Microservice 4.0 Journey

From Spring NetFlix OSS to Istio Service Mesh and Serverless

Daniel Oh / DevOps Evangelist

Open Source Summit Japan 2018

About All of You

Straw Poll Time!

- How many of you are in charge of enterprise developer and application architect?
- How many of you have developed Microservices application based Spring Boot?
- How many of you have deployed Microservices app through containers in development?
- How many of you have heard about service mesh and istio before?
- How many of you have fingers on keyboard, played with service mesh via Istio?
- How many of you have deployed service mesh with istio in production?
- How many of you have developed serverless or FaaS(function as a service) in development?
- And so on and so forth

About Me

Daniel Oh

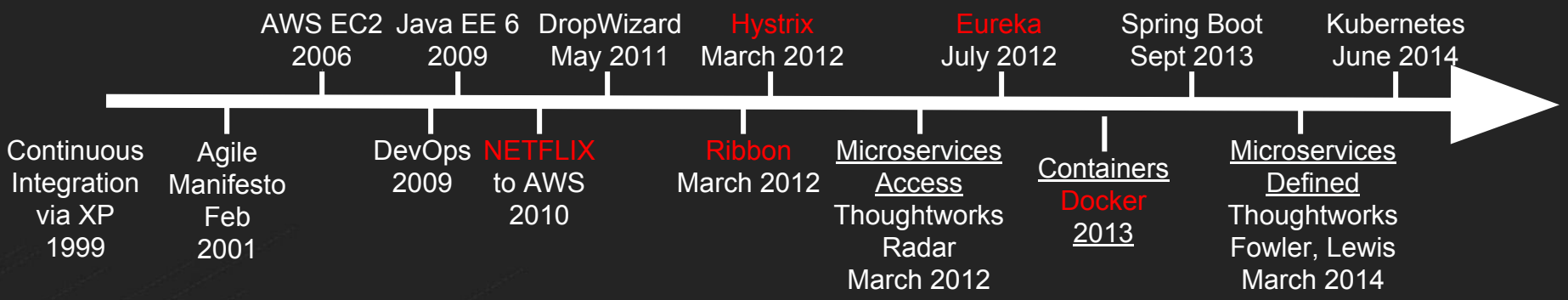
- DevOps Evangelist at Red Hat
  - Cloud Native App Practitioner
  - Agile Coach
  - Container Geek
- Java Developer
- Opensource.com DevOps Team
- Speaker & Writer


doh@redhat.com


@danieloh30


danieloh30

Short History of Microservices



NETFLIX | OSS

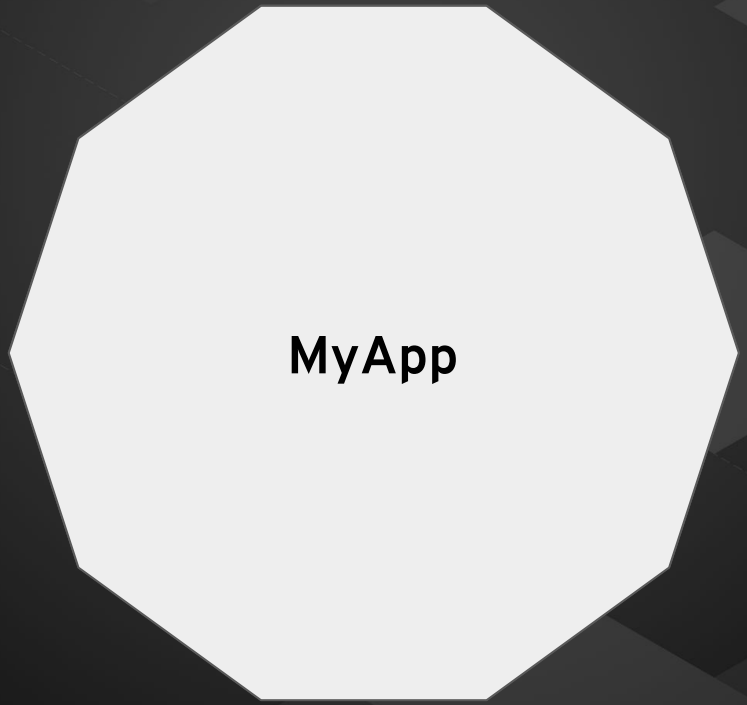
What is a microservice ?

The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

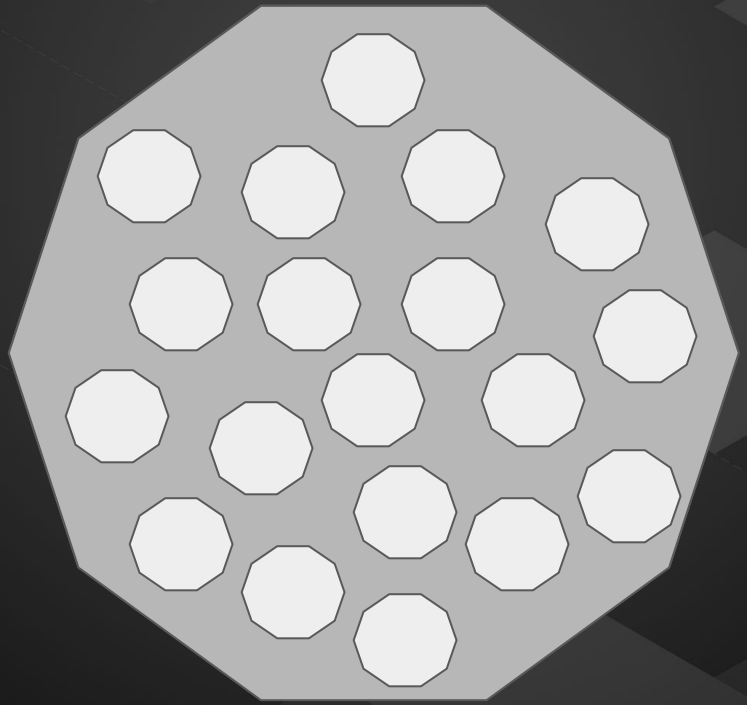
These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

