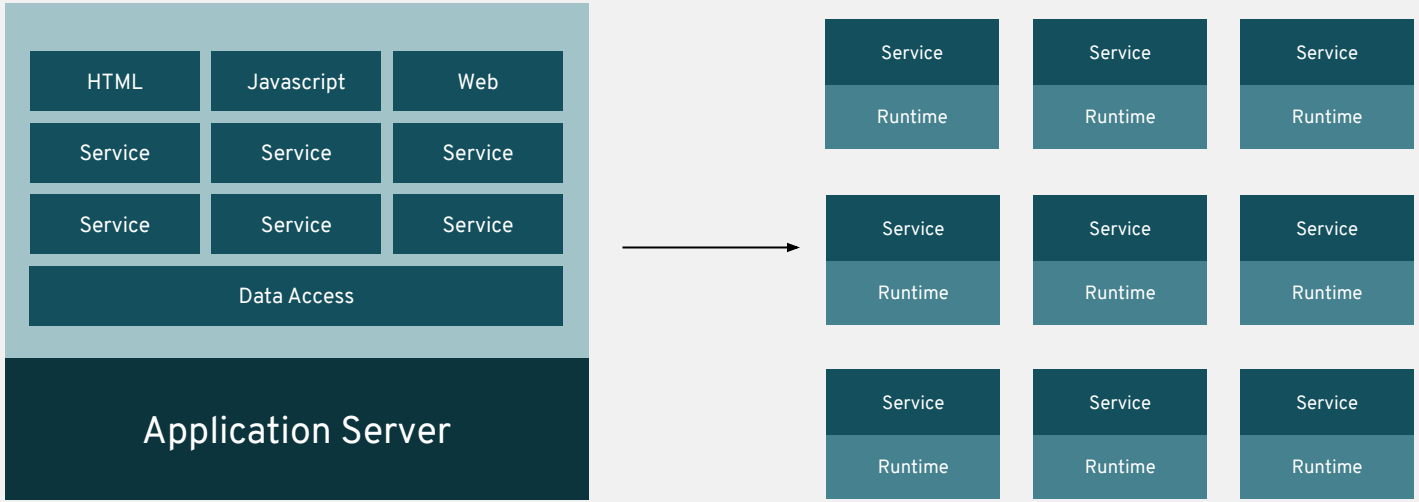
Developing Microservices with Istio Service Mesh

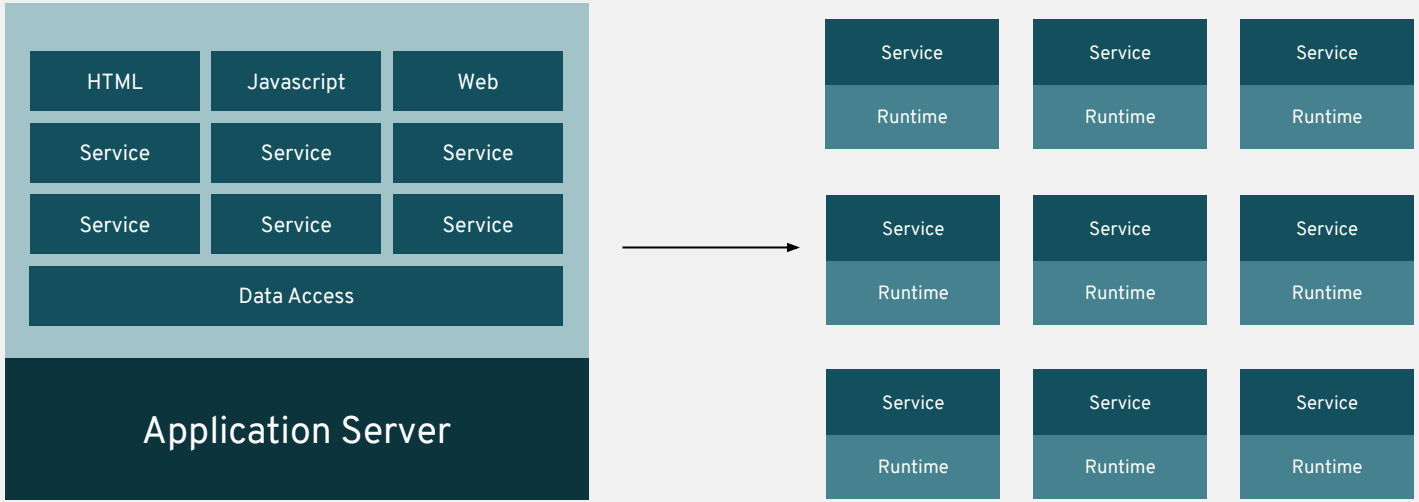
Kevin Conner
Engineering Manager
Istio Engineering
Red Hat

MICROSERVICES ARCHITECTURE

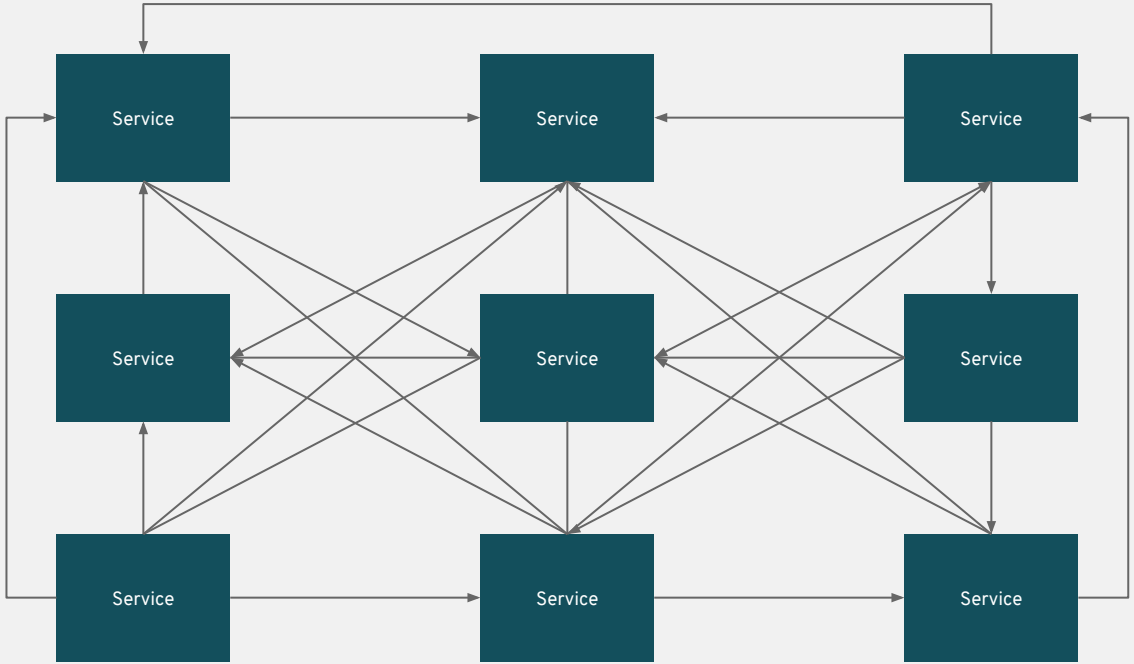


~~MICROSERVICES ARCHITECTURE~~

DISTRIBUTED



DISTRIBUTED ARCHITECTURE



EIGHT FALLACIES OF DISTRIBUTED COMPUTING

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

Source: https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing
Photo: Icon made by Freepik from www.flaticon.com

