Service Mesh: hype or reality?

Tech4People session at the Red Hat Tech Day 2020

January 24th, 2020
Author: Filip Lenaerts

devoteam

Innovative technology consulting for business

Agenda

devoteam

1

Intro

Intro

devoteam

15

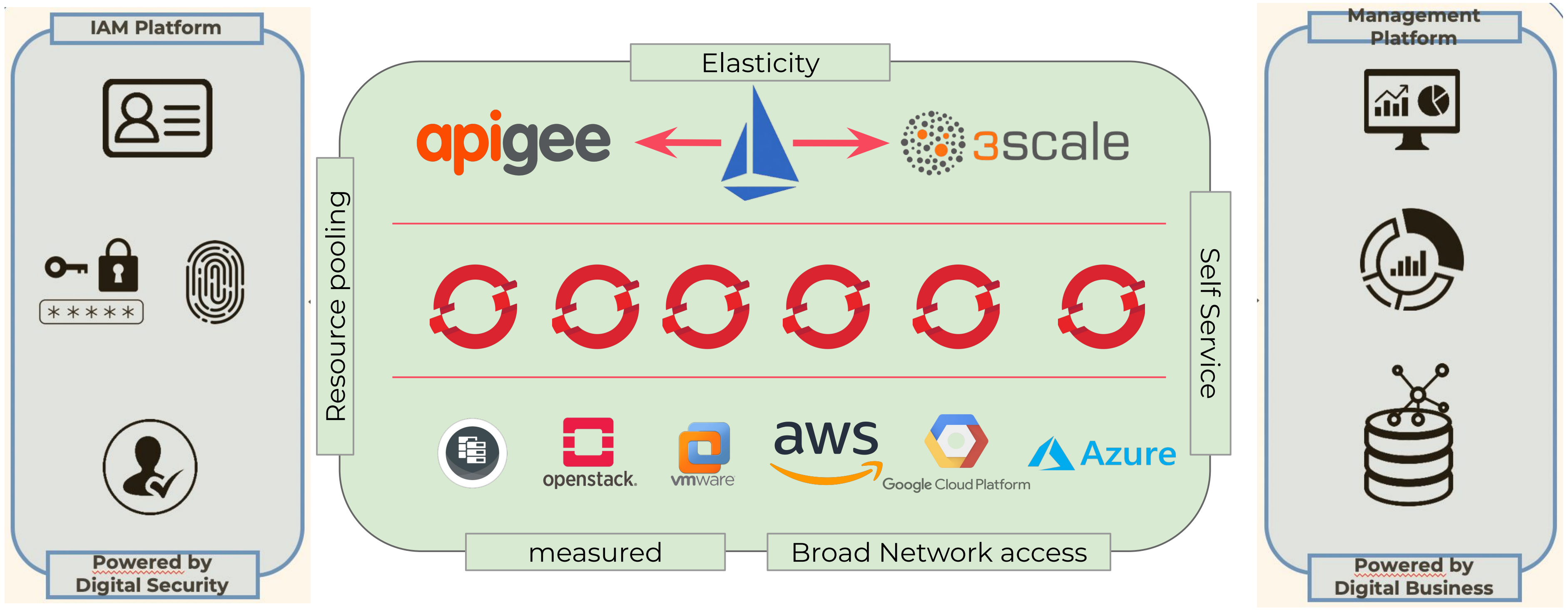
2016

open source

47

redhat

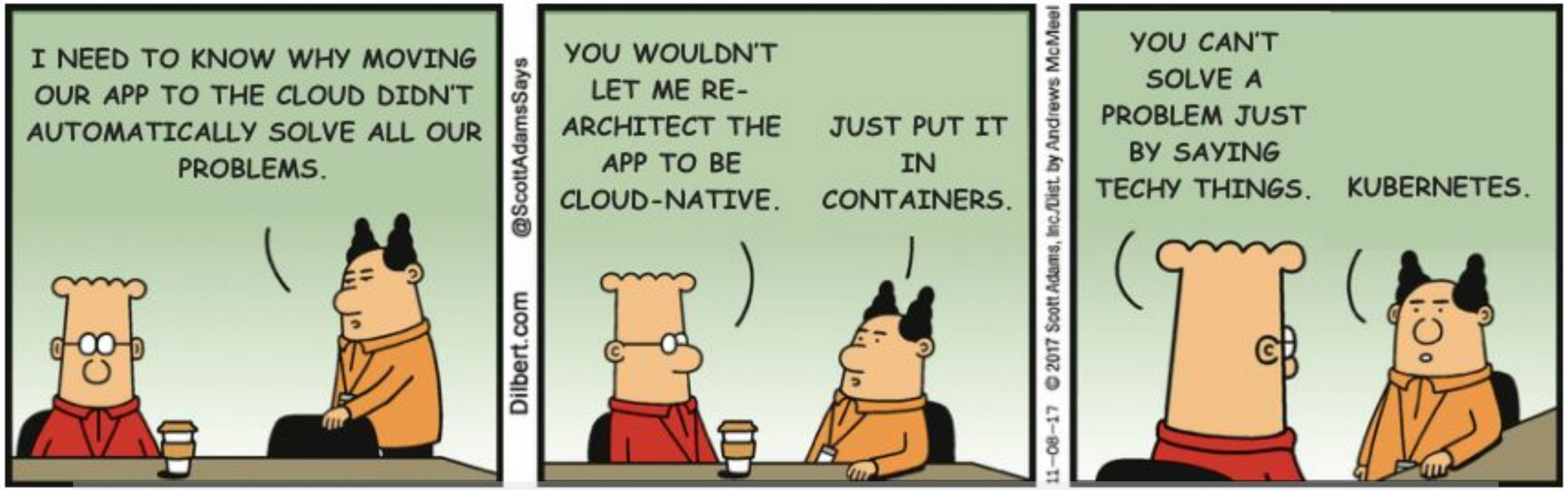
Devops platform



devoteam

Source:
https://nl.m.wikipedia.org/wiki/Bestand:Warning_icon.svg

Source: BINGO generator


I NEED TO KNOW WHY MOVING OUR APP TO THE CLOUD DIDN'T AUTOMATICALLY SOLVE ALL OUR PROBLEMS.
@ScottAdamsSays
Dilbert.com
YOU WOULDN'T LET ME RE-ARCHITECT THE APP TO BE CLOUD-NATIVE.
JUST PUT IT IN CONTAINERS.
11-08-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel
YOU CAN'T SOLVE A PROBLEM JUST BY SAYING TECHY THINGS.
KUBERNETES.

My sincere apologies!

Hype?

Sidecars?

Yet another new paradigm

Easy development?

Hard to get started with

Istio?

Yet another **new greek work** to remember!

New Infrastructure?

My ops already get apesh*t crazy with those containers

Source: http://www.gregerwikstrand.com/cargo-cult-innovation/innovation-bingo/

Cloudnative

Applications

Platform

Cloud Native

CNCF Cloud Native Definition v1.0

Scalable Applications

Cloud native technologies empower organizations to build and run scalable applications in m

Dynamic Environment

s such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

Loosely coupled systems

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combine

Robust automation

y allow engineers to make high-impact changes frequently and predictably with minimal toil.

Resilient

Manageable

Observable

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.

CLOUD NATIVE
COMPUTING FOUNDATION

https://github.com/cncf/toc/blob/master/DEFINITION.md

Cloud native apps

Microservices - one of many implementation options

- Service-oriented architecture
- Each functionality is one services (anti-monolith)
- Communication between services: queueing service

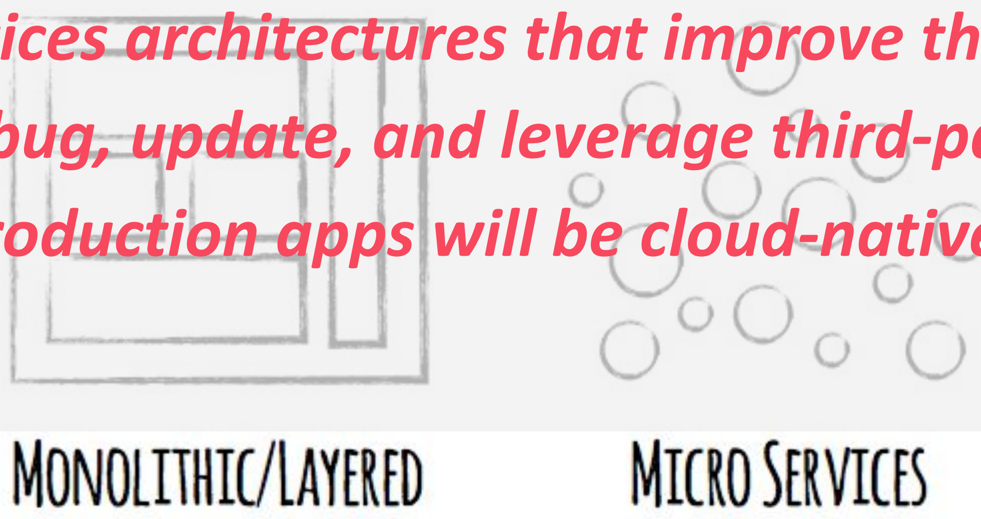
Enables:

- Parallel development
- Short lifecycles
- Fast release cycles
- Module based upgrades (

Requires

- Devops approach
- Versioned APIs

“According to IDC, by 2022, 90% of all new apps will feature microservices architectures that improve the ability to design, debug, update, and leverage third-party code; 35% of all production apps will be cloud-native”