Martin Fowler

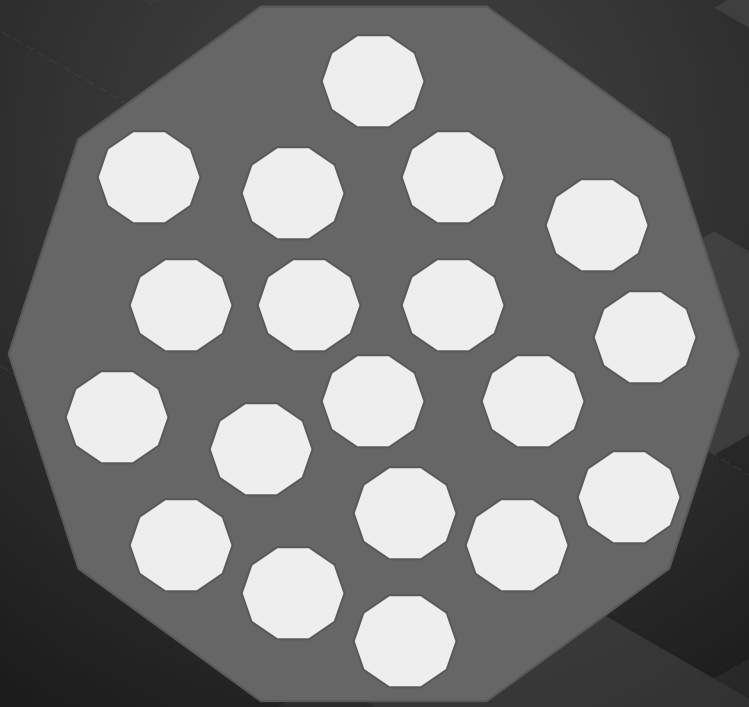
Monolith



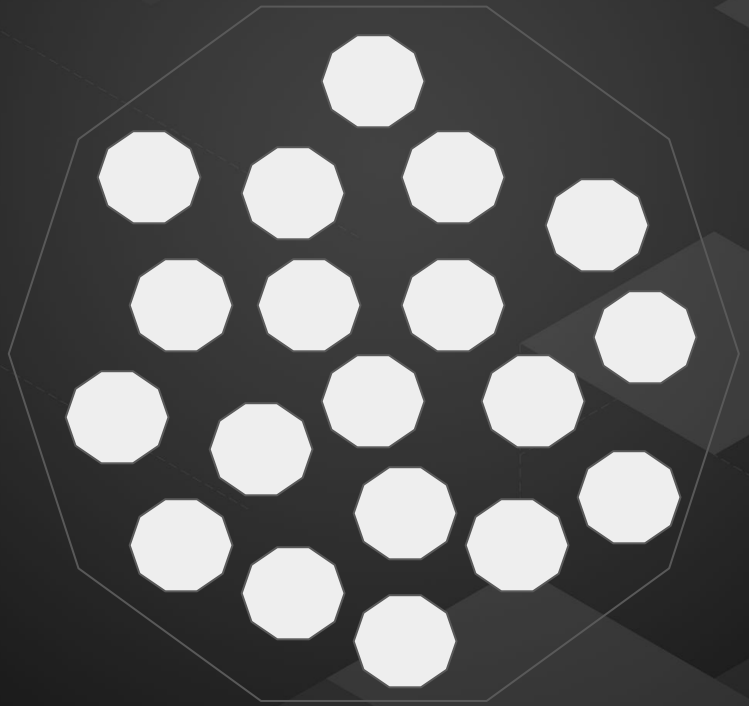
Modules



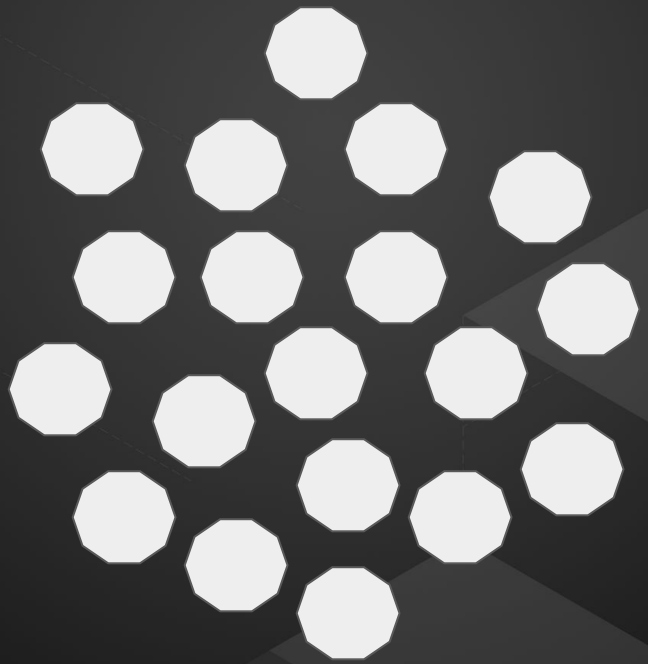
Microservices



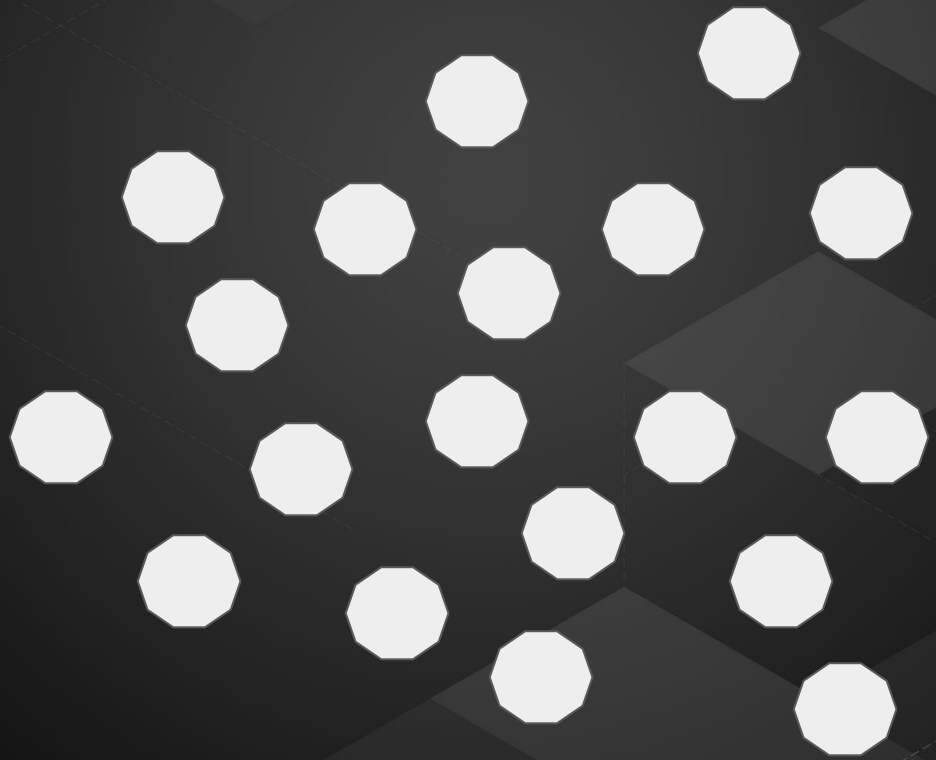
Microservices



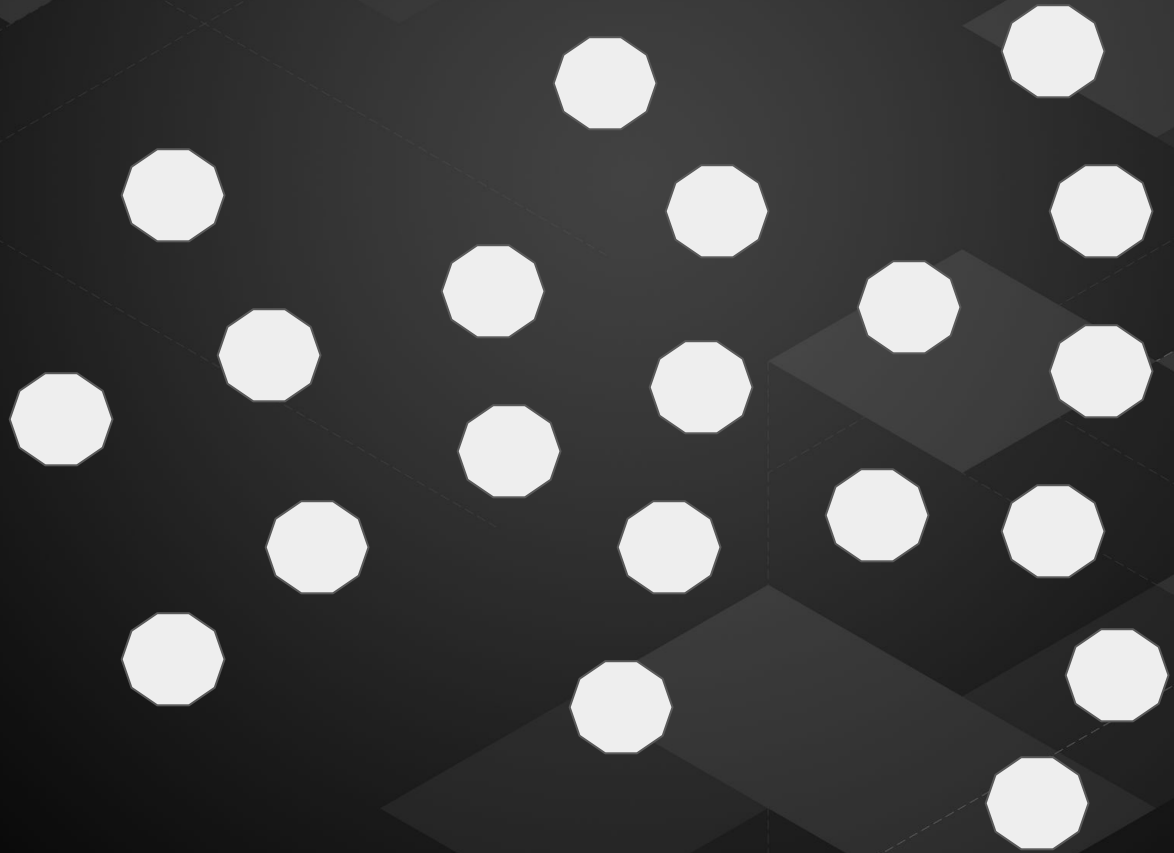
Microservices



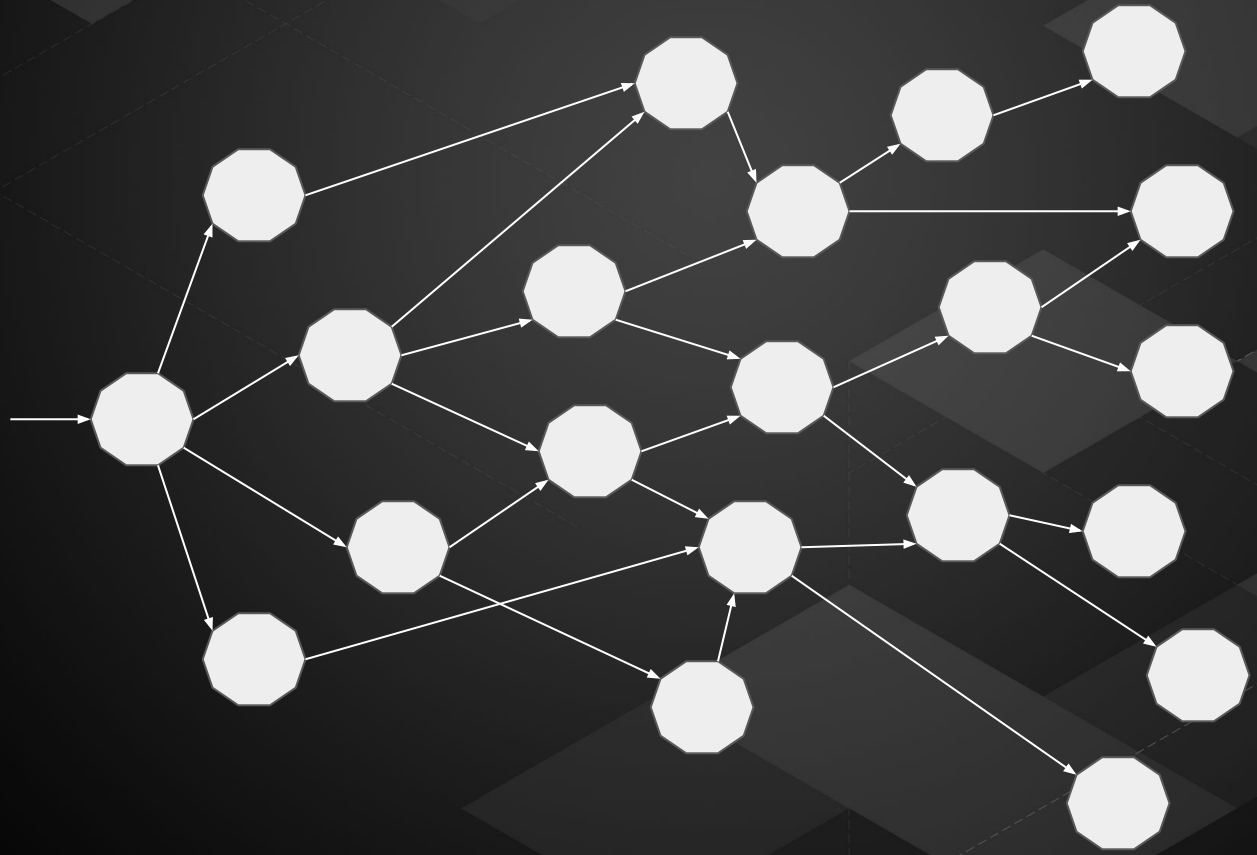
Microservices



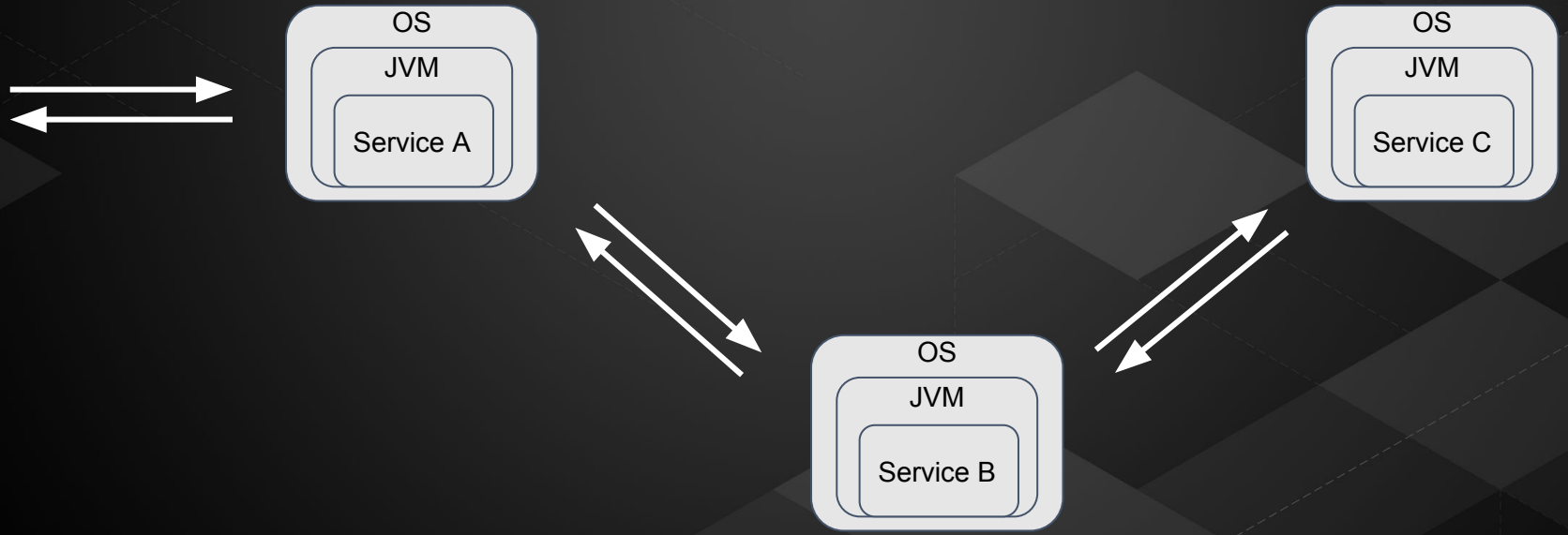
Microservices



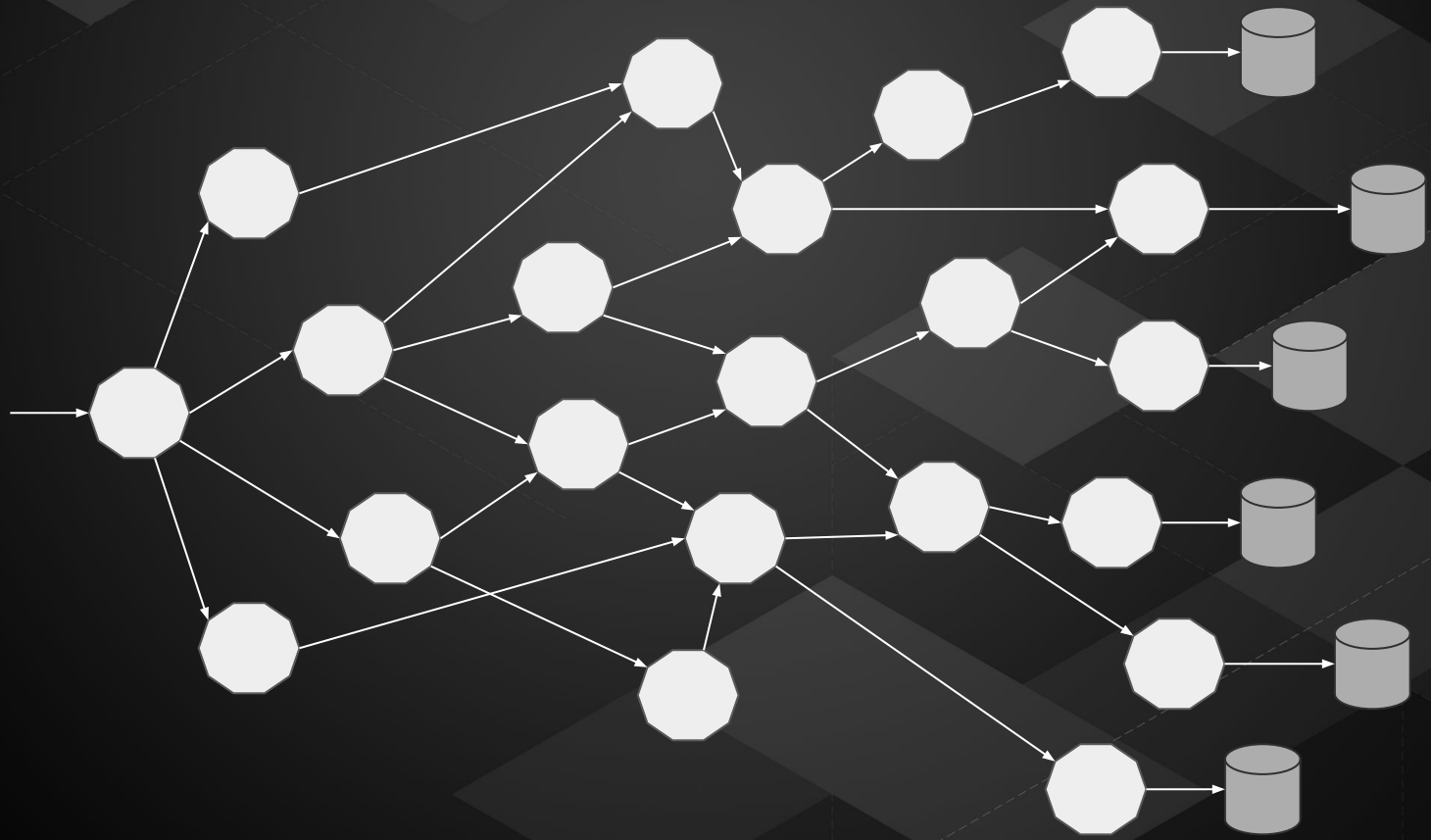
Microservices == Distributed Computing



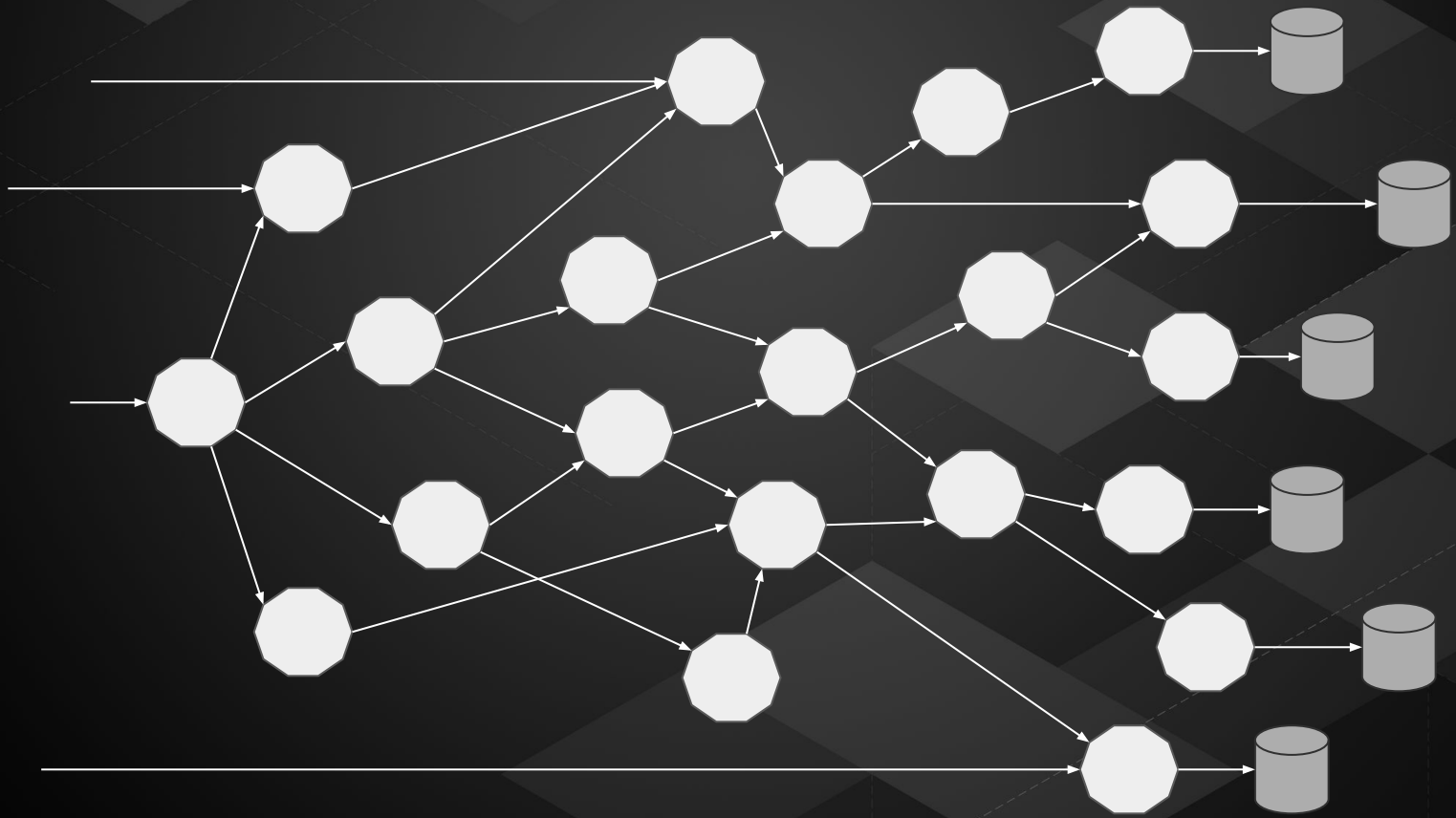
Distributed Computing == Network of Services



