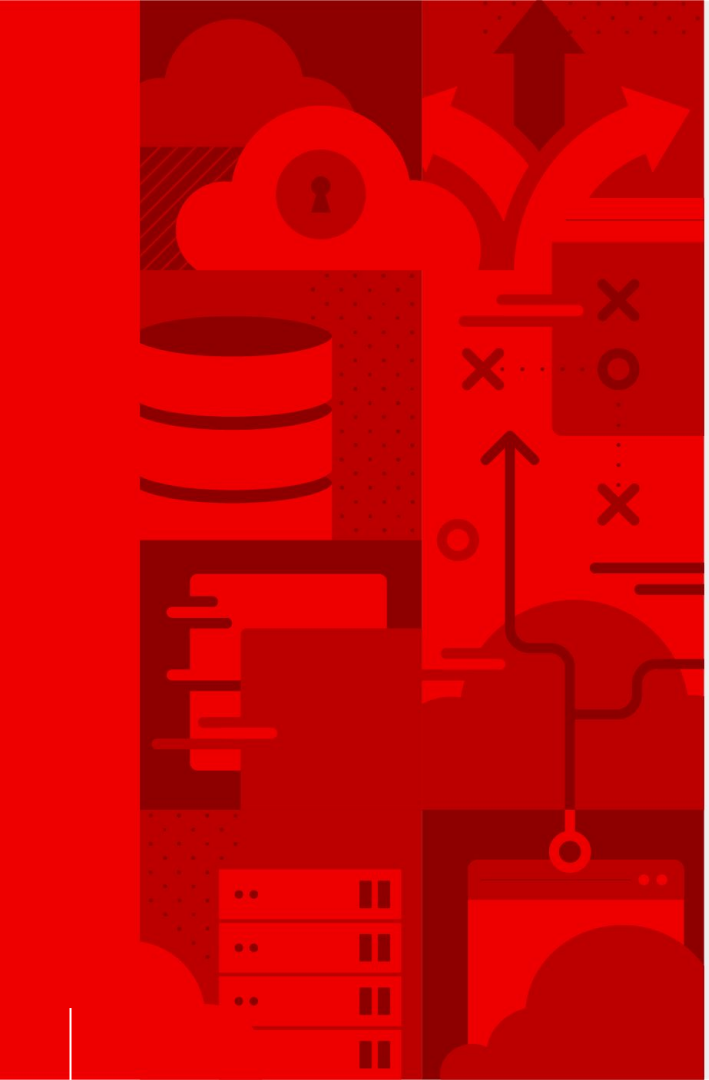


OpenShift Developer

Architecture Workshop

ServiceMesh & Serverless

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat

Self introduction

Name: Wanja Pernath

Email: wpernath@redhat.com

Base: Germany (very close to the Alps)

Role: EMEA Technical Partner Development Manager - OpenShift and MW

Experience: Years of Consulting, Training, PreSales at Red Hat and before

Agenda

Agenda

- Serverless
- ServiceMesh
- Summary & Thank you

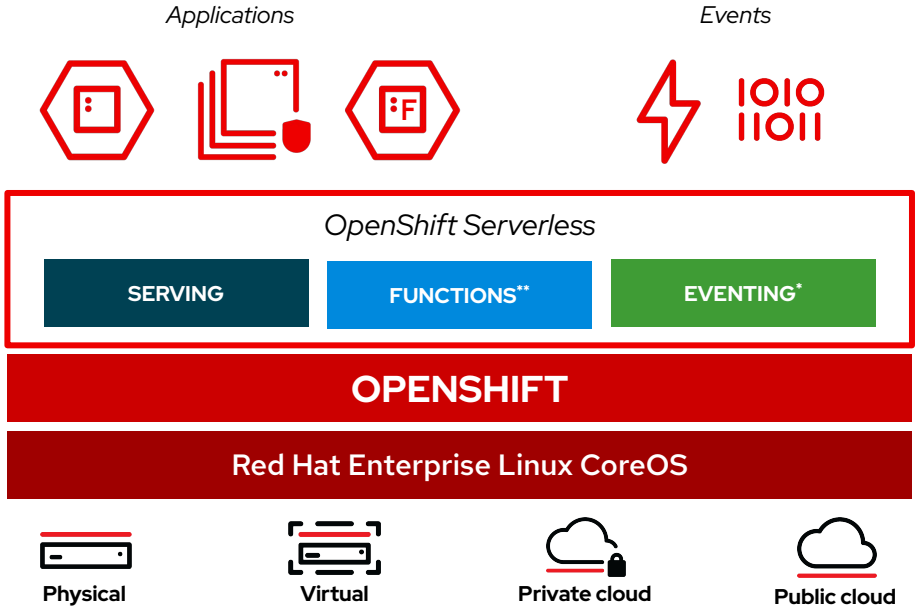
Serverless / knative

Generally Available

Openshift Serverless

Event-driven serverless containers and functions

- Deploy and run **serverless containers**
- Use any programming language or runtime
- Modernize existing applications to run serverless
- Powered by a rich ecosystem of event sources
- Manage serverless apps natively in Kubernetes
- Based on open source project **Knative**
- Run anywhere OpenShift runs



* Eventing is currently in Technology Preview

** Functions are currently a work in progress initiative

Red Hat

OpenShift Serverless

Key Features

Containers made easy

Simplified developer experience to deploy applications/code on serverless containers abstracting infrastructure & focusing on what matters.

Ready for the Hybrid Cloud

Truly portable serverless running anywhere OpenShift runs, that is on-premises or on any public cloud. Leverage data locality and SaaS when needed.

Immutable revisions

Deploy new features: performing canary, A/B or blue-green testing with gradual traffic rollout with no sweat and following best practices.

Any programming language

Use any programming language or runtime of choice. From Java, Python, Go and JavaScript to Quarkus, SpringBoot or Node.js.

Automatic scaling

No need to configure number of replicas, or idling. Scale to zero when not in use, auto scale to thousands during peak, with built-in reliability and fault-tolerance.

Event Driven Architectures

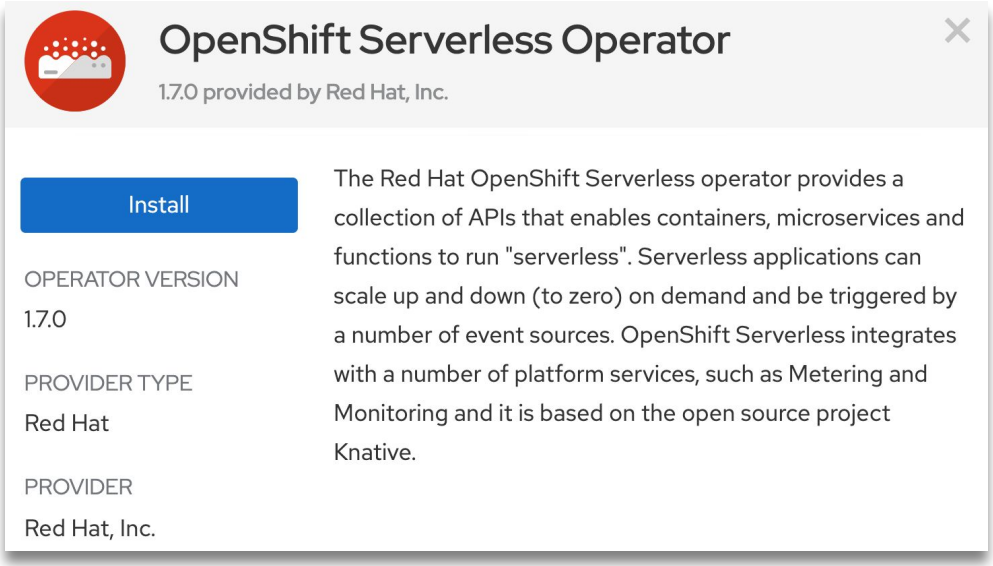
Build loosely coupled & distributed apps connecting with a variety of built-in or third-party event sources or connectors powered by Operators.

Red Hat

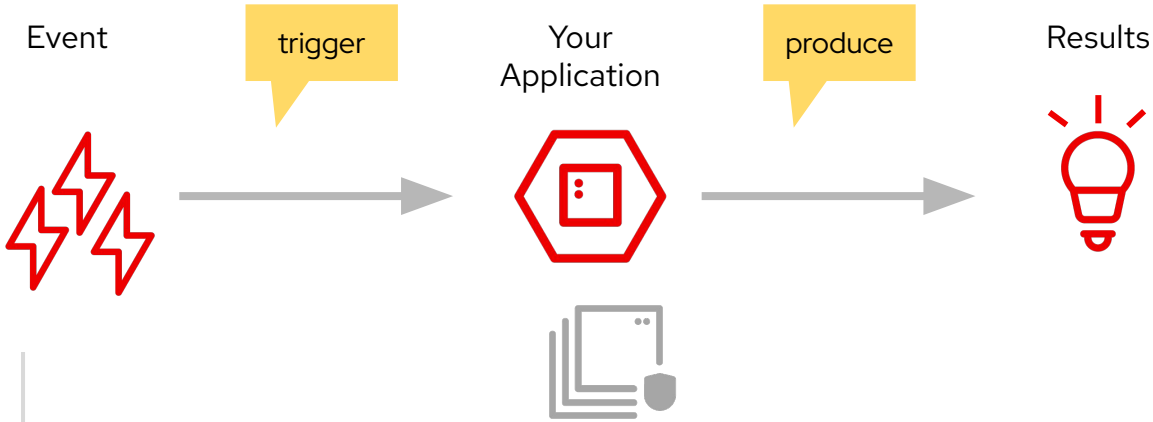
Installation experience

"Easy day 1 and even better for day 2"

- Click Install experience
- Developer & admin experience in Console
- Built-in event sources
- No external dependencies.



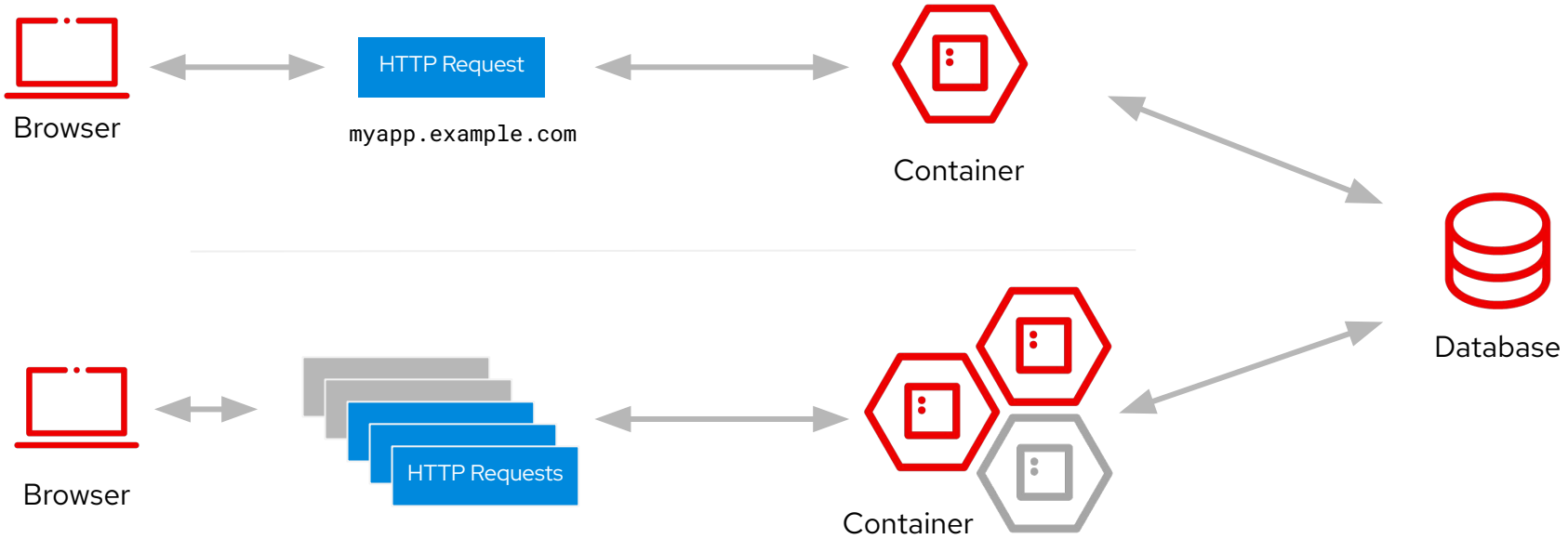
The "Serverless Pattern"



- HTTP Requests
- Kafka Messages
- Image Uploaded
- New Order
- Login from user

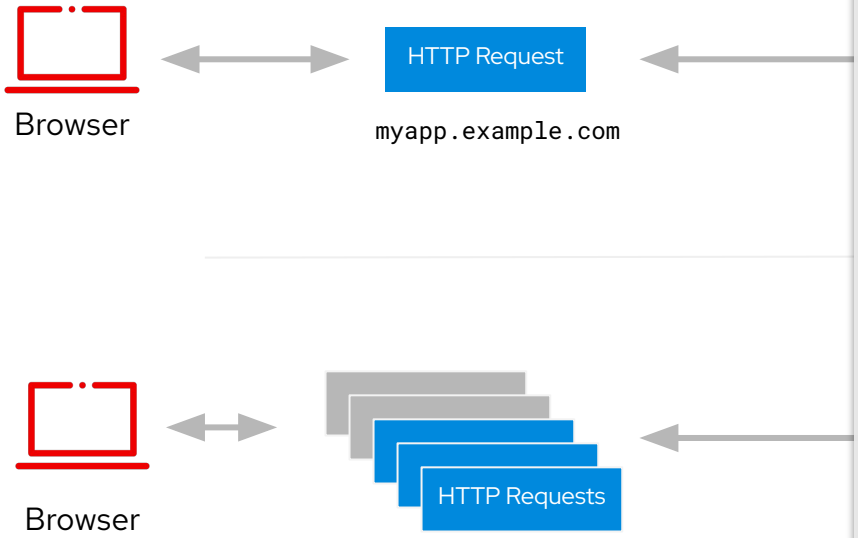
The "Serverless Pattern"