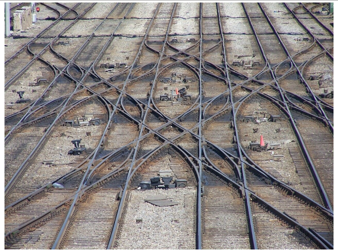


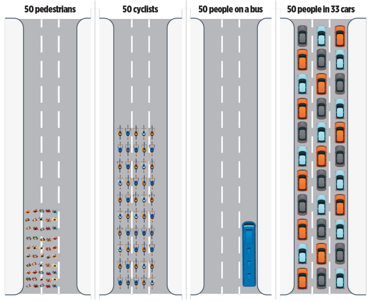
devoteam

Cloud native platform

NIST 800-145

On demand self-service


Broad network access

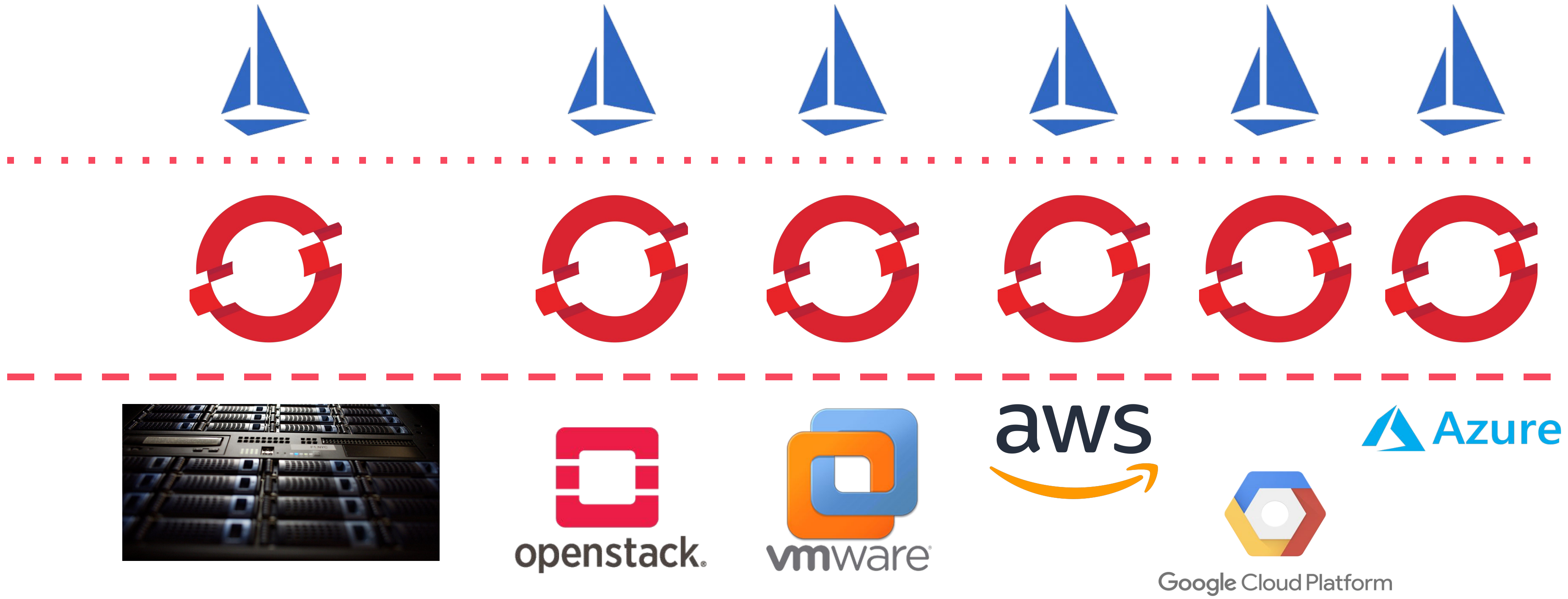


Resource Pooling

Rapid Elasticity

Measured Service

Infrastructure
Common ground



devoteam

Service mesh

Infrastructure layer for microservices communication

Alleviates microservice (code and its developers) from

- Encryption (S2S - zero trust network)
- Authentication
- Authorisation
- Circuit breaker
- Load balancing
- Any non-functionals

Allow focus on the core uService functionality

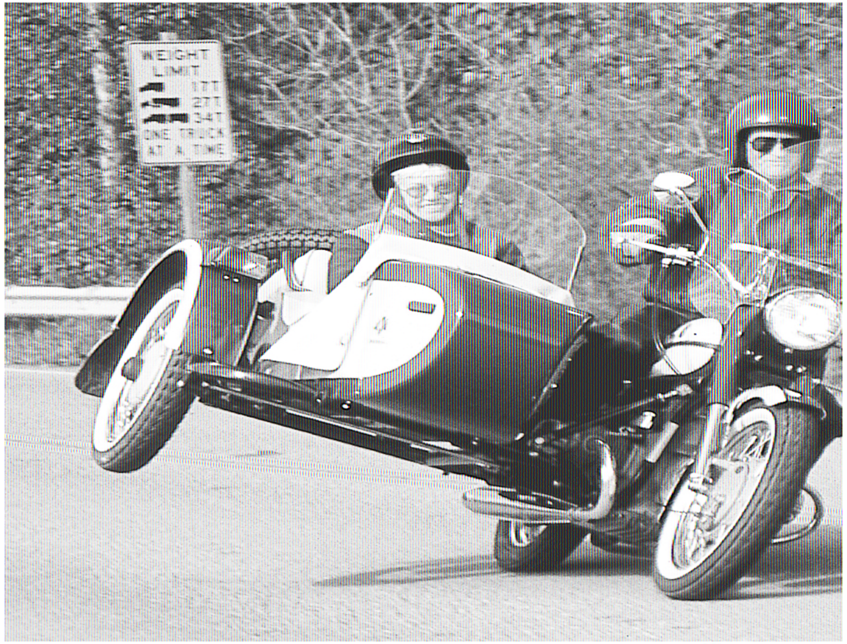
Often implemented as **side car pattern**

Offers additional:

- Monitoring and Traceability(for Ops)
- Enforcement (for SecOps)

Business drivers

- non-functional features/bugfixes **without impacting** the core business functionality
- Faster time to market
- Enables ***shift left*** for Security
- **Dev+Ops** Happy:
    - Dev: doesn't care (!) about non-functionals
    - Ops: implementing changes without impacting service
- Full **visibility** on traffic and versions



devoteam

2

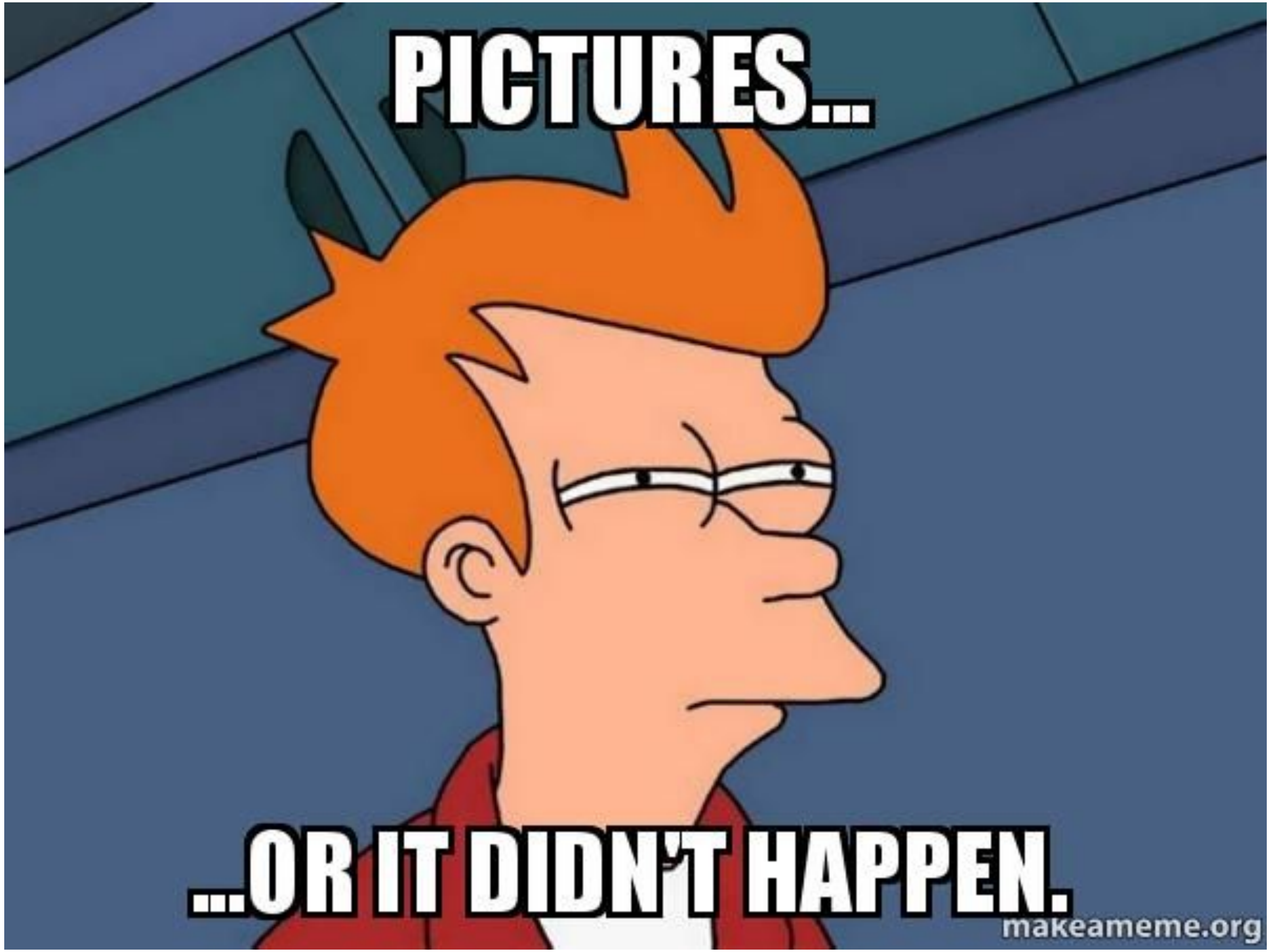
Service Mesh: concepts

A service mesh is a dedicated infrastructure layer for handling **service-to-service communication**. It’s responsible for the reliable delivery of requests through the complex **topology** of services that comprise a modern, cloud native application. In practice, the service mesh is typically implemented as an array of **lightweight network proxies** that are deployed alongside application code, without the application needing to be aware. *(Buoyant.io)*

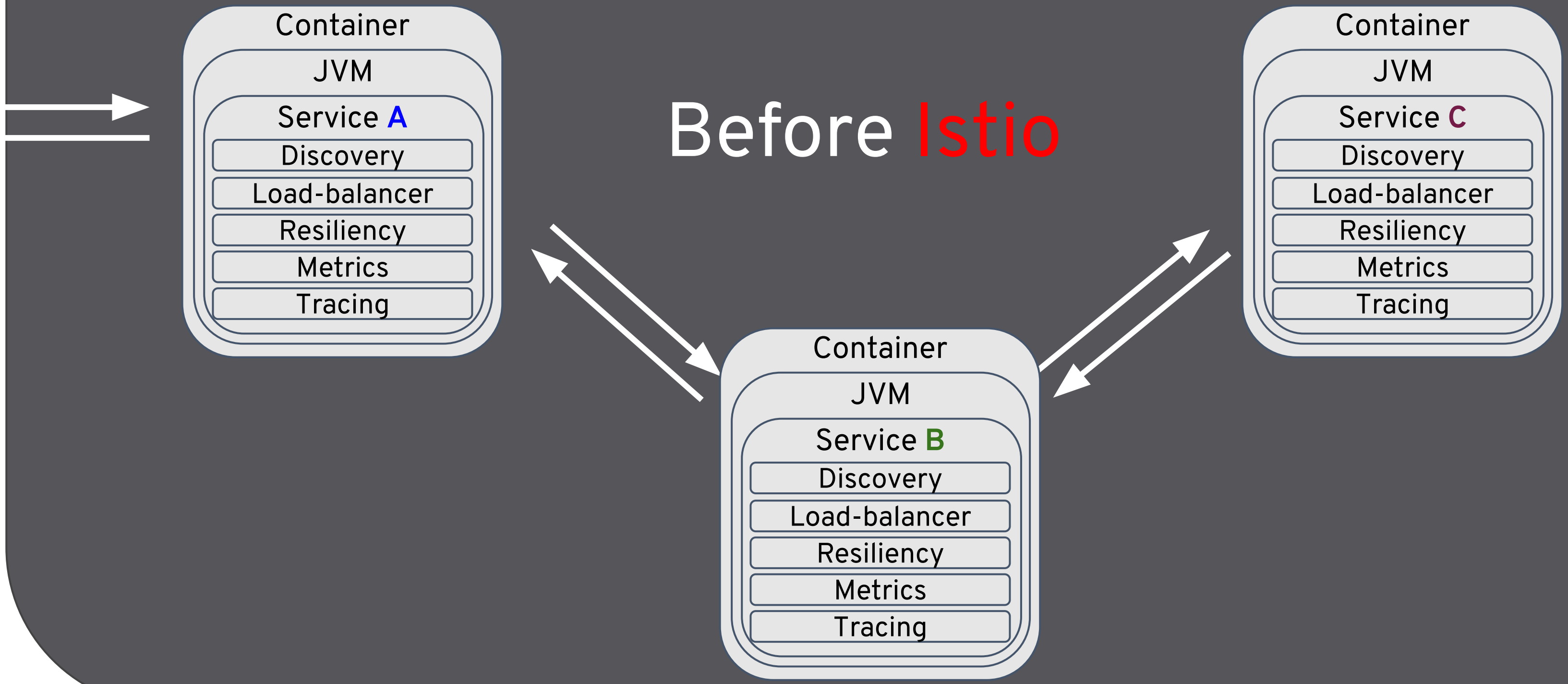
What is a service mesh?

A microservices architecture isolates software functionality into multiple independent services that are independently deployable, highly maintainable and testable, and organized around specific business capabilities. [...]
On a technical level, microservices enable **continuous delivery** and deployment of large, complex applications. On a higher business level, microservices help deliver speed, **scalability**, and **flexibility** to companies trying to achieve agility in rapidly evolving markets. *(New Relic)*

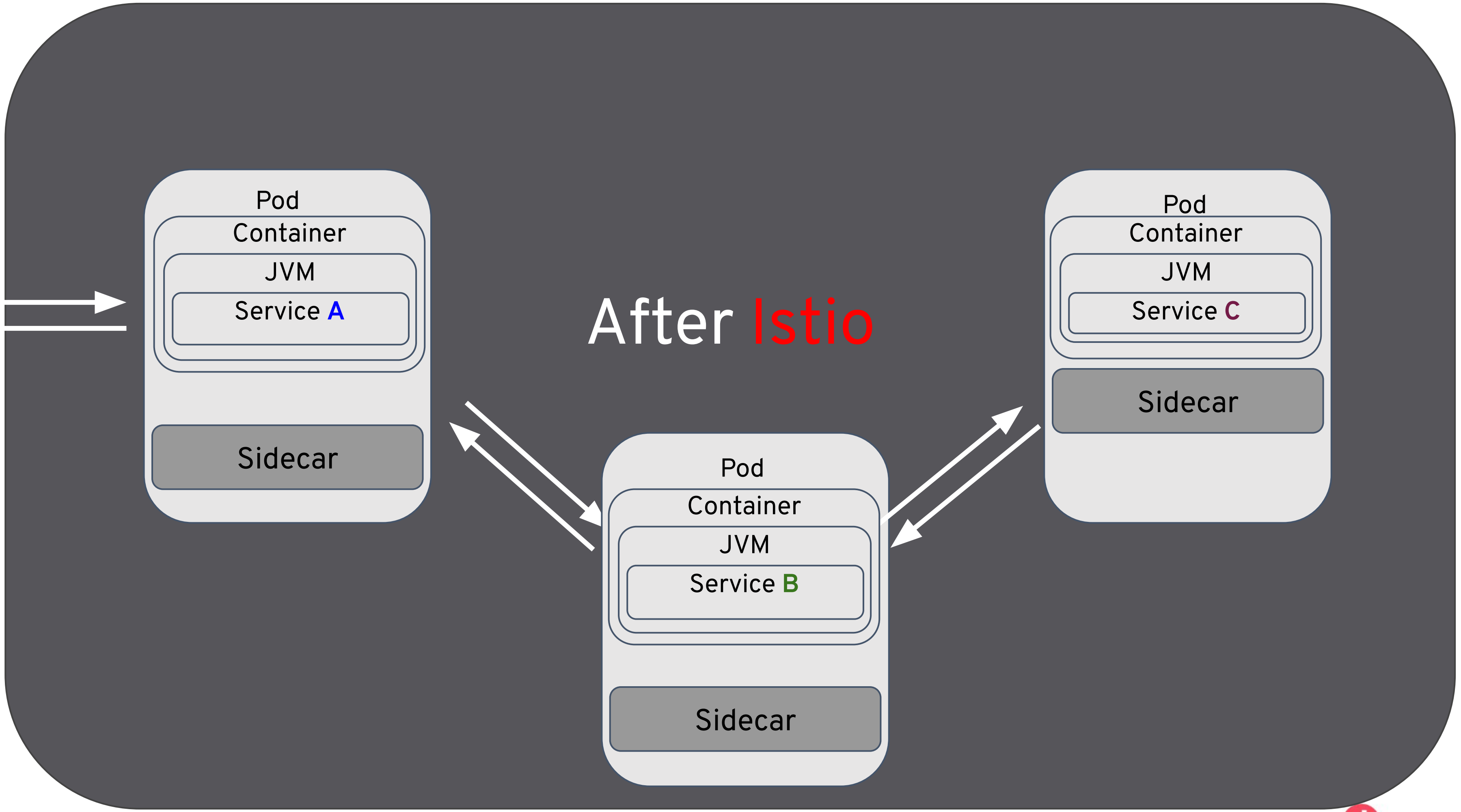
A service mesh is an emerging architecture for **dynamically** linking to one another the chunks of server-side applications -- most notably, the **microservices** *(ZDNet)*


PICTURES...
...OR IT DIDN'T HAPPEN.
makeameme.org

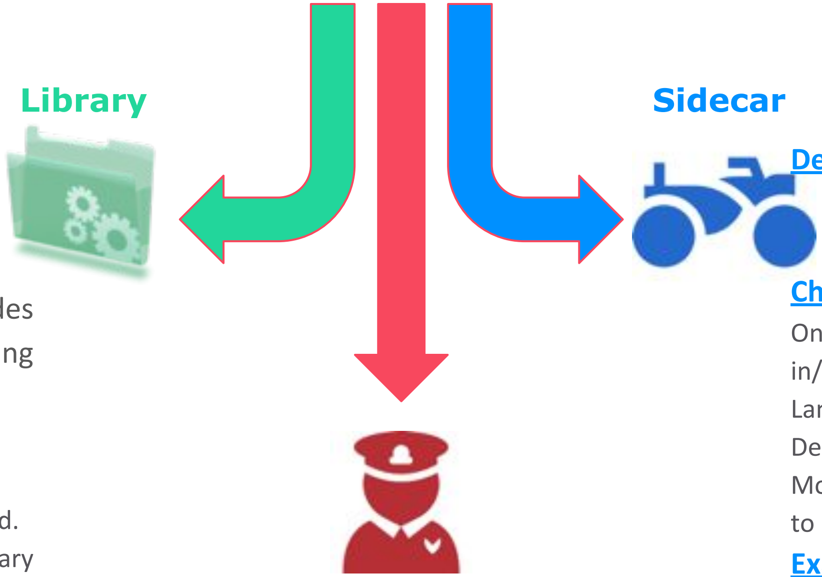
Before Istio



After Istio


Pod
Container
JVM
Service A
Sidecar
Pod
Container
JVM
Service B
Sidecar
Pod
Container
JVM
Service C
Sidecar

Implementation types of service mesh



Library

Definition

Each µservice includes library code implementing service mesh features

Characteristics

Languages Dependencies.
Original, simple and straightforward.
Trust boundary is small (call library inside the process).
Dedicated Resources.

Examples

Hystrix & Ribbon

Node Agent

Definition

Agent (running in a User space process) installed per node/machine managing all the containers on a particular node/machine.

Characteristics

Servicing an heterogenous mix of workloads
Language Agnostic
Resources Sharing à less complex for the configuration (one config per node).

Examples

Consul - Linkerd

Sidecar

Definition

Container running adjacent to the container in the same pod.

Characteristics

One sidecar per pod which manages the traffic in/out of the application container.
Language agnostic
Dedicated resource (per pod)
More flexibility (Security – routing - …) and easier to secure

Examples

Istio –AVI networks – Tigera – Aspen Mesh

Standardisation

A standard interface for service meshes

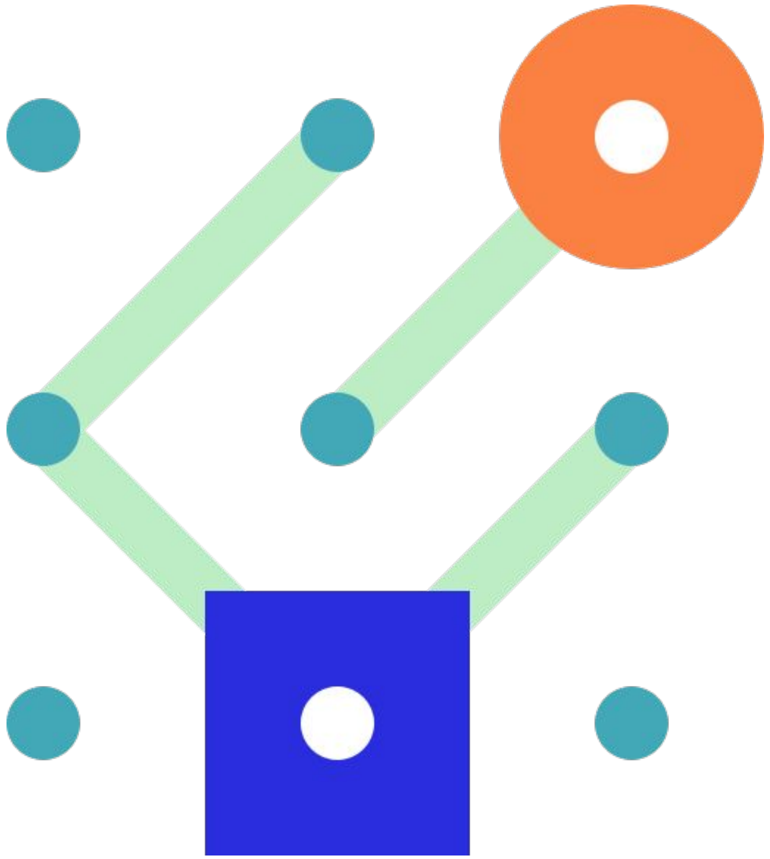
Basic feature set for most common features

- Traffic policy
- Traffic telemetry
- Traffic management

Kubernetes native
specified as a collection of Kubernetes Custom Resource Definitions (CRD)

Provider agnostic

https://smi-spec.io/



devoteam

SMI: partners

Microsoft LINKERD HashiCorp solo.io

Red Hat RANCHER docker meshery

weaveworks ASPEN MESH Pivotal vmware

kinvolk CANONICAL kubecost

3

Service Mesh with Istio

Istio @ GitHub
14,500 stars
6,400 commits
300 contributors

Integrations
Aspen Mesh
Avi Networks
Cisco
OpenShift
NGINX
Rancher
Tufin Orca
Tigera
Twistlock
VMware.

Features

Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic.

Fine-grained *control of traffic* behavior with rich routing rules, retries, failovers, and fault injection.

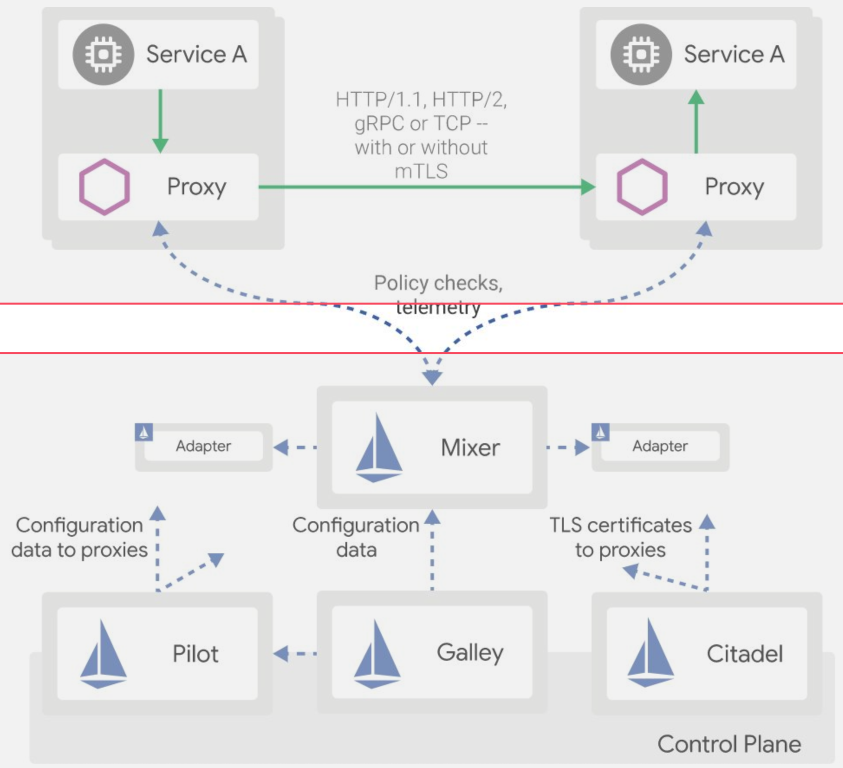
A *pluggable policy layer* and configuration API supporting access controls, rate limits and quotas.

Automatic *metrics, logs, and traces* for all traffic within a cluster, including cluster ingress and egress.

Secure service-to-service communication in a cluster with strong identity-based authentication and authorization.

Main architecture

Separation Data / Control plane



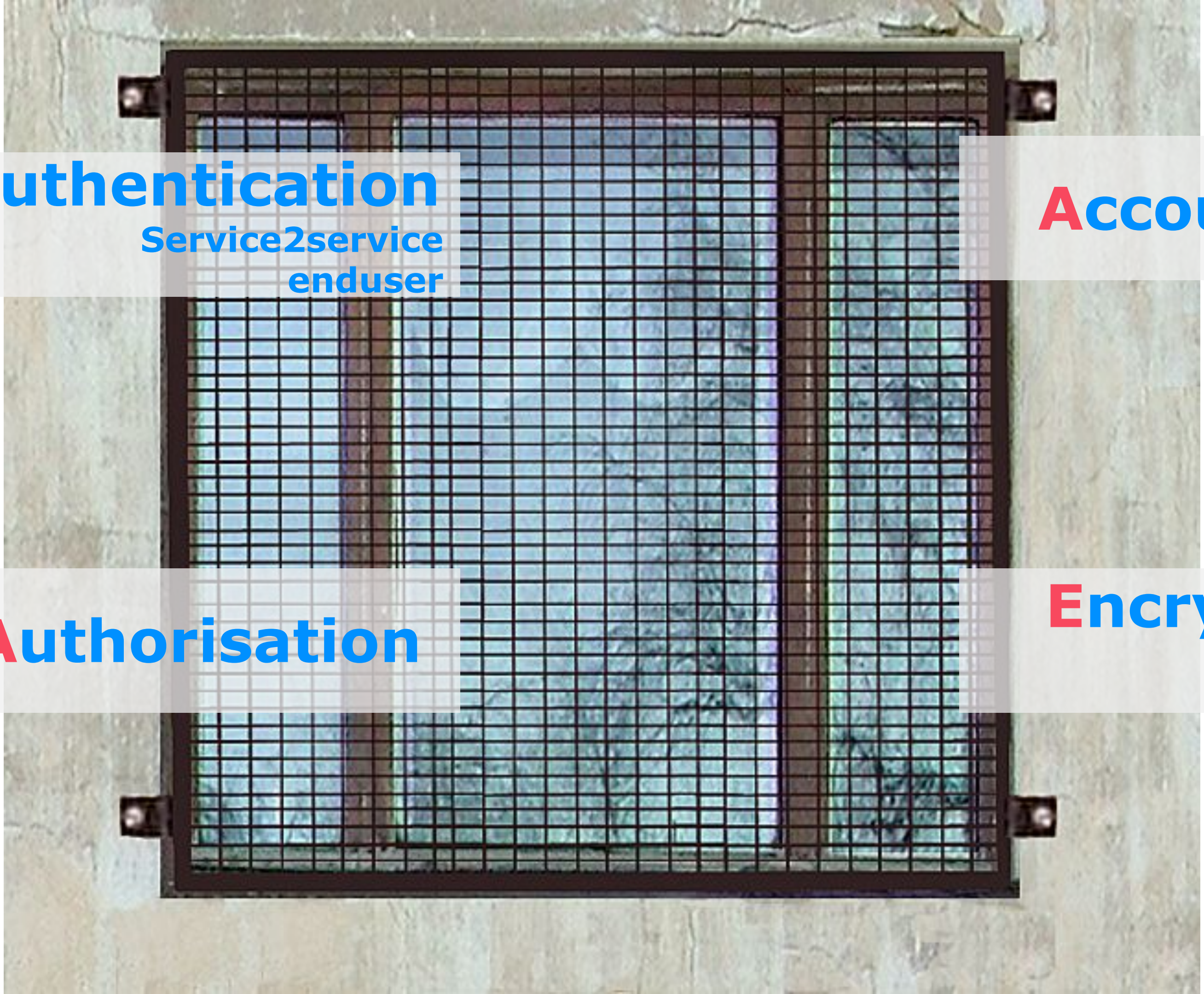
Data Plane:
Intelligent sidecar **envoy** proxies

Control Plane:
Sets routes
Configures policy and telemetry
mixer hub

Istio - Envoy proxy

A real working horse

- a **high-performance proxy** developed in C++
- built-in features, for example:
  - Dynamic service discovery
  - Load balancing
  - TLS termination
  - HTTP/2 and gRPC proxies
  - Circuit breakers
  - Health checks
  - Staged rollouts with %-based traffic split
  - Fault injection
  - Rich metrics
- Customization / Extending
  - Lua scripting
  - C++ envoy
- Platform independent



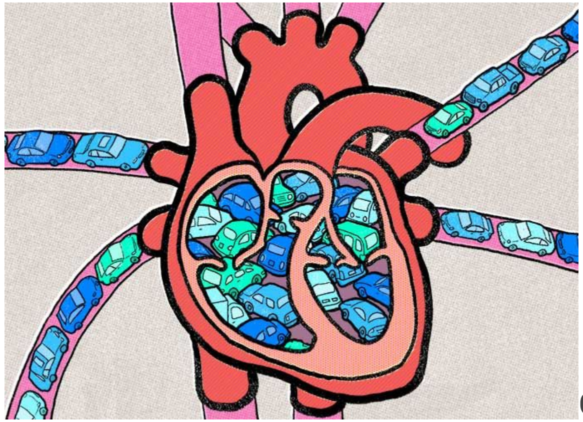
Authentication
Service2service
enduser

Accounting

Authorisation

Encryption
mTLS

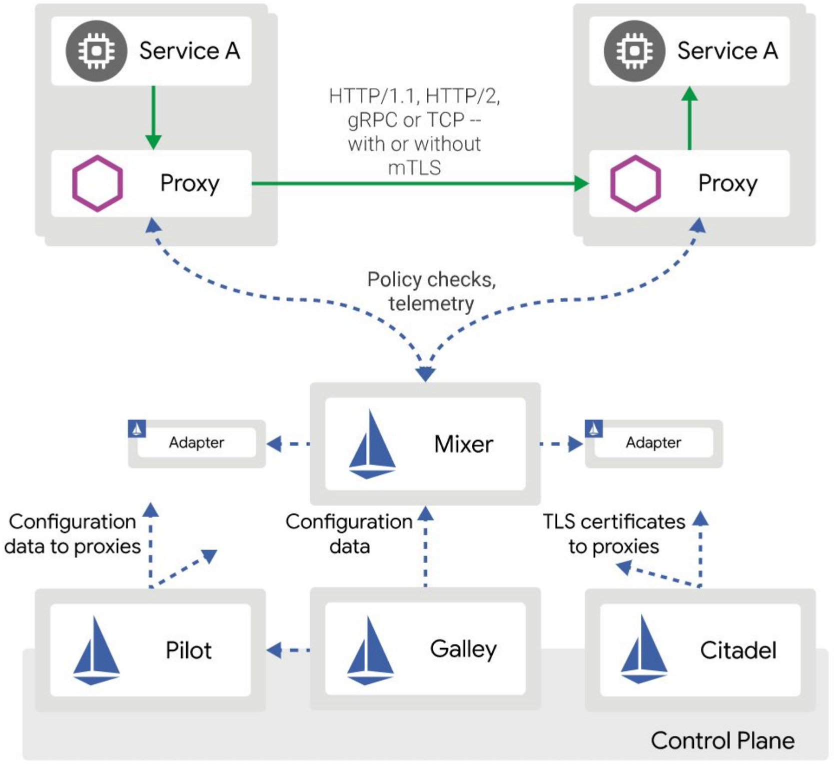
Istio - Envoy proxy at the core of your traffic


Credit: Jess Wheelock

- as a **sidecar**
  - **extract a wealth of signals** about **traffic behavior as attributes**.
  - use these attributes in Mixer to **enforce policy decisions**,
  - send them to monitoring systems to provide information about the behavior of the entire mesh.

- The sidecar proxy model also allows you to **add Istio capabilities** to an existing deployment with **no need to rearchitect or rewrite code**.

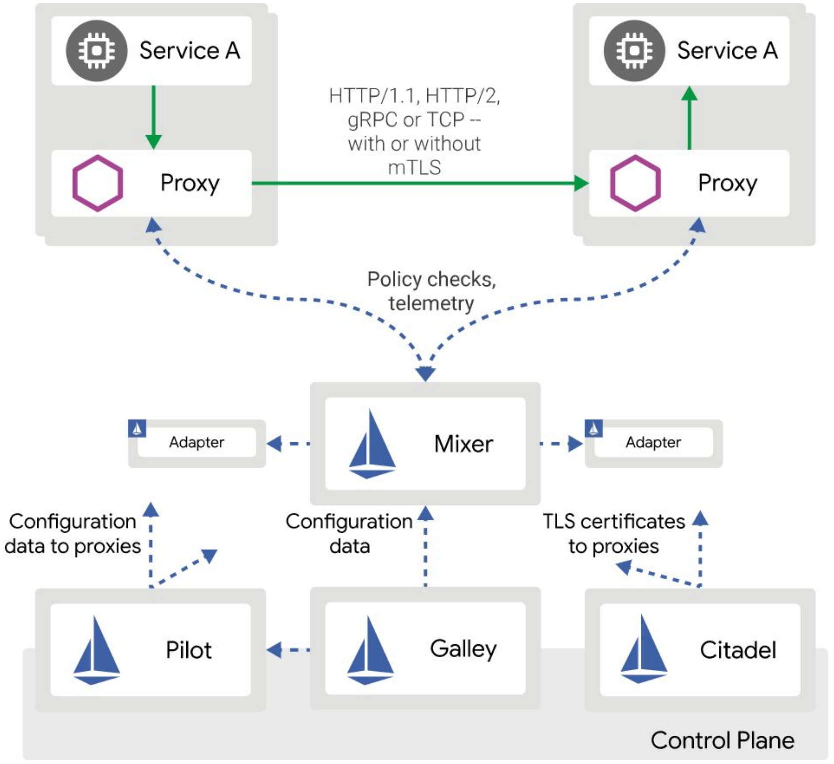
Istio - Mixer



Mixer

- a platform-independent component.
- Mixer **enforces access control** and usage policies across the service mesh, and **collects telemetry** data from the Envoy proxy and other services. The proxy extracts request level attributes, and sends them to Mixer for evaluation.