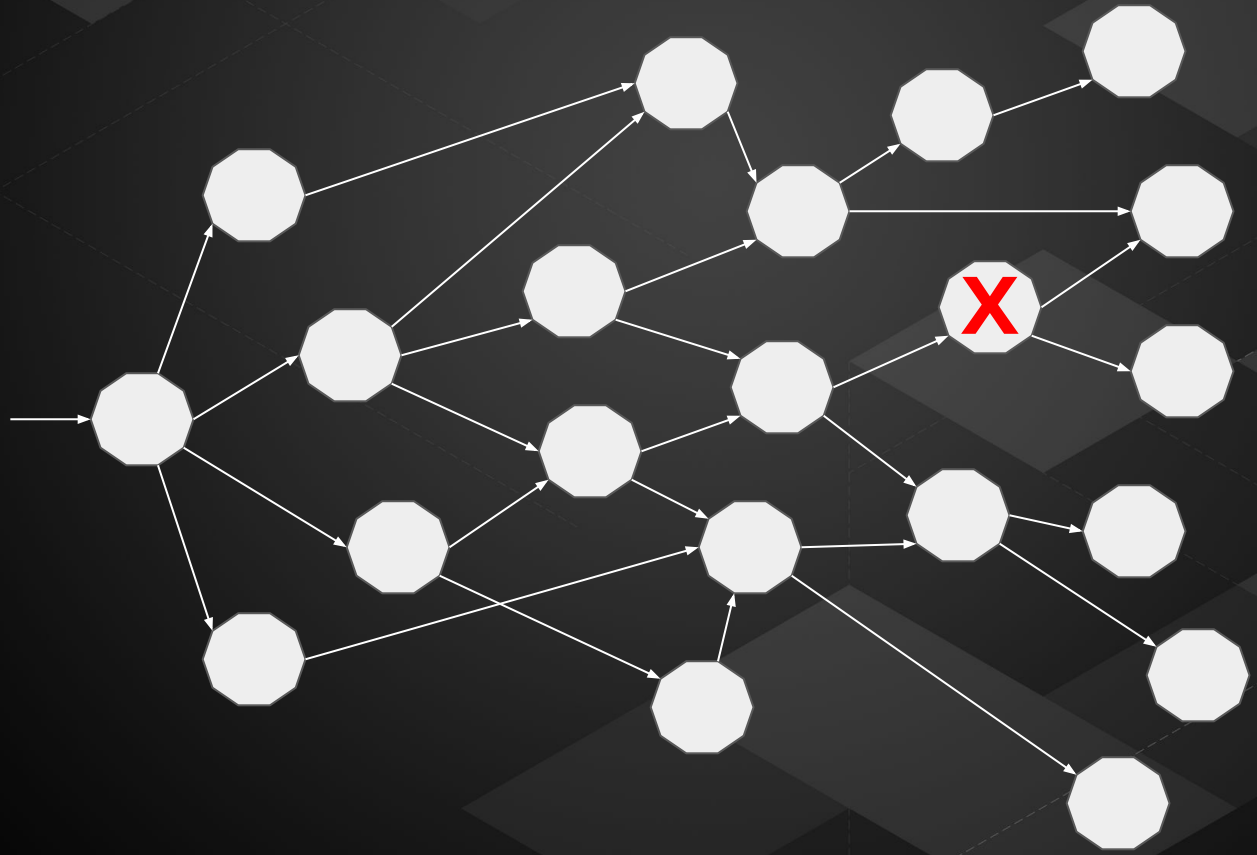
Microservices own their Data



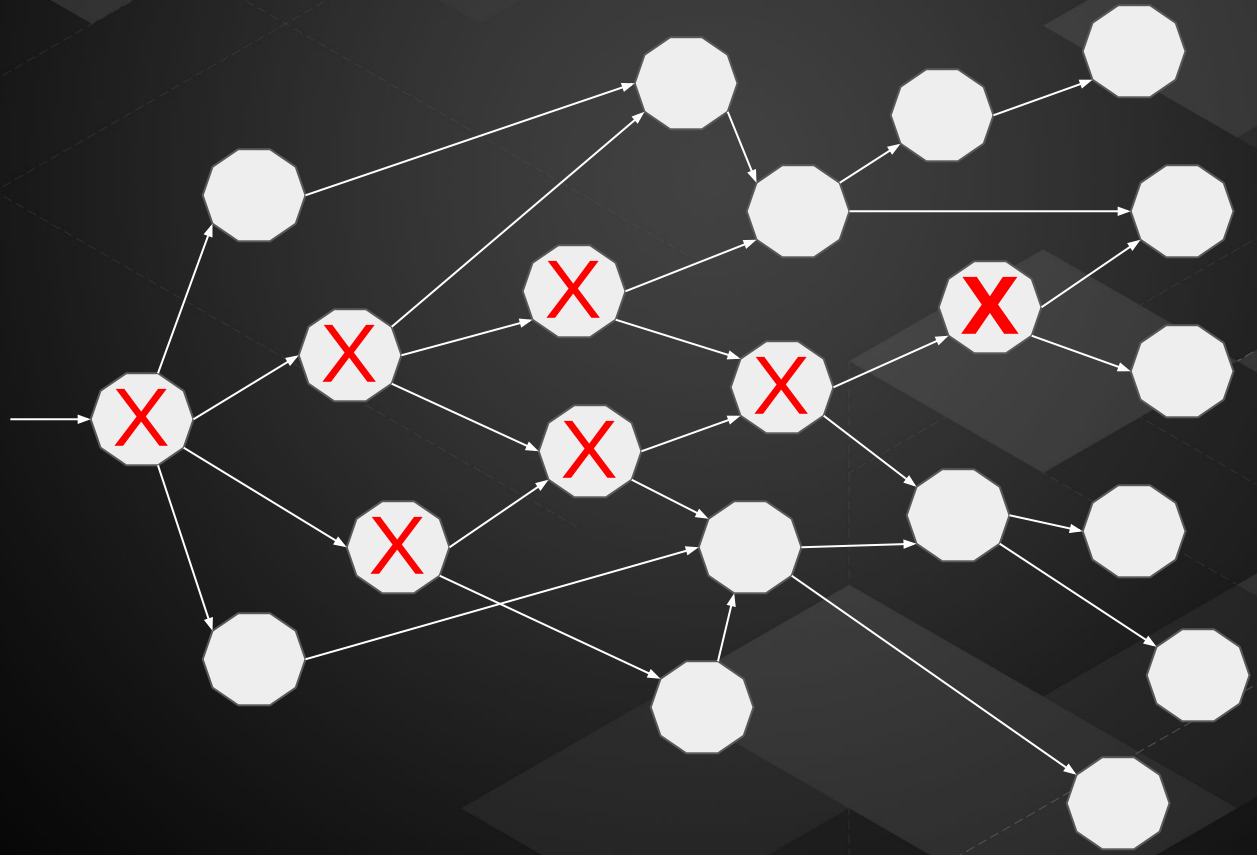
Multiple Points of Entry



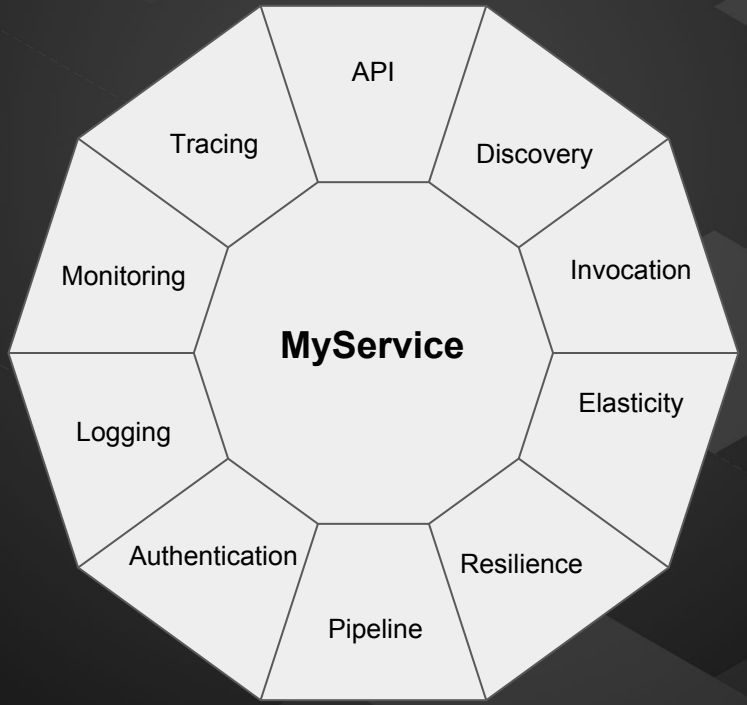
Failure of a Service



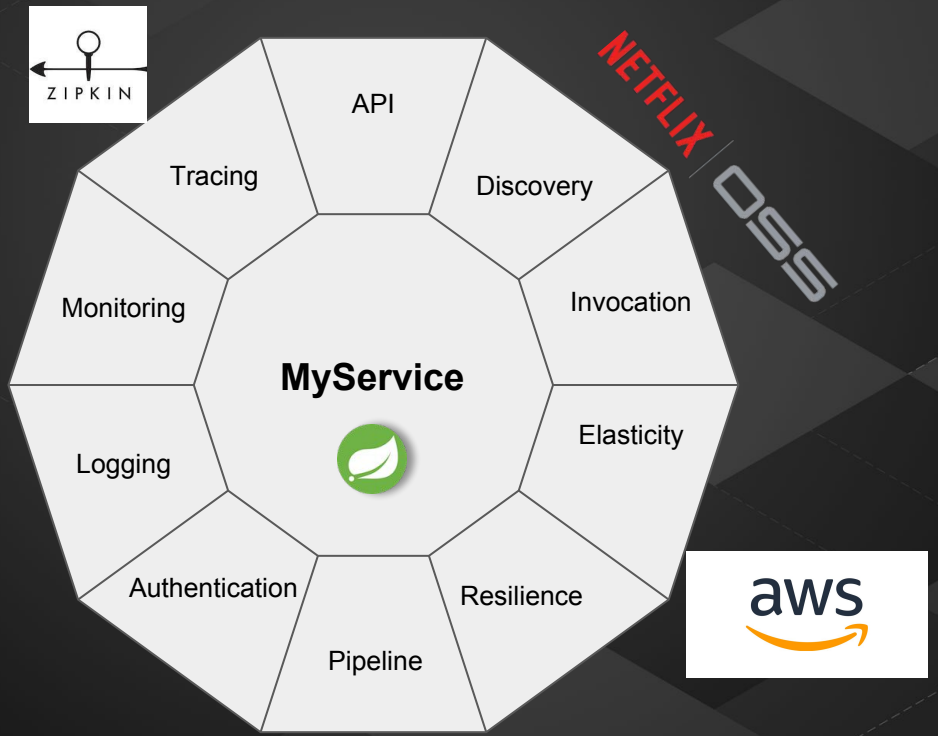
Cascading Failure



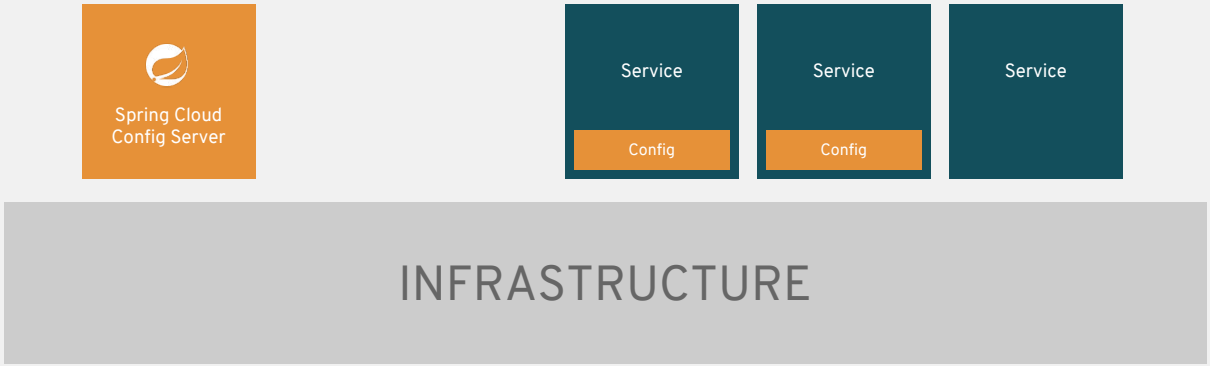
Microservices'ilities



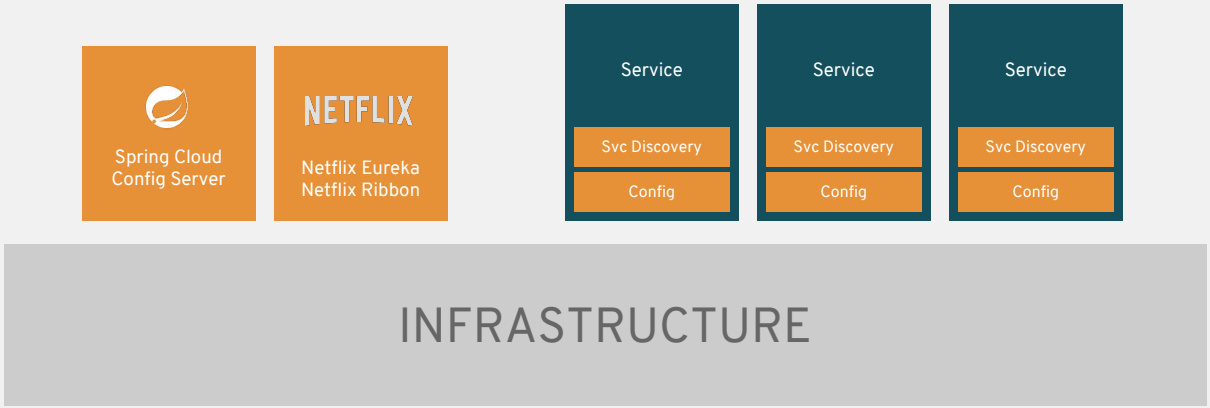
Microservices'ilities + Netflix OSS == 1.0



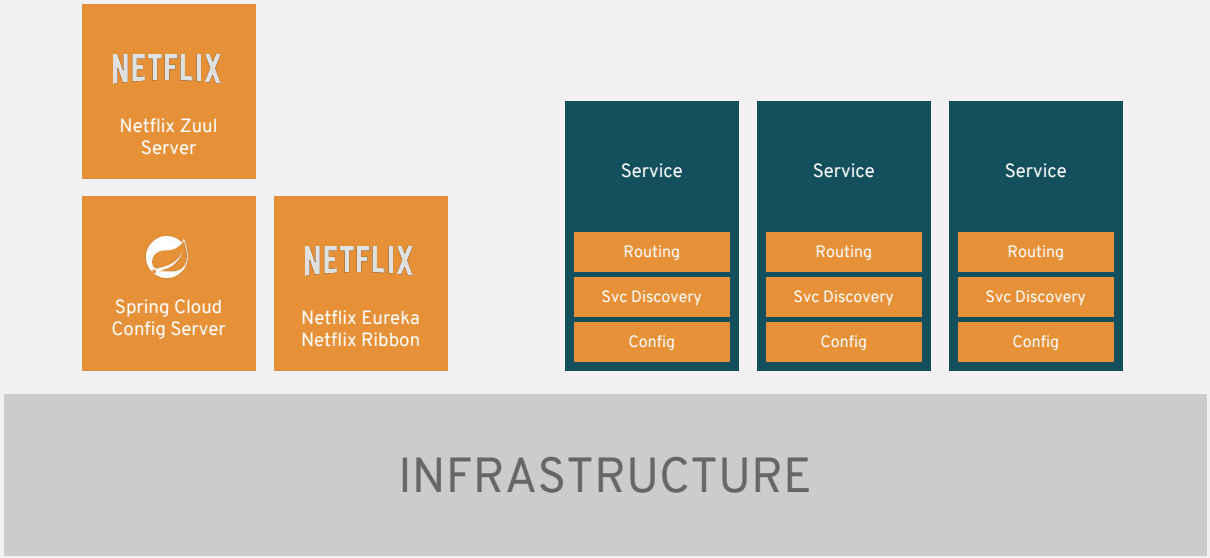
CONFIGURATION



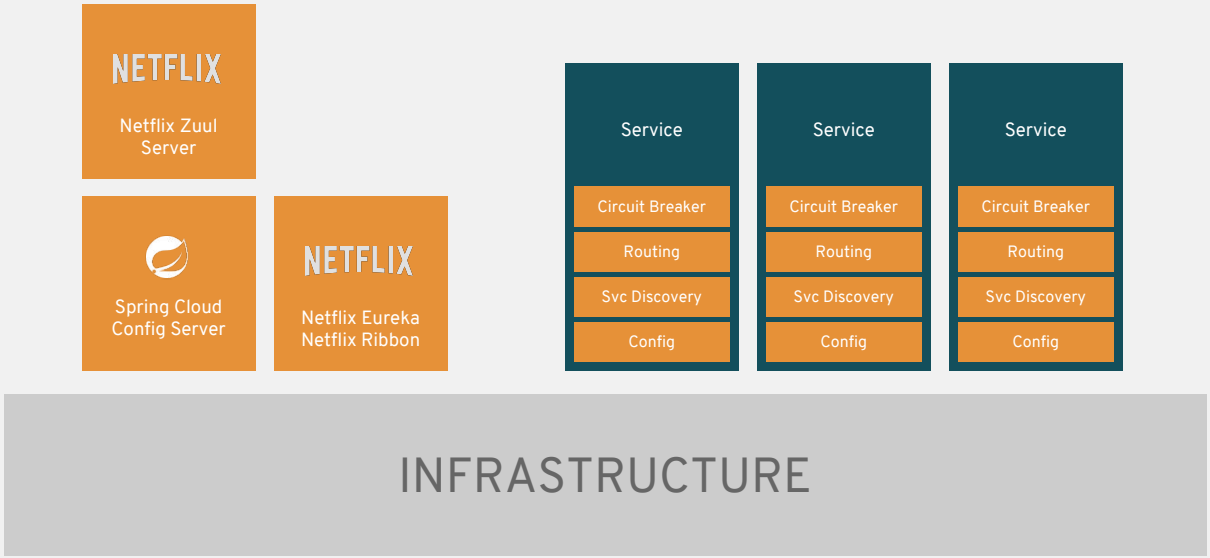
SERVICE DISCOVERY



DYNAMIC ROUTING

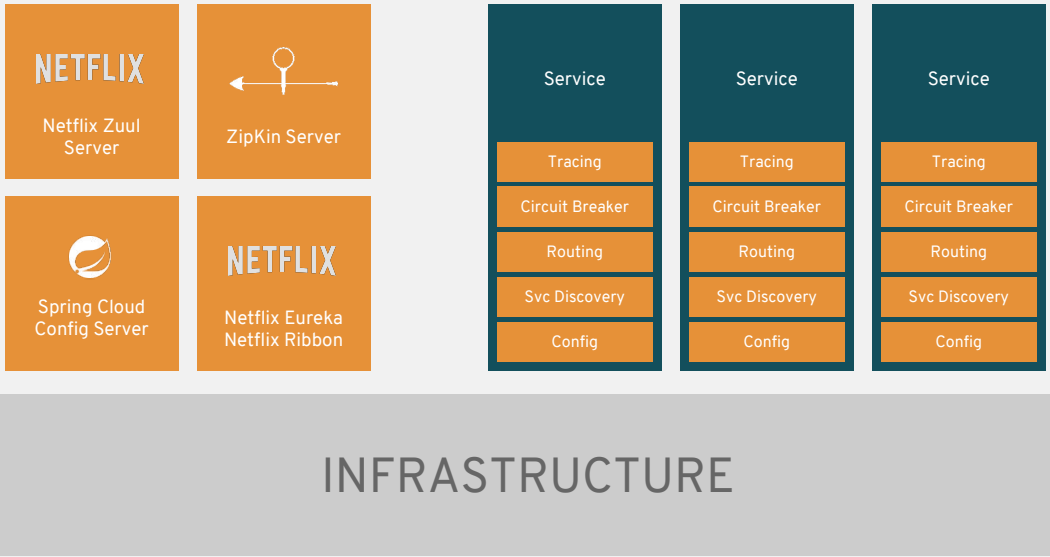


FAULT TOLERANCE



NETFLIX | OSS

TRACING AND VISIBILITY



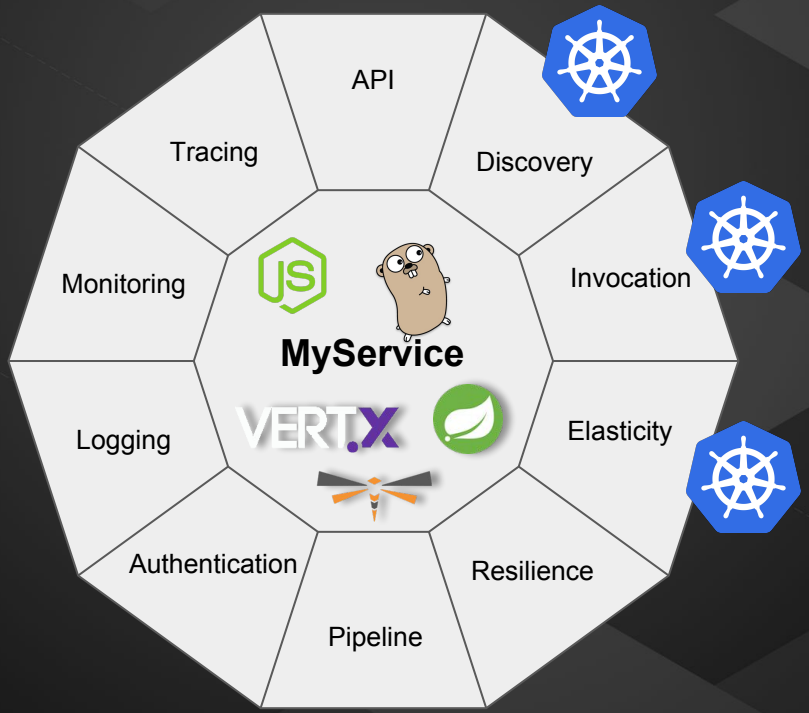
What's Wrong with Netflix OSS?

Java Only

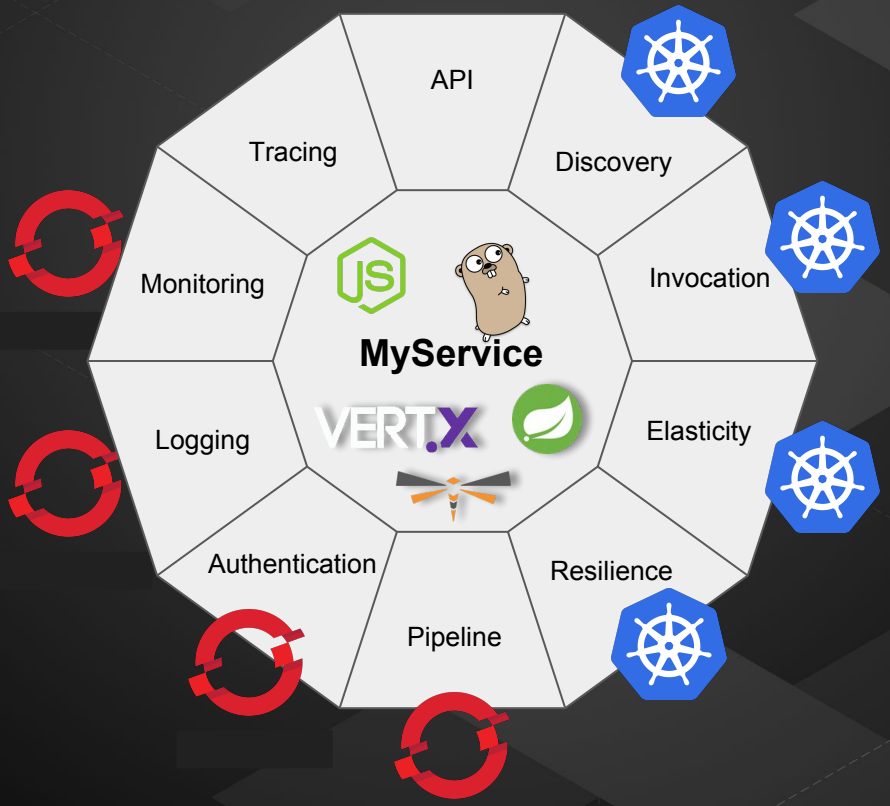
Adds a lot of libraries to YOUR code

OPENSHIFT

Microservices'ilities + Kubernetes == 2.0



Microservices'ilities + OpenShift == 2.0



SERVICE MESH WITH ISTIO

@danieloh30

Service Mesh Defined

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

https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/

Next Generation Microservices - Service Mesh

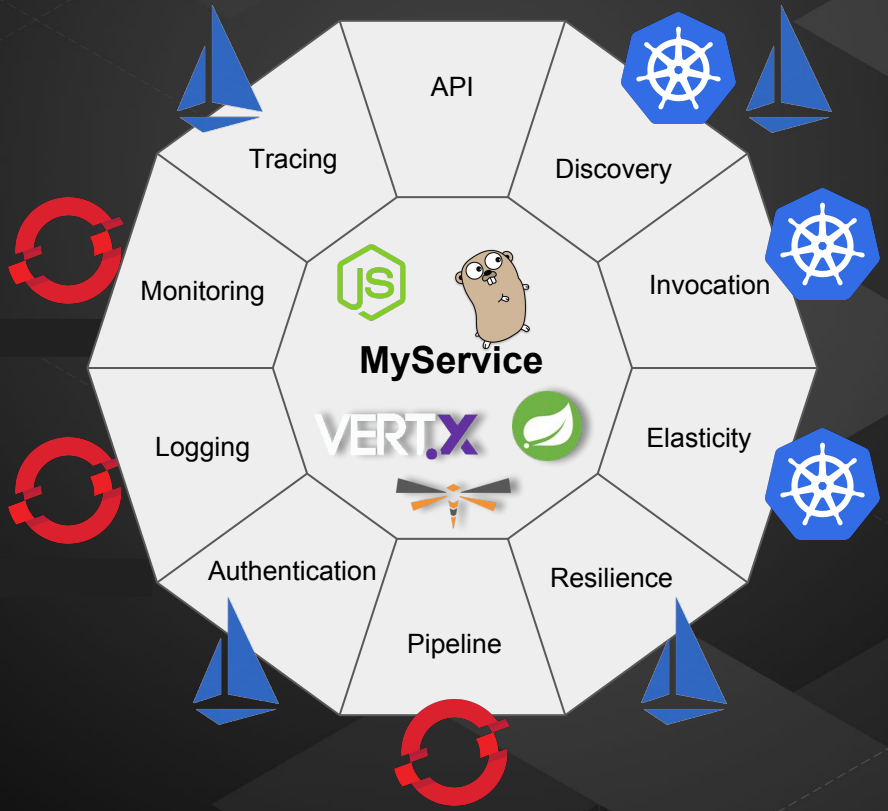
Code Independent (Polyglot)