Difficulty of deploying microservices

- Service discovery
- Retries
- Timeouts
- Circuit breaking
- Rate limiting
- Load balancing

Difficulty of deploying on cloud infrastructure

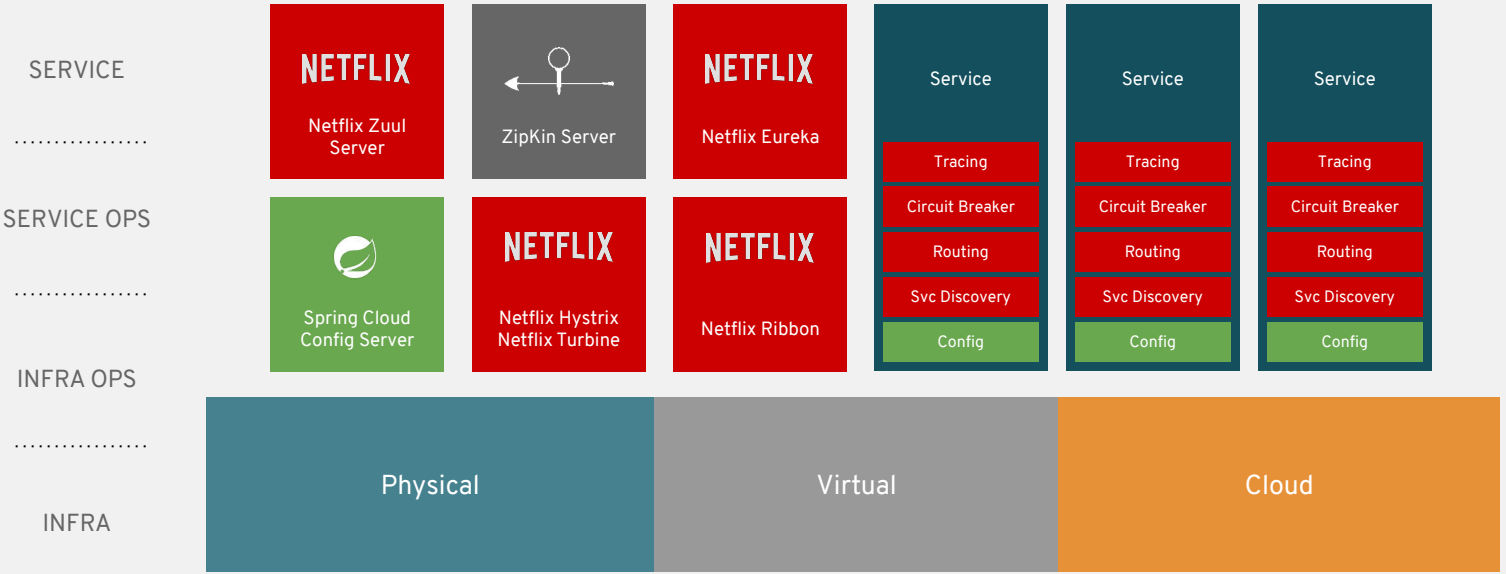
- Edge / DMZ routing (ingress/egress)
- Surgical / fine / per-request routing
- A/B testing
- Traffic shaping
- Internal release / dark launch
- Request shadowing
- Fault injection

Difficulty of deploying on cloud infrastructure

- Adaptive, zone-aware load balancing
- Health checking (active/passive)
- Stats/metrics/telemetry
- Logging
- Distributed tracing

Security!!

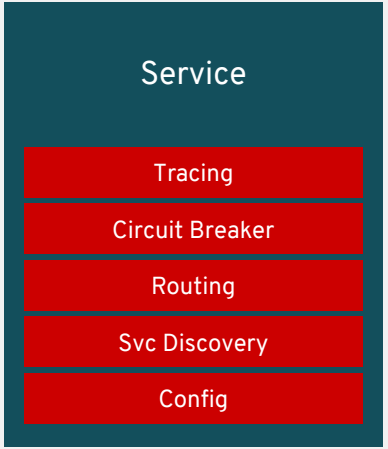
WHAT DO DISTRIBUTED SYSTEMS NEED?



AN OPERATION NIGHTMARE!

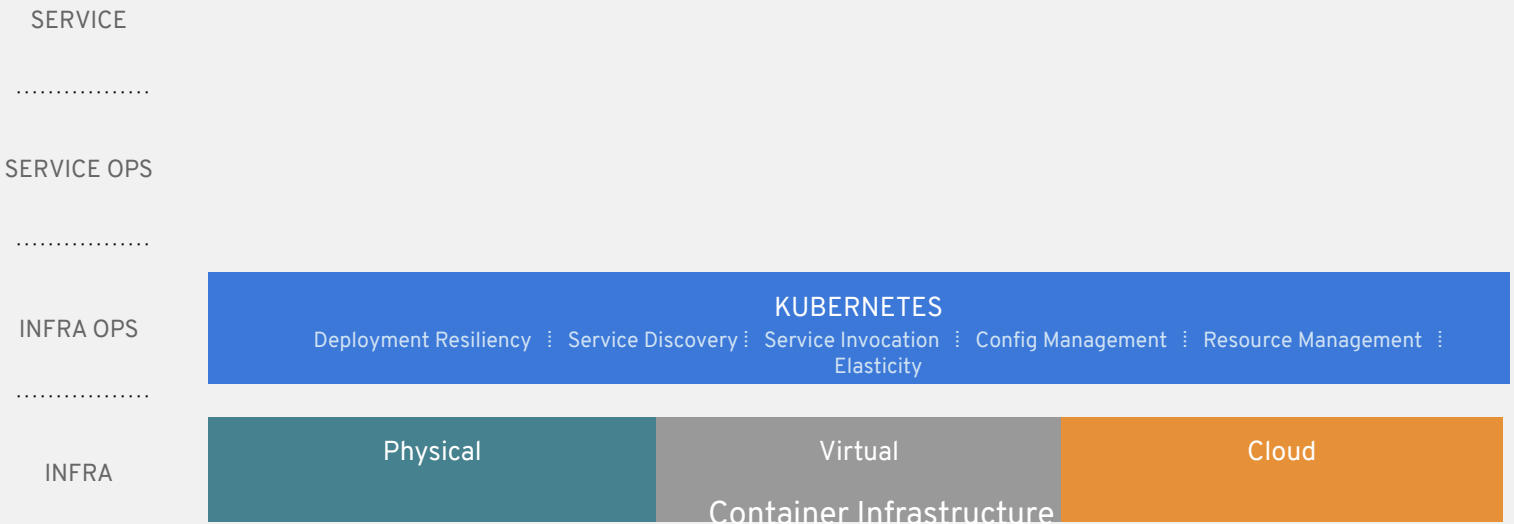
Infra capabilities are tightly coupled with applications and services

- ❌ Incompatible across languages and frameworks
- ❌ Existing apps require refactoring
- ❌ Upgrades needs tight coordination - libraries

