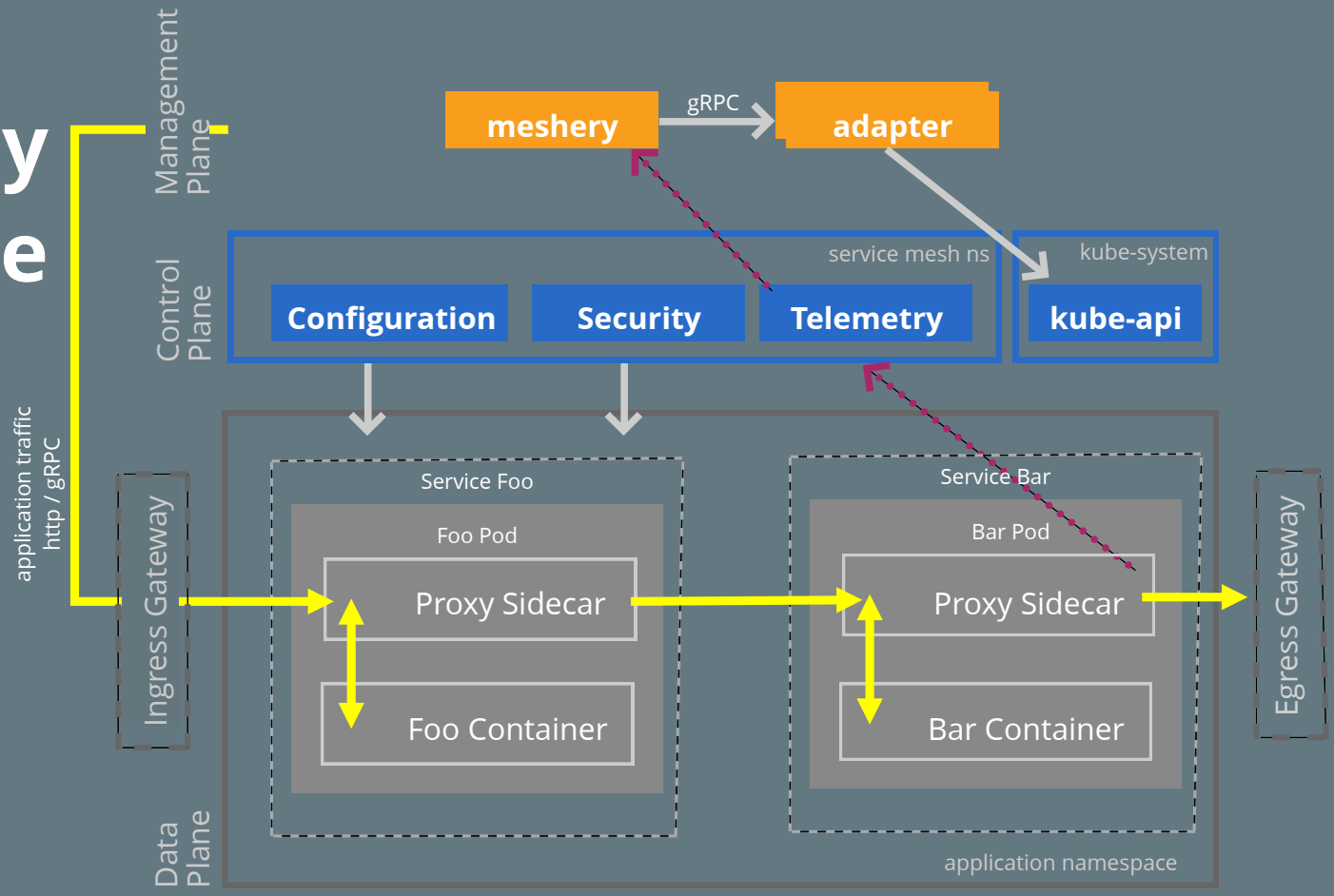
Control flow

Application traffic

@lcalcote



Meshery Architecture



Out-of-band telemetry propagation

Control flow

Application traffic

@lcalcote

Lab 4

Deploy Management Plane

Management Plane

Provides expanded governance, backend system integration, expanded policy, and dynamic application and mesh configuration.

Control Plane

Data Plane

- Generate load on Bookinfo
- View in monitoring tools

1. Install Meshery

```
sudo curl -L https://git.io/meshery -o /usr/local/bin/meshery
sudo chmod a+x /usr/local/bin/meshery
meshery start
```

Mixer Adapters

types: logs, metrics, access control, quota

- Uses pluggable adapters to extend its functionality
  - Adapters run within the Mixer process
- Adapters are modules that interface to infrastructure backends
  - (logging, metrics, quotas, etc.)
- Multi-interface adapters are possible
  - (e.g., SolarWinds® adapter sends logs and metrics)