- Intelligent Routing and Load-Balancing
  - A/B Tests
  - Smarter Canary Releases
- Chaos: Fault Injection
- Resilience: Circuit Breakers
- Observability: Metrics and Tracing
- Fleet wide policy enforcement

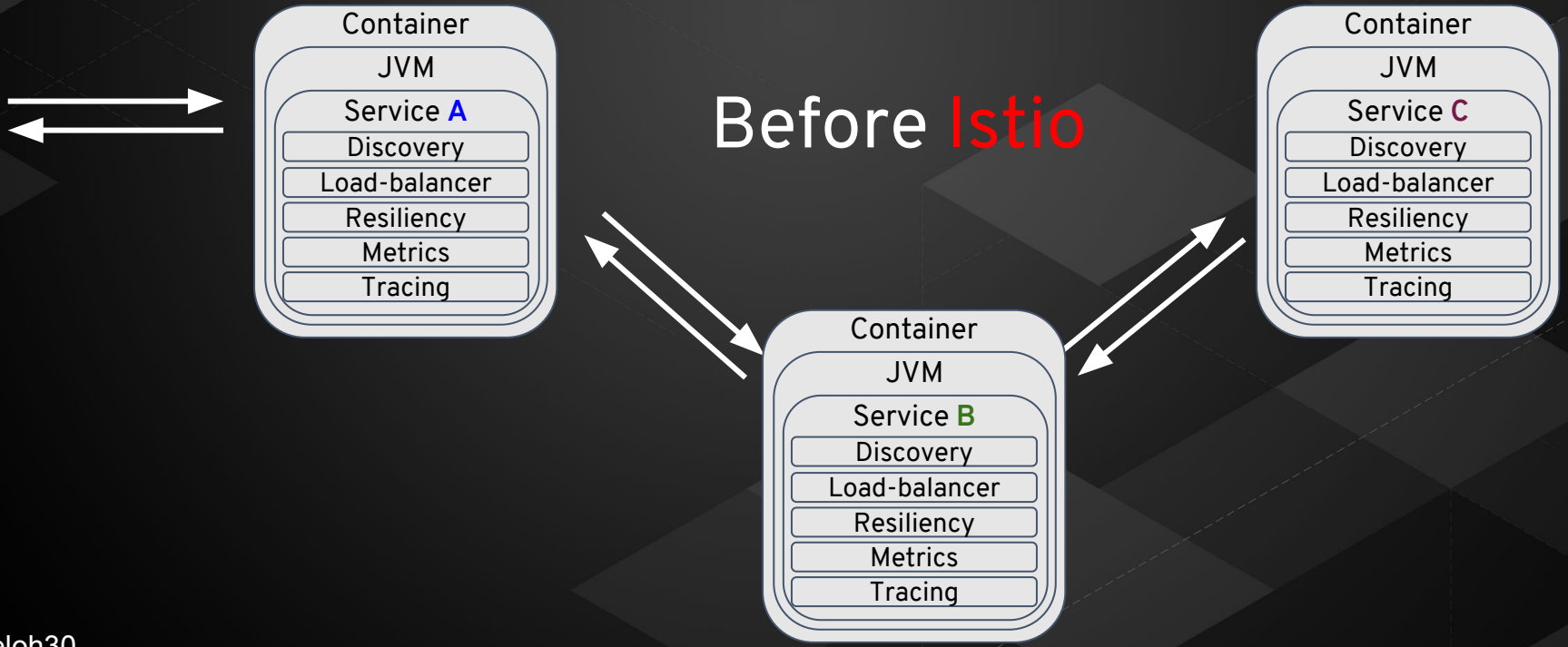
Istio - Sail

(Kubernetes - Helmsman or ship’s pilot)

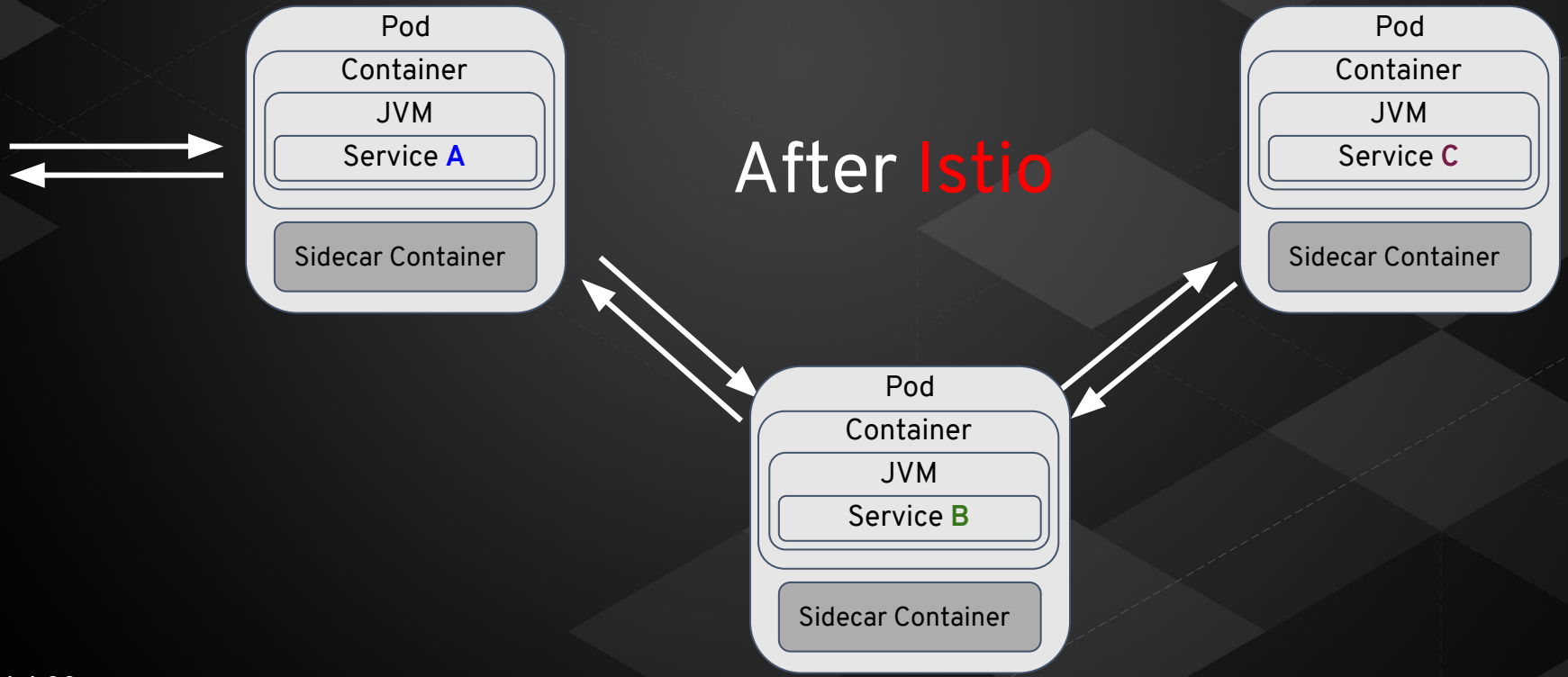
Microservices'ilities + Istio == 3.0



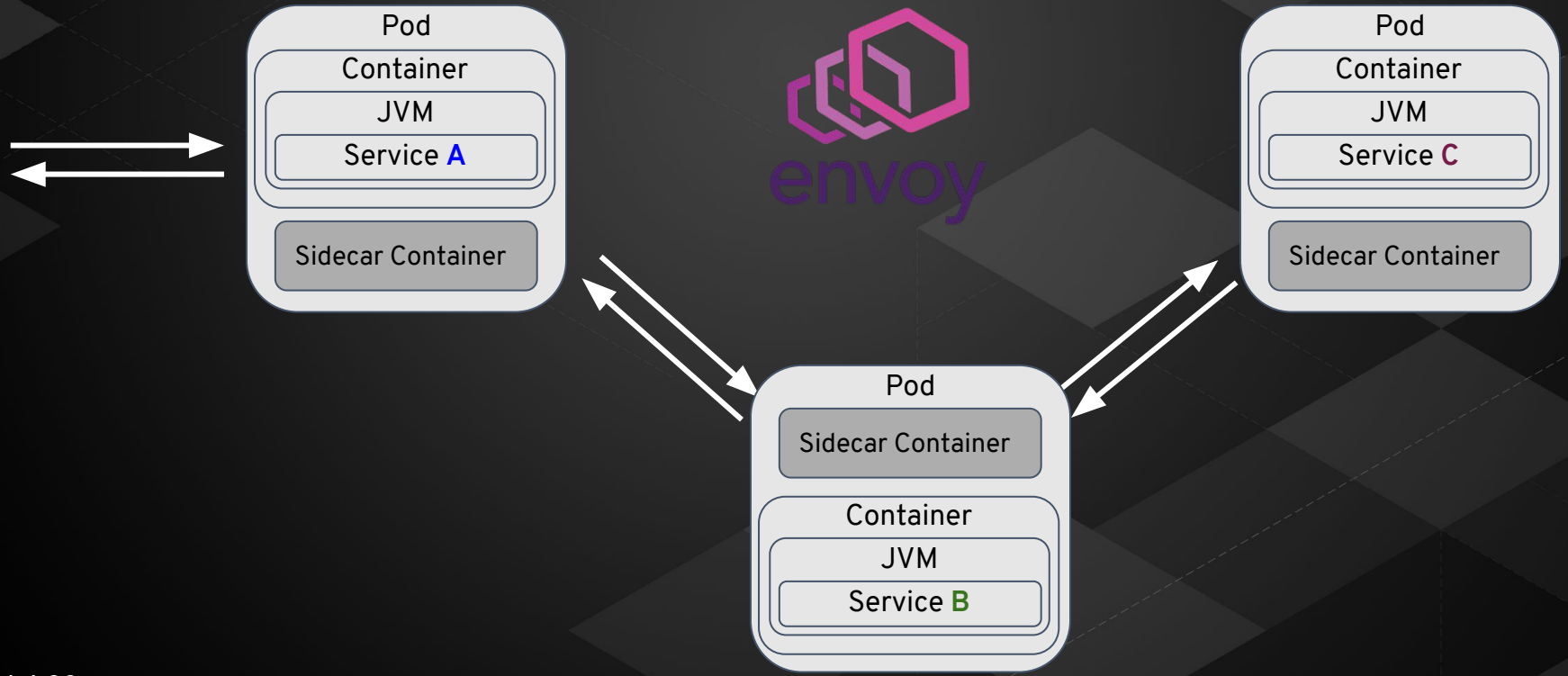
Microservices embedding Capabilities

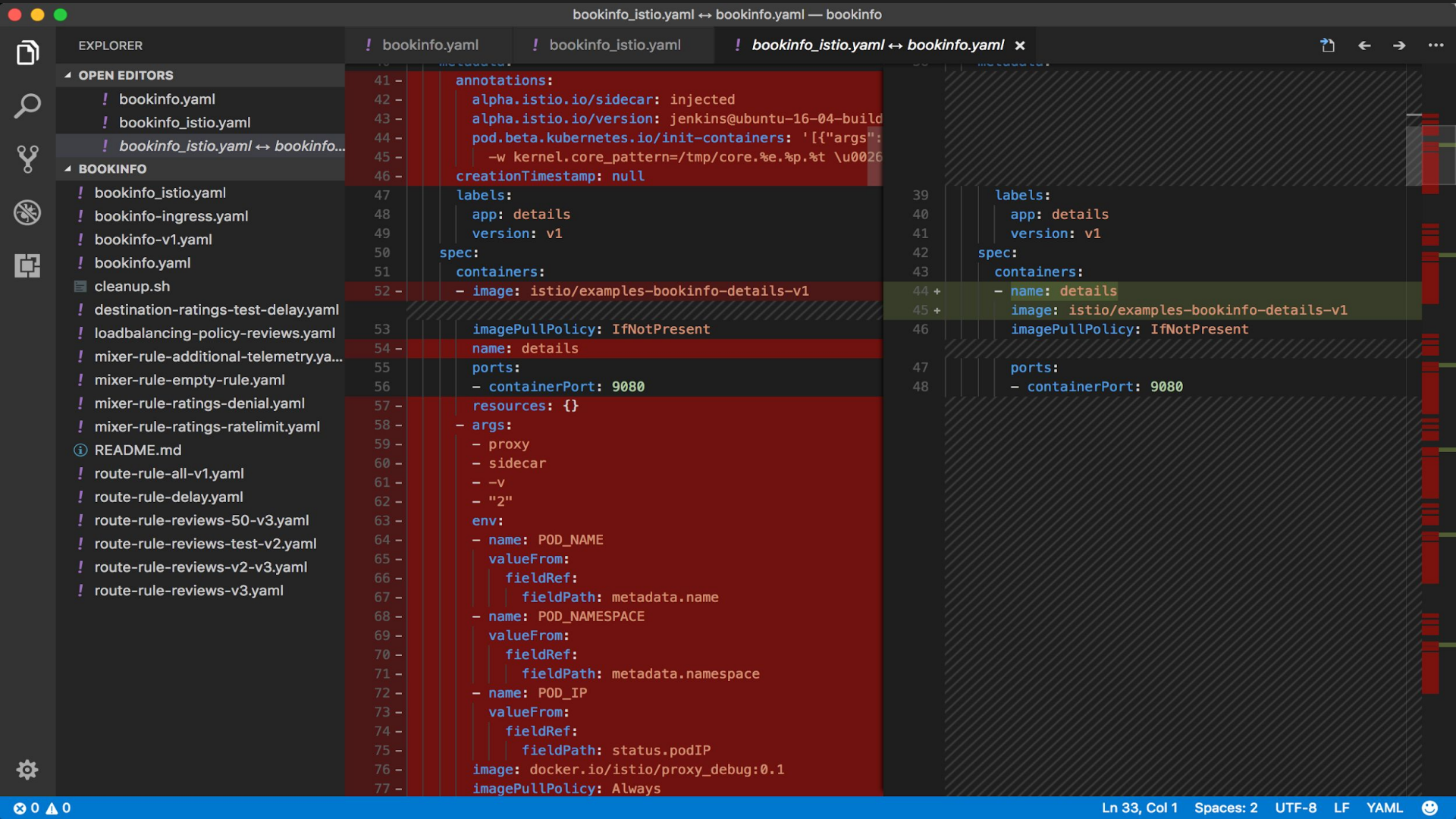


Microservices externalizing Capabilities

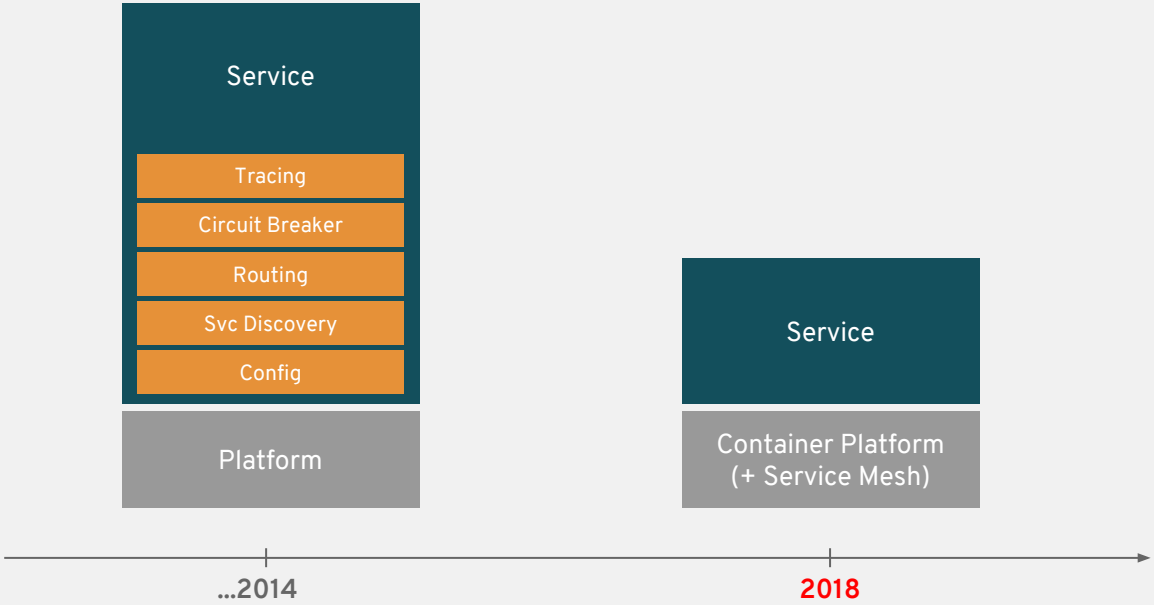


Envoy is the current sidecar

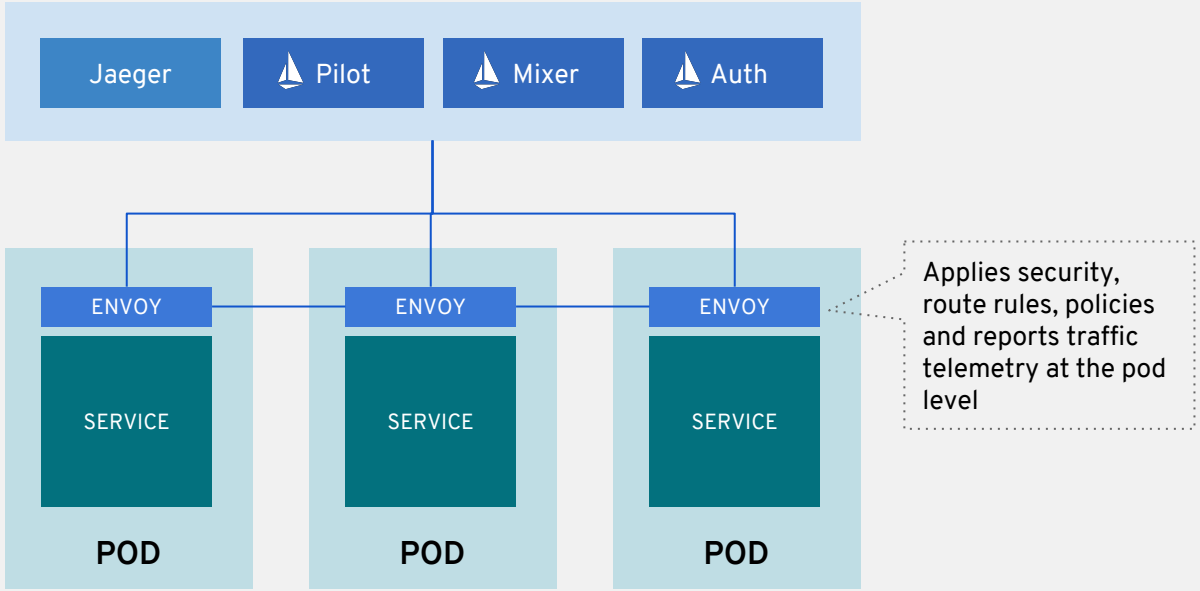



bookinfo_istio.yaml ↔ bookinfo.yaml — bookinfo
EXPLORER
OPEN EDITORS
bookinfo.yaml
bookinfo_istio.yaml
bookinfo_istio.yaml ↔ bookinfo...
BOOKINFO
bookinfo_istio.yaml
bookinfo-ingress.yaml
bookinfo-v1.yaml
bookinfo.yaml
cleanup.sh
destination-ratings-test-delay.yaml
loadbalancing-policy-reviews.yaml
mixer-rule-additional-telemetry.ya...
mixer-rule-empty-rule.yaml
mixer-rule-ratings-denial.yaml
mixer-rule-ratings-ratelimit.yaml
README.md
route-rule-all-v1.yaml
route-rule-delay.yaml
route-rule-reviews-50-v3.yaml
route-rule-reviews-test-v2.yaml
route-rule-reviews-v2-v3.yaml
route-rule-reviews-v3.yaml
bookinfo.yaml
bookinfo_istio.yaml
bookinfo_istio.yaml ↔ bookinfo.yaml
annotations:
alpha.istio.io/sidecar: injected
alpha.istio.io/version: jenkins@ubuntu-16-04-build
pod.beta.kubernetes.io/init-containers: '[{"args":
-w kernel.core_pattern=/tmp/core.%e.%p.%t \u0026
creationTimestamp: null
labels:
app: details
version: v1
spec:
containers:
- image: istio/examples-bookinfo-details-v1
imagePullPolicy: IfNotPresent
name: details
ports:
- containerPort: 9080
resources: {}
- args:
- proxy
- sidecar
- -v
- "2"
env:
- name: POD_NAME
valueFrom:
fieldRef:
fieldPath: metadata.name
- name: POD_NAMESPACE
valueFrom:
fieldRef:
fieldPath: metadata.namespace
- name: POD_IP
valueFrom:
fieldRef:
fieldPath: status.podIP
image: docker.io/istio/proxy_debug:0.1
imagePullPolicy: Always
labels:
app: details
version: v1
spec:
containers:
- name: details
image: istio/examples-bookinfo-details-v1
imagePullPolicy: IfNotPresent
ports:
- containerPort: 9080
Ln 33, Col 1
Spaces: 2
UTF-8
LF
YAML

MICROSERVICES EVOLUTION

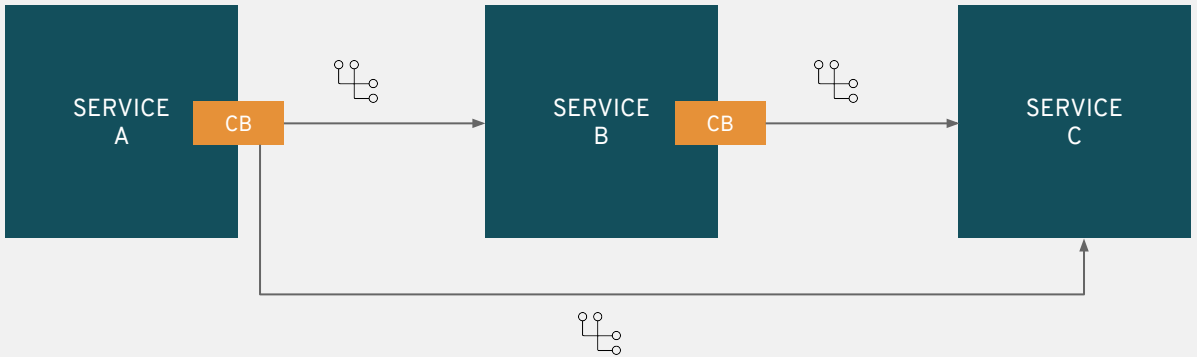


SERVICE MESH ARCHITECTURE



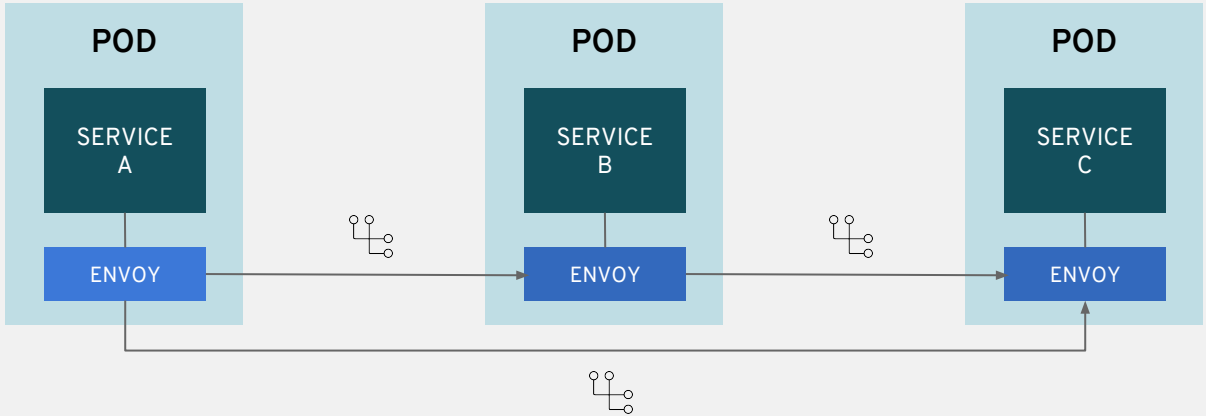
FAULT TOLERANCE

CIRCUIT BREAKERS WITHOUT ISTIO



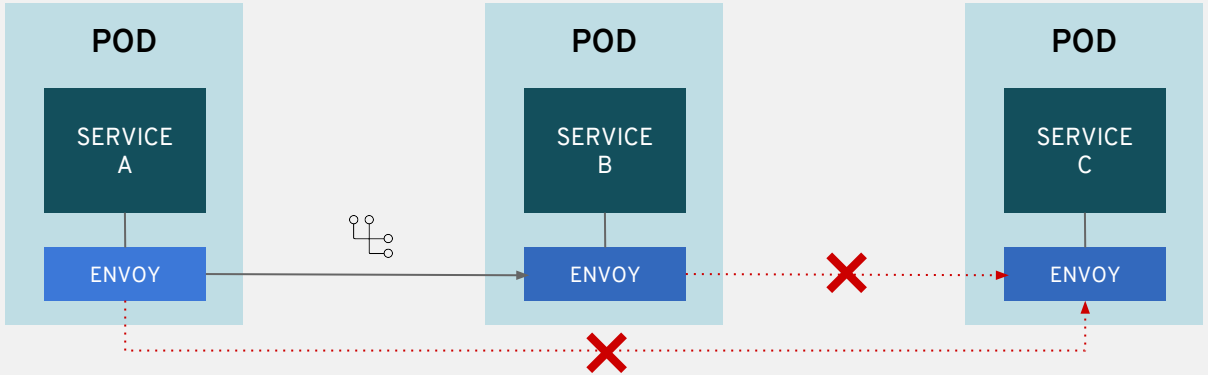
coupled to the service code

CIRCUIT BREAKERS WITH ISTIO



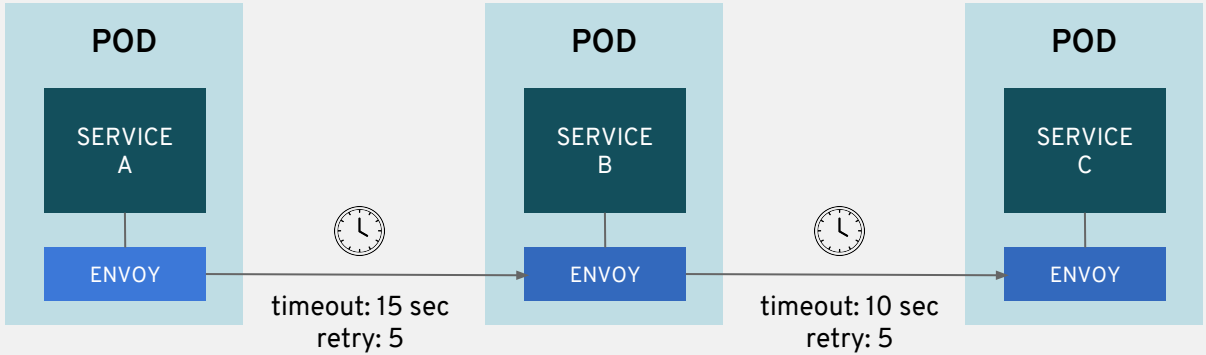
transparent to the services

CIRCUIT BREAKERS WITH ISTIO



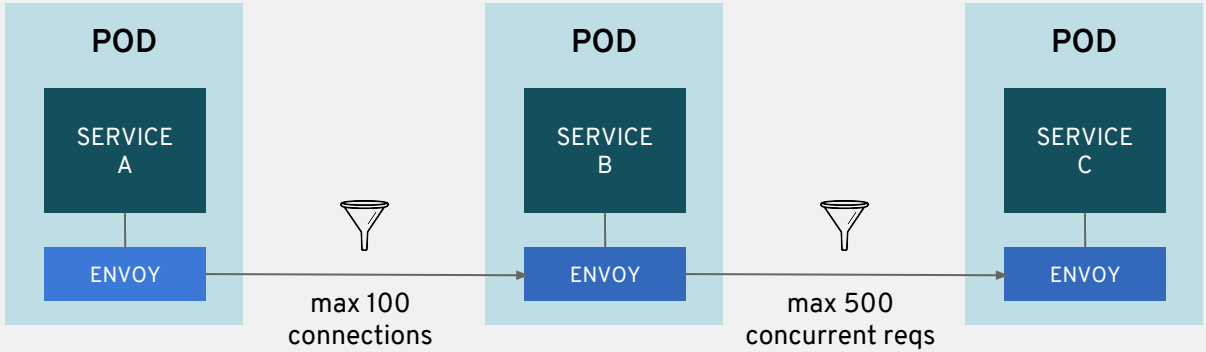
improved response time with global circuit status

TIMEOUTS AND RETRIES WITH ISTIO



configure timeouts and retries, transparent to the services

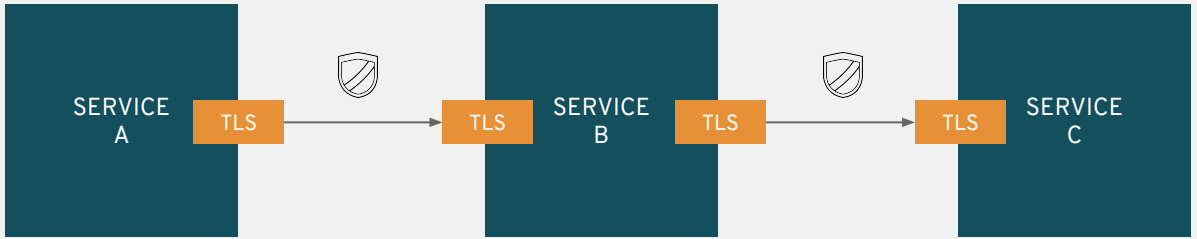
RATE LIMITING WITH ISTIO



limit invocation rates, transparent to the services

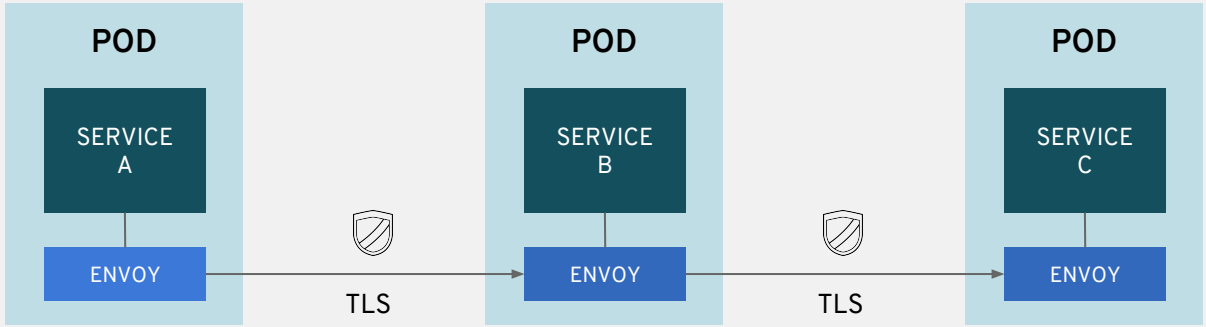
SERVICE SECURITY

SECURE COMMUNICATION WITHOUT ISTIO



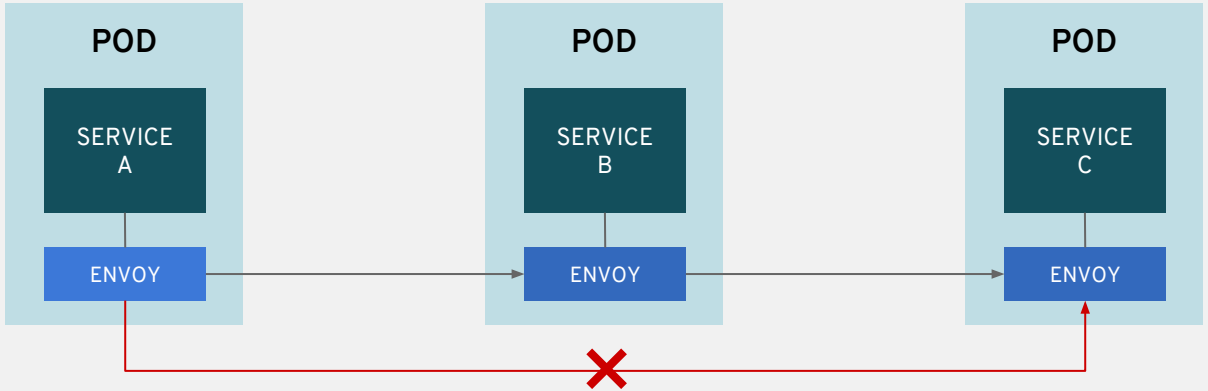
coupled to the service code

SECURE COMMUNICATION WITH ISTIO



mutual TLS authentication, transparent to the services

CONTROL SERVICE ACCESS WITH ISTIO

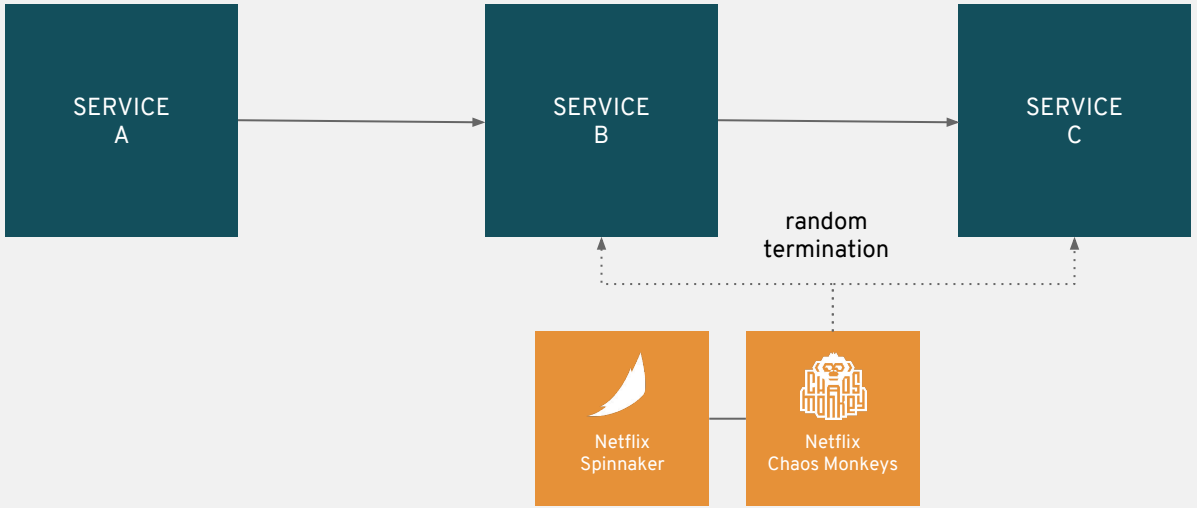


control the service access flow, transparent to the services

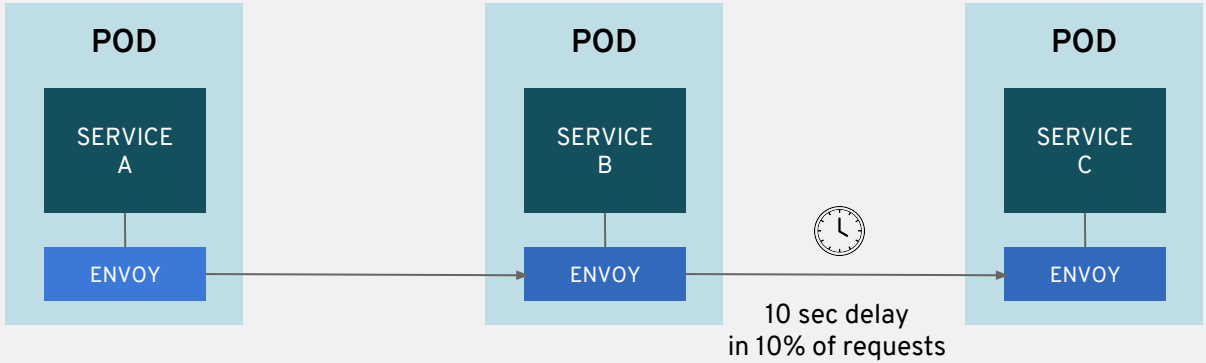
CHAOS ENGINEERING

NETFLIX | OSS

CHAOS ENGINEERING WITHOUT ISTIO

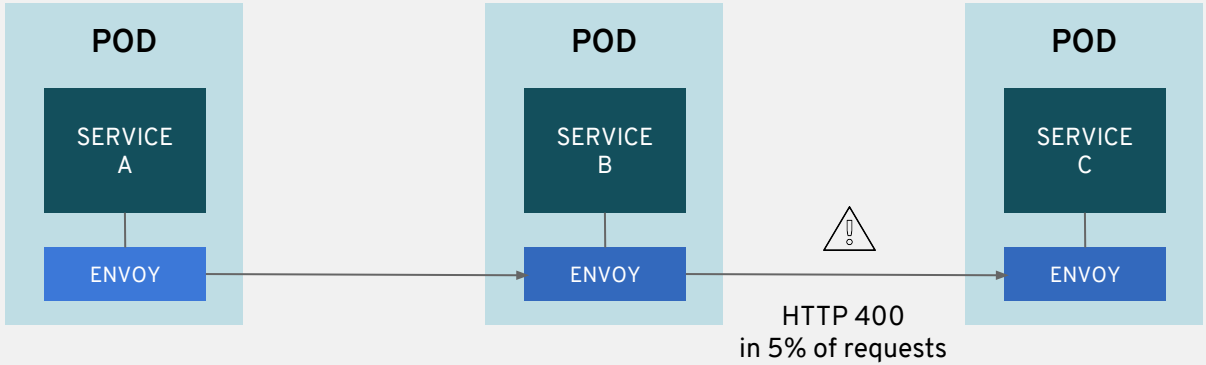


CHAOS ENGINEERING WITH ISTIO



inject delays, transparent to the services

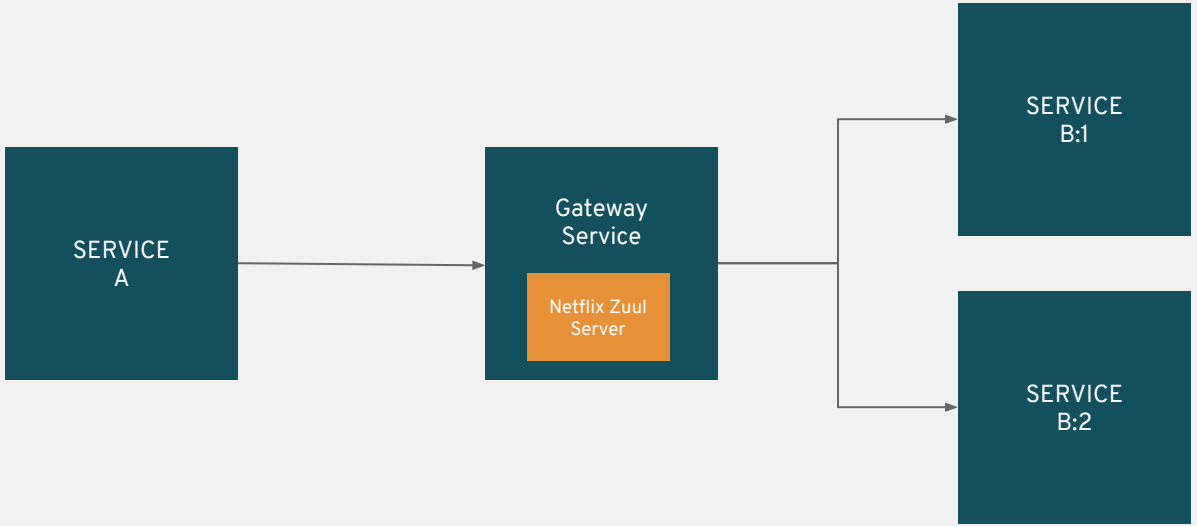
CHAOS ENGINEERING WITH ISTIO



inject protocol-specific errors, transparent to the services

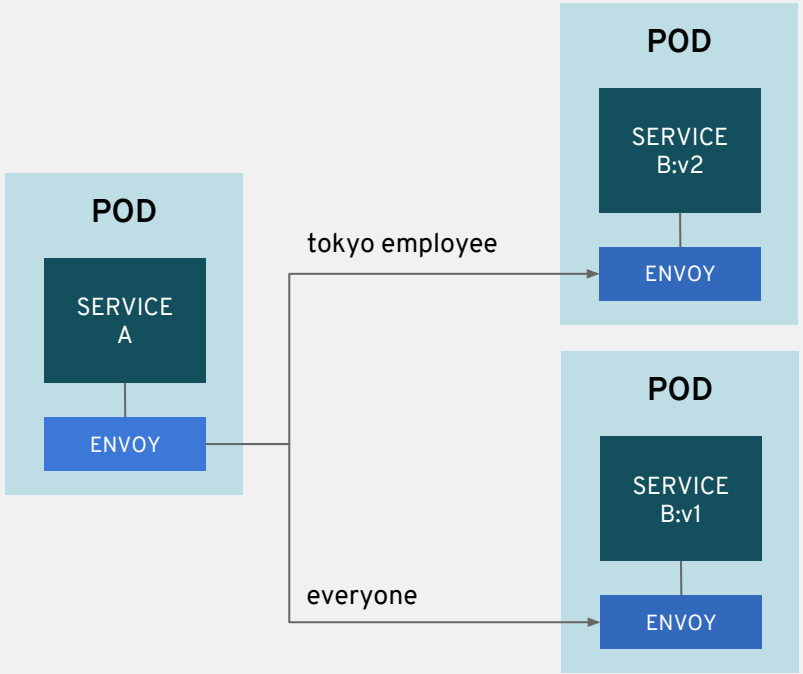
DYNAMIC ROUTING

DYNAMIC ROUTING WITHOUT ISTIO

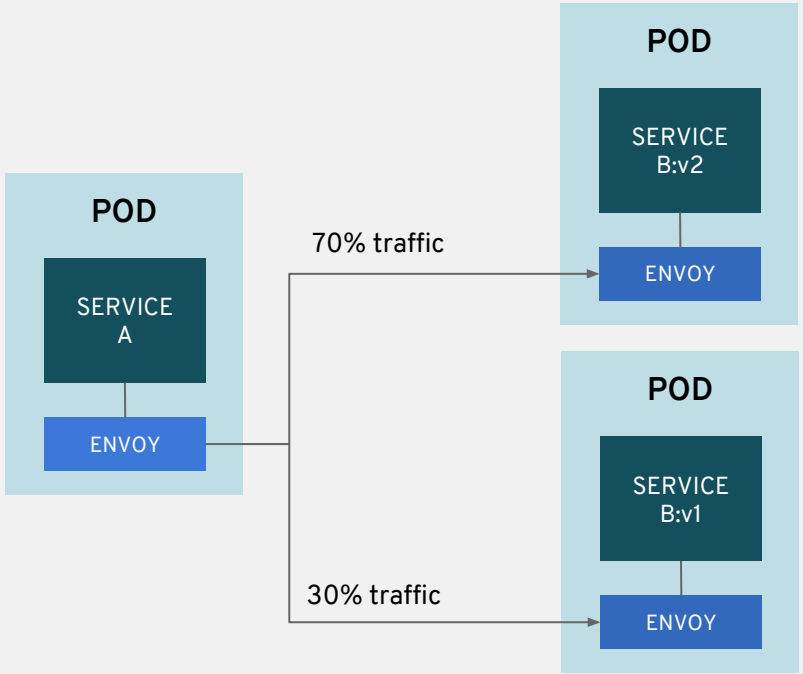


custom code to enable dynamic routing

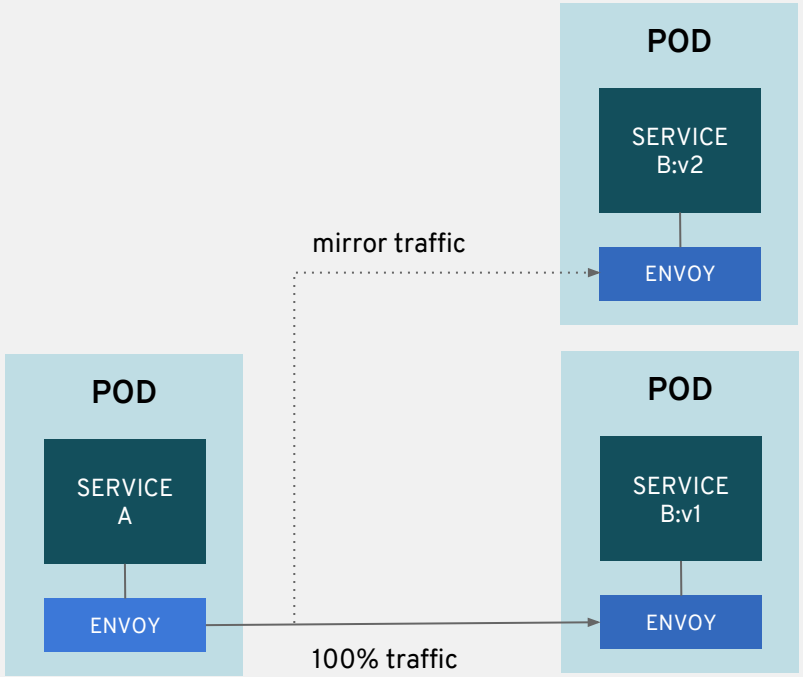
CANARY DEPLOYMENT WITH ISTIO



A/B DEPLOYMENT WITH ISTIO

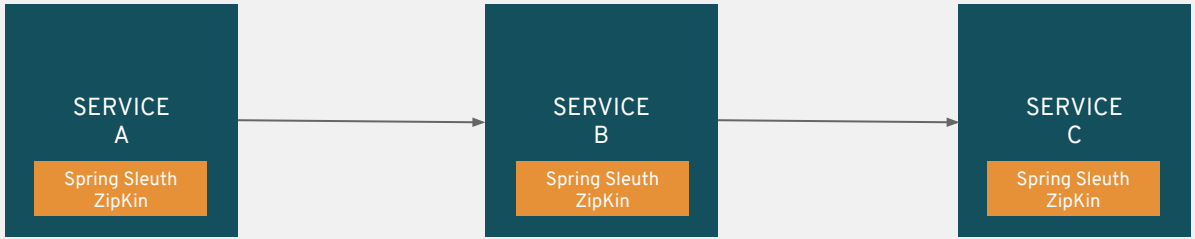


DARK LAUNCHES WITH ISTIO



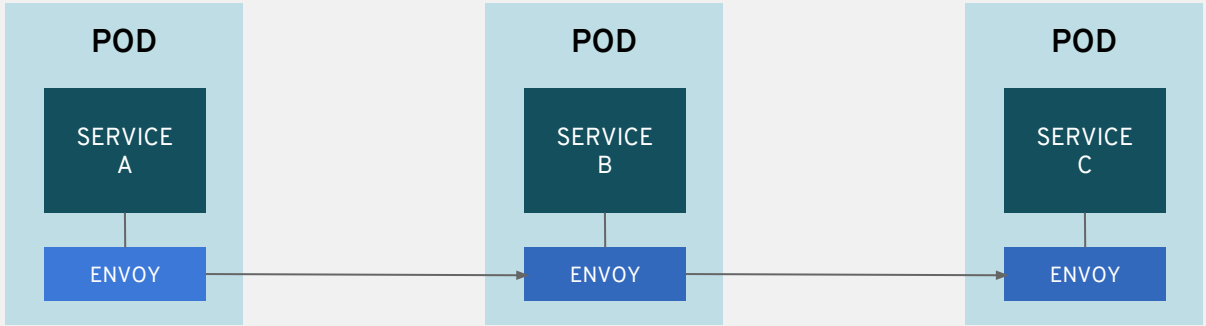
DISTRIBUTED TRACING

DISTRIBUTED TRACING WITHOUT ISTIO

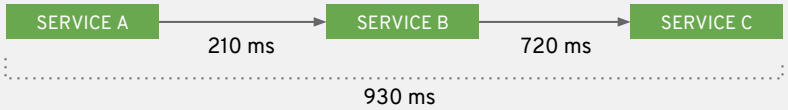


code to enable dynamic tracing

DISTRIBUTED TRACING WITH ISTIO & JAEGER



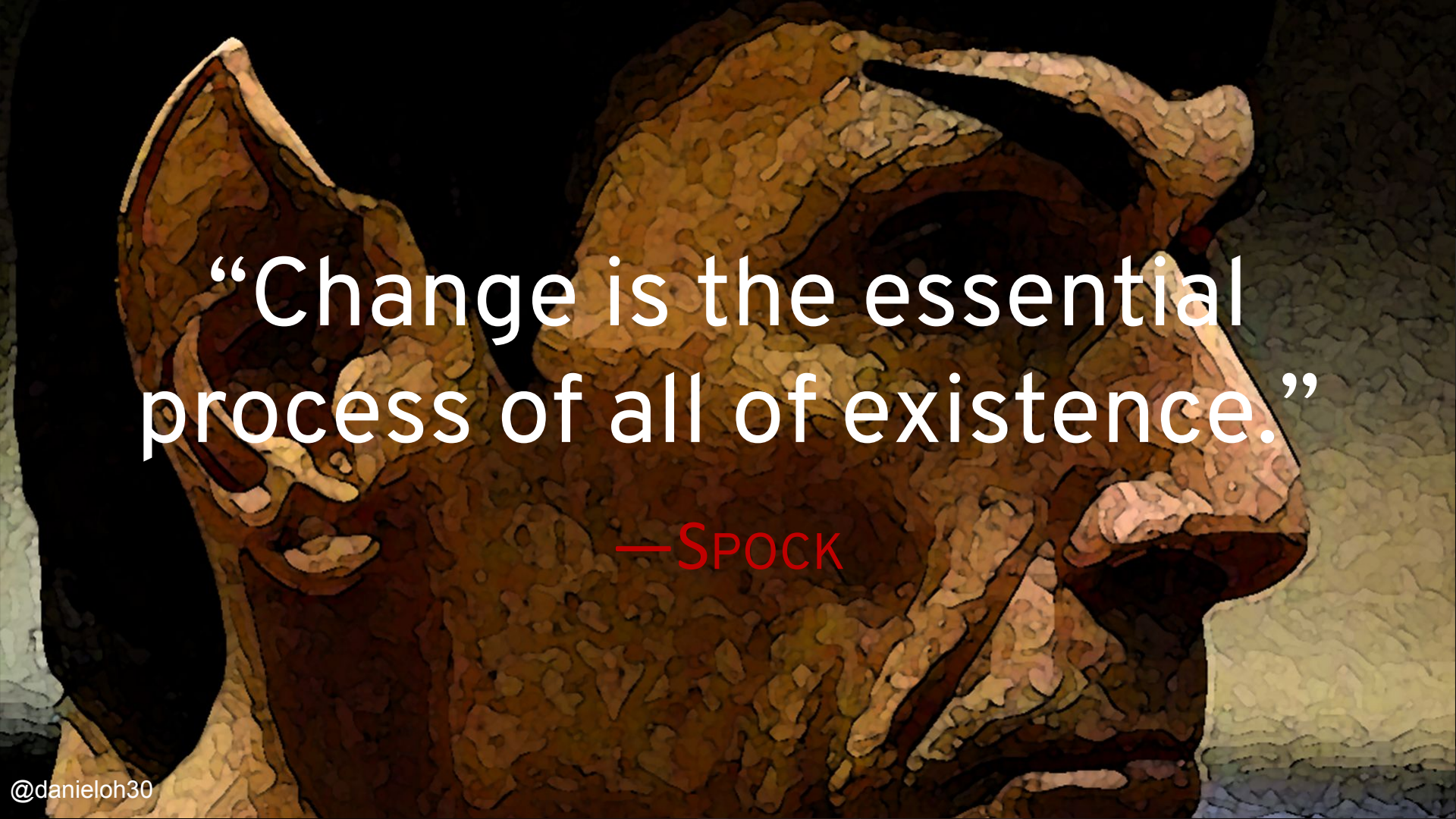
discovers service relationships and process times, transparent to the services



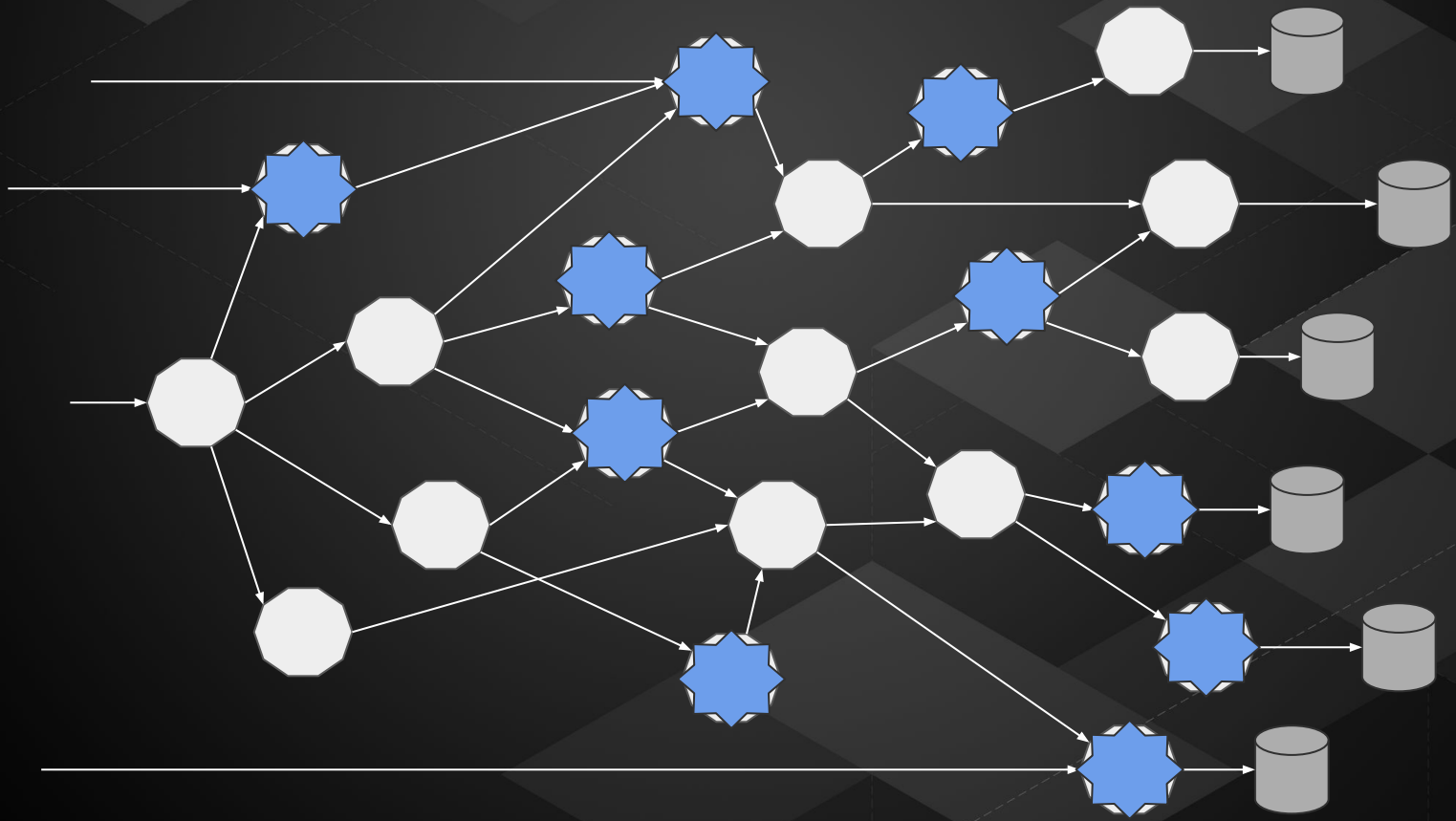
Demo

bit.ly/istio-tutorial

learn.openshift.com/servicemesh


“Change is the essential process of all of existence.”
—Spock

Let there be Functions == 4.0

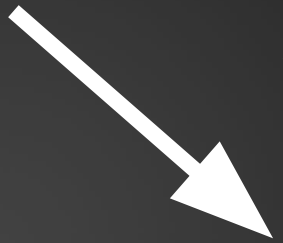


Microservices

Your Control
Long-Lived Processes
Known Programming Model
Often Sync Request-Response

Mature:
IDE Integration
Debuggers
Tracers
Monitoring
CI/CD

Serverless Functions



Cloud Control
Short-Lived Processes
New Programming Model
Event-Driven Async

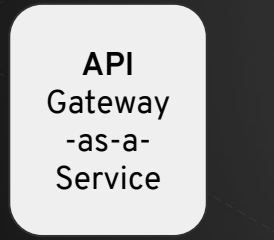
Immature:
?

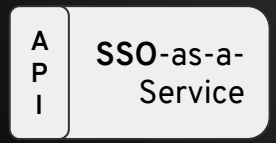
It is Serverless, because of SaaS (managed by another party services).

It is all about the Services

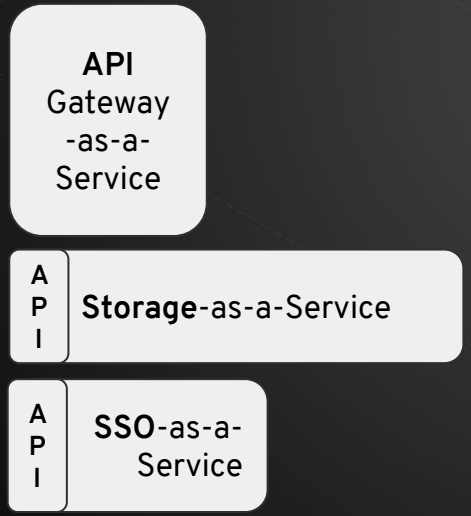
HTTP Input/Output Service

Authentication Service

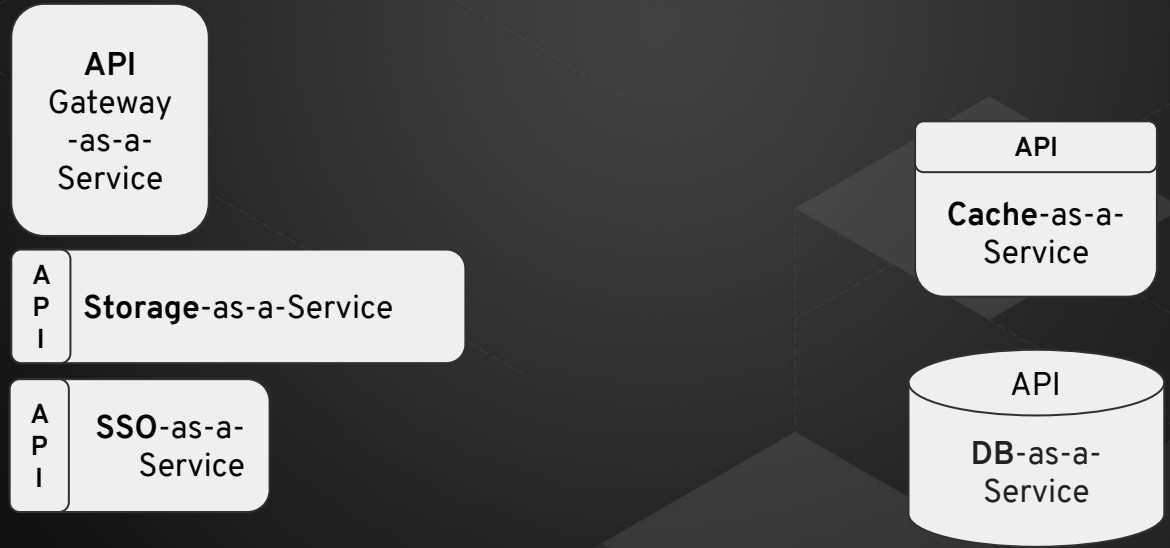




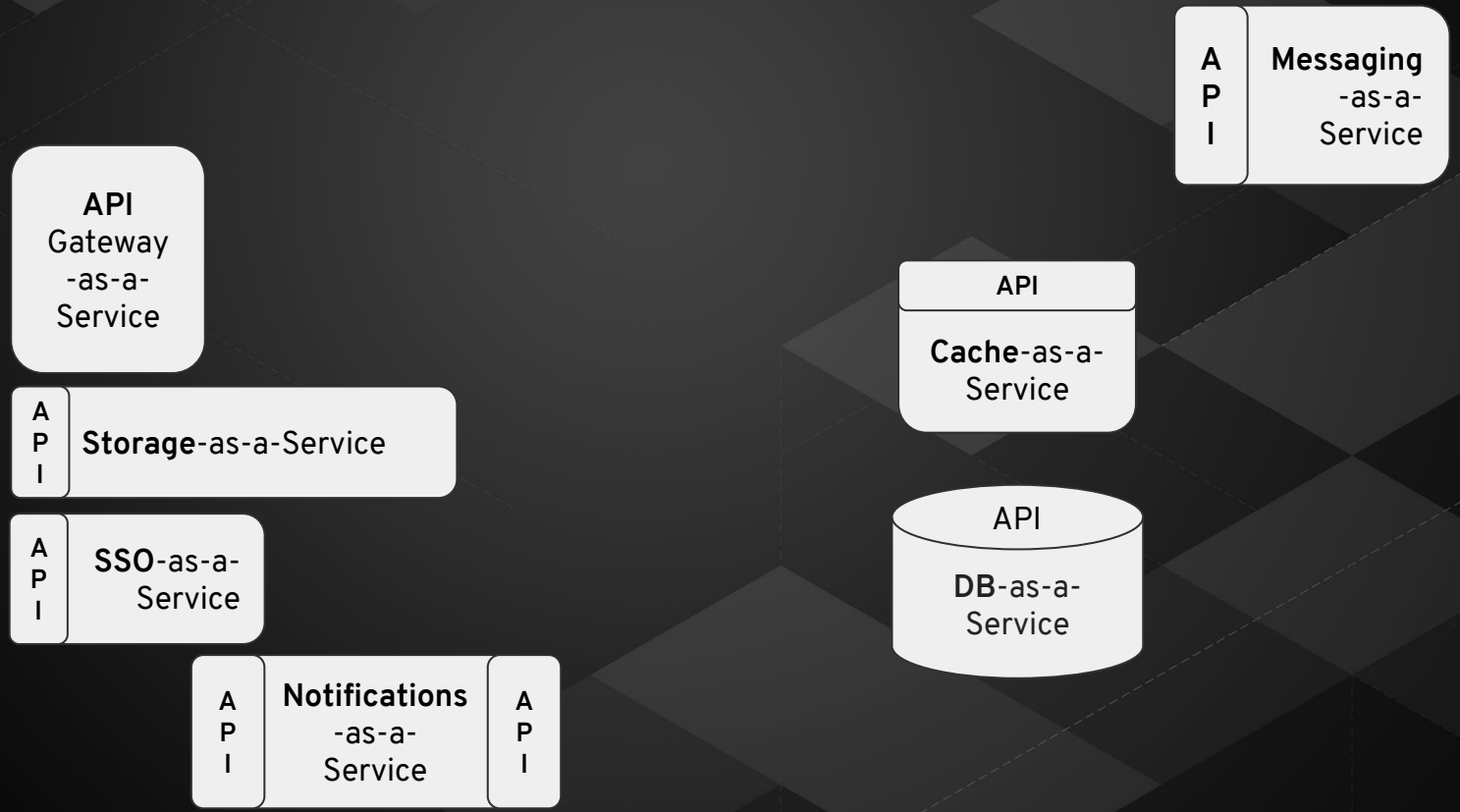
File Storage Service



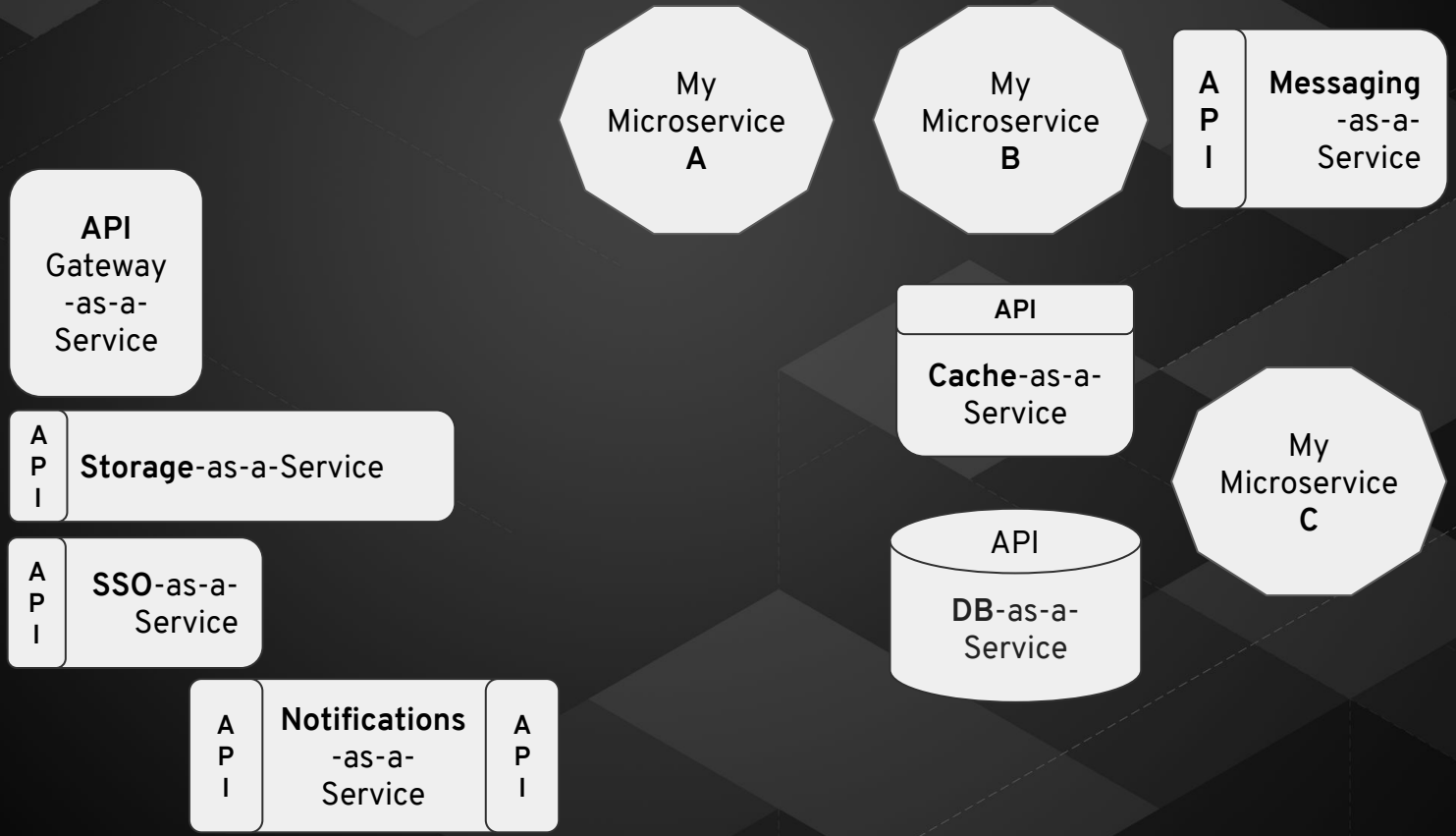
Data Services



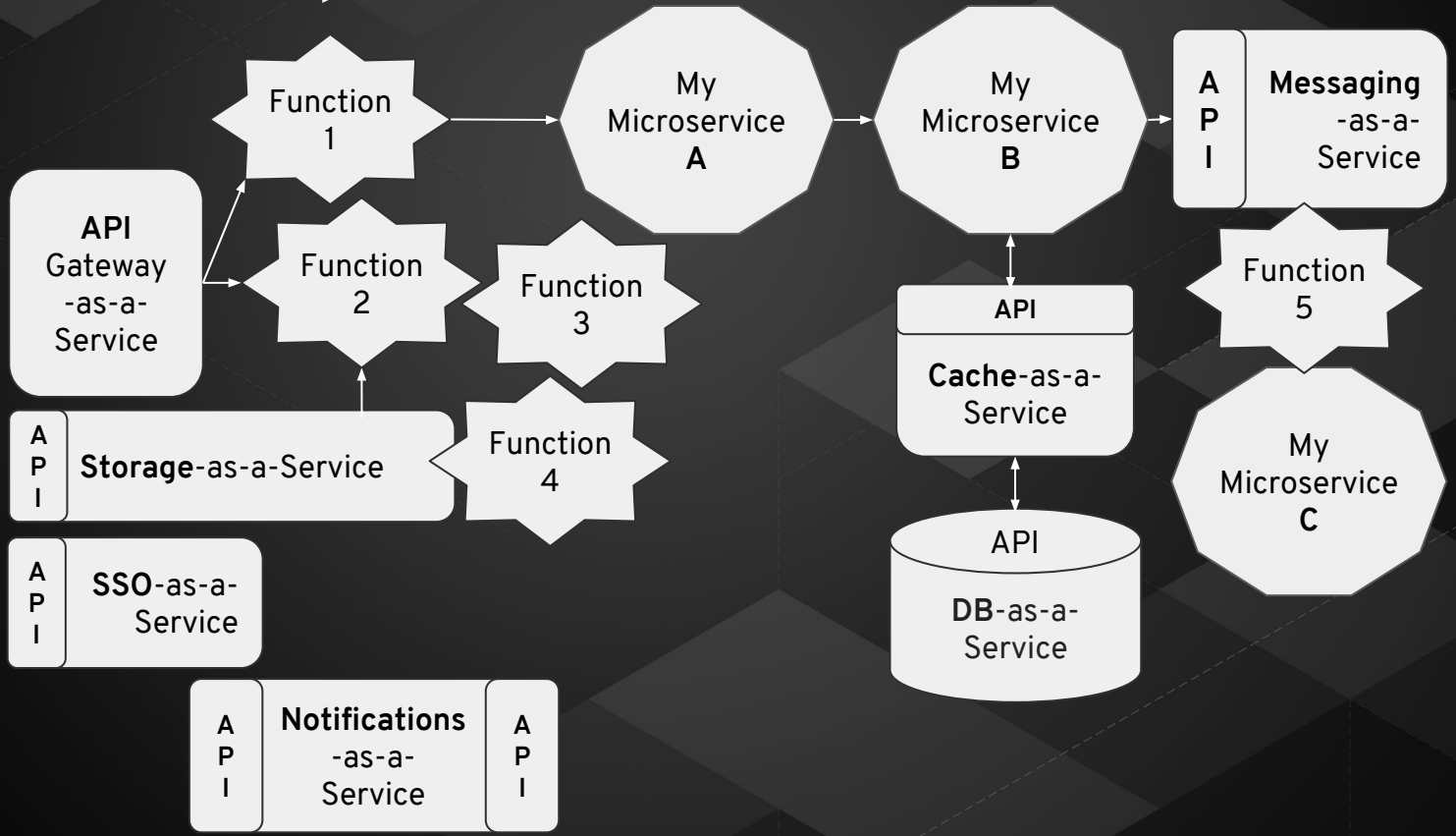
Connectivity Services



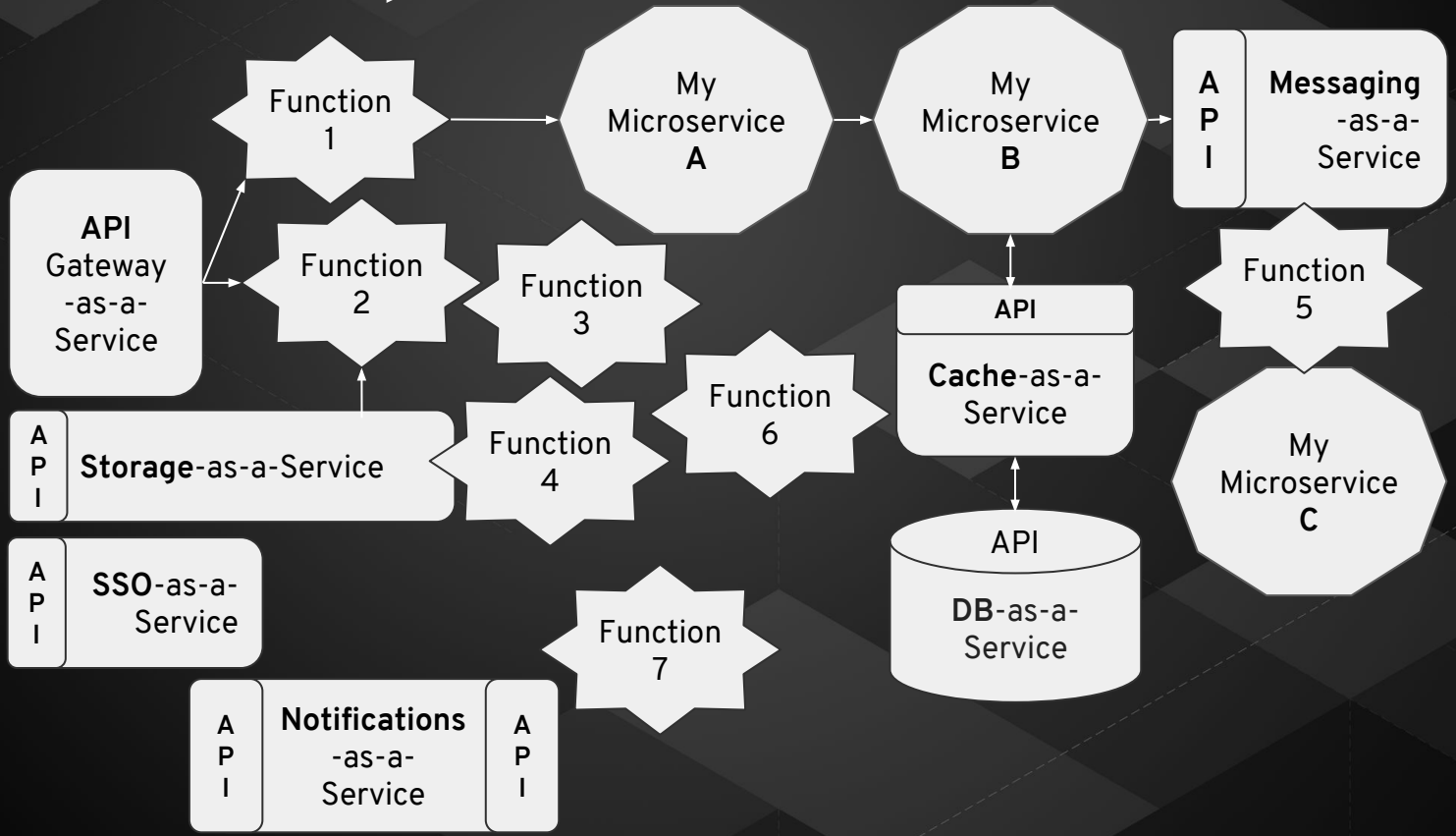
Your Containerized Services



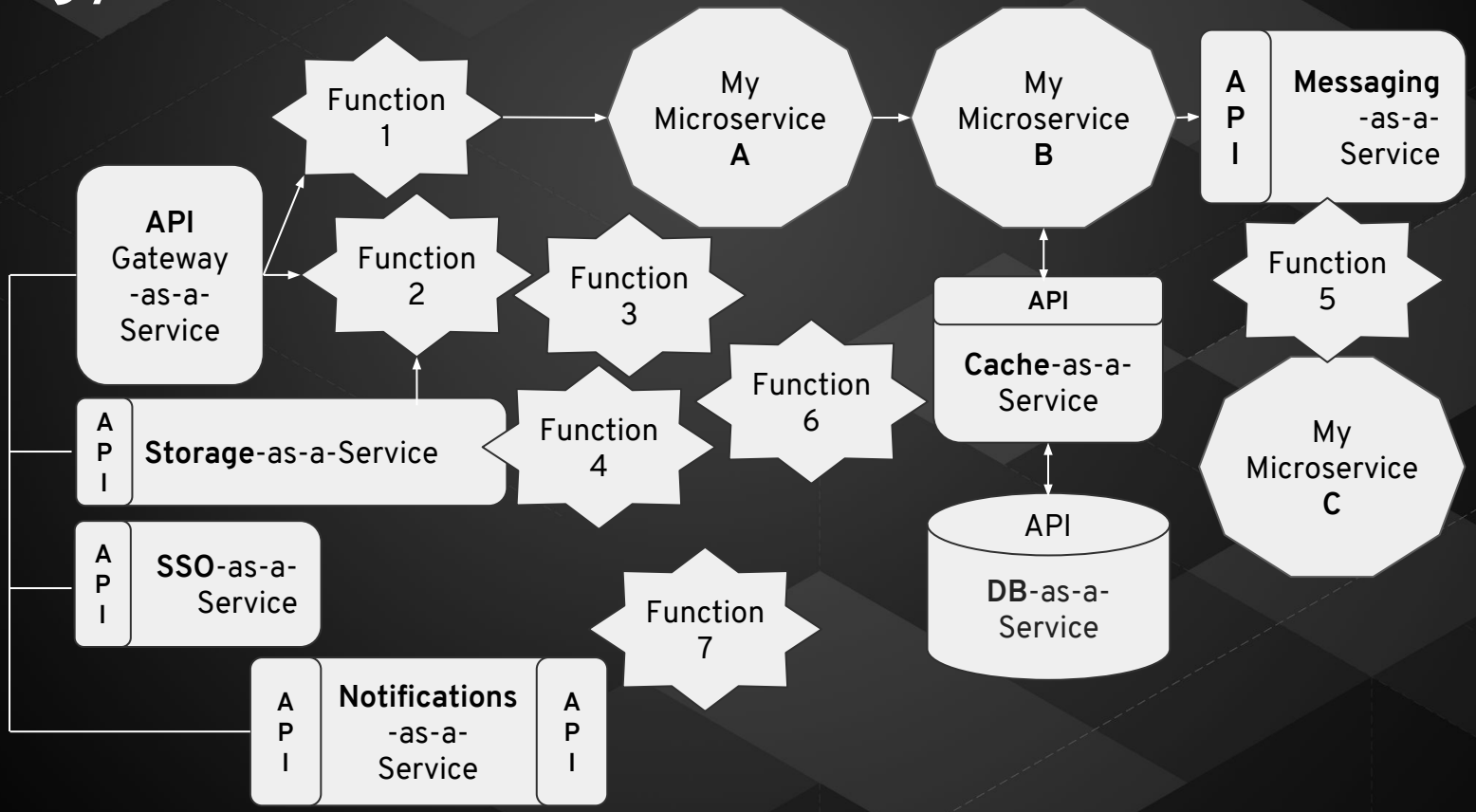
Event-Driven Input



Event-Driven Output



Synergy

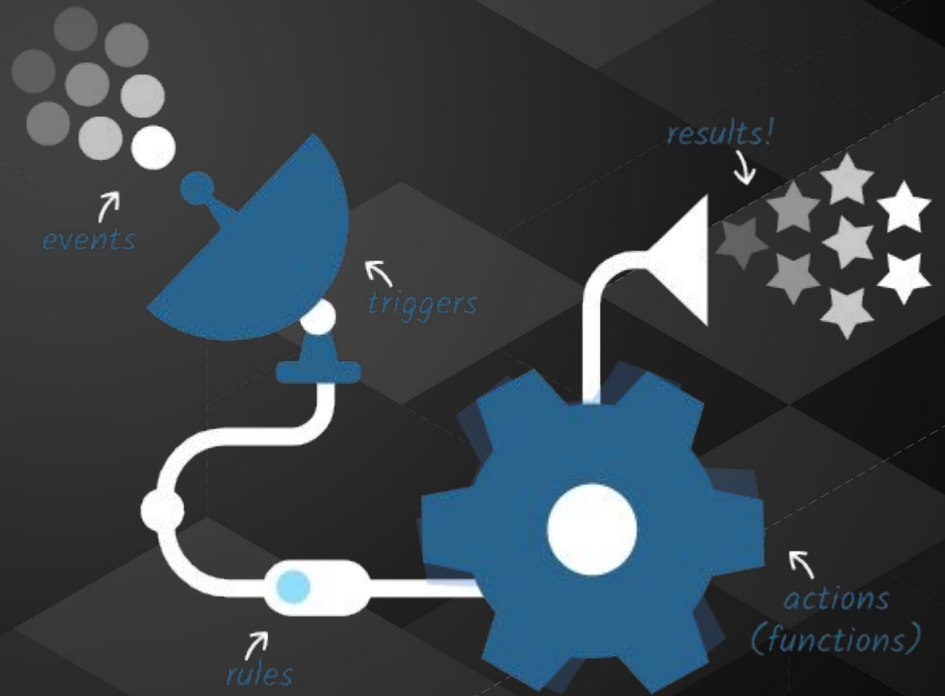

Function 1
My Microservice A
My Microservice B
API
Messaging -as-a-Service
API Gateway -as-a-Service
Function 2
Function 3
Function 5
API
Cache-as-a-Service
Function 6
API
Storage-as-a-Service
Function 4
My Microservice C
API
SSO-as-a-Service
API
DB-as-a-Service
Function 7
API
Notifications -as-a-Service
API

FaaS Kubernetes Players

APACHE
OpenWhisk™

Apache OpenWhisk

- Open Source incubating under Apache
- A Cloud platform to execute functions written in:
  - JavaScript
  - Swift
  - Java
  - Python
  - PHP
  - Docker
  - Go
- Deployable on