Istio - Pilot

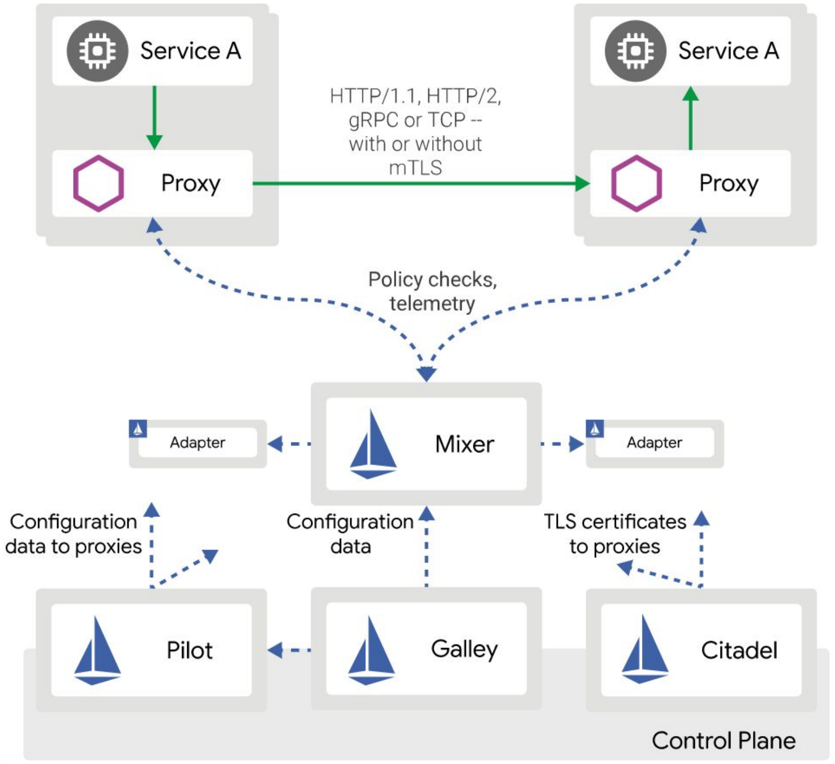


Pilot

- provides **service discovery** for the Envoy sidecars, **traffic management** capabilities for intelligent routing (e.g., A/B tests, canary rollouts, etc.), and **resiliency** (timeouts, retries, circuit breakers, etc.).

- **converts high level routing** rules that control traffic behavior into Envoy-specific configurations, and propagates them to the sidecars at runtime.
- **abstracts platform-specific service discovery** mechanisms and synthesizes them into a standard format that any sidecar conforming with the Envoy data plane APIs can consume.

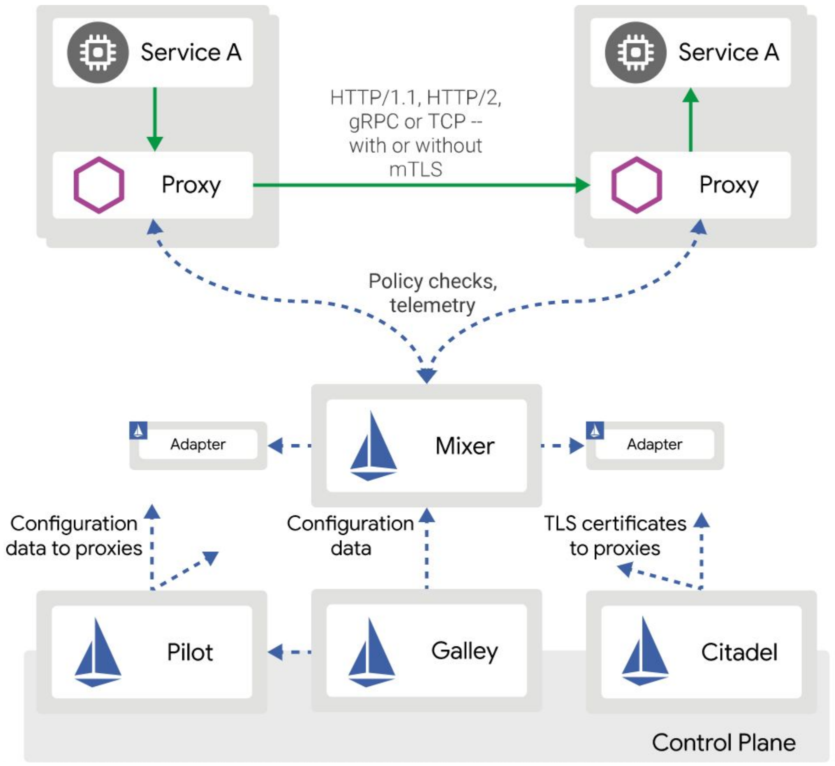
Istio - Citadel



Citadel

- Identity component
- enables strong service-to-service and end-user authentication with built-in identity and credential management.
- Built-in PKI

Istio - Galley



Galley

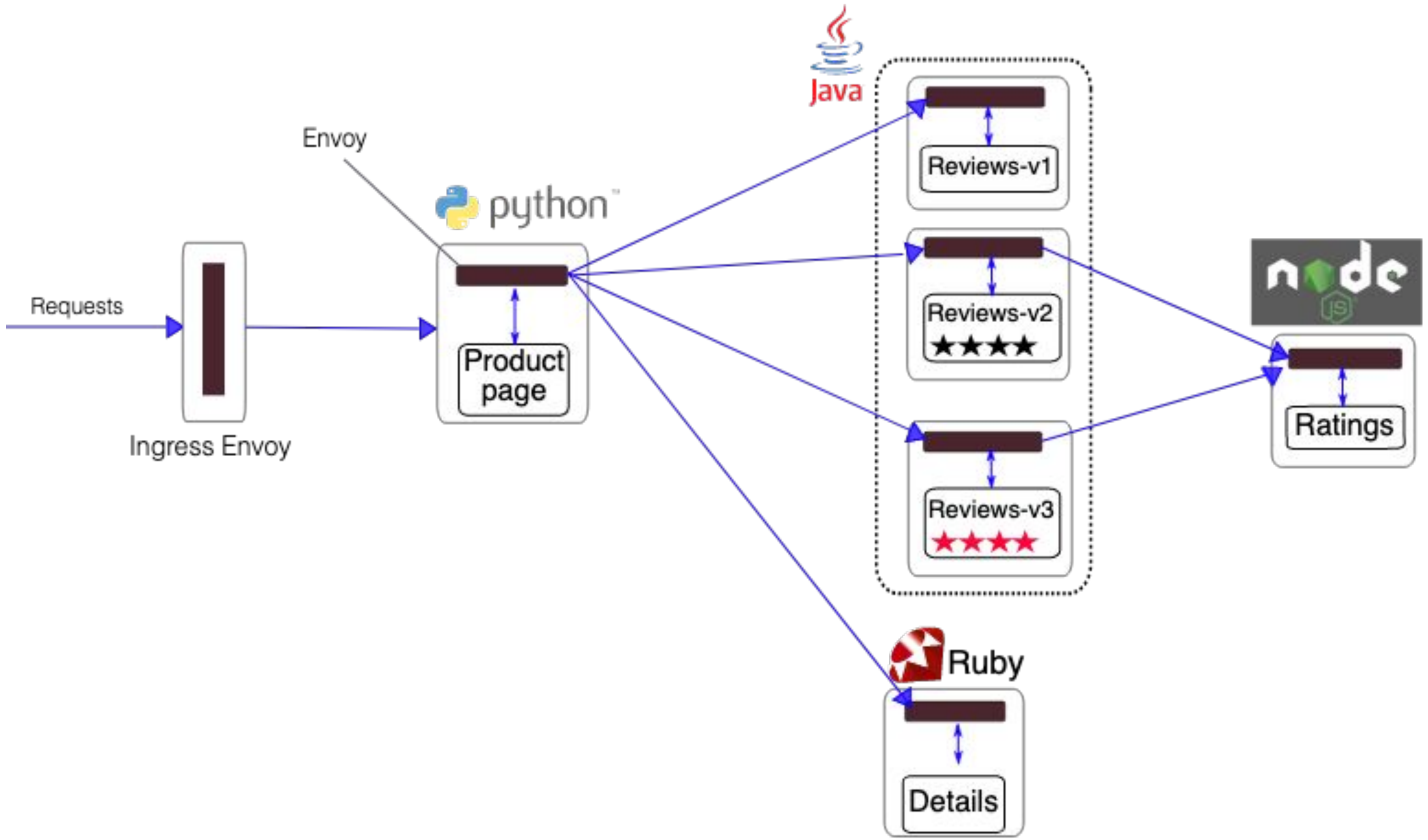
- is Istio's **configuration validation**, ingestion, processing and distribution component.
- insulating the rest of the Istio components from the details of **obtaining user configuration** from the underlying platform (e.g. Kubernetes).

JAEGER

kiali

Prometheus

Grafana


Java
Envoy
python
Reviews-v1
Reviews-v2
★★★★
node
Requests
Product page
Ratings
Ingress Envoy
Reviews-v3
★★★★
Ruby
Details

devoteam

Istio comparaison

Traefik

- Several native backends/integrations
- Dynamic configuration
- Really simple to deploy
- Ideal for small ITs
- Not really a service mesh

Kong

- Plugin support
- Flexible
- Easy to maintain

Linkerd

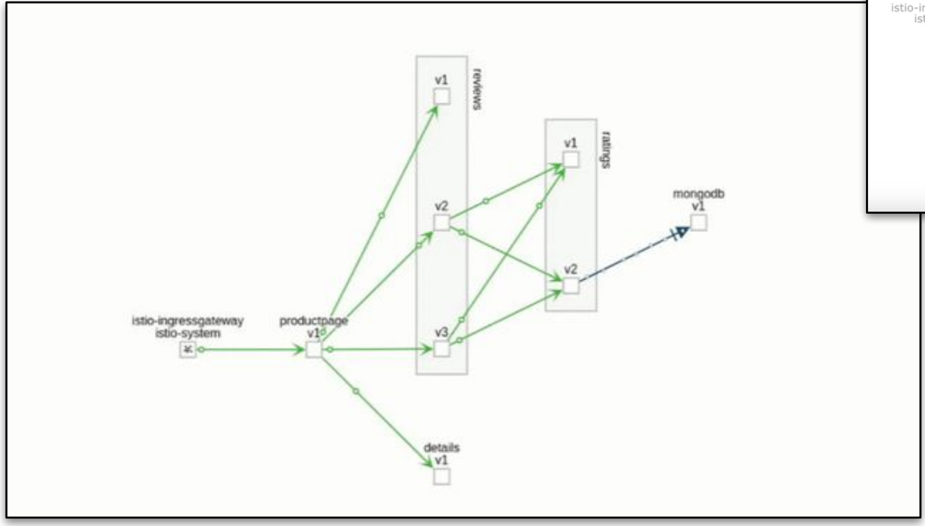
- Node agent
- High performance/traffic load balancer
- No TCP support
- Per request/function routing

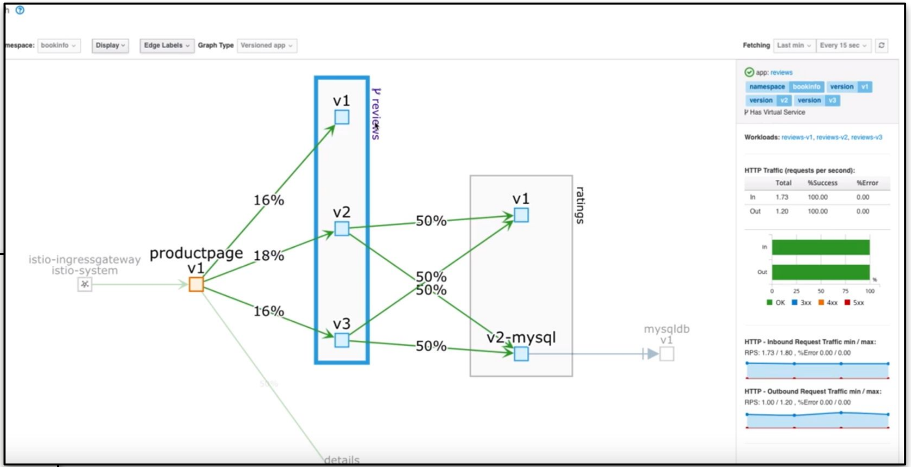
Linkerd2 (formerly Conduit)

- Sidecar
- Low complexity
- TCP Support
- Commercial support available
- Low latency
- Kubernetes specific

Also Consul connect, Mesher, Ambassador...