@lcalcote

Mixer Adapters

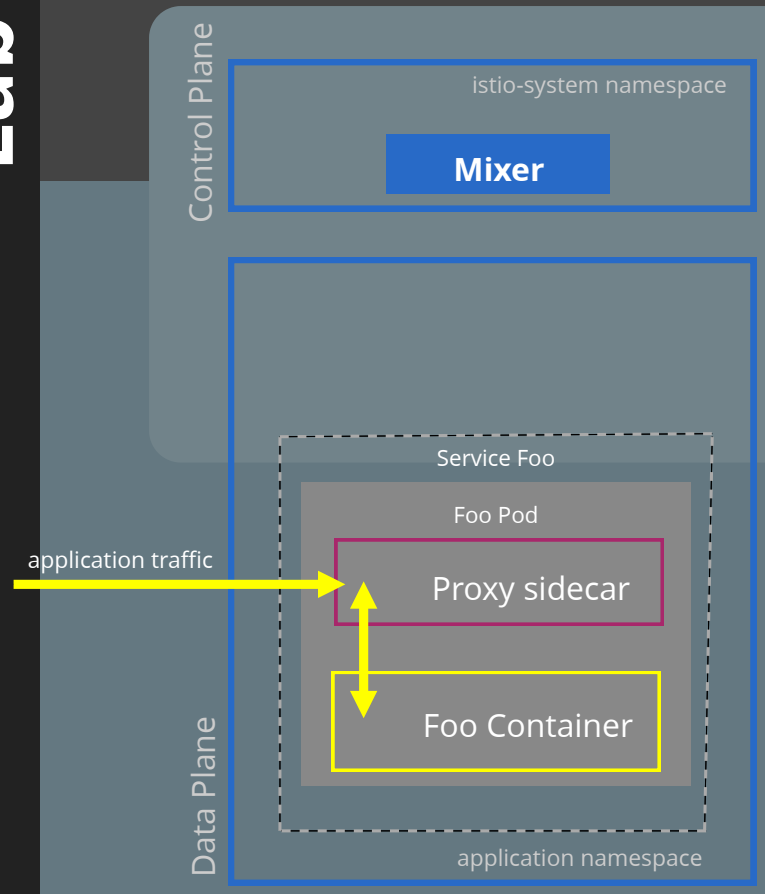
types: logs, metrics, access control, quota

- Uses pluggable adapters to extend its functionality
  - Adapters run within the Mixer process
- Adapters are modules that interface to infrastructure backends
  - (logging, metrics, quotas, etc.)
- Multi-interface adapters are possible
  - (e.g., SolarWinds® adapter sends logs and metrics)