A serverless web application



The "Serverless Pattern"

A serverless web application

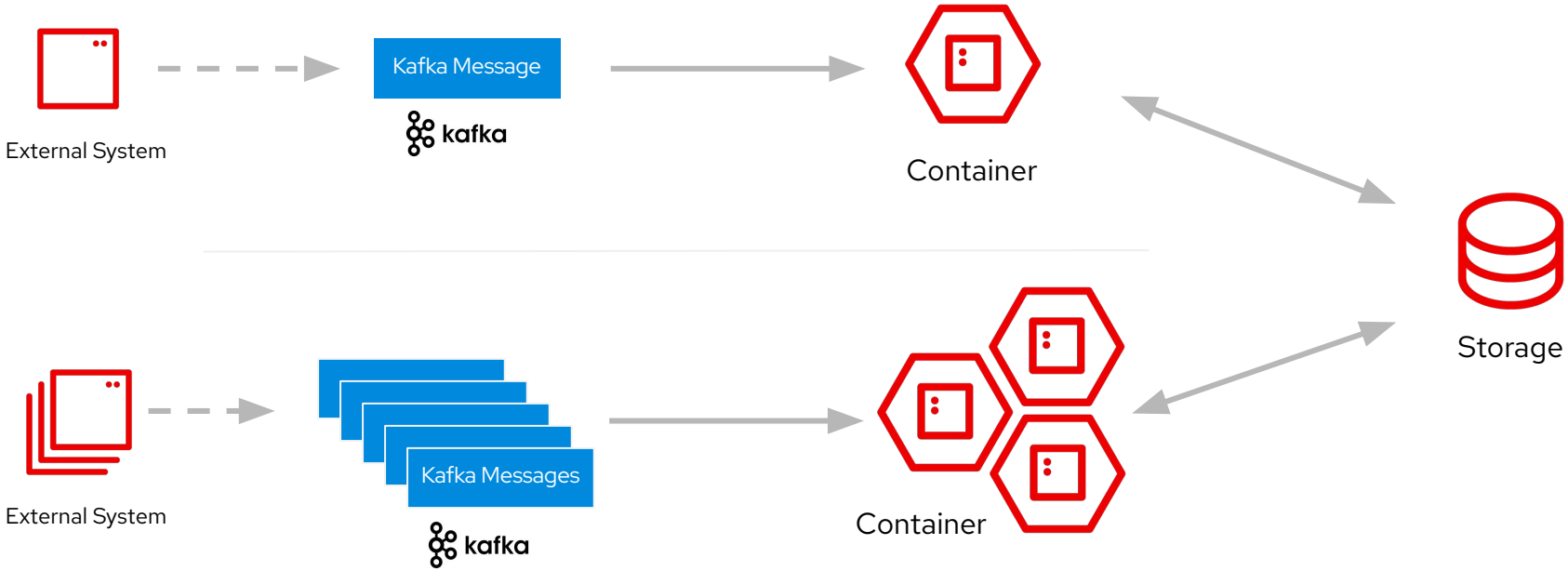


Benefits of this model:

- No need to setup auto-scaling and load balancers
  - Scale down and save resources when needed.
  - Scale up to meet the demand.
- No tickets to configure SSL for applications
- Enable Event Driven Architectures (EDA) patterns
- Enable teams to associate cost with IT
- Modernize existing applications to run as serverless containers

The "Serverless Pattern"

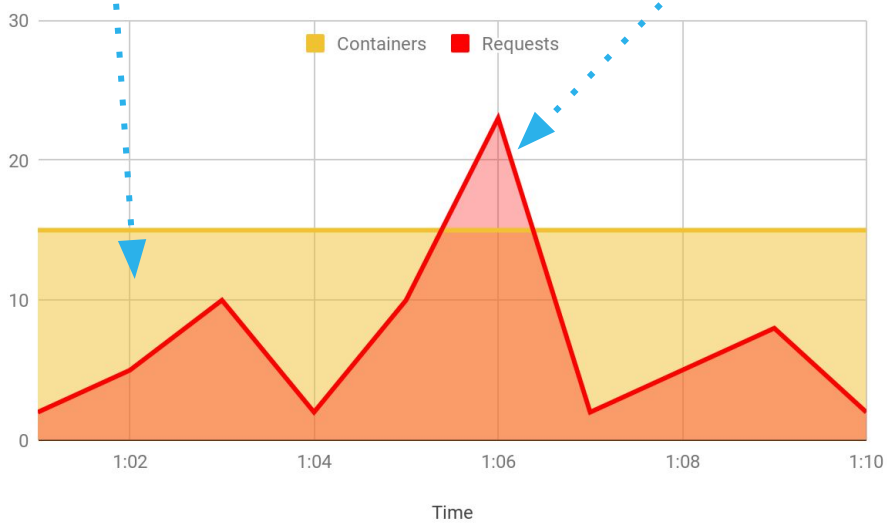
Processing a Kafka message



Serverless Operational Benefits

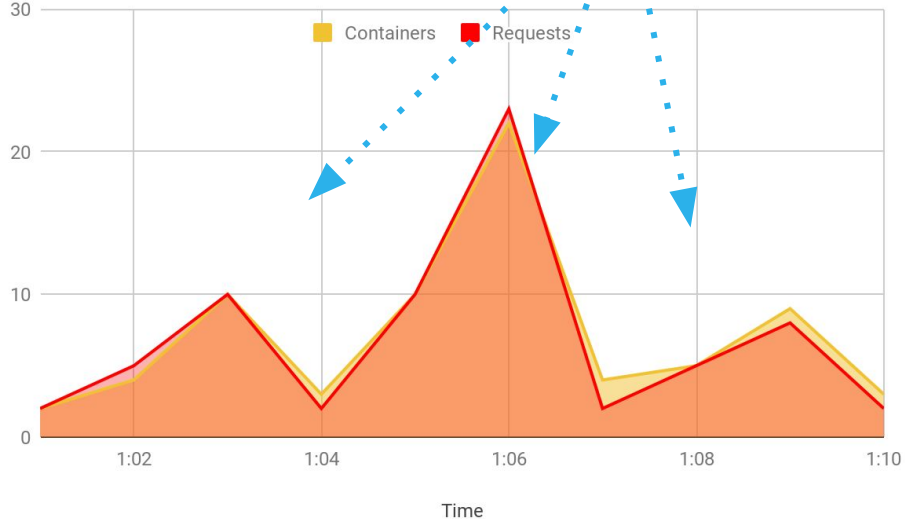
Over provisioning
Time in capacity planning
IT cost of idle resources

Under provisioning
Lost business revenue
Poor quality of service



NOT Serverless

More applications
Direct line between IT
costs & business revenue



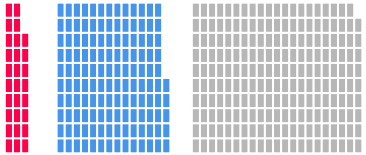
with Serverless

Choosing the Right Tool

Your Team

The Ecosystem

Performance

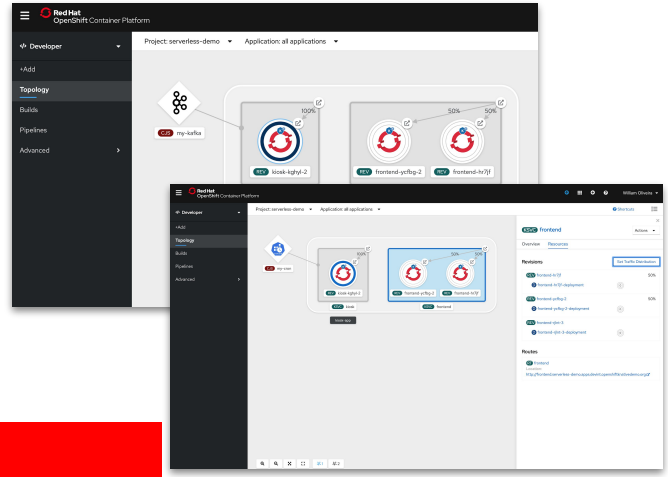


LOGIC

Developer experience

- Java
- Node
- Python
- Go
- Ruby
- Ruby on Rails
- PHP
- Perl
- .NET Core

```
$ kn service create --image=
```



CLI

UI

OpenShift Serverless

OPENSHIFT

Red Hat Enterprise Linux CoreOS

Physical

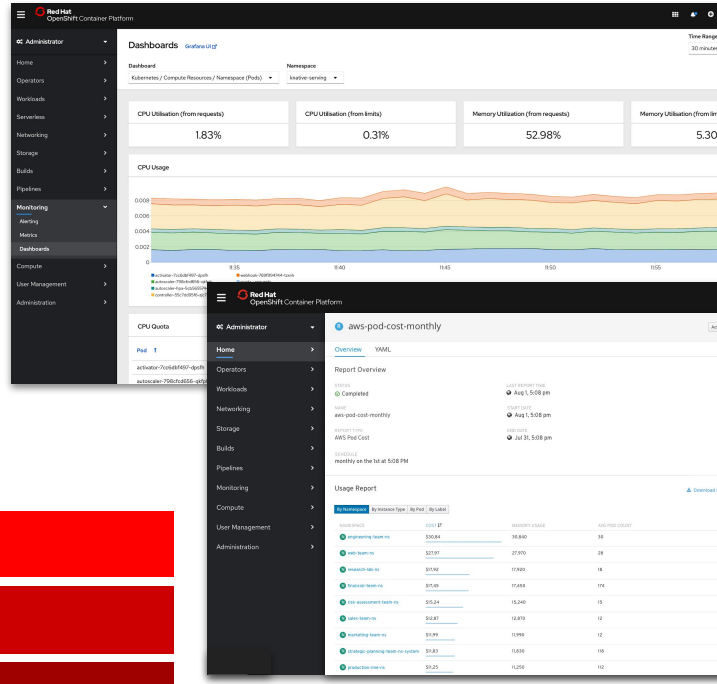
Virtual

Private cloud

Public cloud

Admin experience

- Monitoring, Metering and Logging
- Disconnected install support (air-gapped)
- Egress proxy with TLS support
- Over the air updates and patches



UI

OpenShift Serverless

OPENSHIFT

Red Hat Enterprise Linux CoreOS

Physical | Virtual | Private cloud | Public cloud

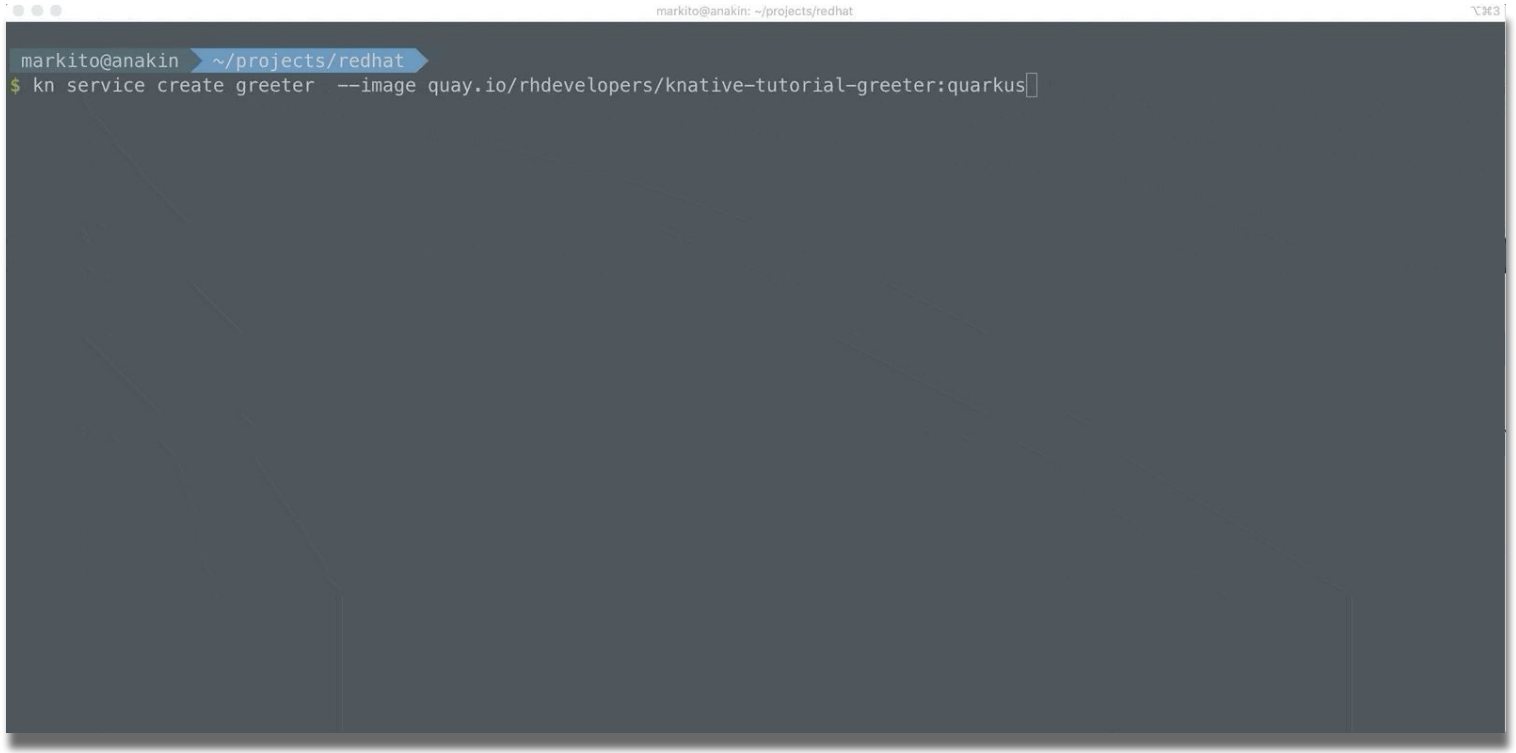
Command Line Experience

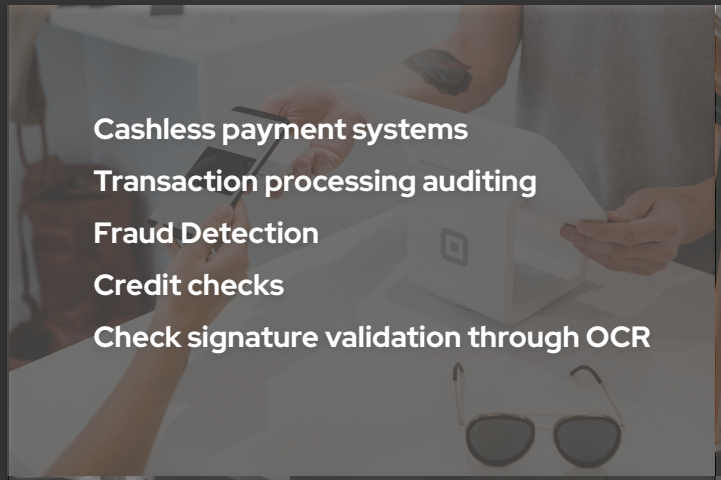
```
$ kn service create myService --image=[registry/mycontainer:v1] --min-scale=1 --max-scale=100
```

```
$ kn service update myService --traffic myService-rev1=50,myService-rev2=50
```

```
$ kn source cronjob create my-cron --schedule "* * * * */1" --data "ping" --sink svc:myService
```

Hello World with Quarkus!