Kiali





app: reviews
namespace bookinfo version v1 version v2 version v3
Has Virtual Service

Workloads: reviews-v1, reviews-v2, reviews-v3

HTTP Traffic (requests per second):

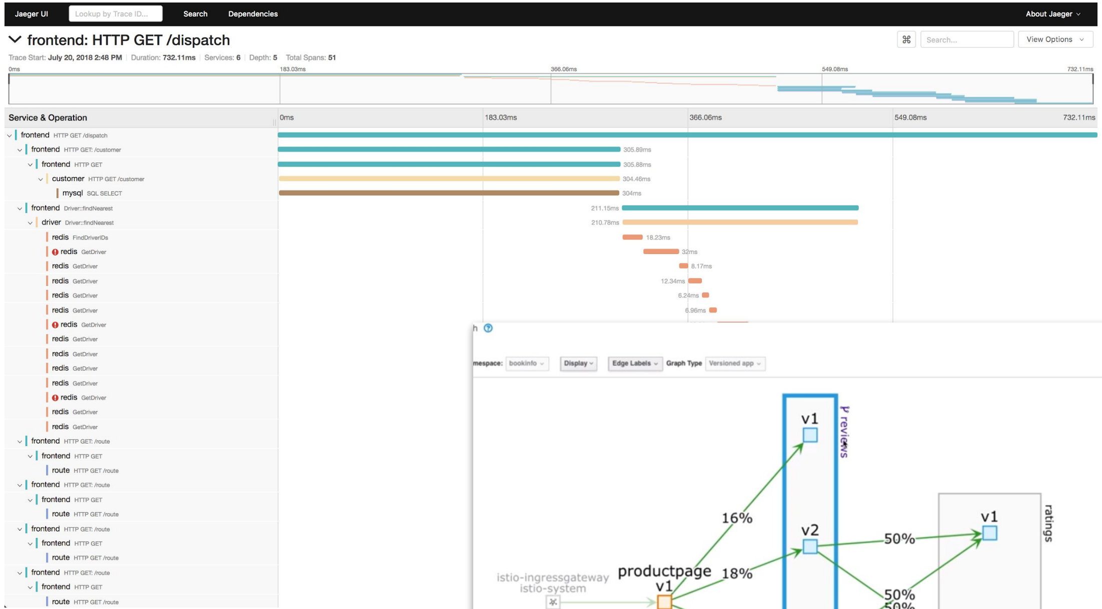
	Total	%Success	%Error
In	1.73	100.00	0.00
Out	1.20	100.00	0.00

HTTP - Inbound Request Traffic min / max:
RPS: 1.73 / 1.80 , %Error 0.00 / 0.00

HTTP - Outbound Request Traffic min / max:
RPS: 1.00 / 1.20 , %Error 0.00 / 0.00

- Manage, observe and analyzes service mesh, services and related objects as deployments
- Now Openshift integrated

Jaeger tracing

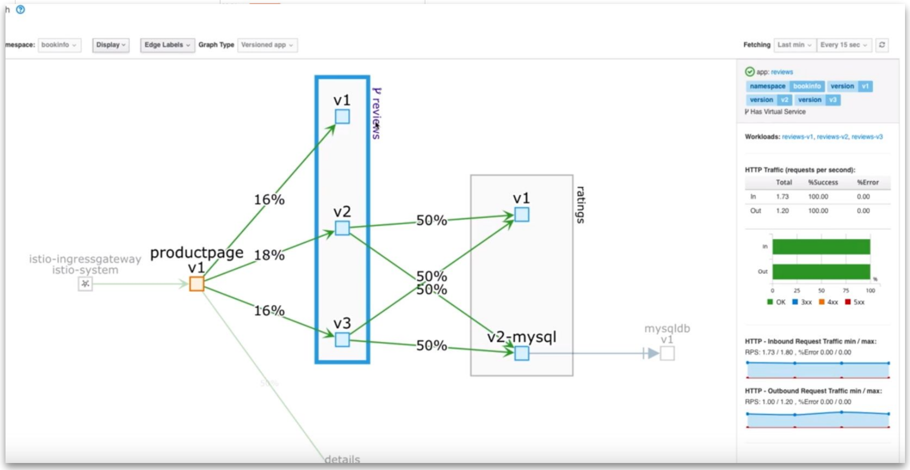


Jaeger UI | Lookup by Trace ID... | Search | Dependencies | About Jaeger

frontend: HTTP GET /dispatch

Trace Start: **July 20, 2018 2:48 PM** | Duration: **732.11ms** | Services: **6** | Depth: **5** | Total Spans: **51**

```
Service & Operation                 0ms   183.03ms   366.06ms   549.08ms   732.11ms
frontend  HTTP GET /dispatch
  frontend  HTTP GET: /customer                 305.89ms
    frontend  HTTP GET                          305.88ms
      customer  HTTP GET /customer              304.46ms
        mysql  SQL SELECT                       304ms
  frontend  Driver::findNearest                 211.15ms
    driver  Driver::findNearest                 210.78ms
      redis  FindDriverIDs                      18.23ms
      redis  GetDriver                          32ms
      redis  GetDriver                          8.17ms
      redis  GetDriver                          12.34ms
      redis  GetDriver                          6.24ms
      redis  GetDriver                          6.96ms
      redis  GetDriver
      redis  GetDriver
      redis  GetDriver
      redis  GetDriver
      redis  GetDriver
      redis  GetDriver
      redis  GetDriver
      redis  GetDriver
  frontend  HTTP GET: /route
    frontend  HTTP GET
      route  HTTP GET /route
  frontend  HTTP GET: /route
    frontend  HTTP GET
      route  HTTP GET /route
  frontend  HTTP GET: /route
    frontend  HTTP GET
      route  HTTP GET /route
  frontend  HTTP GET: /route
    frontend  HTTP GET
      route  HTTP GET /route
```



app: reviews

namespace bookinfo | version v1 | version v2 | version v3

Has Virtual Service

Workloads: reviews-v1, reviews-v2, reviews-v3

HTTP Traffic (requests per second):

	Total	%Success	%Error
In	1.73	100.00	0.00
Out	1.20	100.00	0.00

🟩 OK 🟦 3xx 🟧 4xx 🟥 5xx

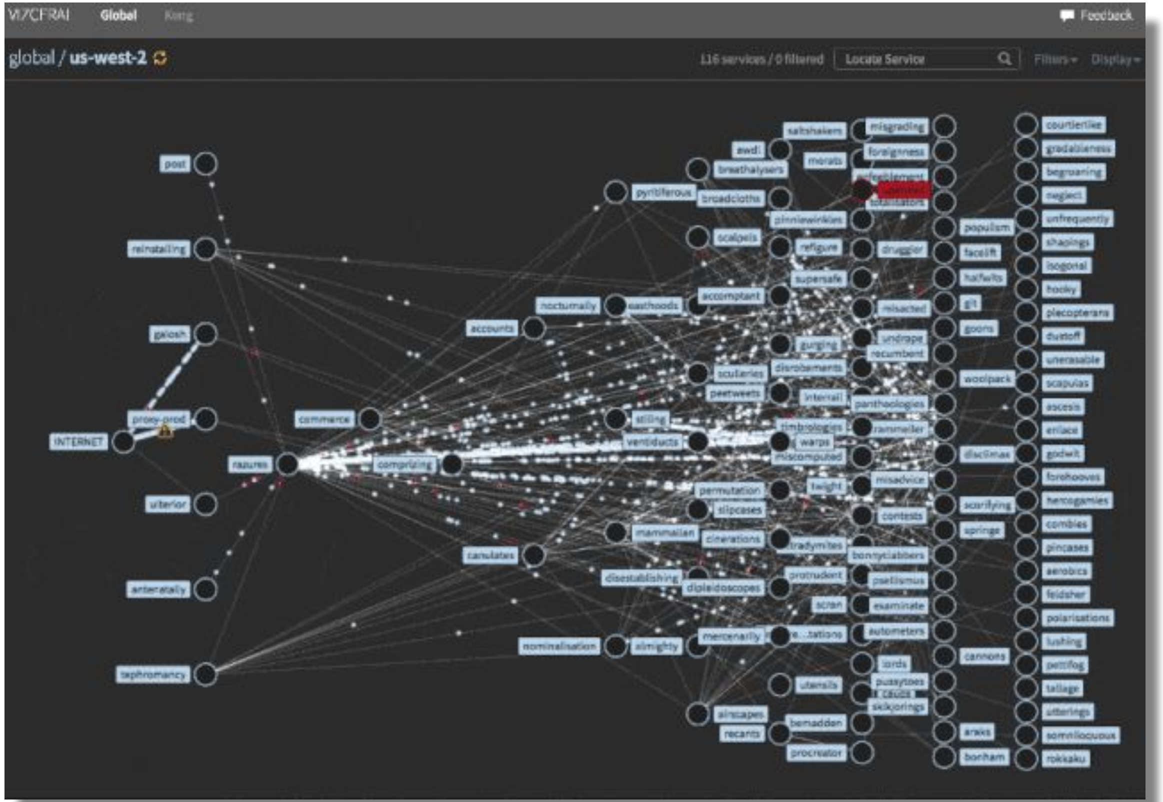
HTTP - Inbound Request Traffic min / max:
RPS: 1.73 / 1.80 , %Error 0.00 / 0.00