@lcalcote

Lab 4

Metrics



1. Expose services with NodePort:

```
kubectl -n istio-system edit svc prometheus
```

2. Find port assigned to Prometheus:

```
kubectl -n istio-system get svc prometheus
```

Out-of-band telemetry propagation

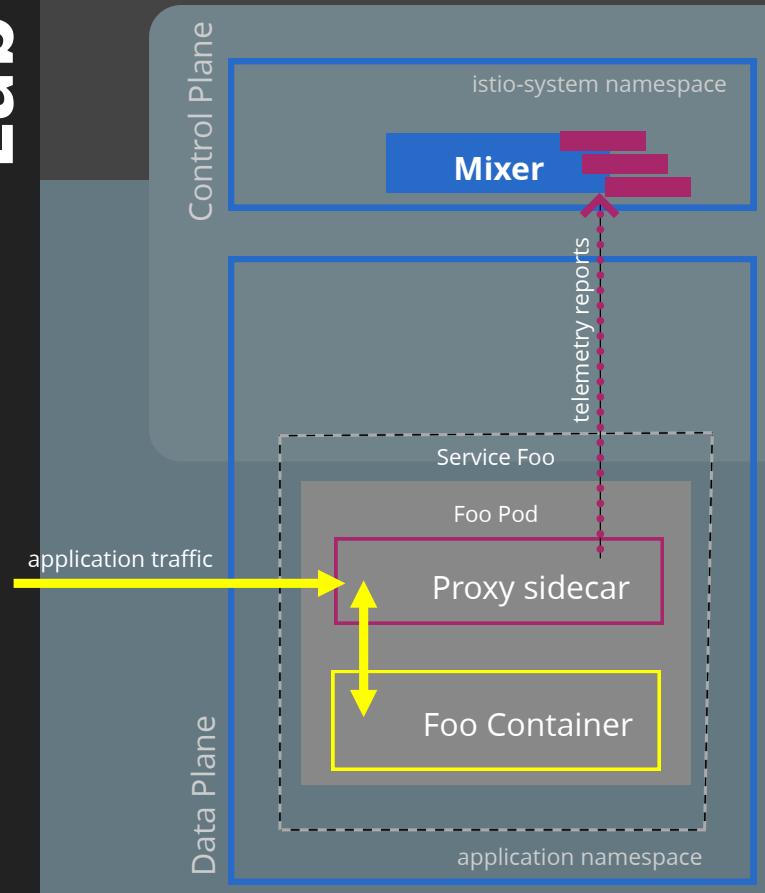
Control flow during request processing

application traffic

@lcalcote

Lab 4

Metrics



1. Expose services with NodePort:

```
kubectl -n istio-system edit svc prometheus
```

2. Find port assigned to Prometheus:

```
kubectl -n istio-system get svc prometheus
```

Out-of-band telemetry propagation

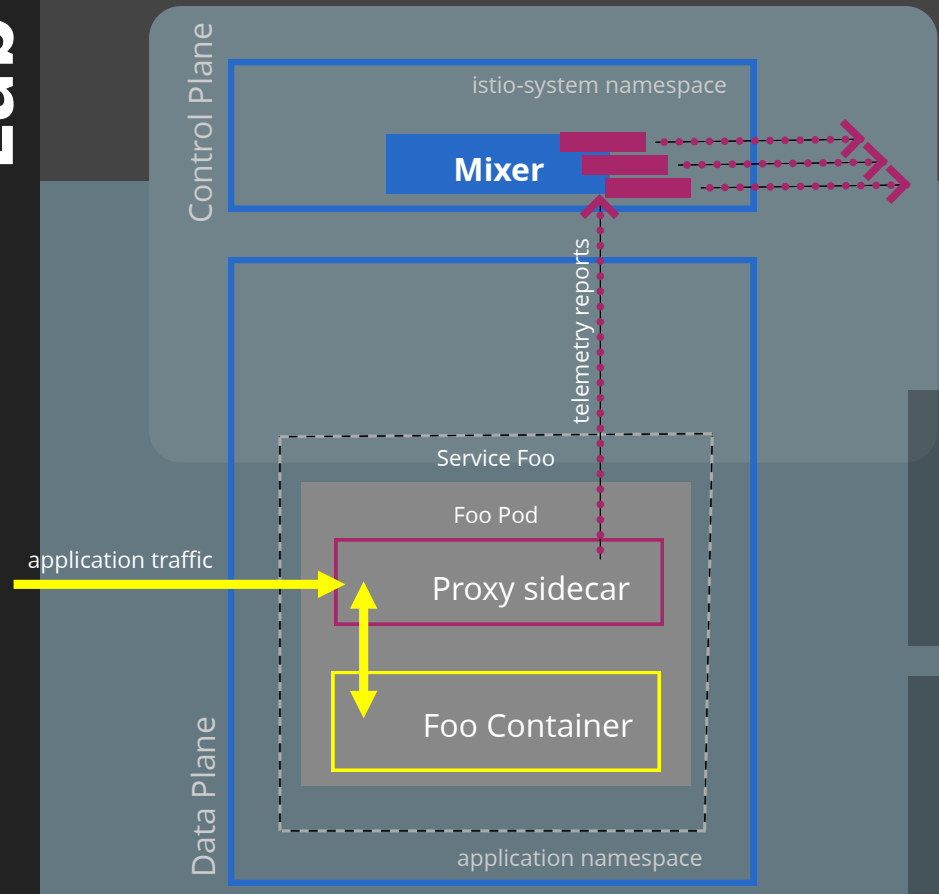
Control flow during request processing

application traffic

@lcalcote

Lab 4

Metrics



1. Expose services with NodePort:

```
kubectl -n istio-system edit svc prometheus
```

2. Find port assigned to Prometheus:

```
kubectl -n istio-system get svc prometheus
```

Out-of-band telemetry propagation

Control flow during request processing

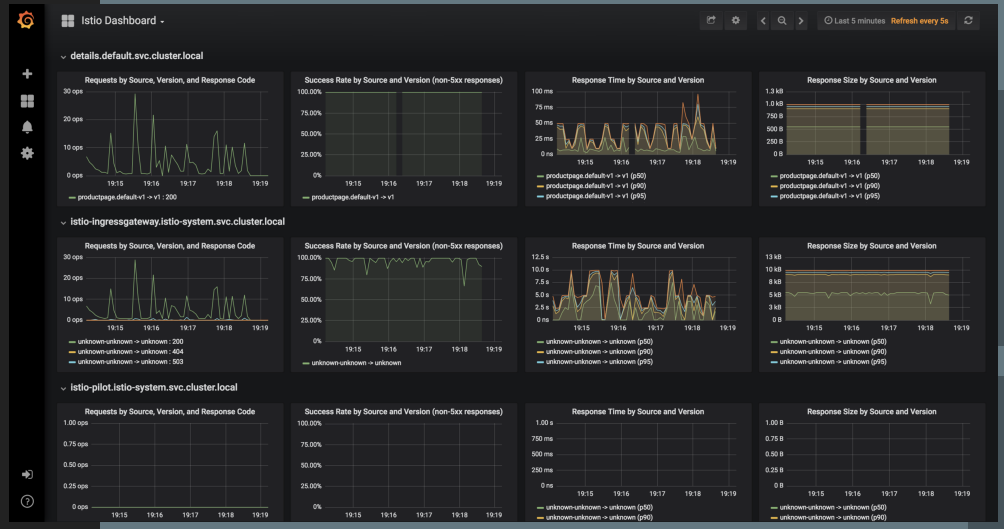
application traffic

@lcalcote

Lab 4

Metrics Dashboard

Grafana



1. Expose services with NodePort:

```
kubectl -n istio-system edit svc grafana
```

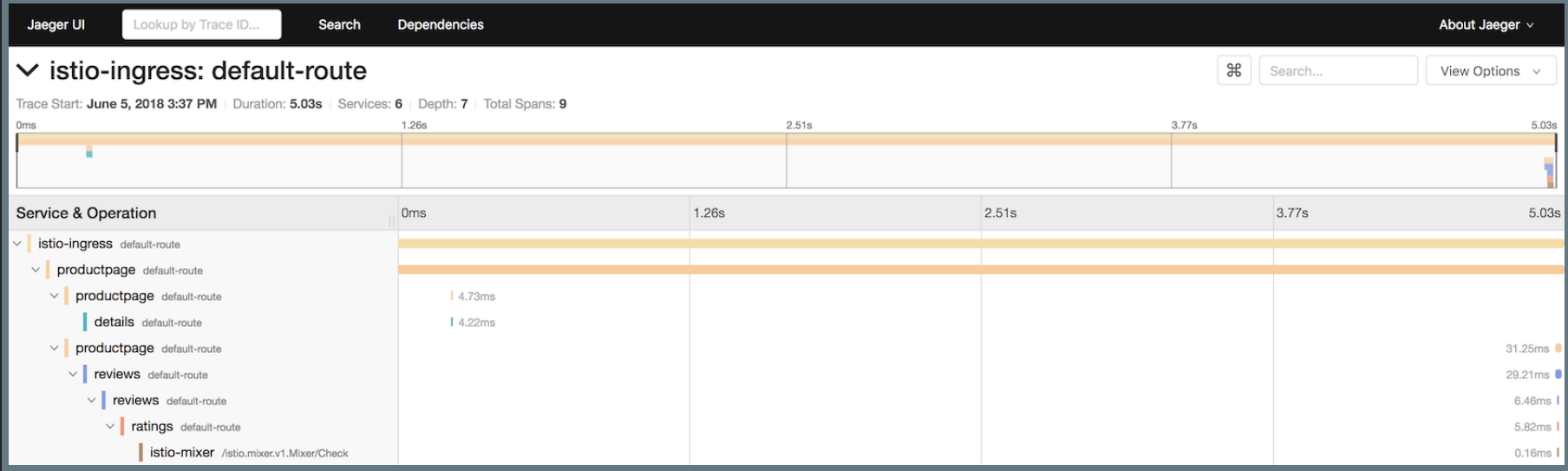
2. Find port assigned to Grafana:

```
kubectl -n istio-system get svc grafana
```