Cashless payment systems
Transaction processing auditing
Fraud Detection
Credit checks
Check signature validation through OCR

Product thumbnail generation
Chatbots and CRM functions
Marketing Campaign notifications
Sales Audit
Content Push

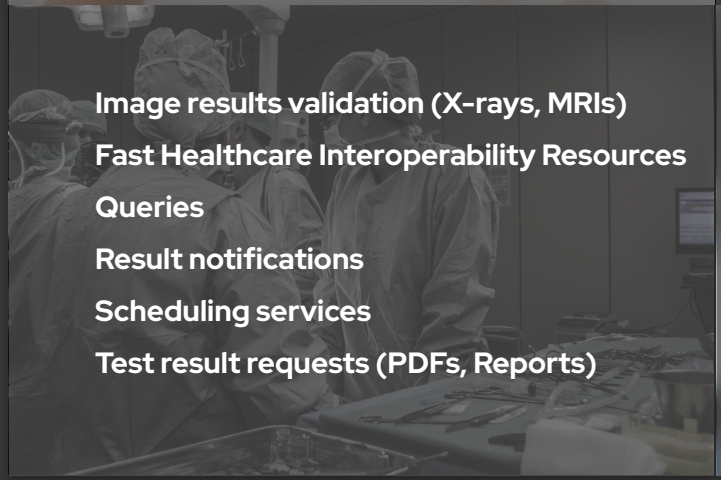
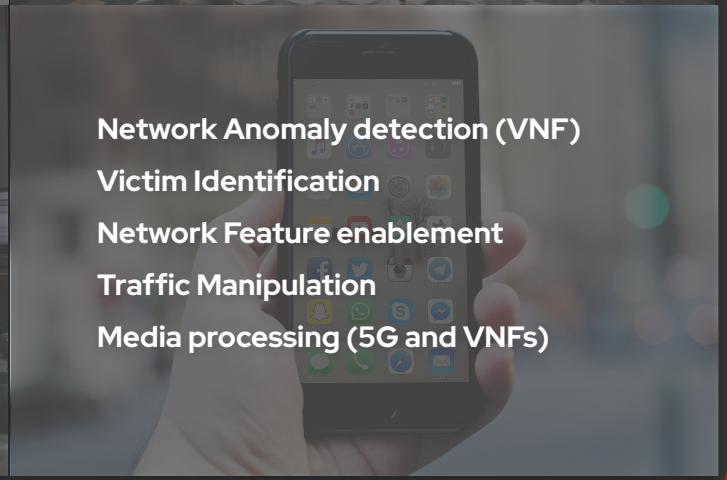
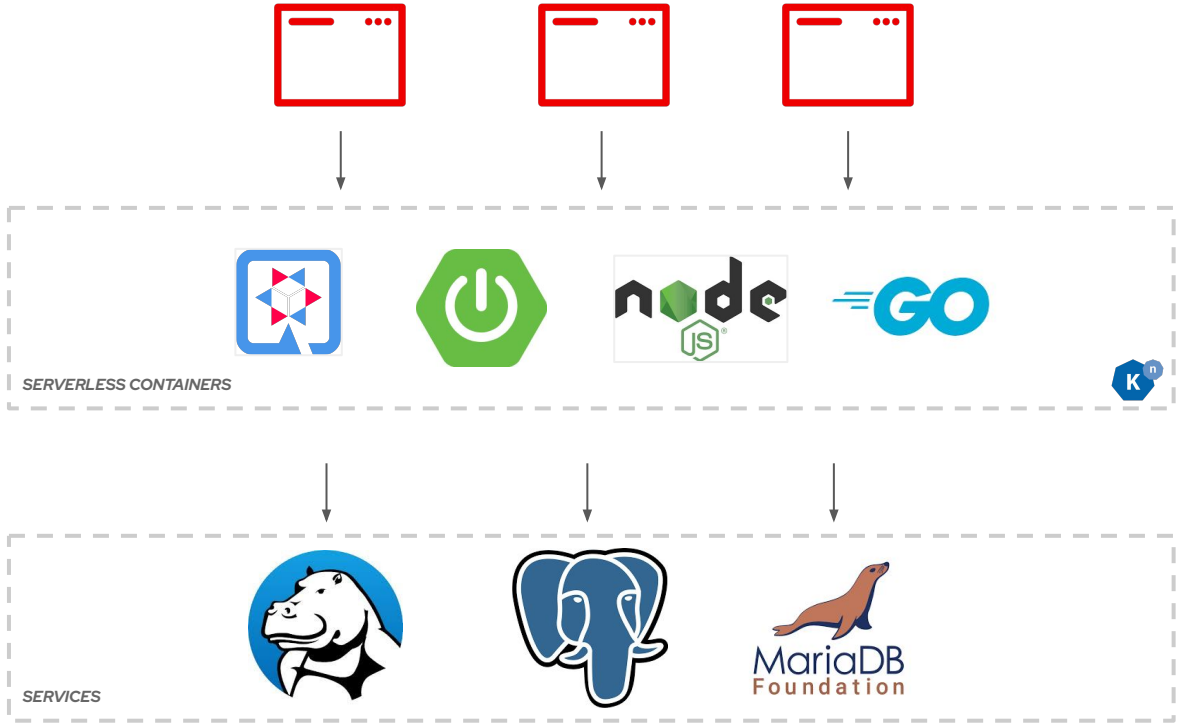


Image results validation (X-rays, MRIs)
Fast Healthcare Interoperability Resources Queries
Result notifications
Scheduling services
Test result requests (PDFs, Reports)



Network Anomaly detection (VNF)
Victim Identification
Network Feature enablement
Traffic Manipulation
Media processing (5G and VNFs)

Web Applications and APIs



Language or runtime of your choice:

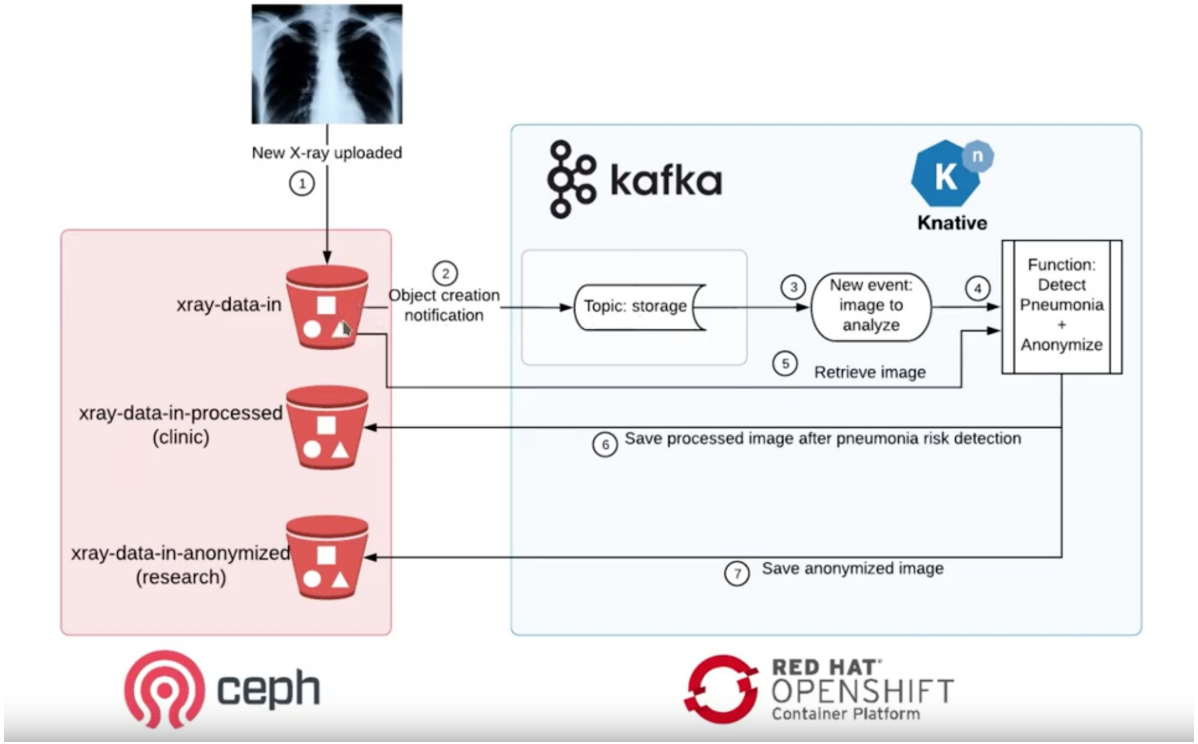
OpenJDK
python
django
VERT.X
JS
RAILS

Red Hat

Data Transformation

Common Use cases

- Image analysis* (medical, AI/ML, media)
- Video format transcoding
- File type conversion (financial, medical)
- Data extraction
- Lightweight Data Transformation
- Invoice Generation



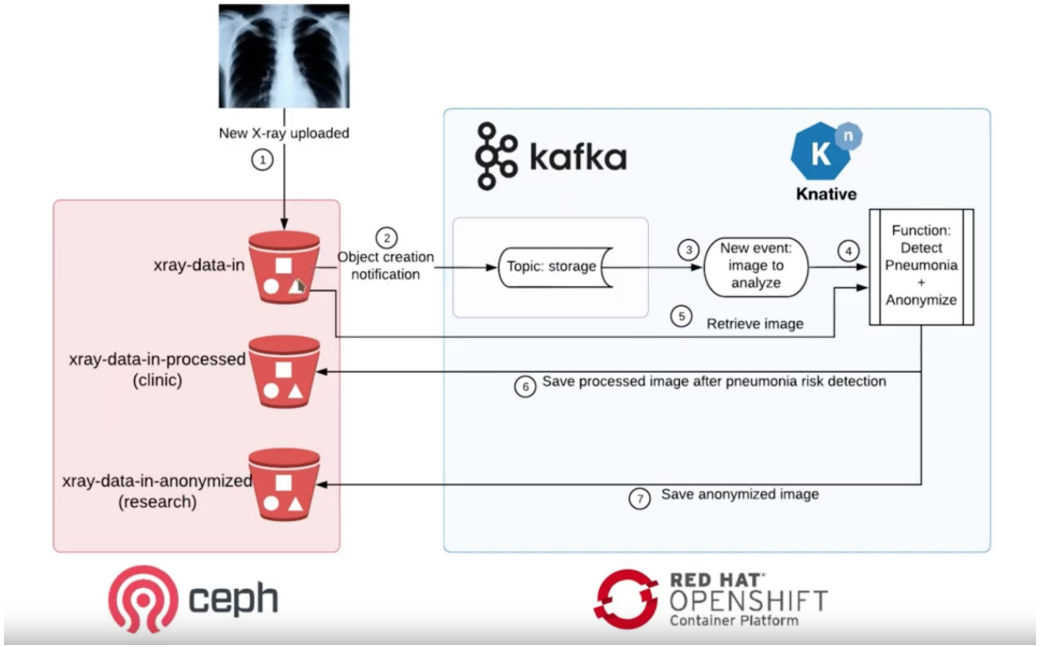
* Article: https://medium.com/analytics-vidhya/automated-data-pipeline-using-ceph-notifications-and-kserving-5e1e9b996661

Ceph + Kafka + Serverless

"Automated AI/ML Data Pipelines"

Common Use cases

- Image analysis* (medical, AI/ML, media)
- Video format transcoding
- File type conversion (financial, medical)
- Data extraction
- Data Transformation
- Invoice Generation



TensorFlow

https://medium.com/analytics-vidhya/automated-data-pipeline-using-ceph-notifications-and-kserving-5e1e9b996661

Red Hat

Building on OpenShift Serverless with Red Hat Services

Connected Services
How Knative services interact with the outside world.

API Gateway
Next gen APIs still require management.