HTTP - Outbound Request Traffic min / max:
RPS: 1.00 / 1.20 , %Error 0.00 / 0.00

- Jaeger + Kiali integration

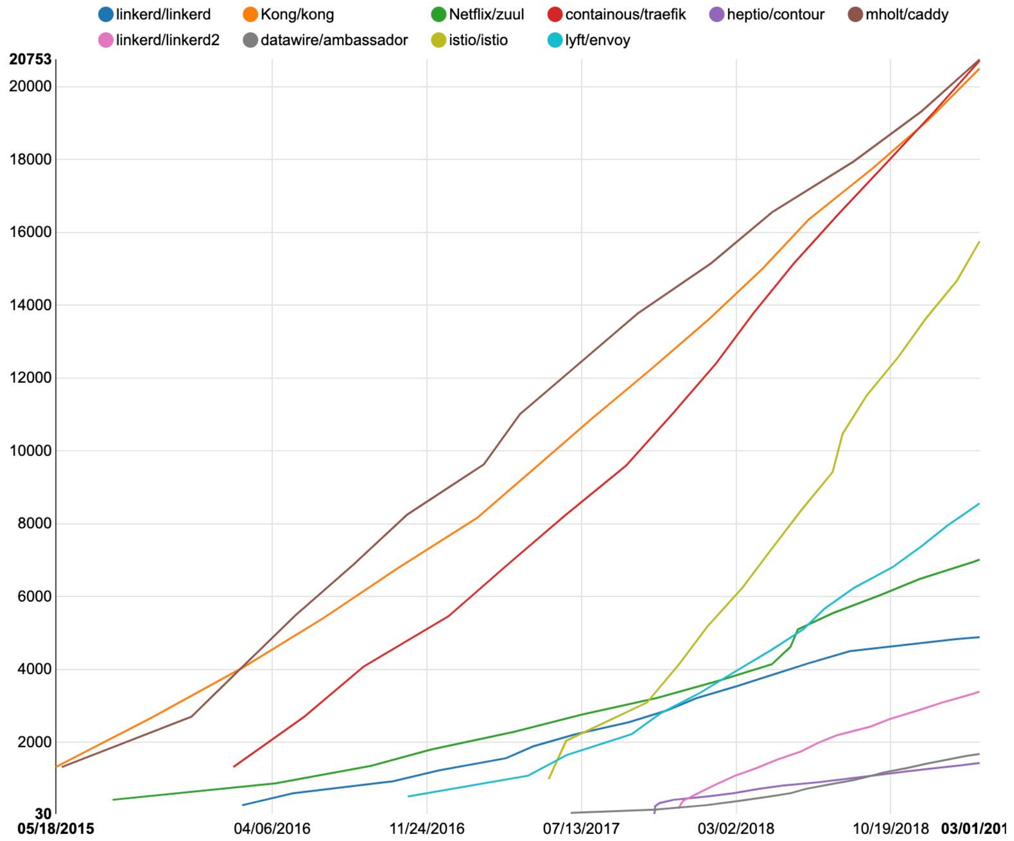
devoteam

Netflix Vizceral



devoteam

Service mesh applications



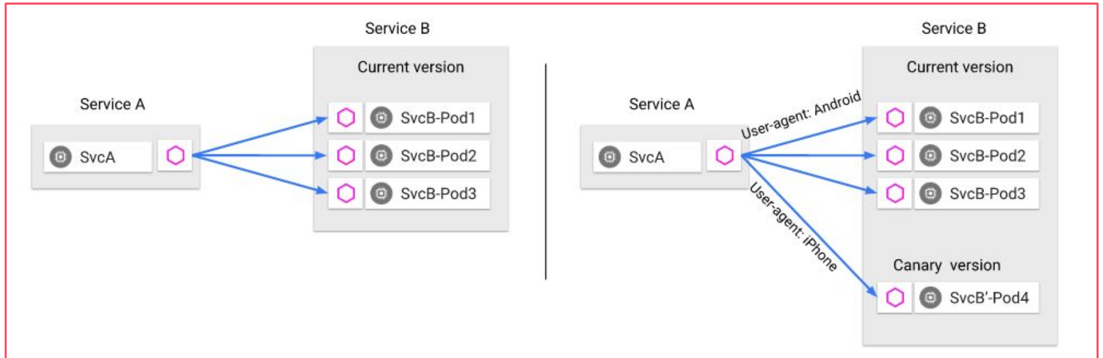
3

Use-Cases

Use-cases: overview

1. Traffic management
2. Canary deployments
3. Environment as service mesh
4. Traffic shadowing
5. Canary Analysis
6. Istio gateway: build real hybrid applications
   a. multi cloud
   b. integration BM/VM

Traffic management



Content-based traffic steering - The content of a request can be used to determine the destination of a request

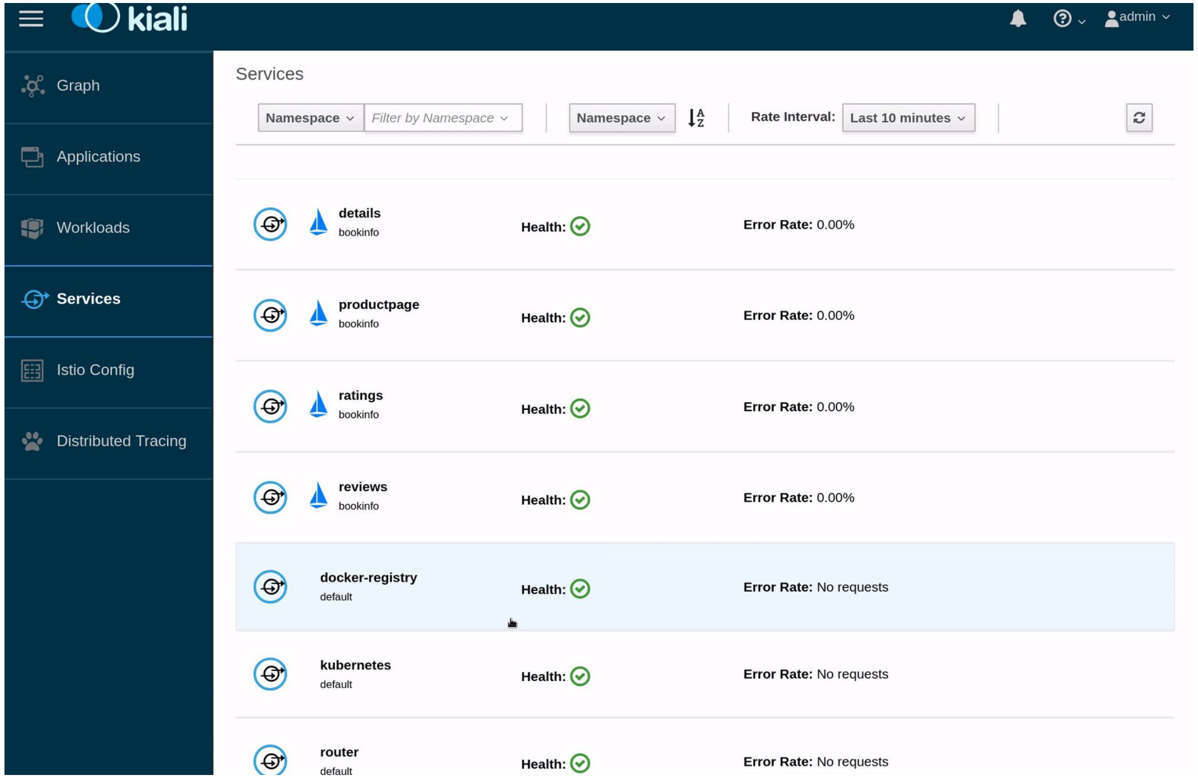
Flexible routing strategies, based on:

- Domain, subdomain
- Url, paths
- Headers
- User-agent
- Geolocalisation

Rule-based traffic control means you can route a specific portion of traffic to a specific instance of a service (for example, specify the percentage of traffic that should hit a canary deploy), or set routing rules based on the content of a request

```
apiVersion:
networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: productpage
spec:
  hosts:
    - productpage
  http:
  - match:
    - uri:
        prefix: /api/v1
    route:
    ...
```

Traffic management - Visibility with Kiali



Services

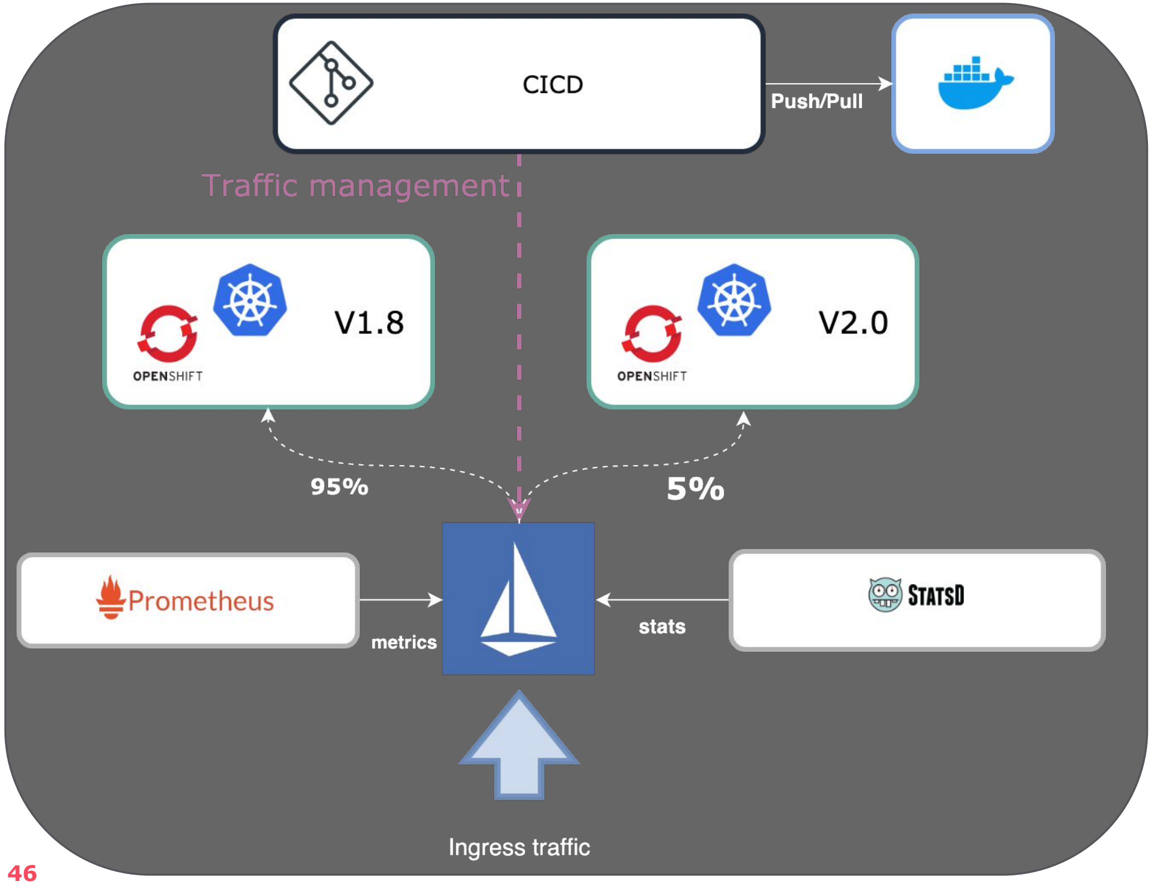
Service	Namespace	Health	Error Rate
details	bookinfo	Health: ✓	Error Rate: 0.00%
productpage	bookinfo	Health: ✓	Error Rate: 0.00%
ratings	bookinfo	Health: ✓	Error Rate: 0.00%
reviews	bookinfo	Health: ✓	Error Rate: 0.00%
docker-registry	default	Health: ✓	Error Rate: No requests
kubernetes	default	Health: ✓	Error Rate: No requests
router	default	Health: ✓	Error Rate: No requests

Flexible routing strategies, based on:

- Domain, subdomain
- Url, paths
- Headers
- User-agent
- Geolocalisation

```
apiVersion:
networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: productpage
spec:
  hosts:
    - productpage
  http:
  - match:
    - uri:
        prefix: /api/v1
    route:
    ...
```