Lab 4

Distributed Tracing

Jaeger



Lab 4

Distributed Tracing

Jaeger

The istio-proxy collects and propagates the following headers from the incoming request to any outgoing requests:

- x-request-id
- x-b3-traceid
- x-b3-spanid
- x-b3-parentspanid
- x-b3-sampled
- x-b3-flags
- x-ot-span-context

1. Expose services with NodePort:

```
kubectl -n istio-system edit svc tracing
```

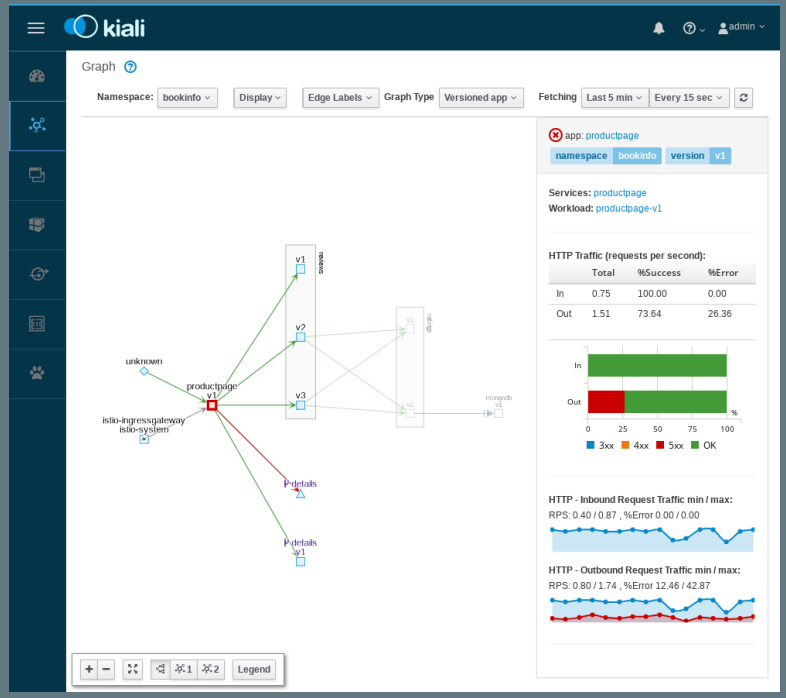
2. Find port assigned to Jaeger:

```
kubectl -n istio-system get svc tracing
```

Lab 4

Services Observation

Kiali



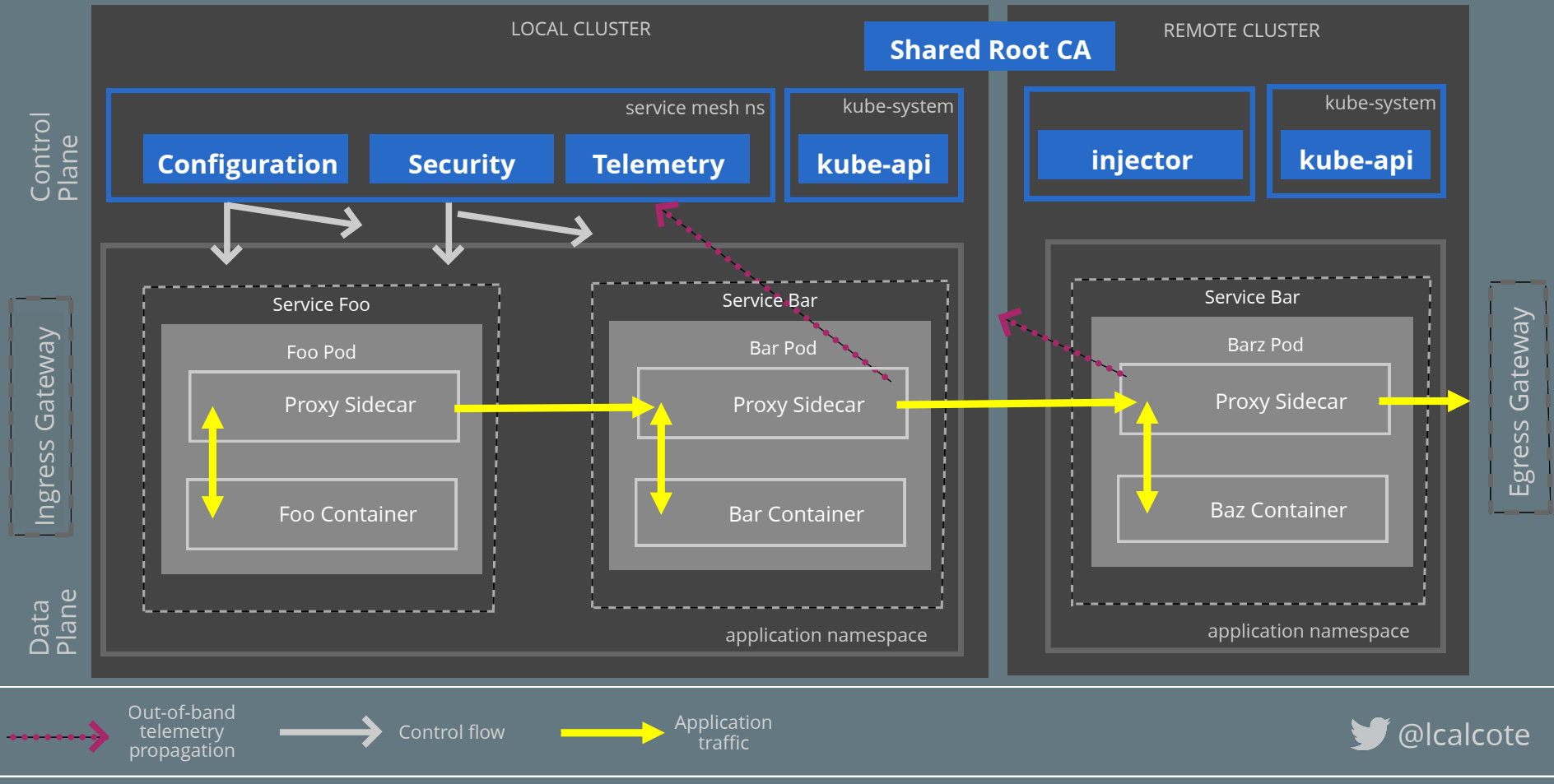
1. Expose services with NodePort:

```
kubectl -n istio-system edit svc kiali
```