  - Any platform where docker can be run
  - Kubernetes/OpenShift

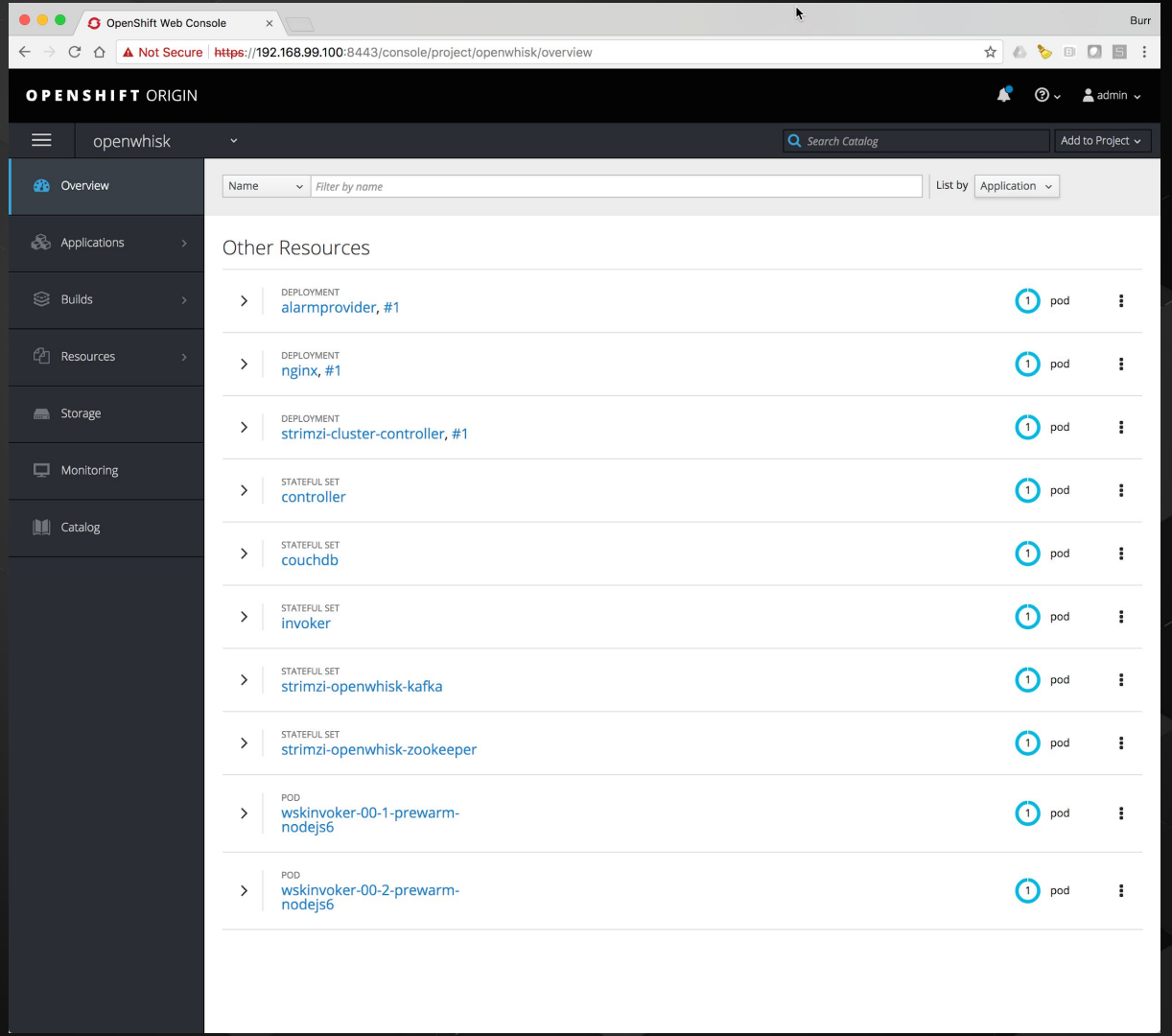


OpenWhisk

On

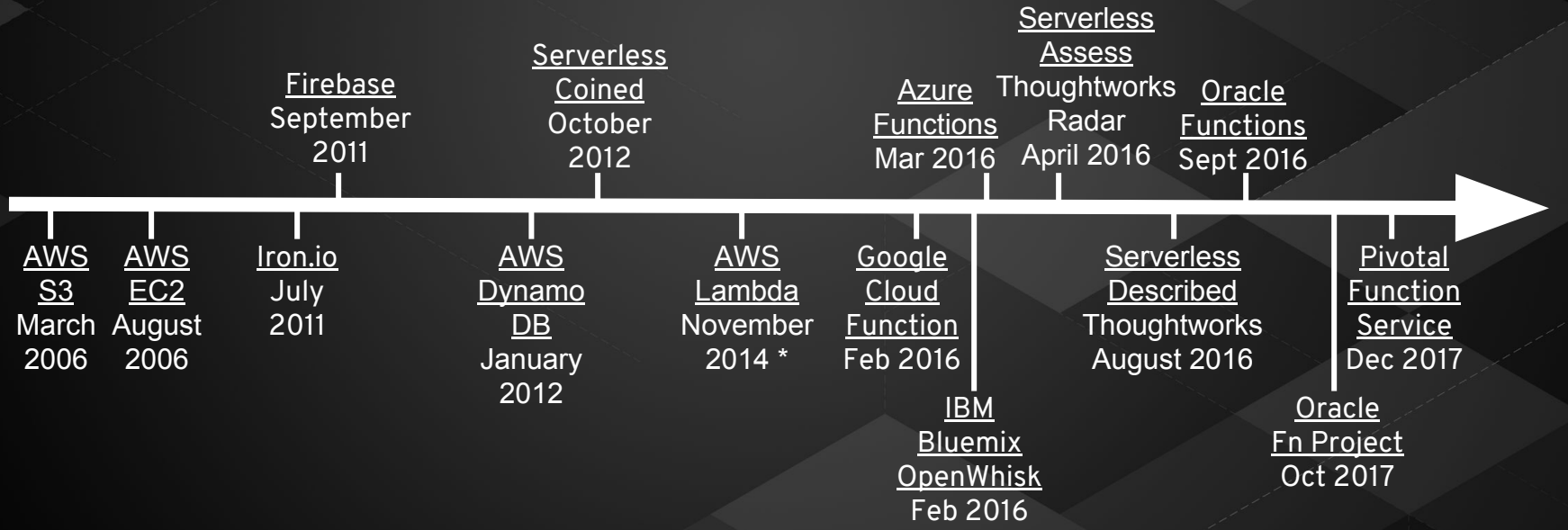
OpenShift

(bit.ly/faas-tutorial)



@danieloh30

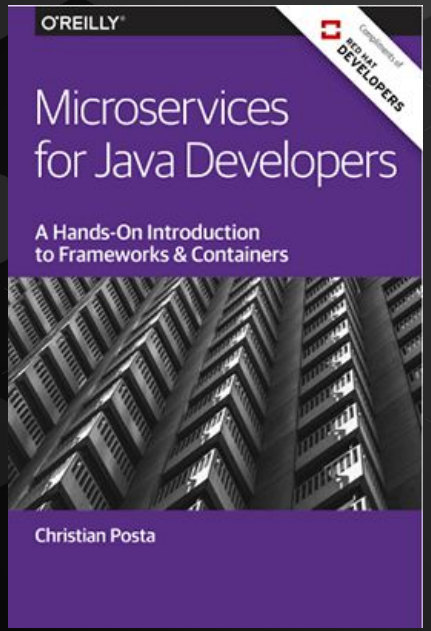
Short History of Serverless



* Only supports JavaScript
Only for stateless, short-lived, simple applications

RESOURCES

bit.ly/javamicroservicesbook



Free eBooks from developers.redhat.com

Microservices Introductory Materials

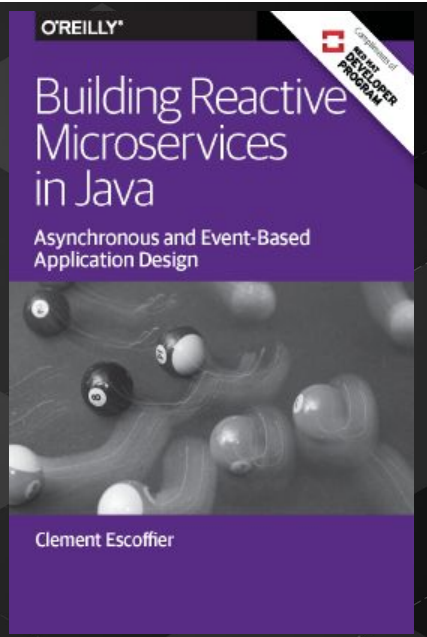
Demo: bit.ly/msa-instructions
Slides: bit.ly/microservicesdeepdive
Video Training: bit.ly/microservicesvideo
Kubernetes for Java Developers

Advanced Materials

bit.ly/istio-tutorial
learn.openshift.com/servicemesh

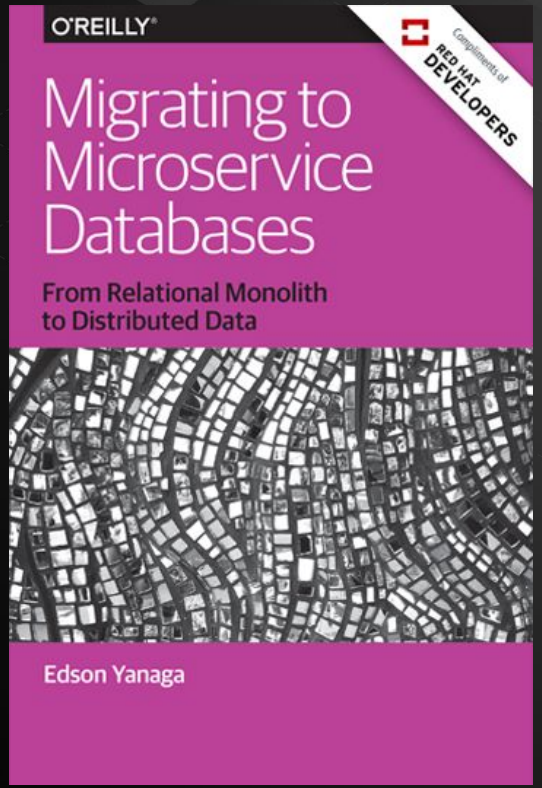
bit.ly/faas-tutorial
learn.openshift.com/serverless

bit.ly/reactivemicroservicesbook

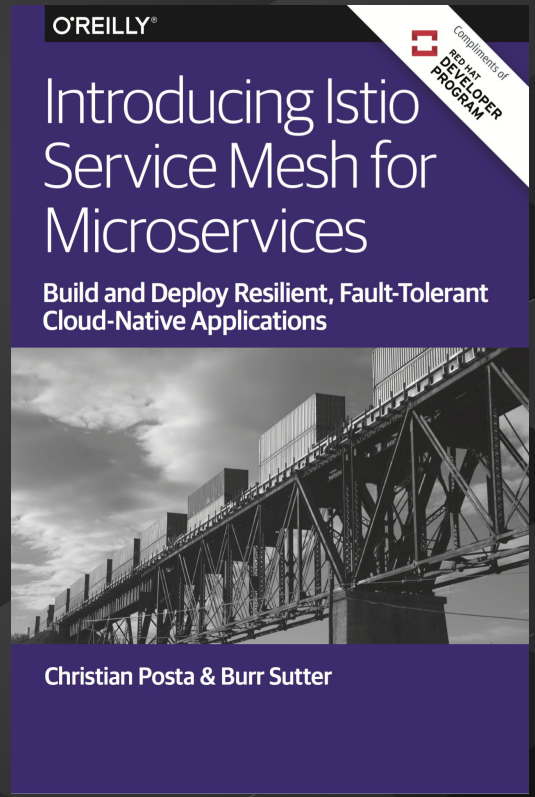


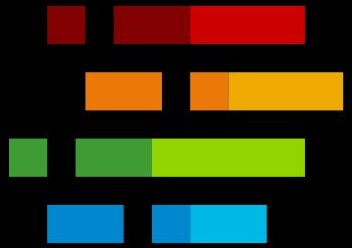
@danieloh30

bit.ly/mono2microdb



bit.ly/istio-book





RED HAT® DEVELOPER

Get software and know-how.
Get started with Red Hat technologies.

Join at **developers.redhat.com**.

@danieloh30

THANK YOU & QUESTION?

Contacting me: doh@redhat.com / @danieloh30