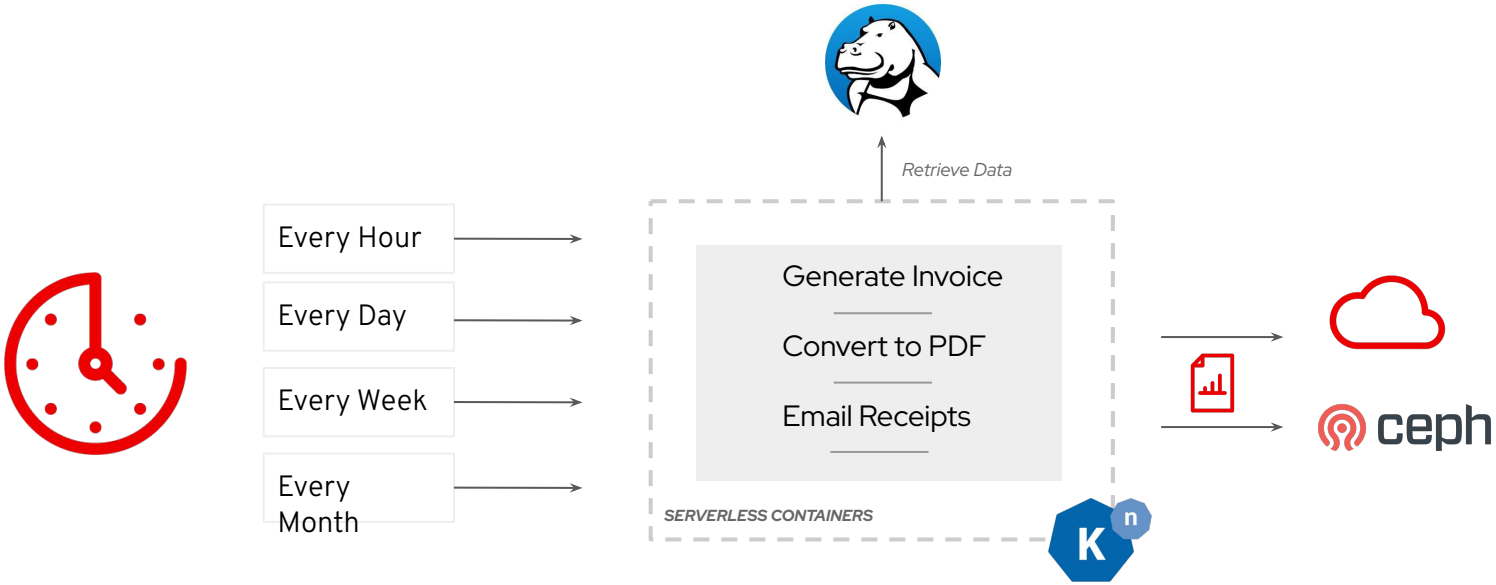
Service Orchestrator
Composing multiple services together into an application.

Implementing Services
Functions, languages, and the vagaries of cold starts.

Event Streaming
All modern architectures need some Kafka.

The Dirty Word in Serverless
Yep, you still need state to handle long-lived orchestration.

Scheduled Apps Cron Jobs



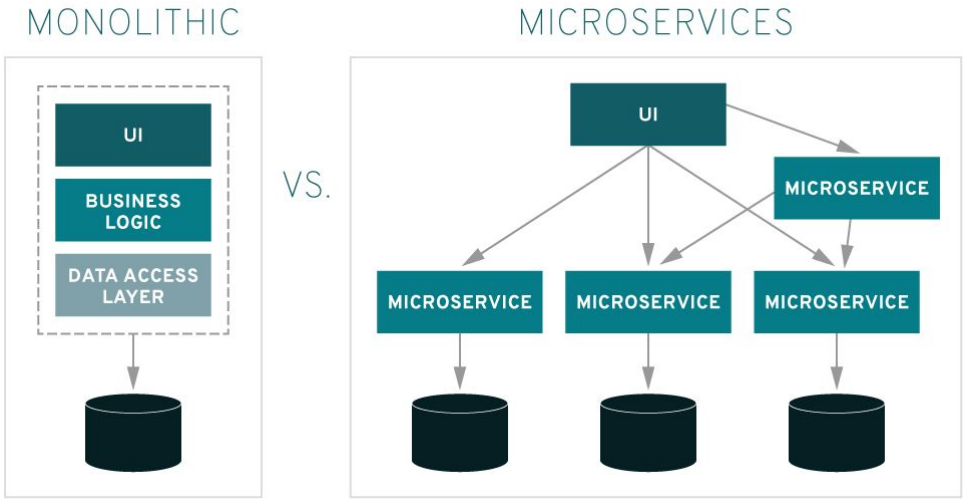
Red Hat

ServiceMesh / Istio

What are Microservices?

an architectural style that structures an application as a collection of services

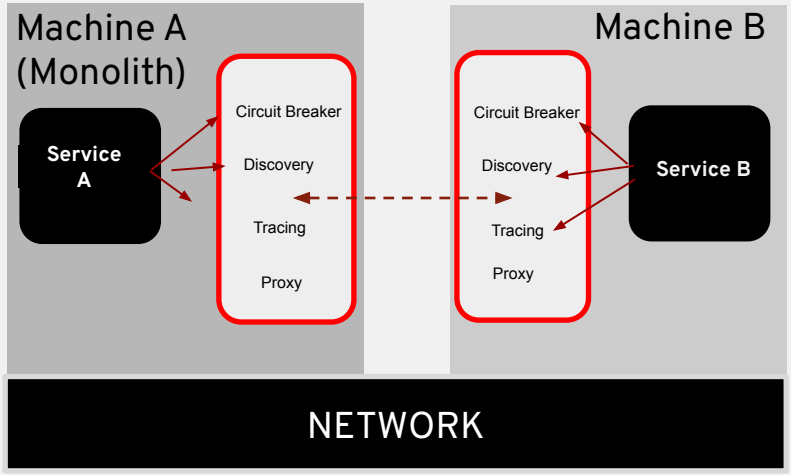
- Single purpose
- Independently deployable
- Have their context bound to a biz domain
- Owned by a small team
- Often stateless



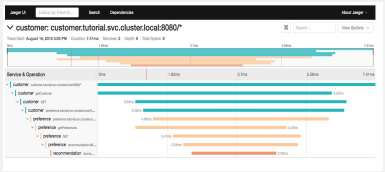
Red Hat

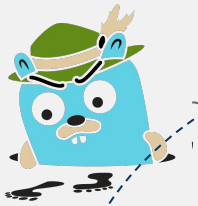
OVERVIEW

WHAT IS A SERVICEMESH ?