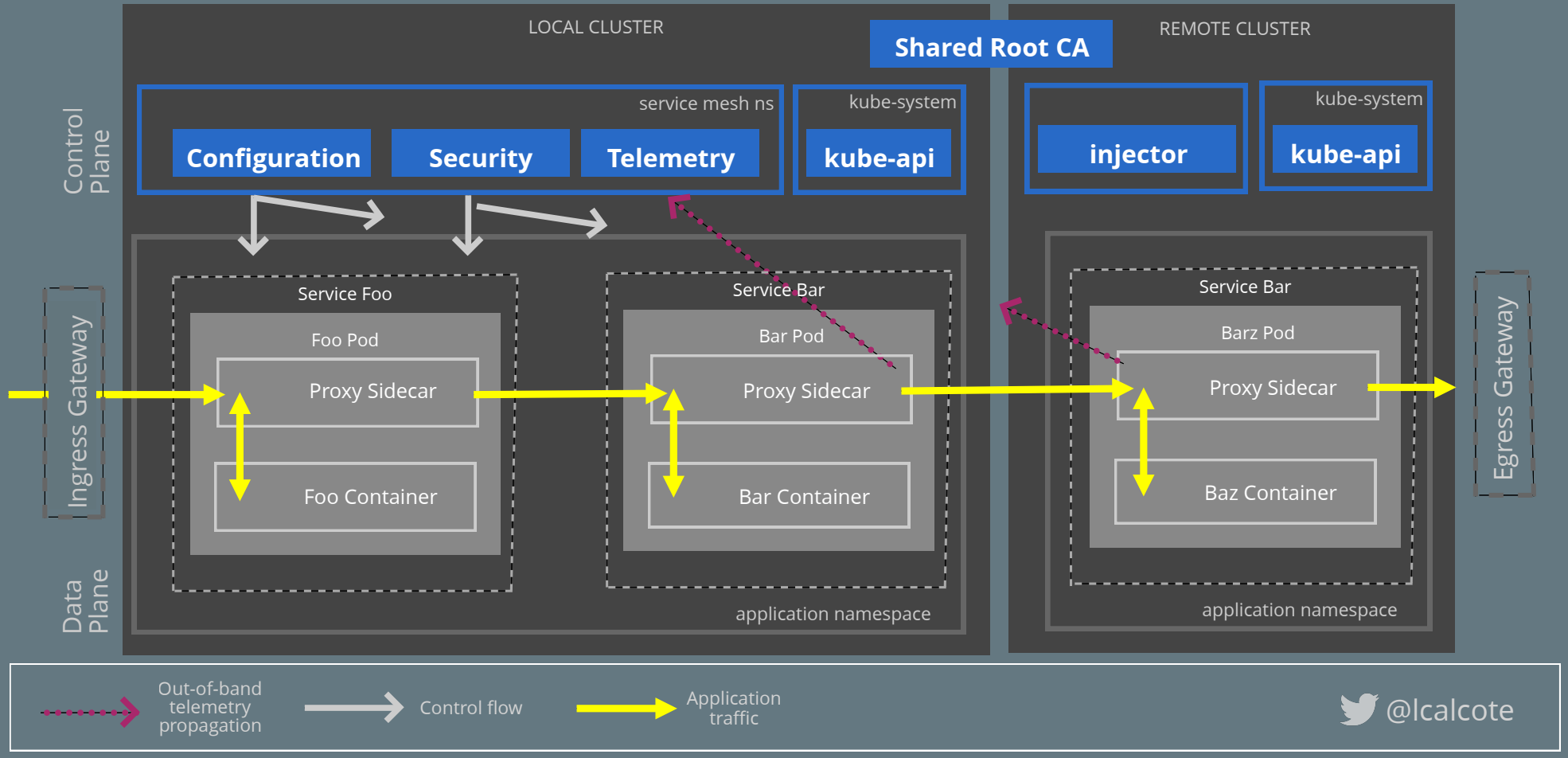
2. Find port assigned to Kiali:

```
kubectl -n istio-system get svc kiali
```

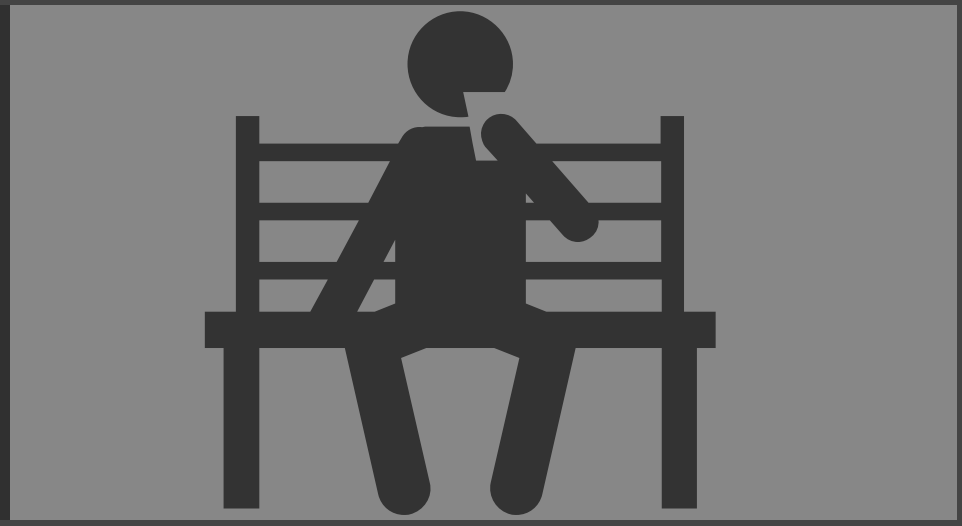
Istio v1.0 Multi-Cluster



Istio v1.0 Multi-Cluster



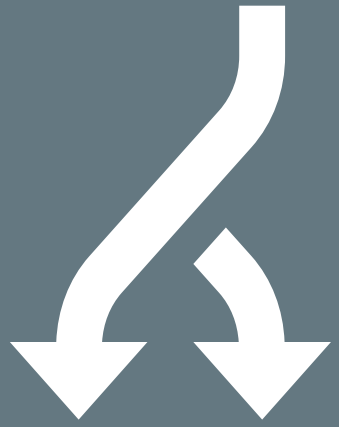
Q&A



Break

Lab 5

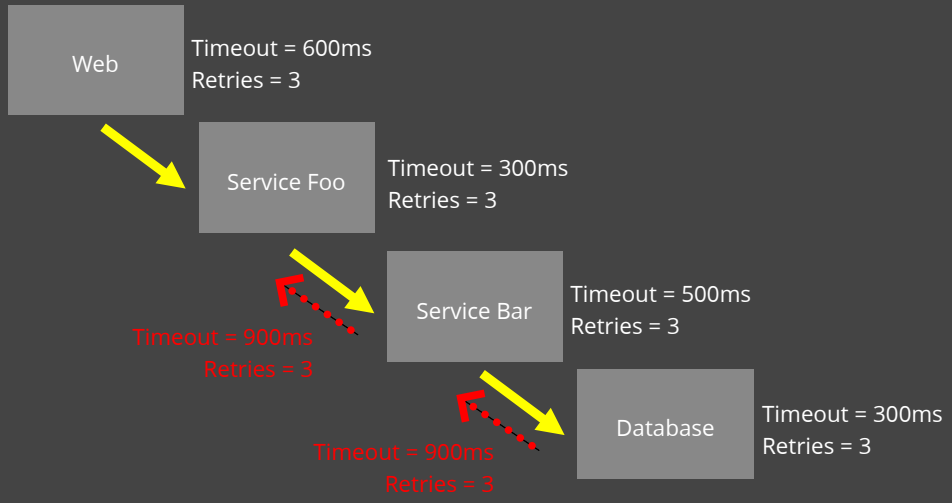
Request Routing and Canary Testing



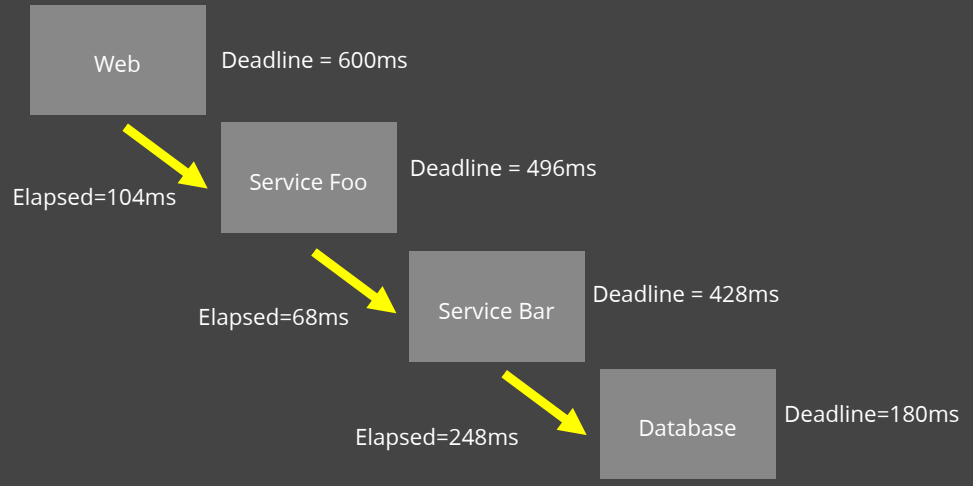
1. Configure the default route for all services to V1
2. Content based routing
3. Canary Testing - Traffic Shifting

```
kubectl apply -f samples/bookinfo/networking/destination-rule-all-mtls.yaml
```

Timeouts & Retries



Deadlines



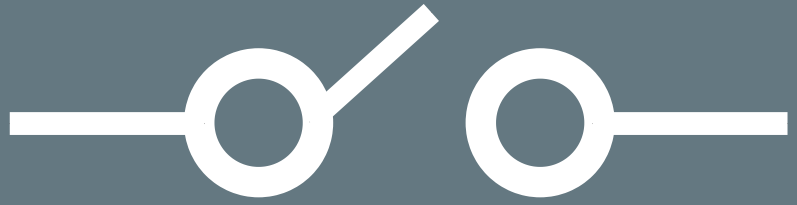
Lab 6

Fault Injection and Rate-Limiting

1. Inject a route rule to create a fault using HTTP delay
2. Inject a route rule to create a fault using HTTP abort
3. Verify fault injection

Lab 7

Circuit Breaking



1. Deploy a client for the app
    1. With manual sidecar injection:
2. Initial test calls from client to server
3. Time to trip the circuit

Q&A

Lab 8

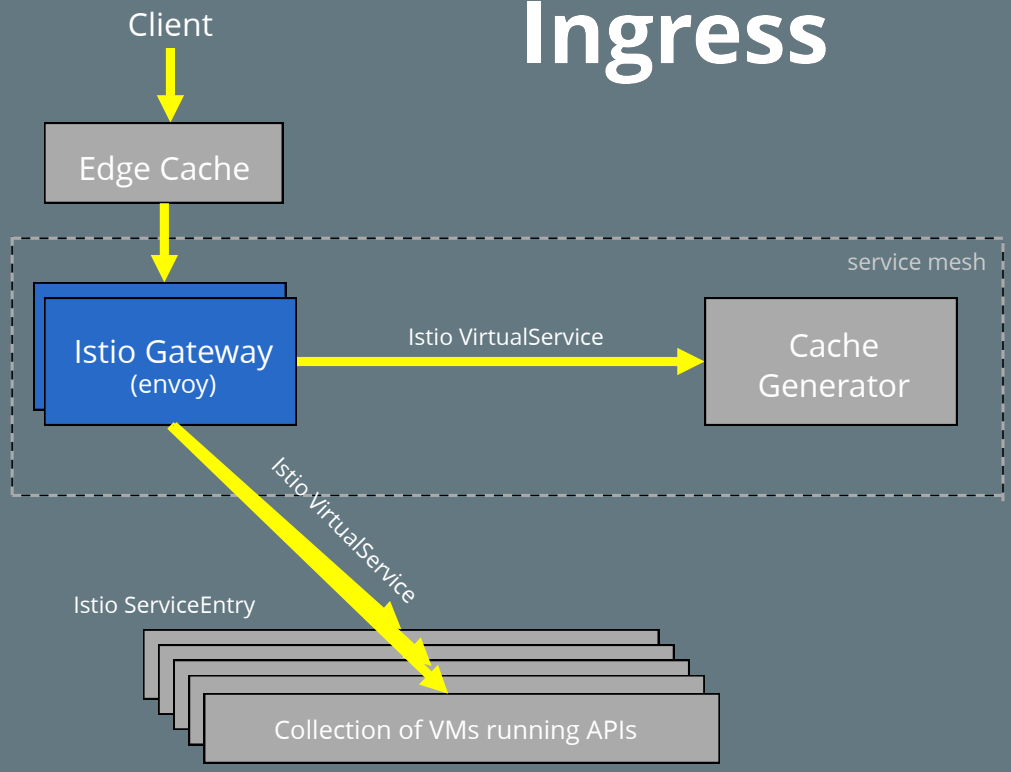
Mutual TLS & Identity Verification

1. Verify mTLS
2. Understanding SPIFFE

If we have time...

Service Mesh Deployment Models

Ingress



Situation:

- existing services running on VMs (that have little to no service-to-service traffic).
- nearly all traffic flows from client to the service and back to client.

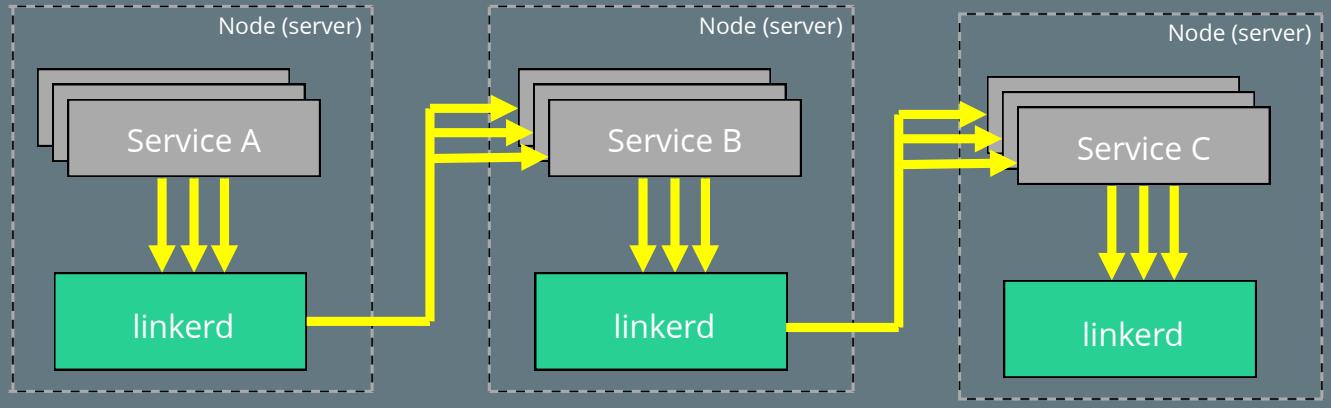
Benefits:

- gain granular traffic control (e.g path rewrites).
- detailed service monitoring without immediately deploying a thousand sidecars.

Istio - The Weather Company's Journey

@lcalcote

Proxy per Node

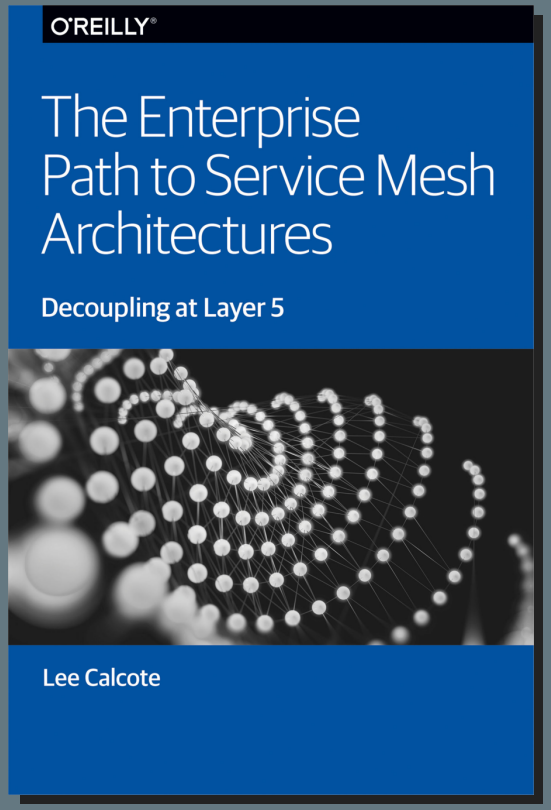


Advantages:

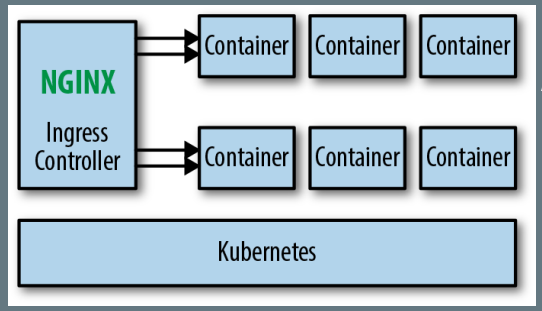
- Less (memory) overhead.
- Simpler distribution of configuration information.
- primarily physical or virtual server based; good for large monolithic applications.

Disadvantages:

- Coarse support for encryption of service-to-service communication, instead host-to-host encryption and authentication policies.
- Blast radius of a proxy failure includes all applications on the node, which is essentially equivalent to losing the node itself.
- Not a transparent entity, services must be aware of its existence.



layer5.io/books



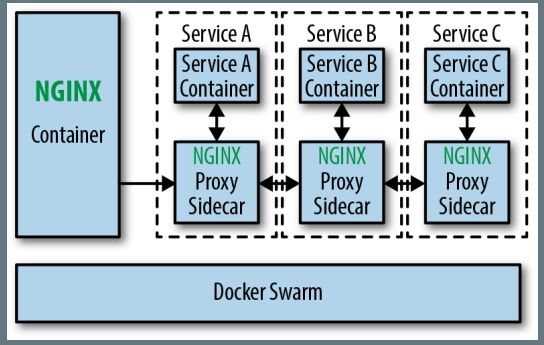
Ingress or Edge Proxy

Advantages:

- Works with existing services that can be broken down over time.

Disadvantages:

- Is missing the benefits of service-to-service visibility and control.



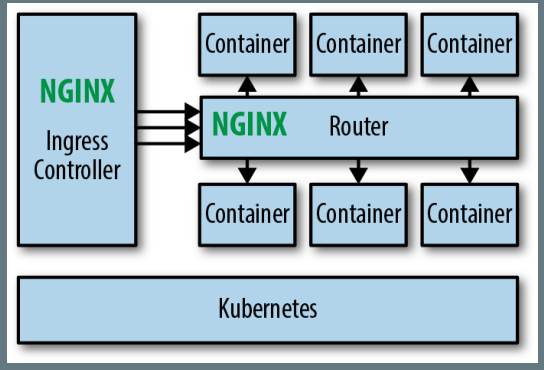
Fabric Model

Advantages:

- Granular encryption of service-to-service communication.
- Can be gradually added to an existing cluster without central coordination.

Disadvantages:

- Lack of central coordination. Difficult to scale operationally.



Router "Mesh"

Advantages:

- Good starting point for building a brand-new microservices architecture or for migrating from a monolith.

Disadvantages:

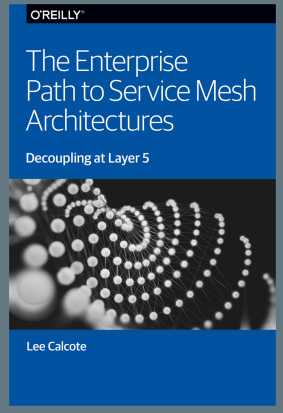
- When the number of services increase, it becomes difficult to manage.

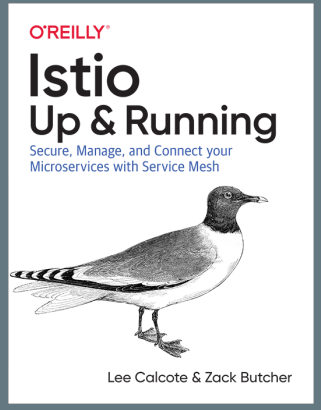
Q&A

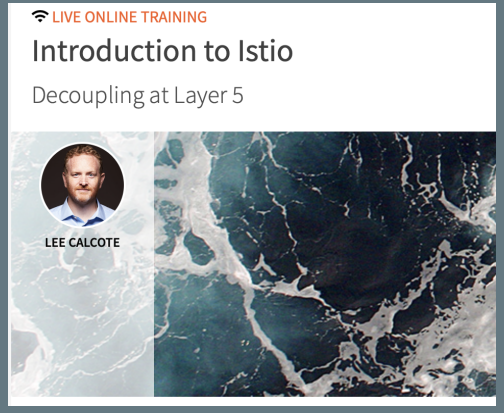
Layer5.io

the Service Mesh Community

SERVICE MESH LANDSCAPE







Meshery

layer5.io/subscribe

@layer5