Traffic management - Visibility with Kiali

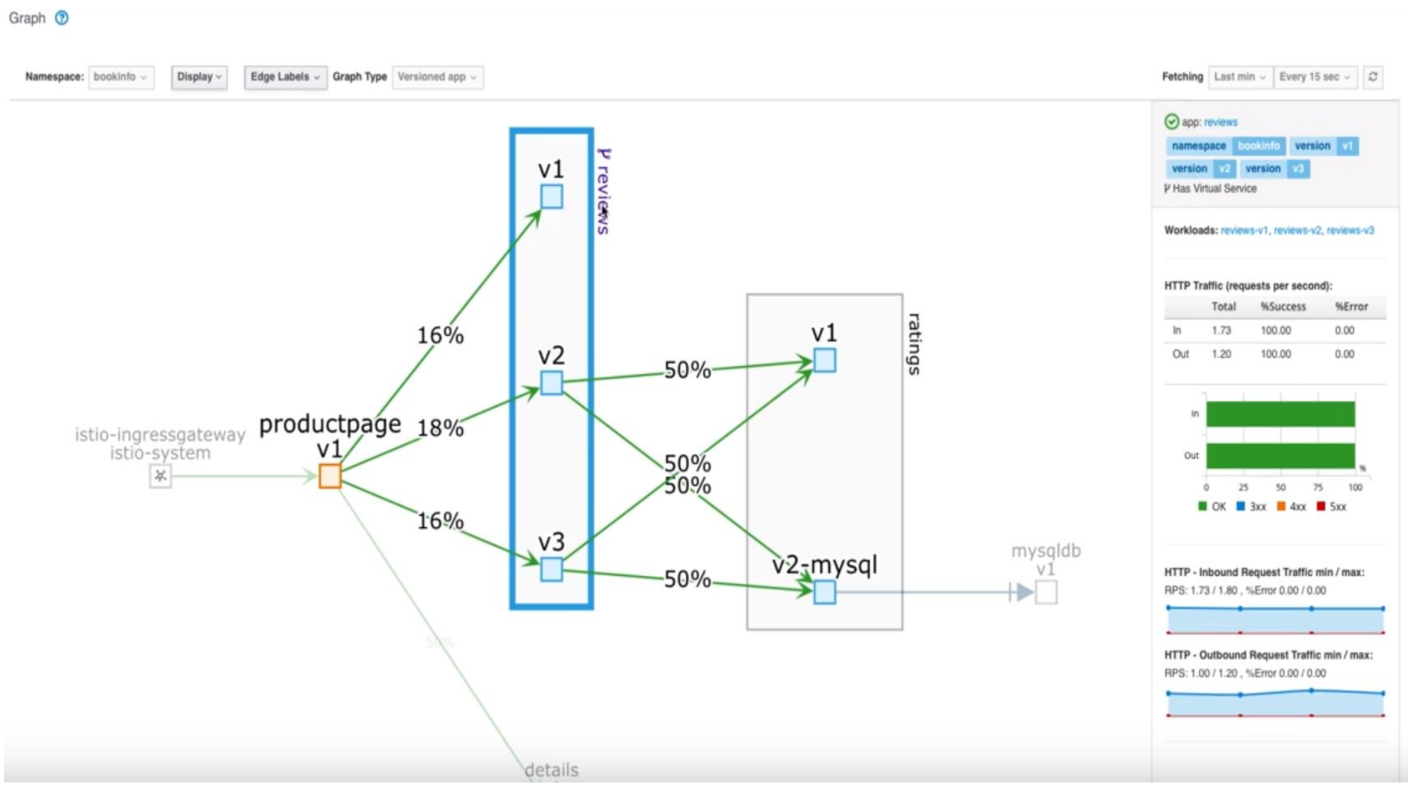


Flexible routing strategies, based on:

- Domain, subdomain
- Url, paths
- Headers
- User-agent
- Geolocalisation

```
apiVersion:
networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: productpage
spec:
  hosts:
    - productpage
  http:
  - match:
    - uri:
        prefix: /api/v1
    route:
    ...
```

Canary Deployment



app: reviews

namespace bookinfo | version v1 | version v2 | version v3

Has Virtual Service

Workloads: reviews-v1, reviews-v2, reviews-v3

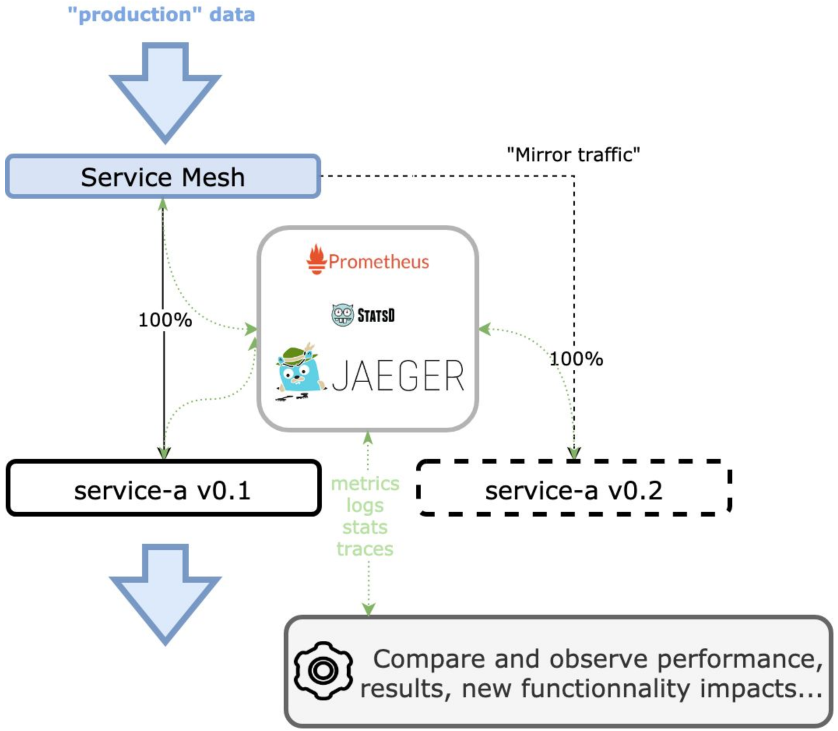
HTTP Traffic (requests per second):

	Total	%Success	%Error
In	1.73	100.00	0.00
Out	1.20	100.00	0.00

HTTP - Inbound Request Traffic min / max:
RPS: 1.73 / 1.80 , %Error 0.00 / 0.00

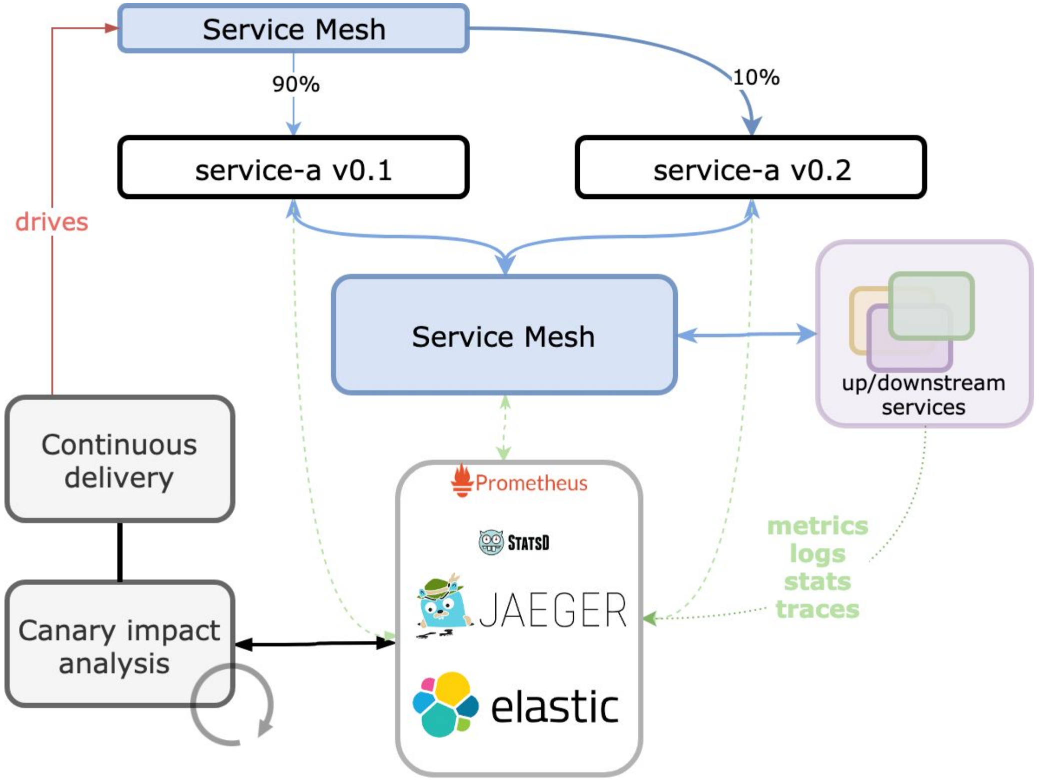
HTTP - Outbound Request Traffic min / max:
RPS: 1.00 / 1.20 , %Error 0.00 / 0.00

Traffic shadowing and sensitive deliveries



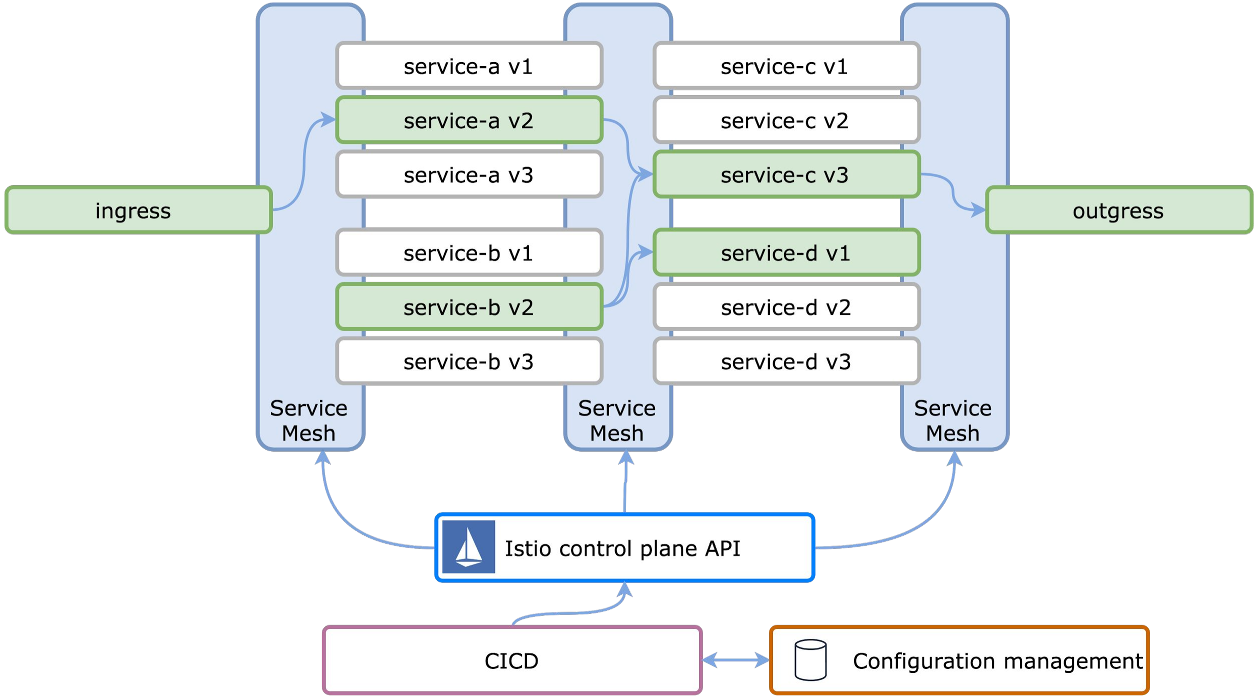
- Test for errors, exceptions, performance, and result parity.
- Mirror 100% of the traffic
- No impact on current traffic
- Compare and observe with production data (Twitter Diffy like)
- Traffic is mirrored as “fire-and-forget”

Deployment driven by canary analysis



- A prerequisite to implementing canary releases is the ability to effectively observe and monitor your infrastructure and application stack.
- Gradual rollout of new functionality limits the potential system blast radius of any operational issues
- Deployment impact analysis by metrics and traces analytics (Harness like)

Environment as a Service mesh



- Service-to-Service as Code :-)
- Define cluster wide routing definitions
- Apply to any environment

devoteam

4

Hype or Reality

devoteam

digipolis

devoteam

Reality?

Sidecars?

Yet another new paradigm?

Proxies are proven technology

Easy development?

Hard to get started with.

Easier, no non-functionals

Istio?

Yet another **new greek work** to remember!

YEP!

New Infrastructure?

My ops already get apesh*t crazy with those containers

Your Ops get F_I_N_A_L_L_Y insight in those containers

devoteam