SERVICE MESH ECOSYSTEM





Jaeger

Secure

Istio

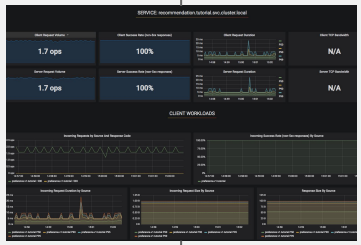
Observe

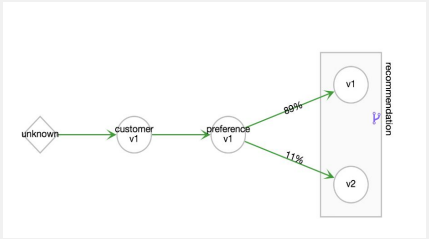
Connect

Control

Prometheus

Observe





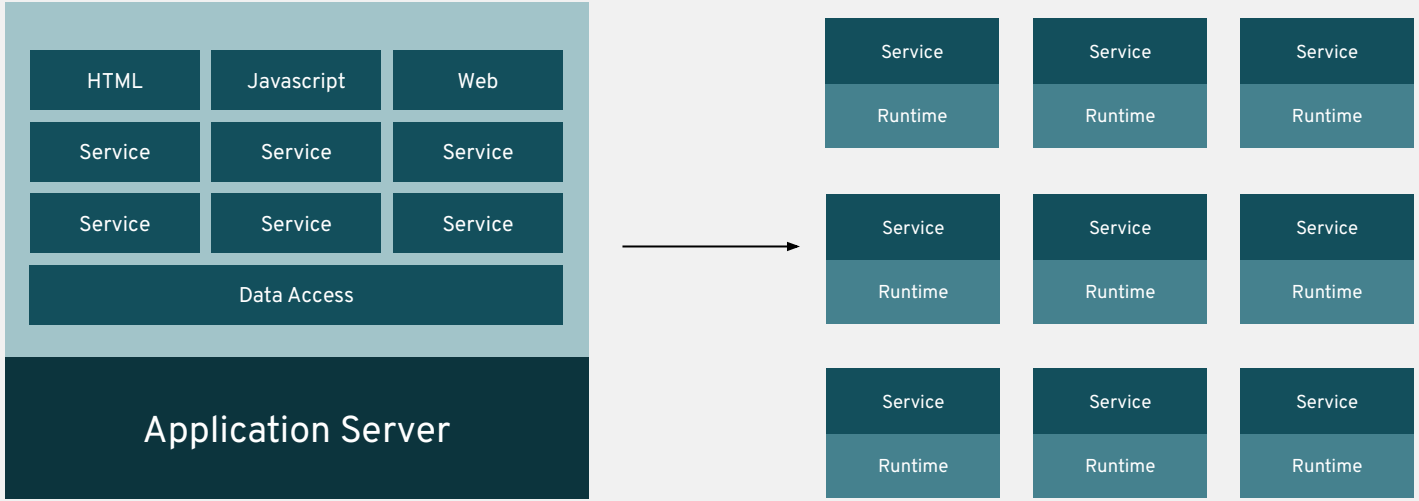
kiali

Kiali

Grafana

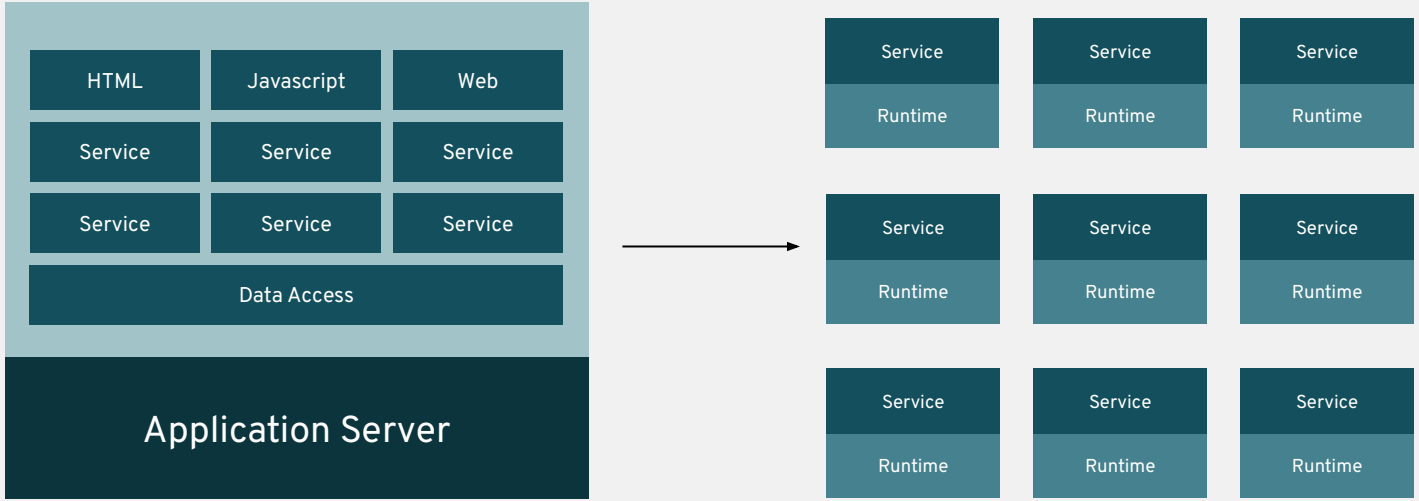
UNDER THE HOOD

MICROSERVICES ARCHITECTURE

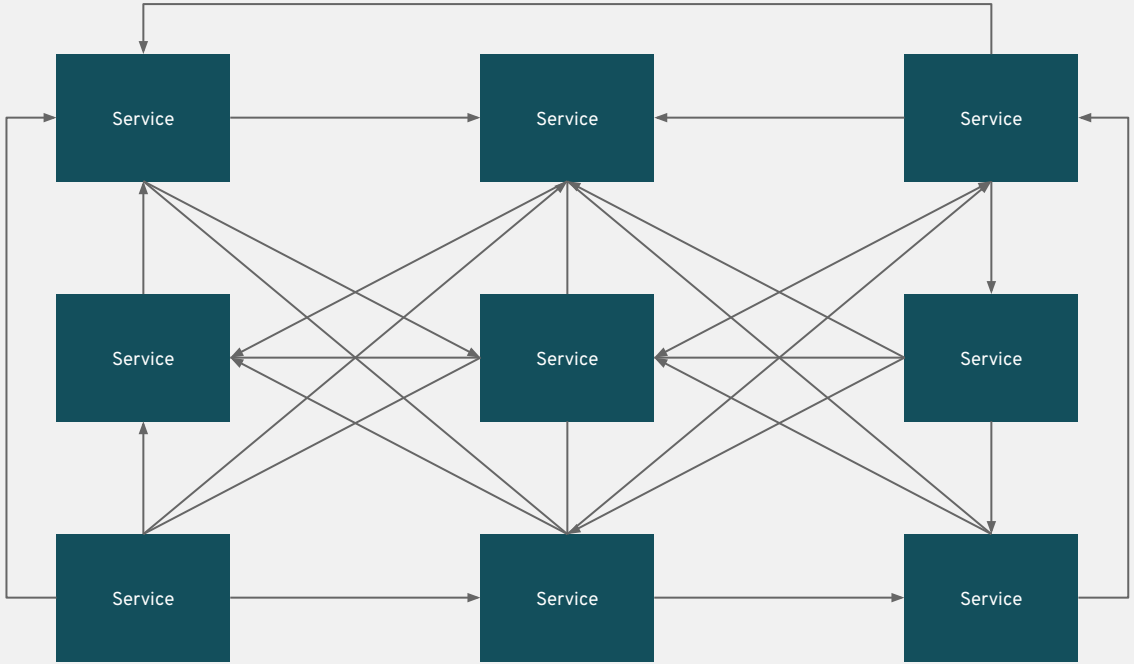


~~MICROSERVICES~~ ARCHITECTURE

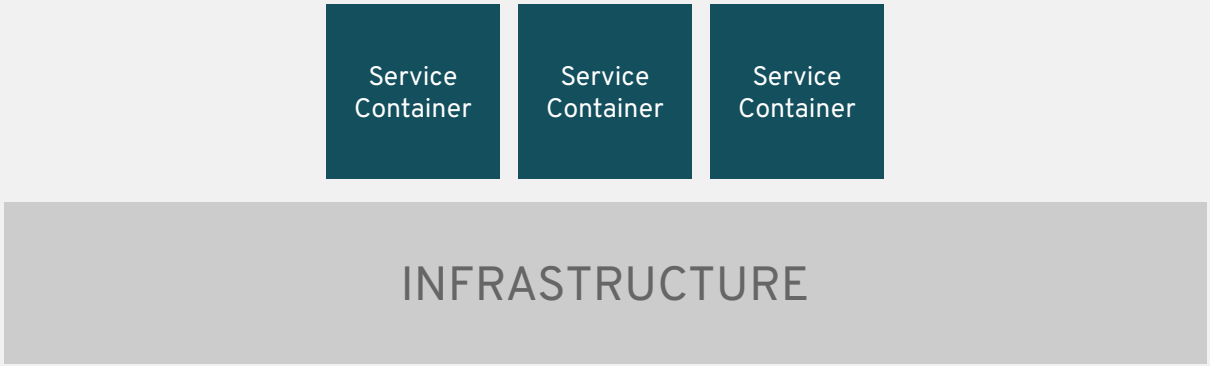
DISTRIBUTED



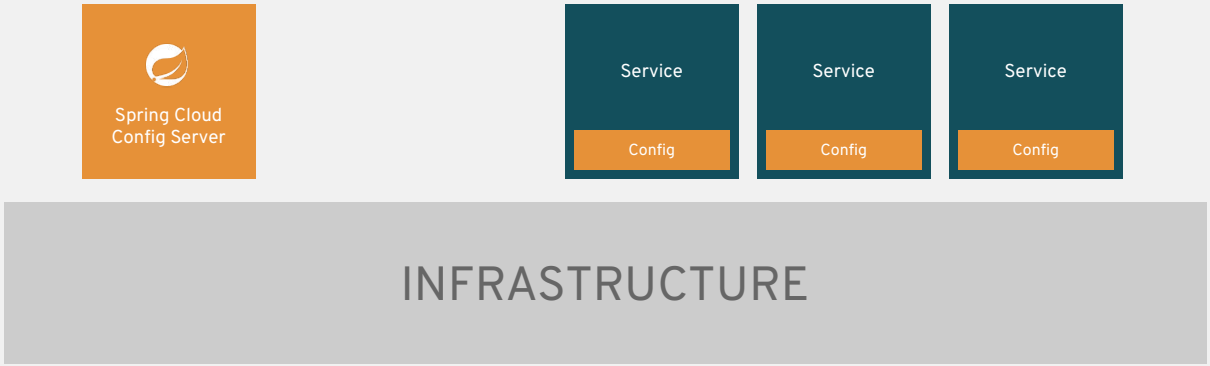
DISTRIBUTED ARCHITECTURE



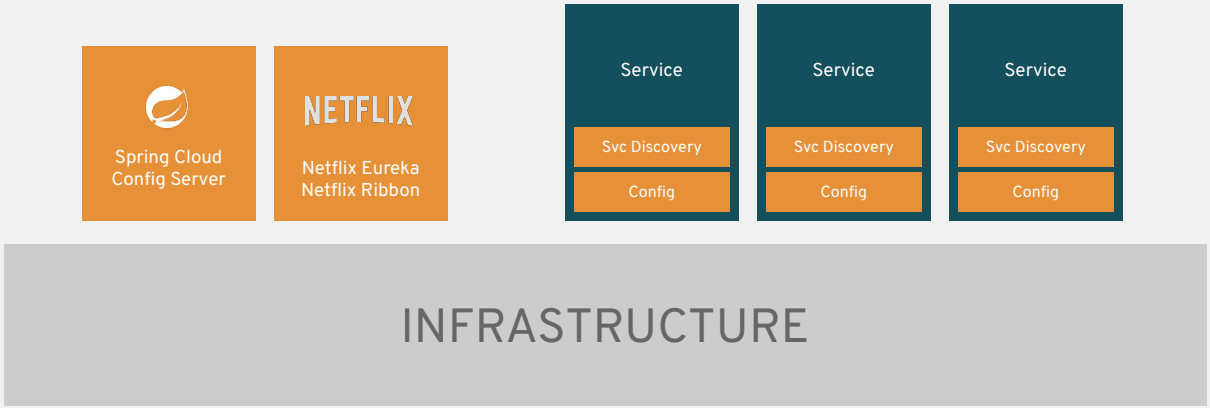
DEPLOYMENT


Service Container
Service Container
Service Container
INFRASTRUCTURE

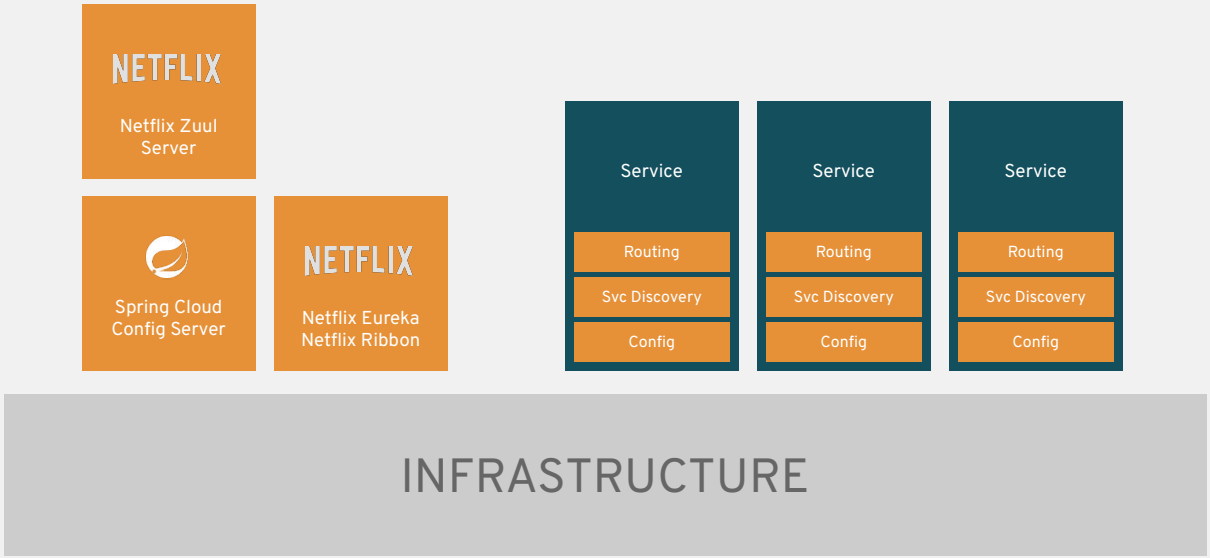
CONFIGURATION



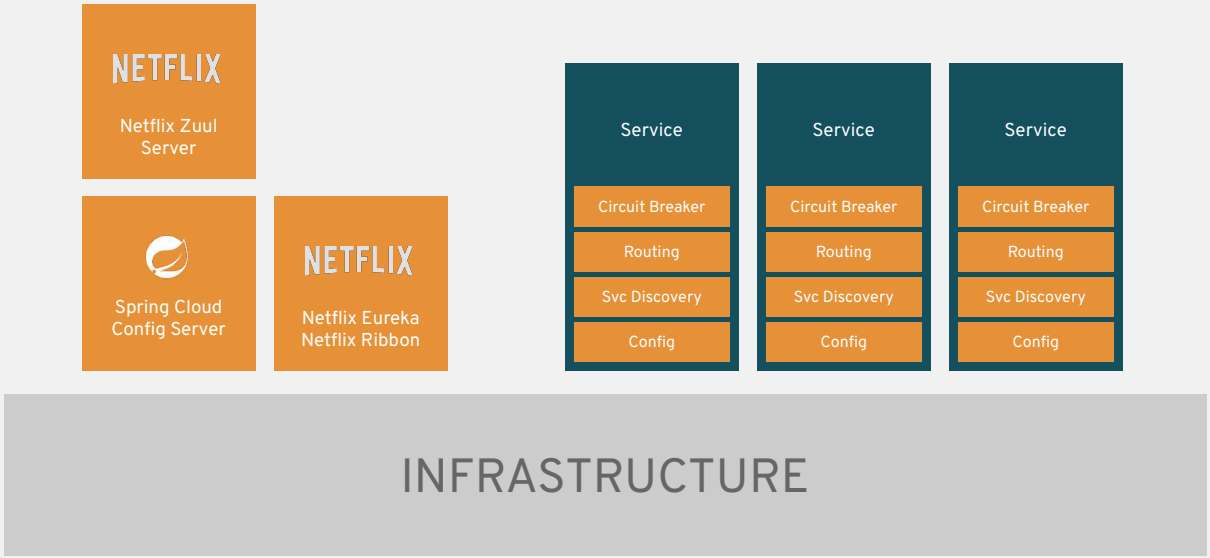
SERVICE DISCOVERY



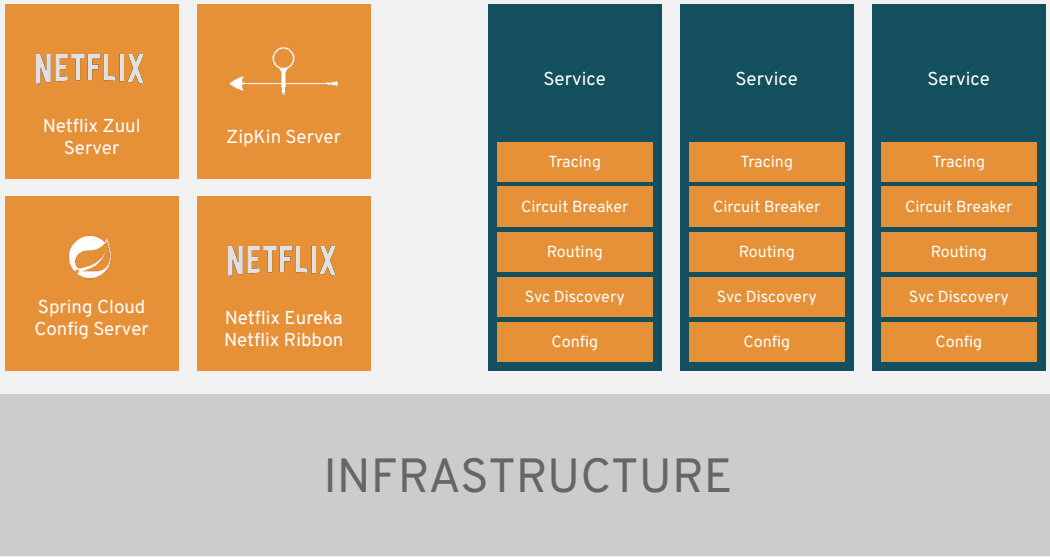
DYNAMIC ROUTING



FAULT TOLERANCE



TRACING AND VISIBILITY



WHAT ABOUT...?