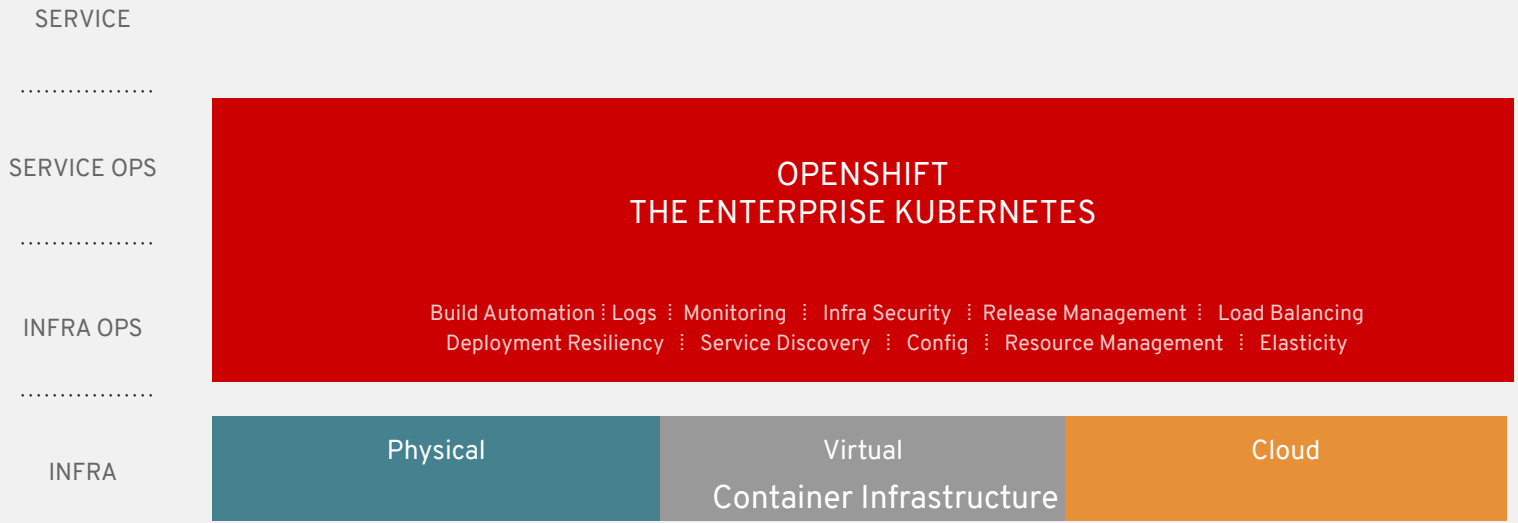


THERE SHOULD BE A BETTER WAY

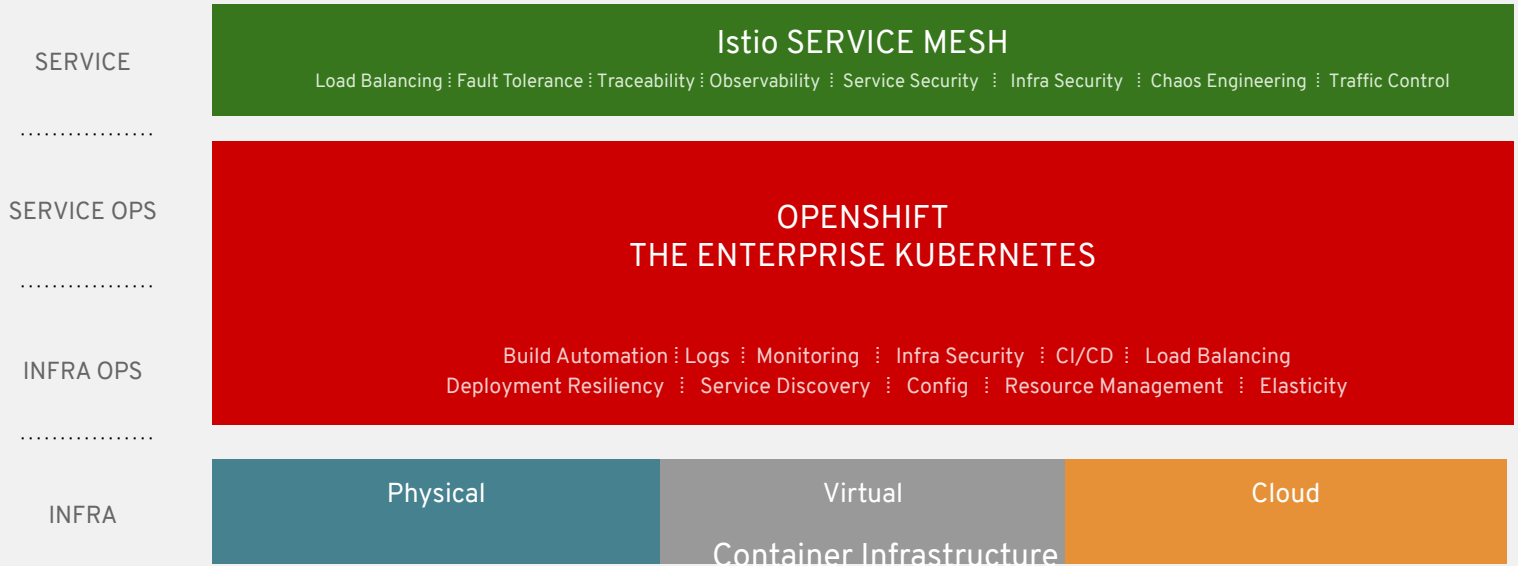
DISTRIBUTED SERVICES WITH KUBERNETES



DISTRIBUTED SERVICES WITH OPENSHIFT



DISTRIBUTED SERVICES WITH ISTIO SERVICE MESH

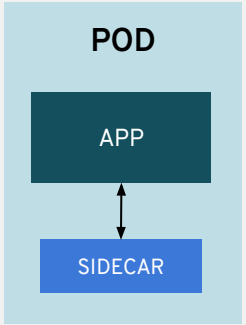


USE A SERVICE PROXY

- Envoy Proxy (used in Istio as Istio proxy)
- L3/4 network filter, out of the box L7 filters (HTTP, HTTP2, gRPC)
- Service discovery, load balancing, circuit breaking, metrics collection, timeouts, retries, rate limiting, distributed tracing, et al
- Written in C++
- Dynamic configuration

SIDECAR PATTERN

- A utility container in the same pod to enhance the main container's functionality
- Share the same network and lifecycle
- Istio uses an Istio Proxy sidecar to proxy all network traffic between apps