POLYGLOT APPS

EXISTING APPS

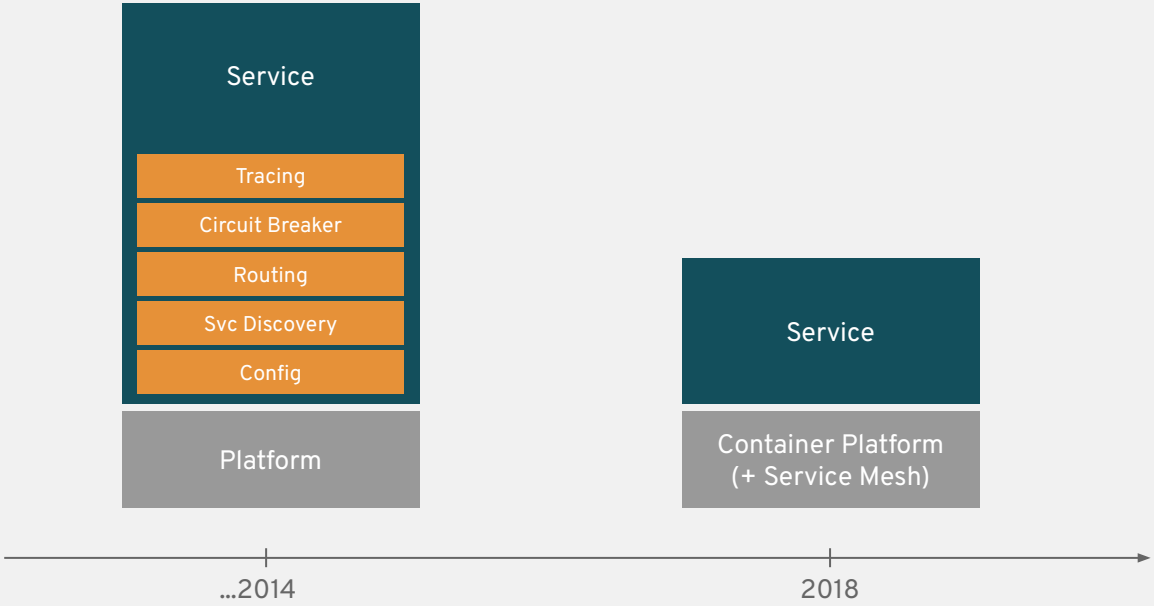
THERE SHOULD BE A BETTER WAY

ADDRESS THE COMPLEXITY IN THE INFRASTRUCTURE

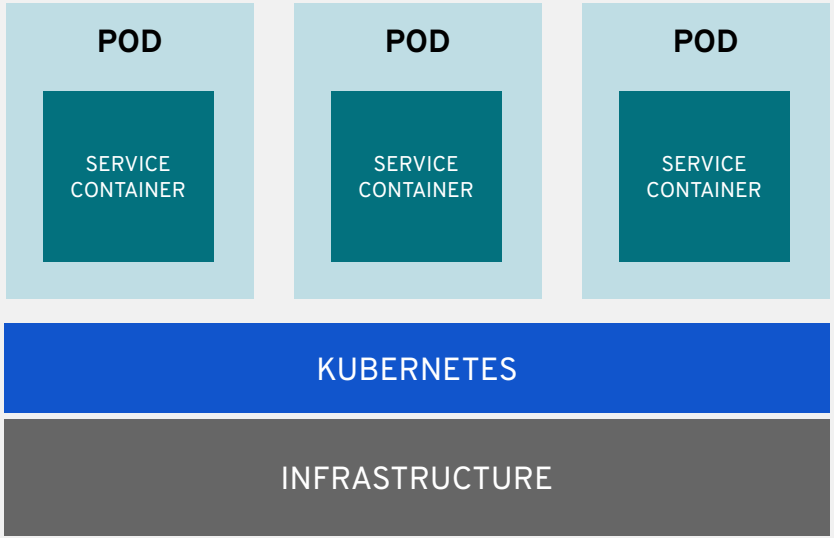
SERVICE MESH

A dedicated infrastructure layer for service-to-service communications

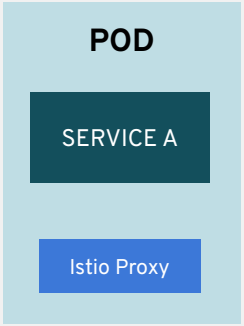
MICROSERVICES EVOLUTION



AUTOMATING CONTAINER DEPLOYMENT



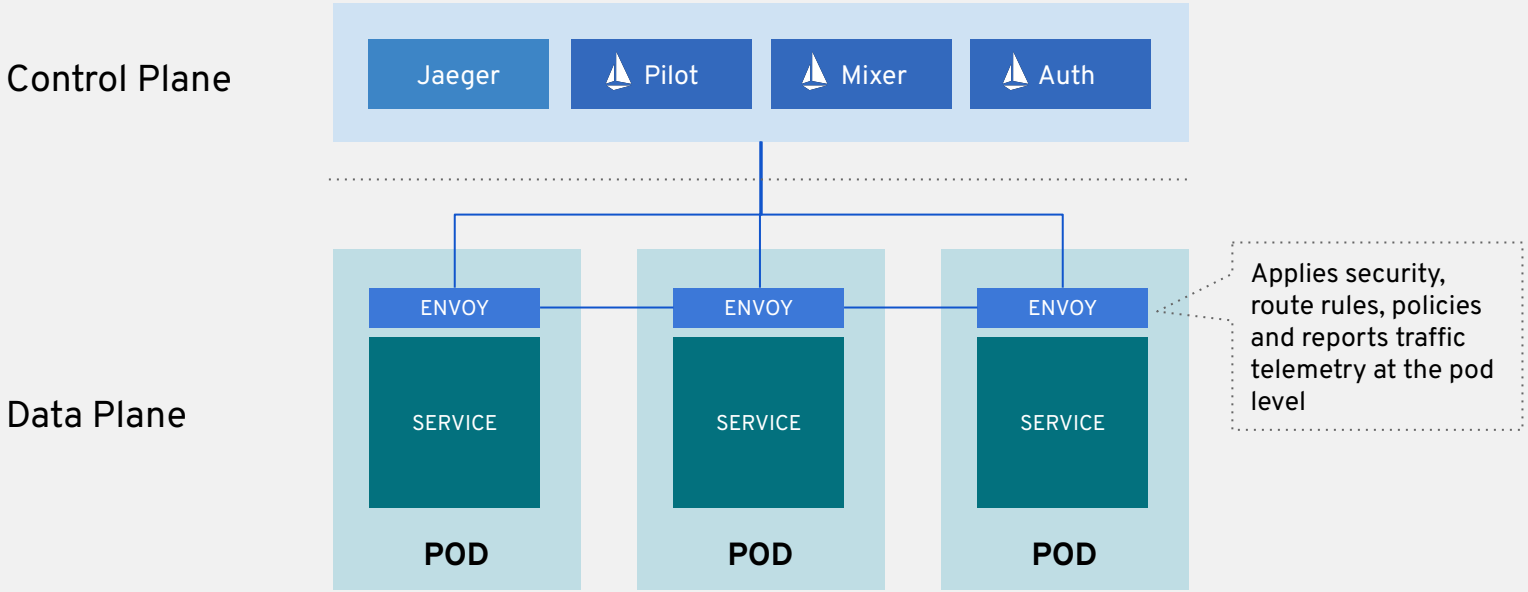
SIDECARS



- Two or more containers deployed to same pod
- Share
  - Same
    - Namespace
    - Pod IP
  - Shared lifecycle
- Used to enhance the co-located containers
- Istio Proxy (L7 Proxy)
  - Proxy all network traffic in and out of the app container

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

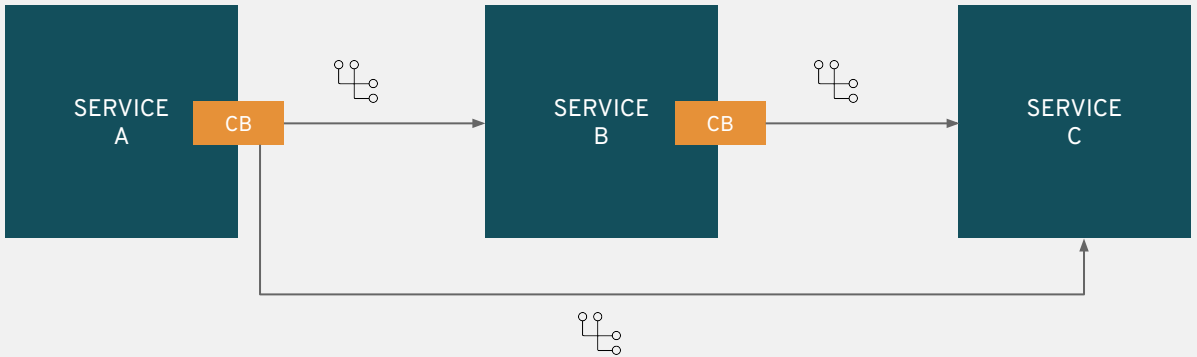
SERVICE MESH ARCHITECTURE



MAJOR FUNCTIONALITY

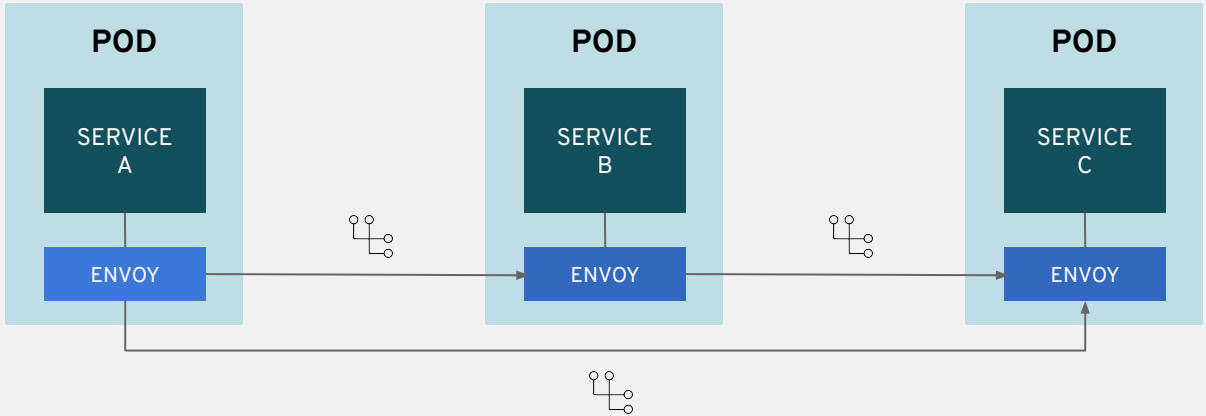
FAULT TOLERANCE

CIRCUIT BREAKERS WITHOUT ISTIO



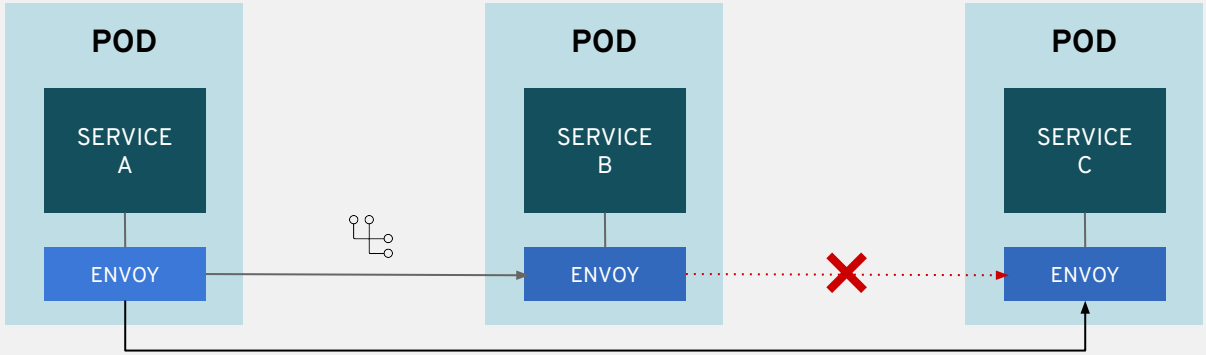
coupled to the service code

CIRCUIT BREAKERS WITH ISTIO



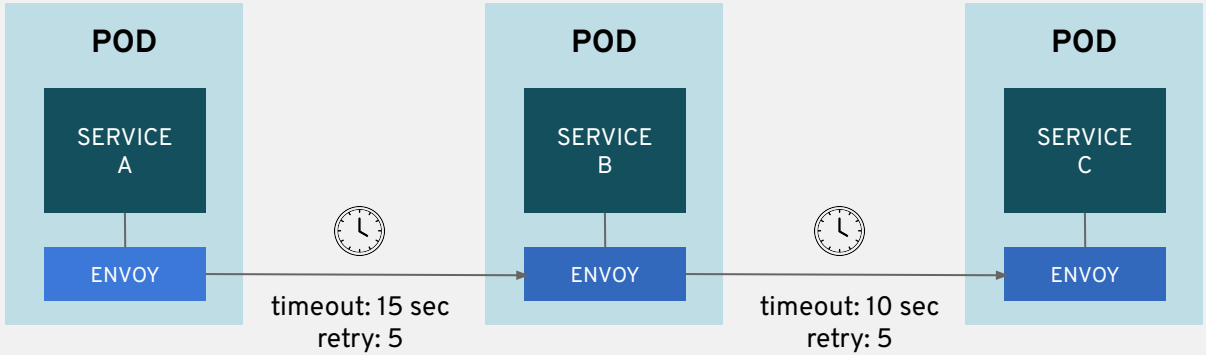
transparent to the services

CIRCUIT BREAKERS WITH ISTIO



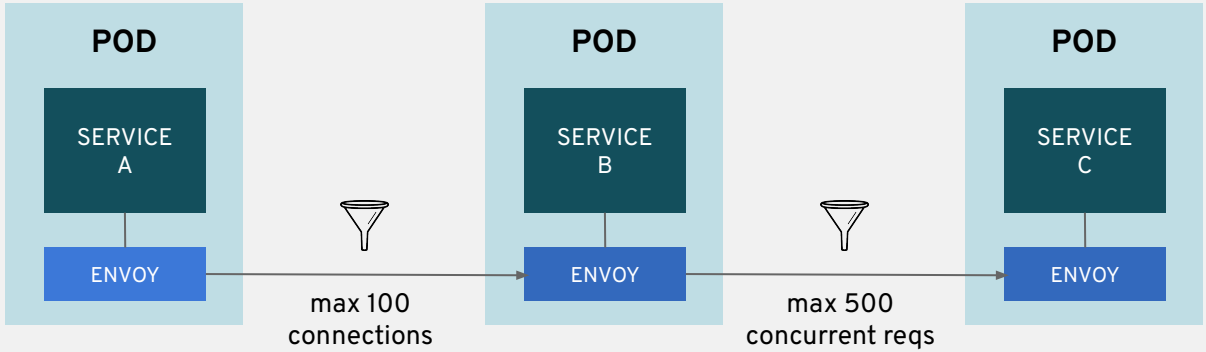
improved response time with global circuit status