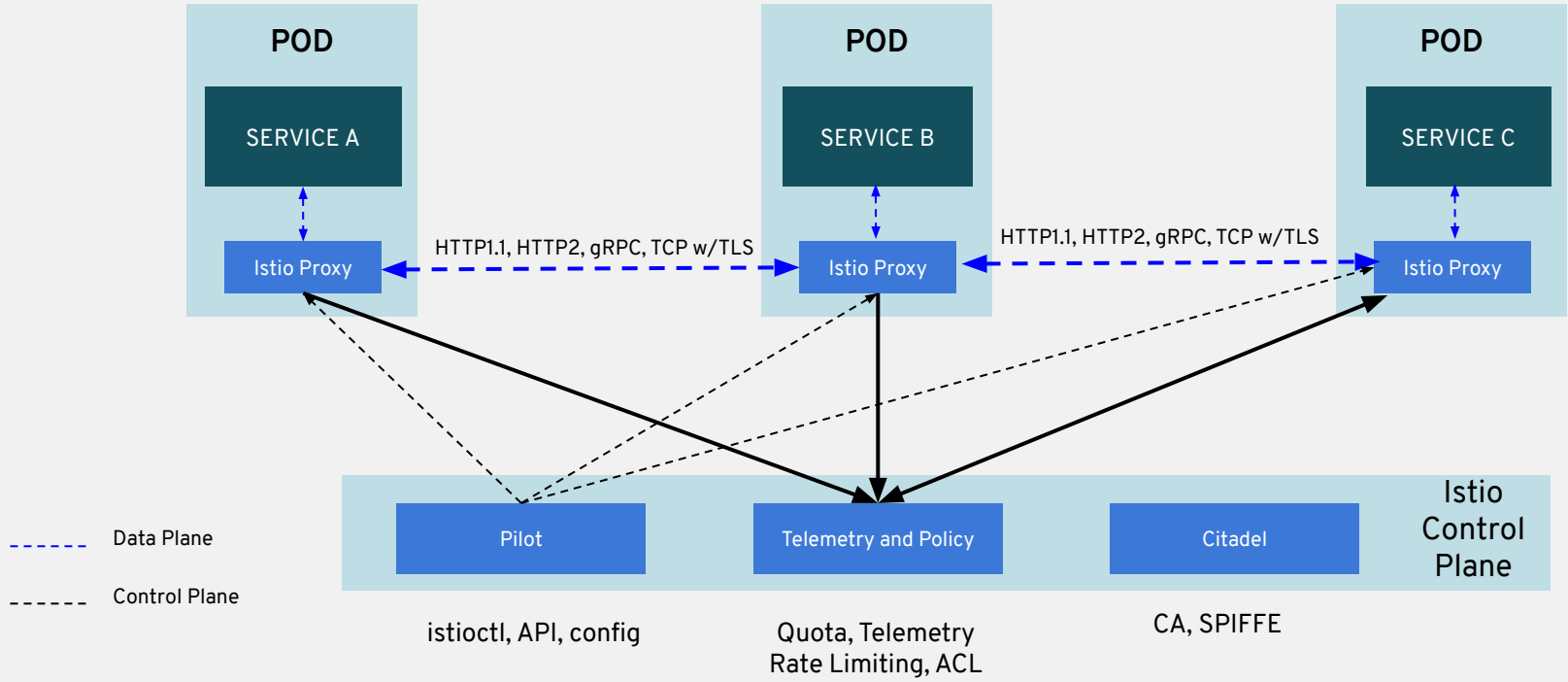


Source: http://blog.kubernetes.io/2015/06/the-distributed-system-toolkit-patterns.html

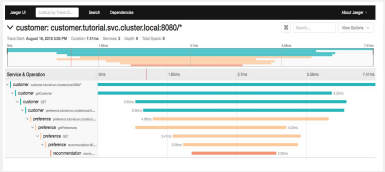
WHAT DOES ISTIO GIVE ME?

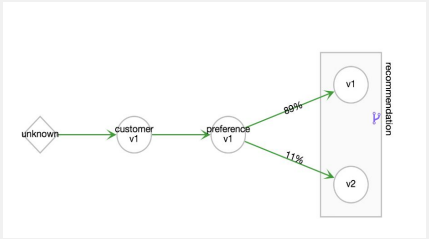
- Service resiliency
- Observability
- Traffic control
- Security
- Policy Enforcement
- Chaos Testing

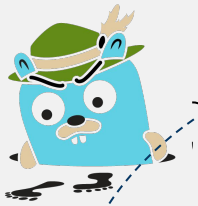
Istio Service Mesh



SERVICE MESH ECOSYSTEM







Jaeger

Secure

Observe

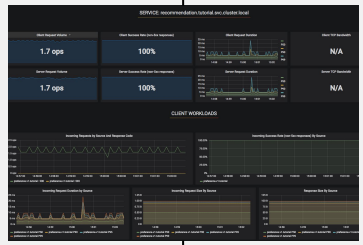
Connect

Control

Kiali

Prometheus

Observe

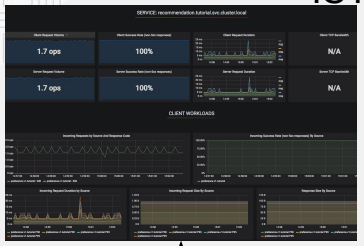


Grafana

Istio In Action

OBSERVABILITY

ISTIO MONITORING



GRAFANA

PROMETHEUS

Prometheus Adapter

POD

SERVICE A

Istio Proxy

POD

SERVICE A

Istio Proxy

POD

SERVICE A

Istio Proxy

Check & Report

Backend

Logging | Telemetry | Auth | Quota

Adapter API

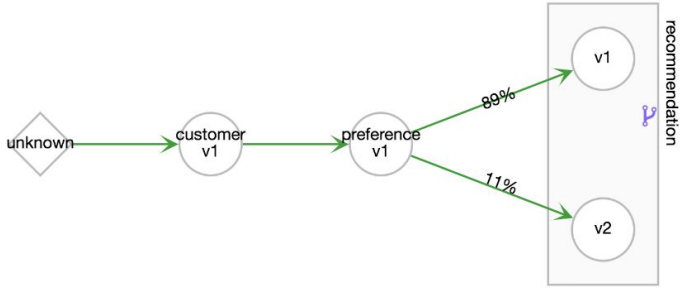
Mixer

Pilot

Citadel

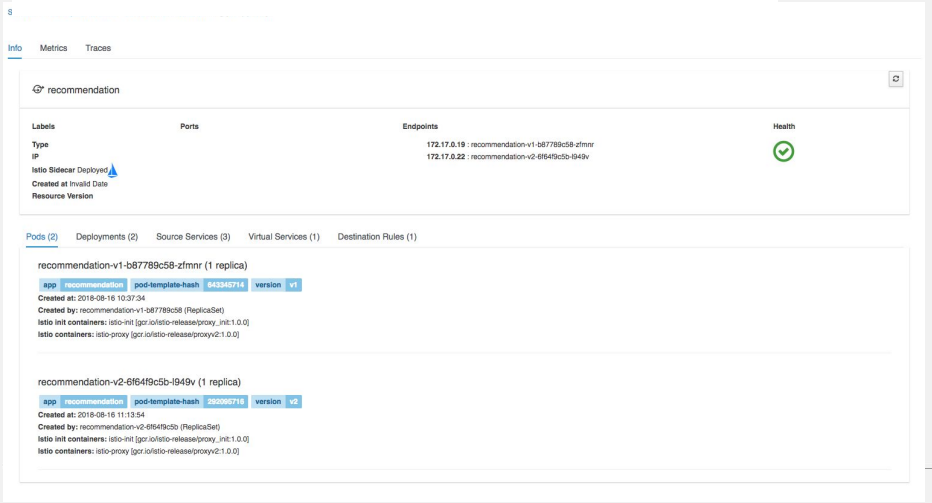
Control Plane

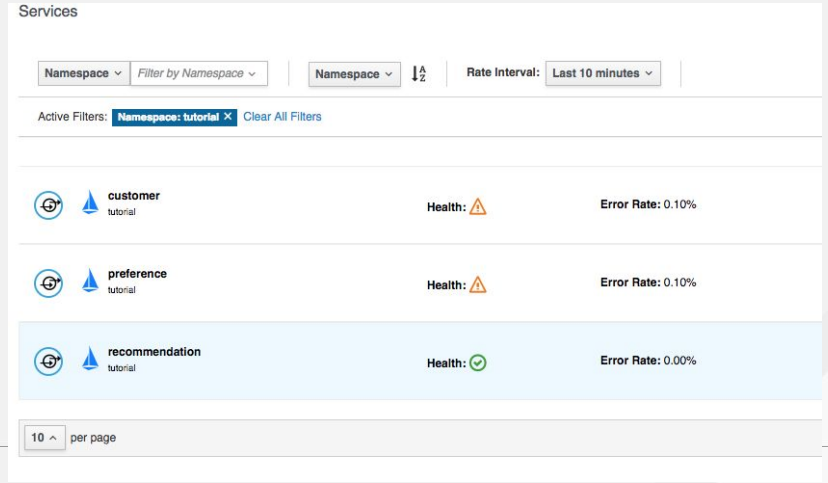
KIALI - SERVICE MESH OBSERVABILITY



FEATURES:

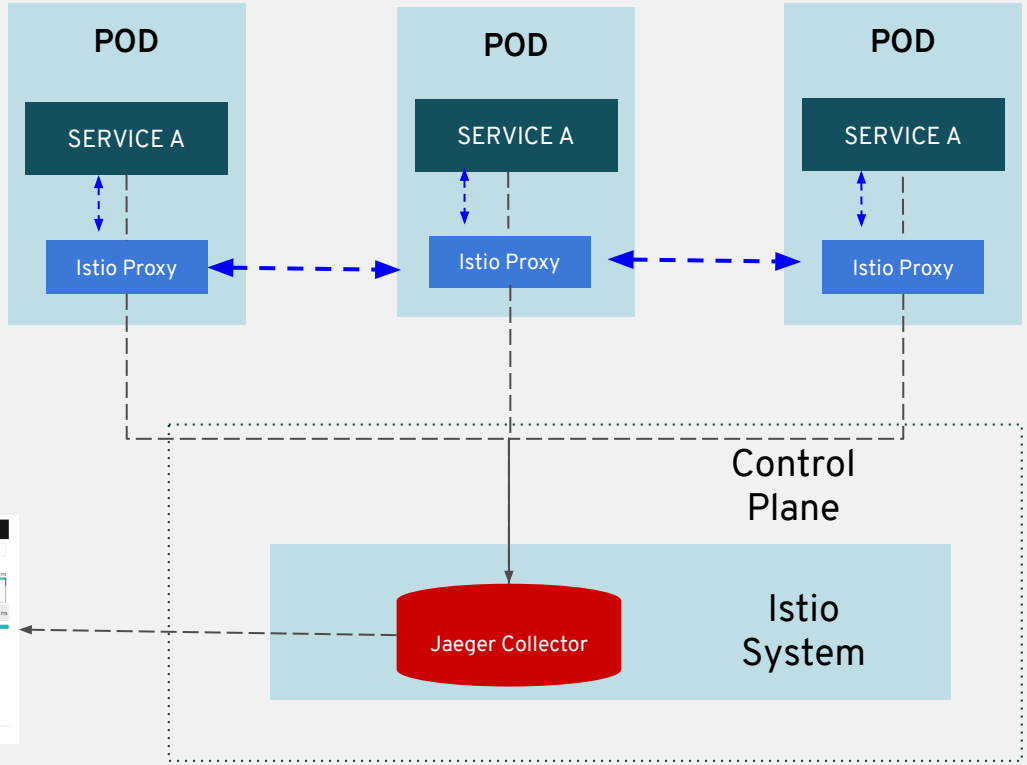
- Service graph representation
- Distributed tracing (via Jaeger)
- Metrics collection and graphs (from Prometheus)
- Configuration validation
- Health computation/display
- Service discovery

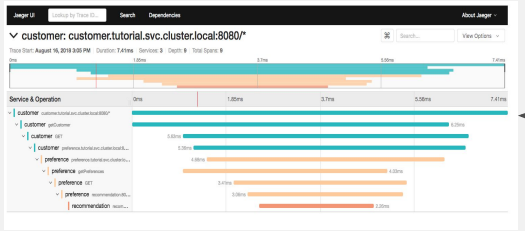




TRACING

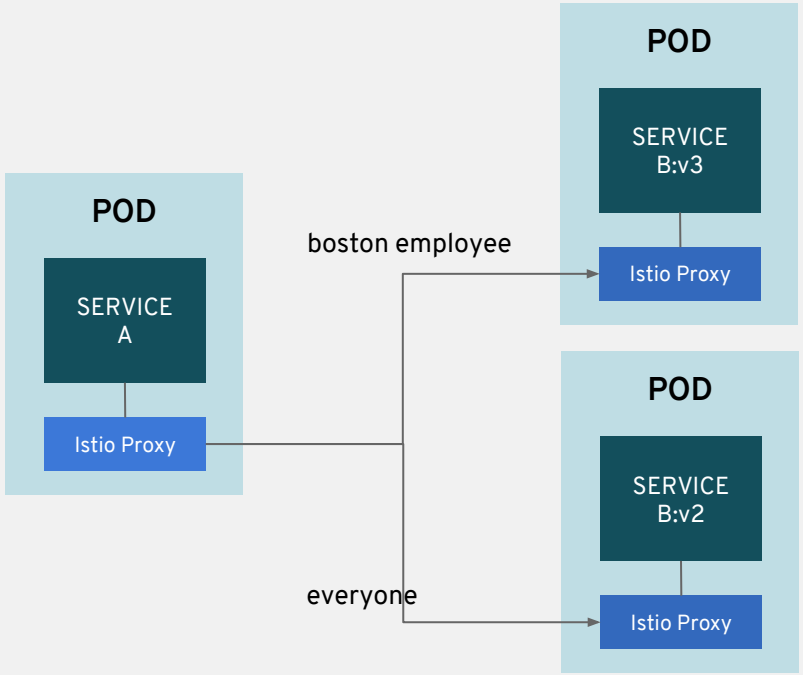
ISTIO TRACING



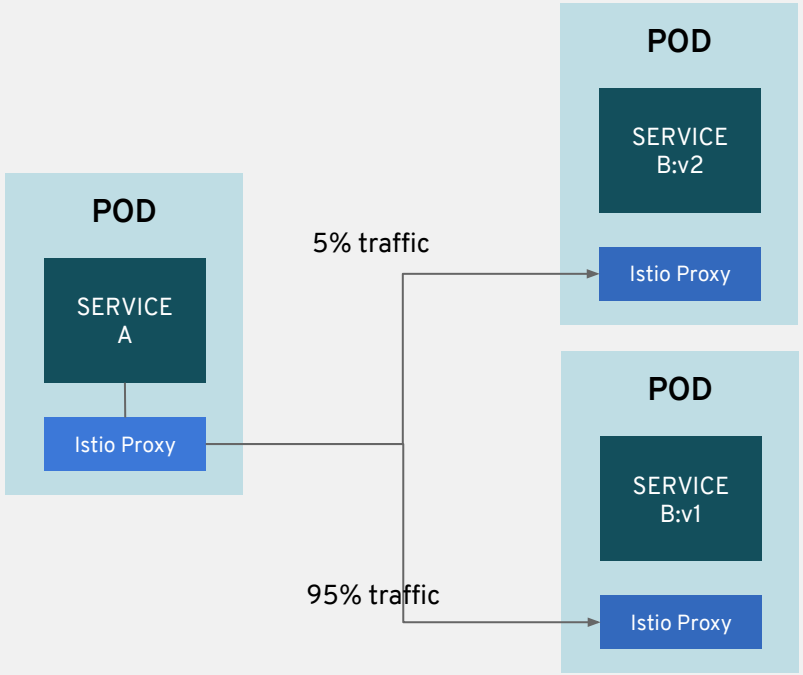


TRAFFIC CONTROL

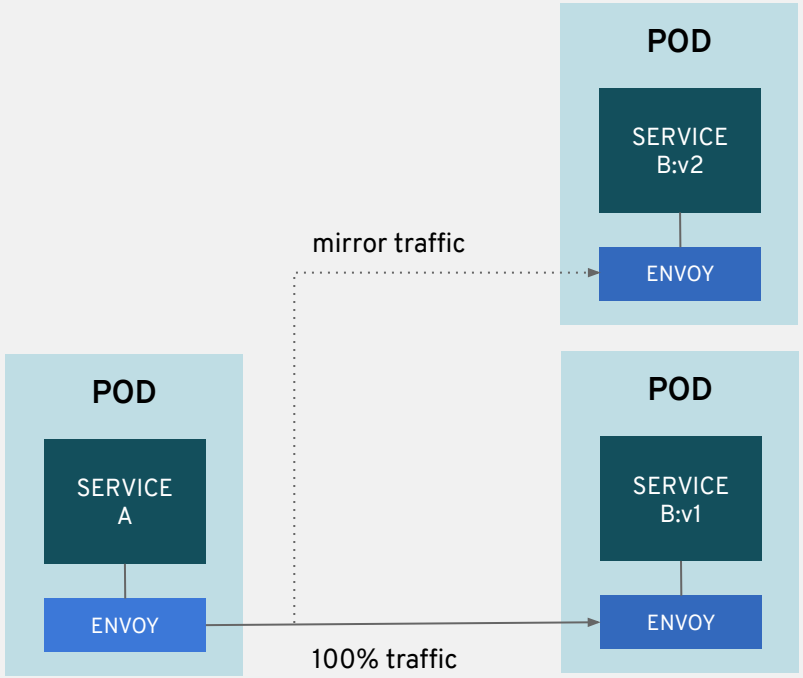
CANARY DEPLOYMENT WITH ISTIO



WEIGHTED ROUTING WITH ISTIO

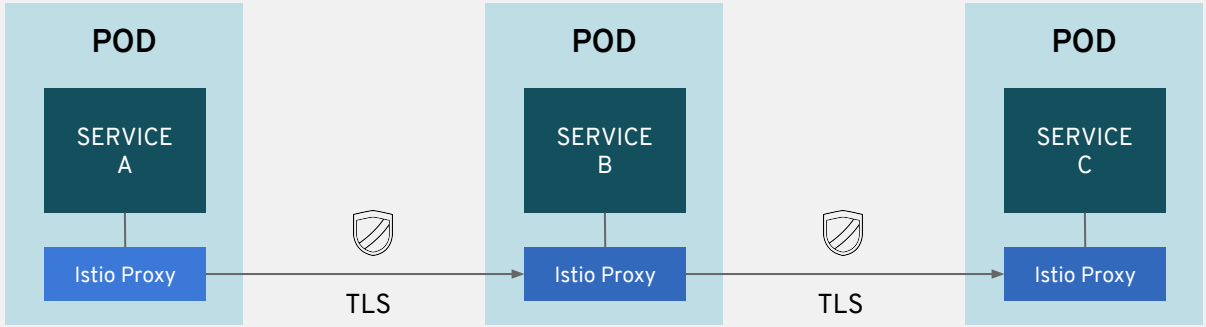


DARK LAUNCHES WITH ISTIO



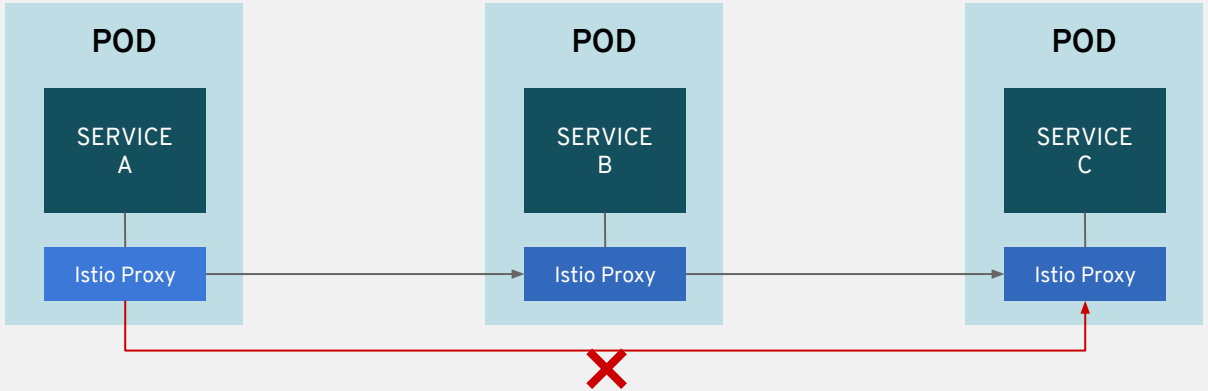
SERVICE SECURITY

SECURE COMMUNICATION WITH ISTIO



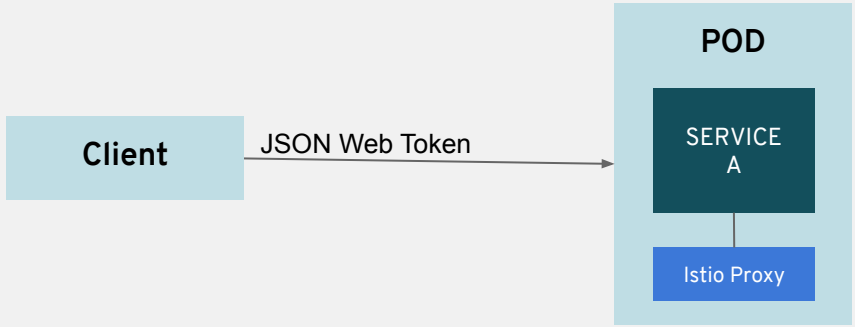
mutual TLS authentication, transparent to the services

CONTROL SERVICE ACCESS WITH MUTUAL TLS



control the service access flow, transparent to the services

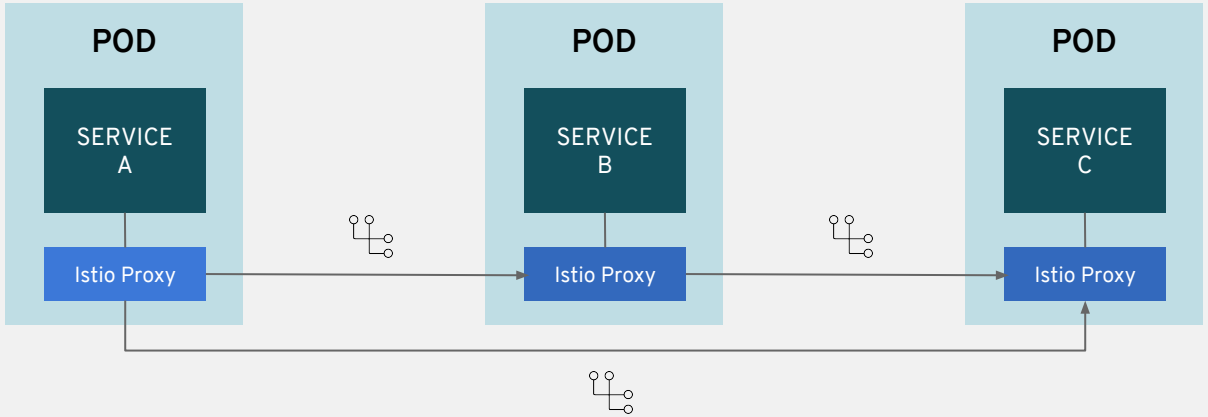
CONTROL SERVICE ACCESS WITH END USER AUTHENTICATION



control the service access flow, transparent to the services

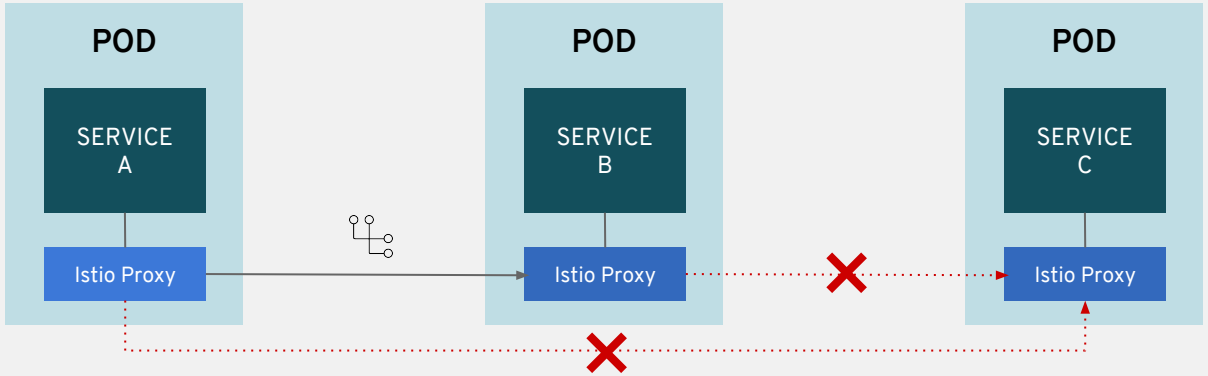
FAULT TOLERANCE

CIRCUIT BREAKERS WITH ISTIO



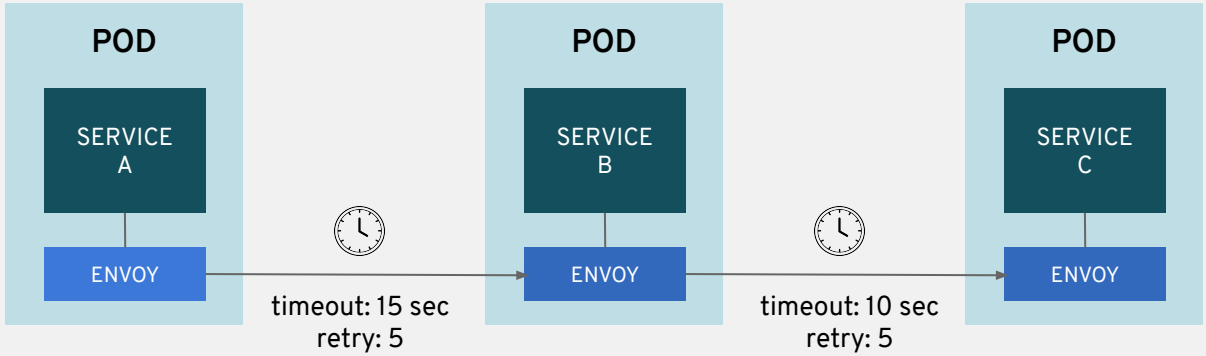
transparent to the services

CIRCUIT BREAKERS WITH ISTIO



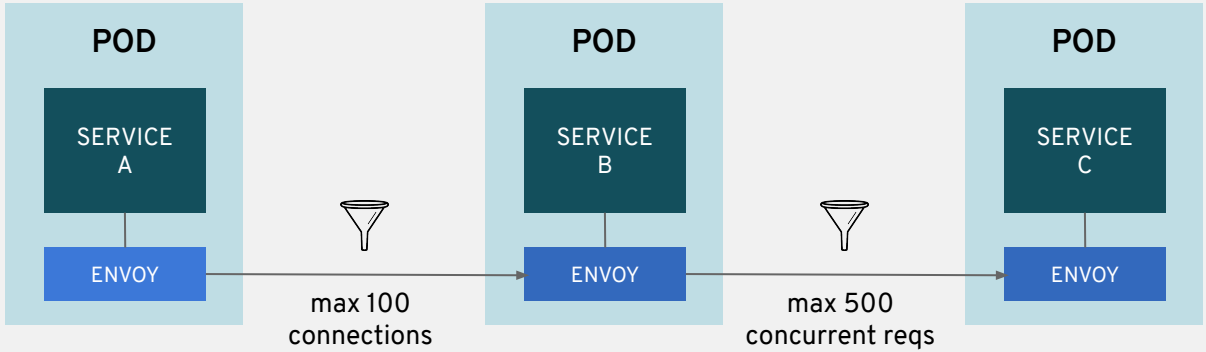
transparent to the services

TIMEOUTS AND RETRIES WITH ISTIO



configure timeouts and retries, transparent to the services

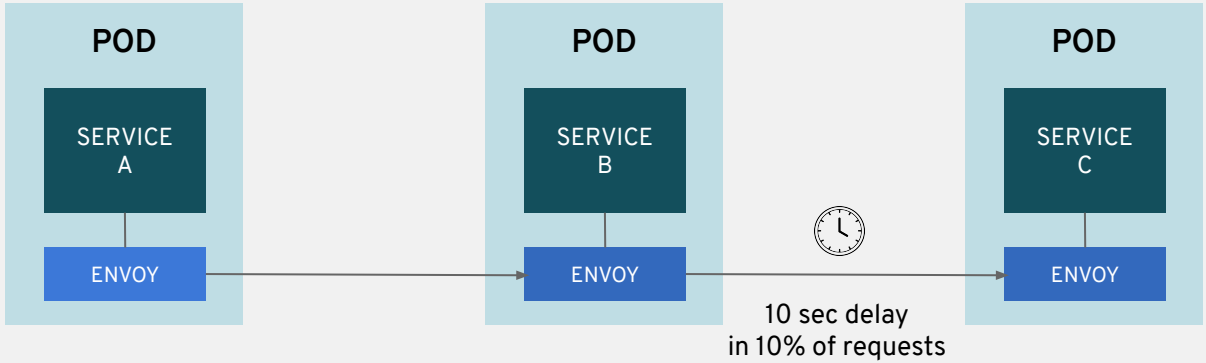
RATE LIMITING WITH ISTIO



limit invocation rates, transparent to the services

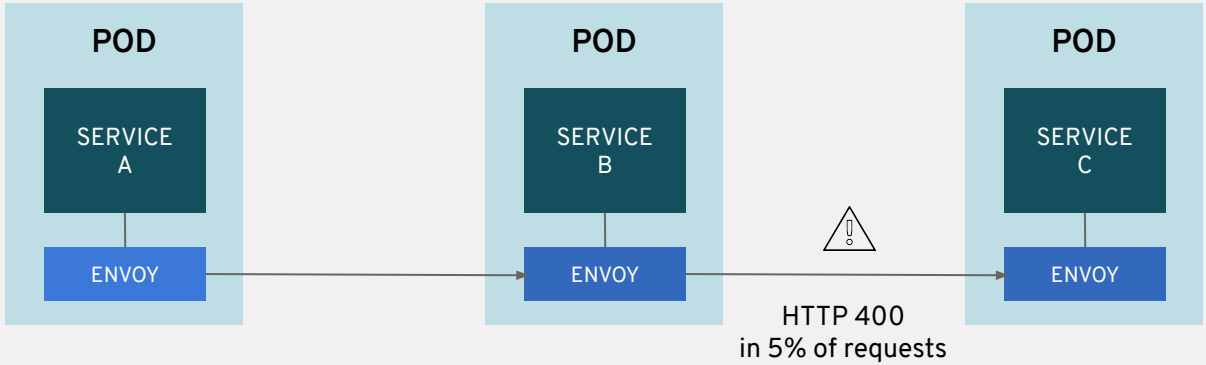
CHAOS ENGINEERING

CHAOS ENGINEERING WITH ISTIO



inject delays, transparent to the services

CHAOS ENGINEERING WITH ISTIO

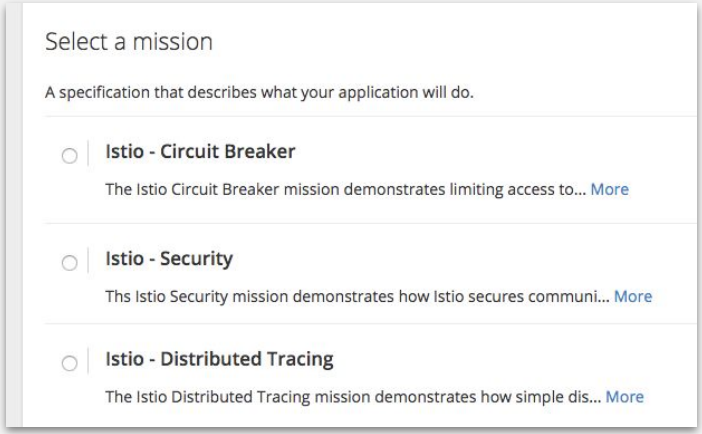


inject protocol-specific errors, transparent to the services

RED HAT OPENSHIFT SERVICE MESH

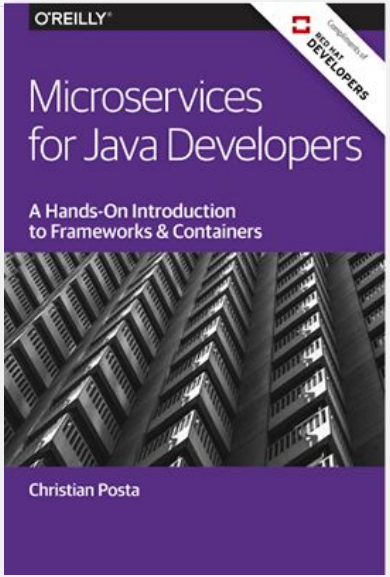
OPENSHIFT SERVICE MESH

- Supported distribution of
  - Istio
  - Jaeger
  - Kiali
  - Prometheus
  - Grafana
- Upstream project called Maistra https://github.com/Maistra
- Integrated with Red Hat OpenShift Application Runtimes (RHOAR)
- Integrated with Red Hat 3scale API Management

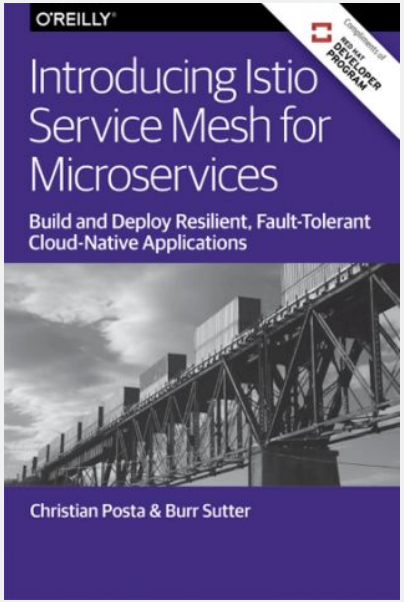


RESOURCES

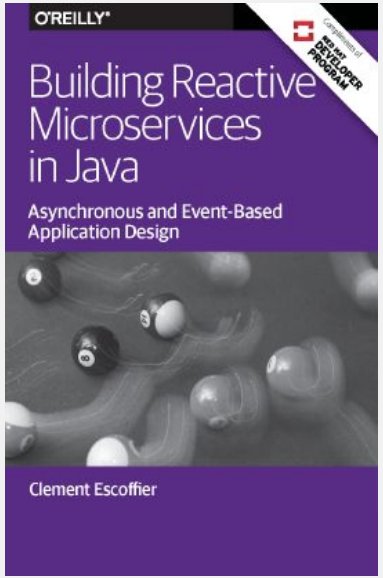
bit.ly/javamicroservicesbook



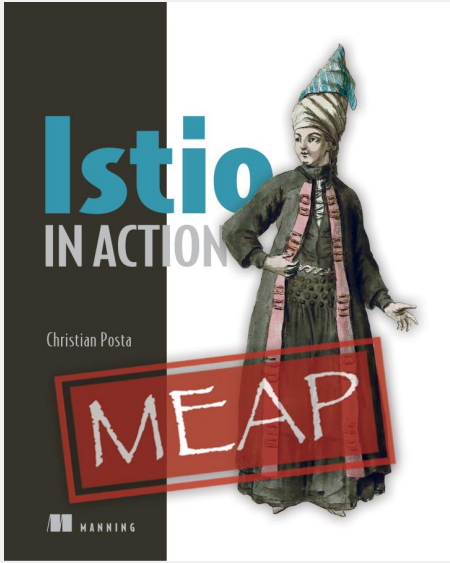
bit.ly/istio-book



bit.ly/reactivemicroservicesbook



https://www.manning.com/books/istio-in-action



Demo: bit.ly/msa-instructions
Slides: bit.ly/microservicesdeepdive
Video Training: bit.ly/microservicesvideo

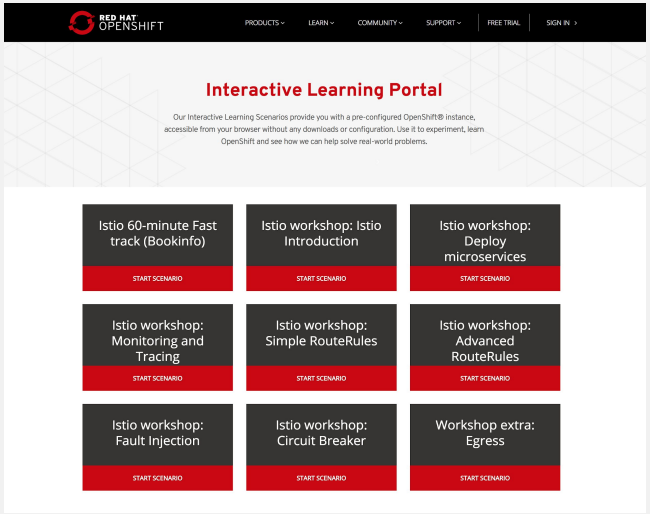
bit.ly/mono2microdb

Kubernetes for Java Developers

WORKSHOPS

Self-paced Labs

learn.openshift.com/servicemesh



GitHub

bit.ly/istio-tutorial

https://github.com/thoraxe/istio-lab-summit-2019

Questions?