TIMEOUTS AND RETRIES WITH ISTIO



configure timeouts and retries, transparent to the services

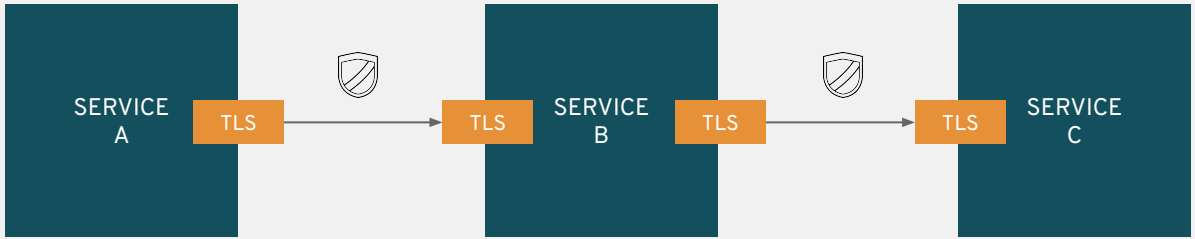
RATE LIMITING WITH ISTIO



limit invocation rates, transparent to the services

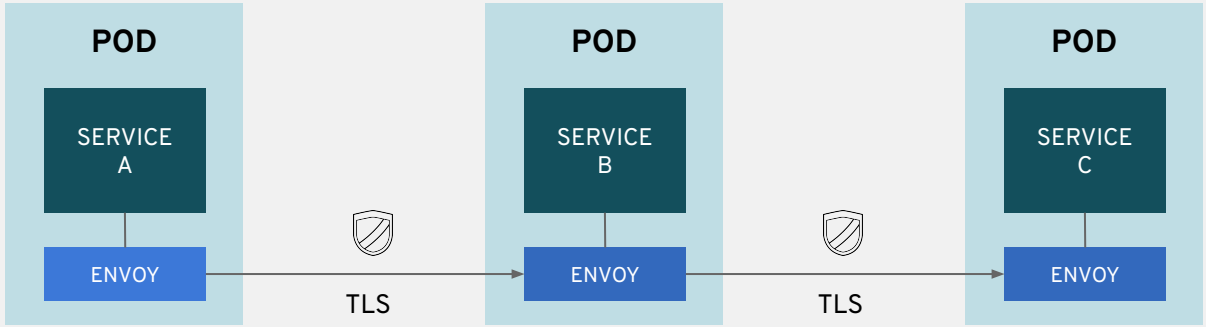
SERVICE SECURITY

SECURE COMMUNICATION WITHOUT ISTIO



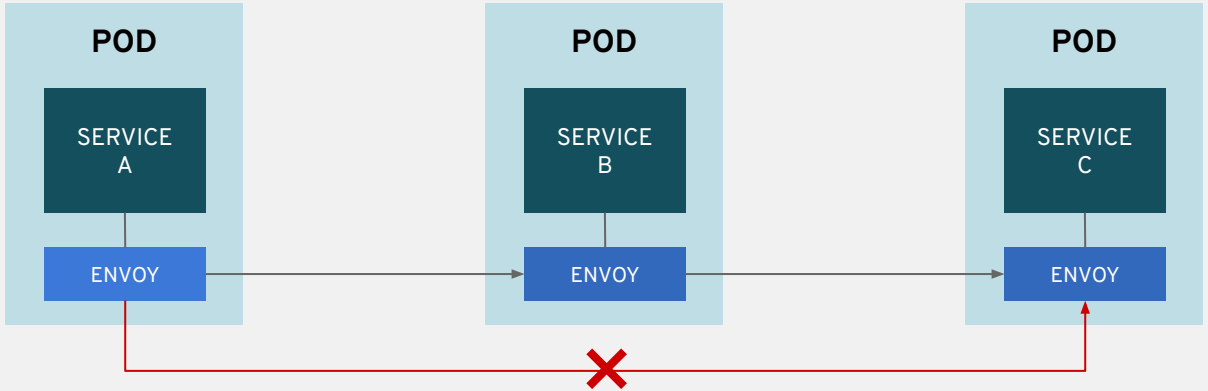
coupled to the service code

SECURE COMMUNICATION WITH ISTIO



mutual TLS authentication, transparent to the services

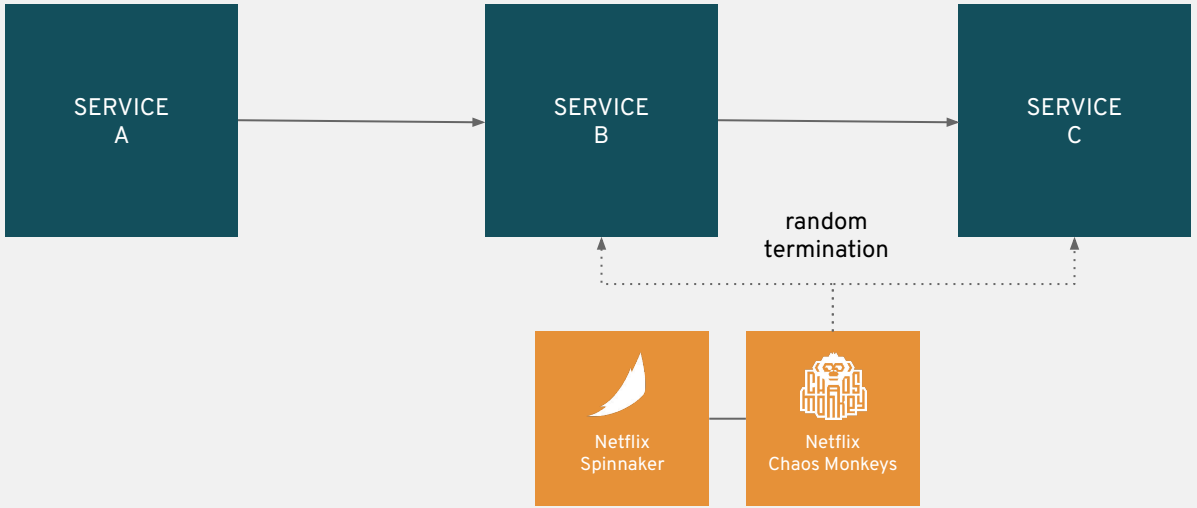
CONTROL SERVICE ACCESS WITH ISTIO



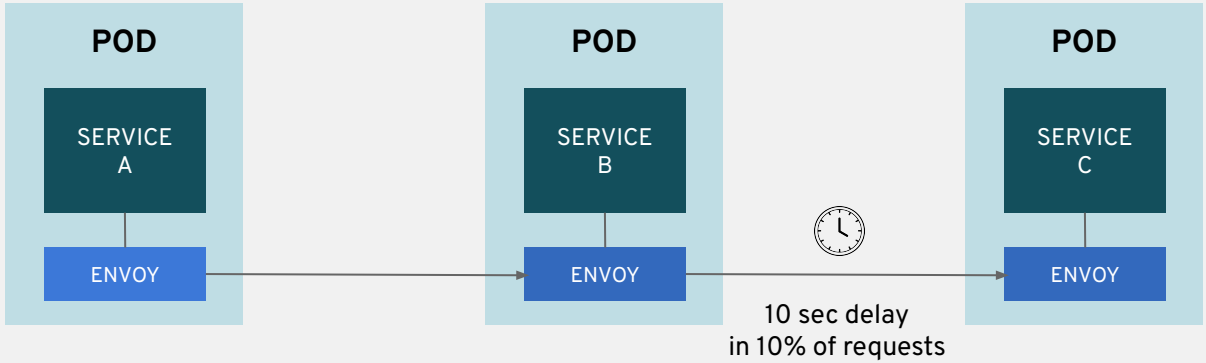
control the service access flow, transparent to the services

CHAOS ENGINEERING

CHAOS ENGINEERING WITHOUT ISTIO

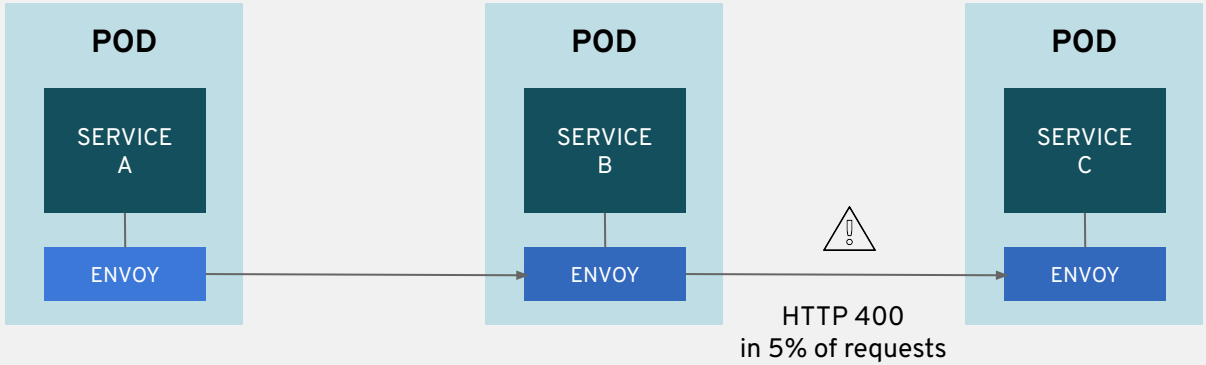


CHAOS ENGINEERING WITH ISTIO



inject delays, transparent to the services

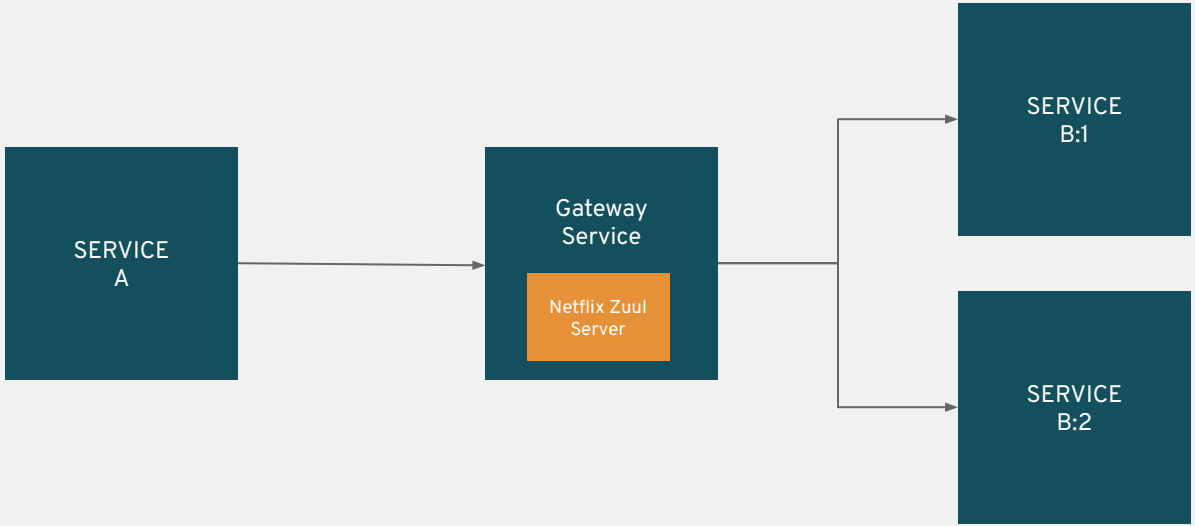
CHAOS ENGINEERING WITH ISTIO



inject protocol-specific errors, transparent to the services

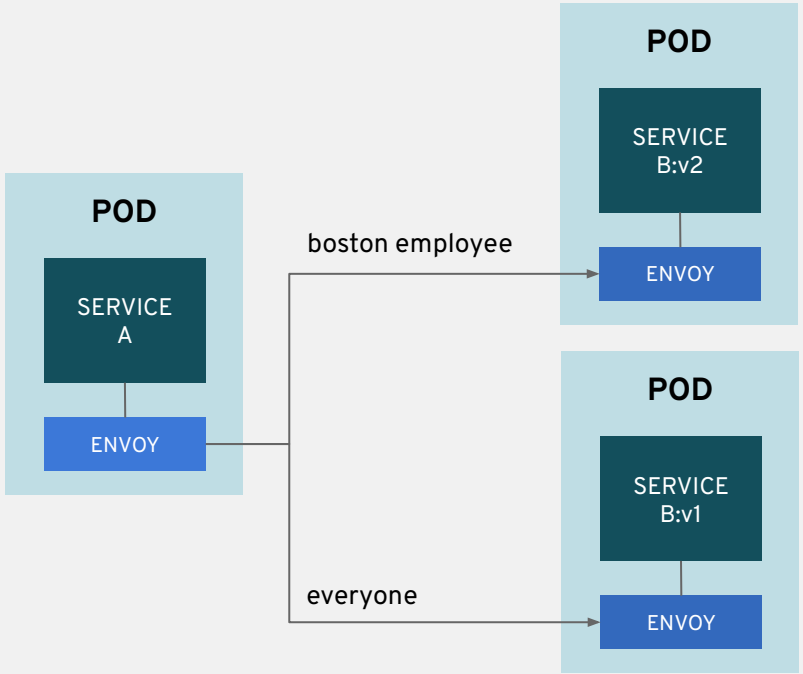
DYNAMIC ROUTING

DYNAMIC ROUTING WITHOUT ISTIO

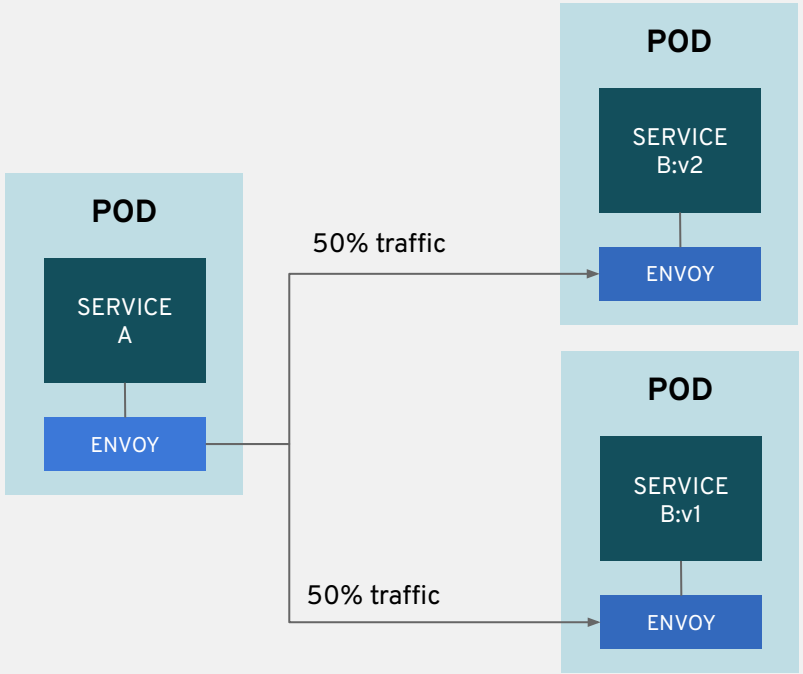


custom code to enable dynamic routing

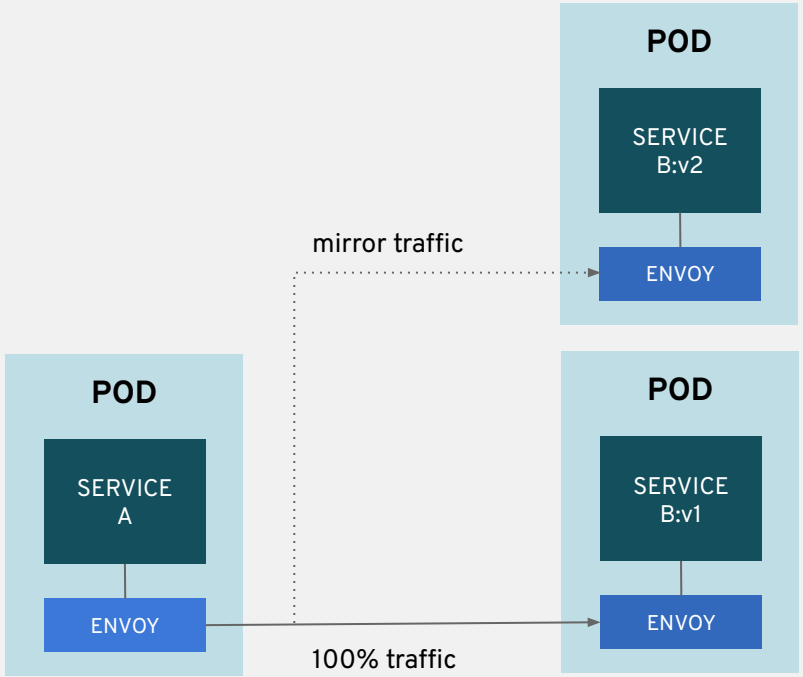
CANARY DEPLOYMENT WITH ISTIO



A/B DEPLOYMENT WITH ISTIO

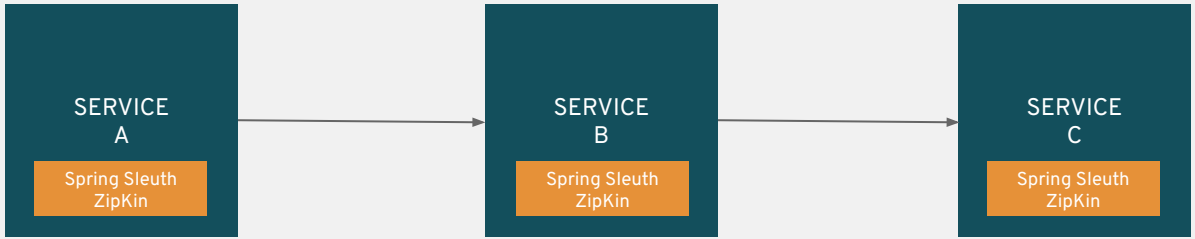


DARK LAUNCHES WITH ISTIO



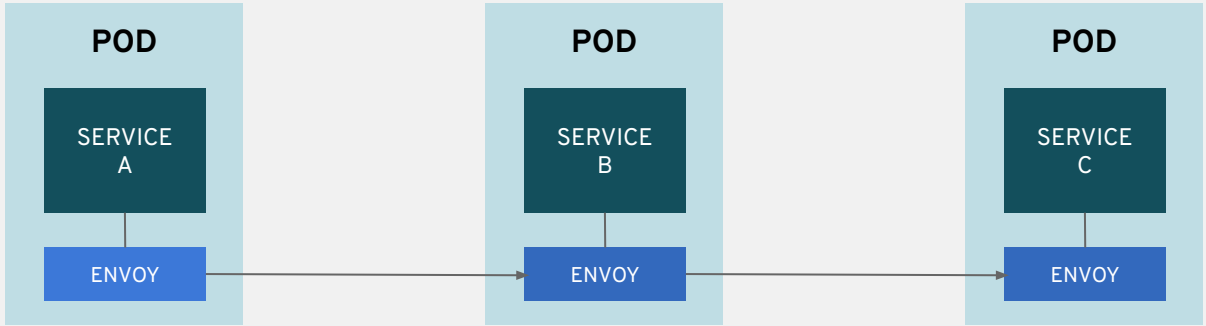
DISTRIBUTED TRACING (JAEGER)

DISTRIBUTED TRACING WITHOUT ISTIO



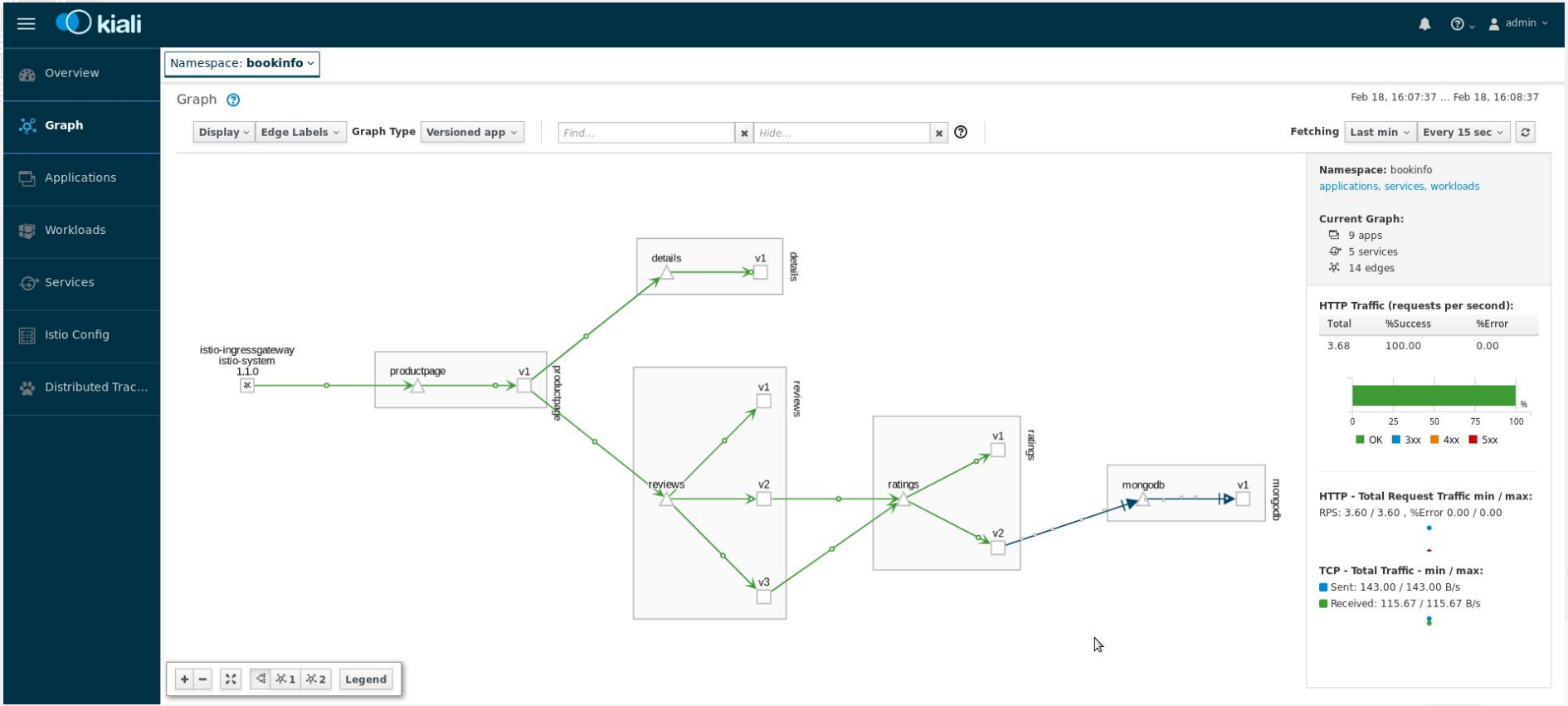
code to enable dynamic tracing

DISTRIBUTED TRACING WITH ISTIO & JAEGER

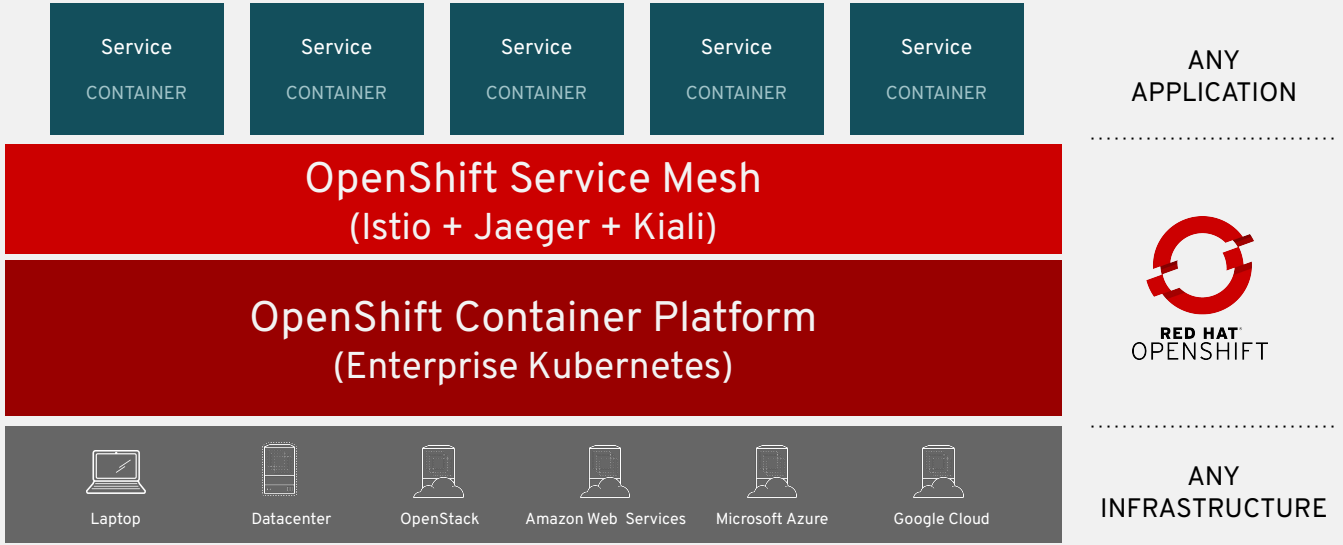


discovers service relationships and process times, transparent to the services

SERVICE MESH OBSERVABILITY (KIALI)


kiali
admin
Overview
Graph
Applications
Workloads
Services
Istio Config
Distributed Trac...
Namespace: bookinfo
Graph
Feb 18, 16:07:37 ... Feb 18, 16:08:37
Display
Edge Labels
Graph Type
Versioned app
Find...
Hide...
Fetching
Last min
Every 15 sec
istio-ingressgateway
istio-system
1.1.0
productpage
v1
details
reviews
v2
v3
ratings
mongodb
Namespace: bookinfo
applications, services, workloads
Current Graph:
9 apps
5 services
14 edges
HTTP Traffic (requests per second):
Total
%Success
%Error
3.68
100.00
0.00
0
25
50
75
100
%
OK
3xx
4xx
5xx
HTTP - Total Request Traffic min / max:
RPS: 3.60 / 3.60 , %Error 0.00 / 0.00
TCP - Total Traffic - min / max:
Sent: 143.00 / 143.00 B/s
Received: 115.67 / 115.67 B/s
Legend

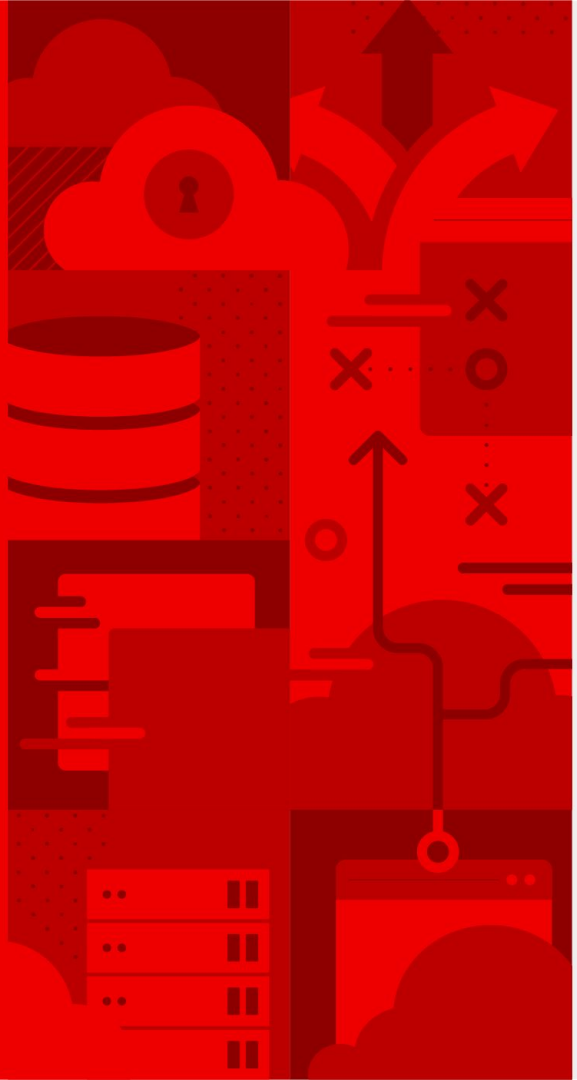
DISTRIBUTED SERVICES PLATFORM



Istio - Caveats

- Caveats
  - Be careful, istio is NOT a ServiceBus
  - Be careful, istio is NOT an API Manager
  - Be careful, istio is NOT meant to be a router
  - Be careful, istio is NOT a BPM tool

- Istio is meant to help solving some challenges
  - Centrally encryption / management of internal service communication
  - Distributed networking / communication of services
  - Increase Fault tolerance
  - Fault simulation



Thank you

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat