



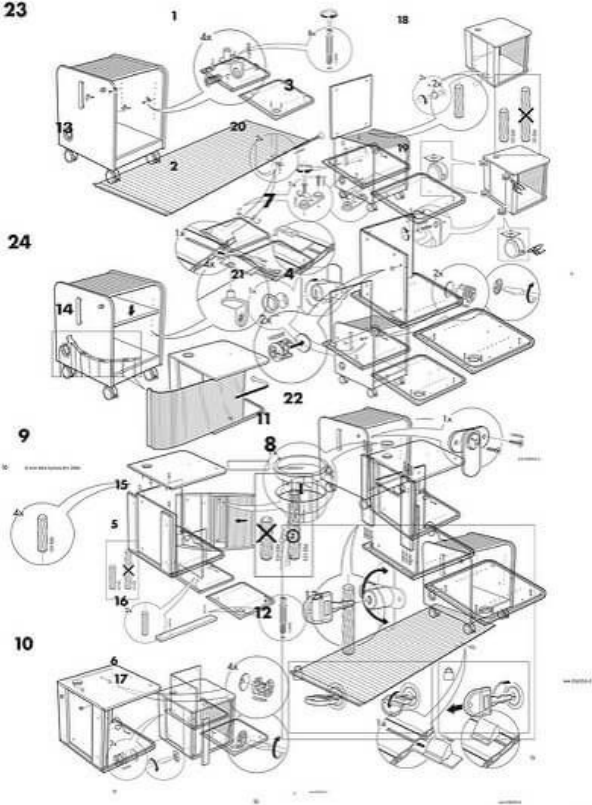
# OpenShift Service Mesh

It was supposed to be simple, but then it was *not*?

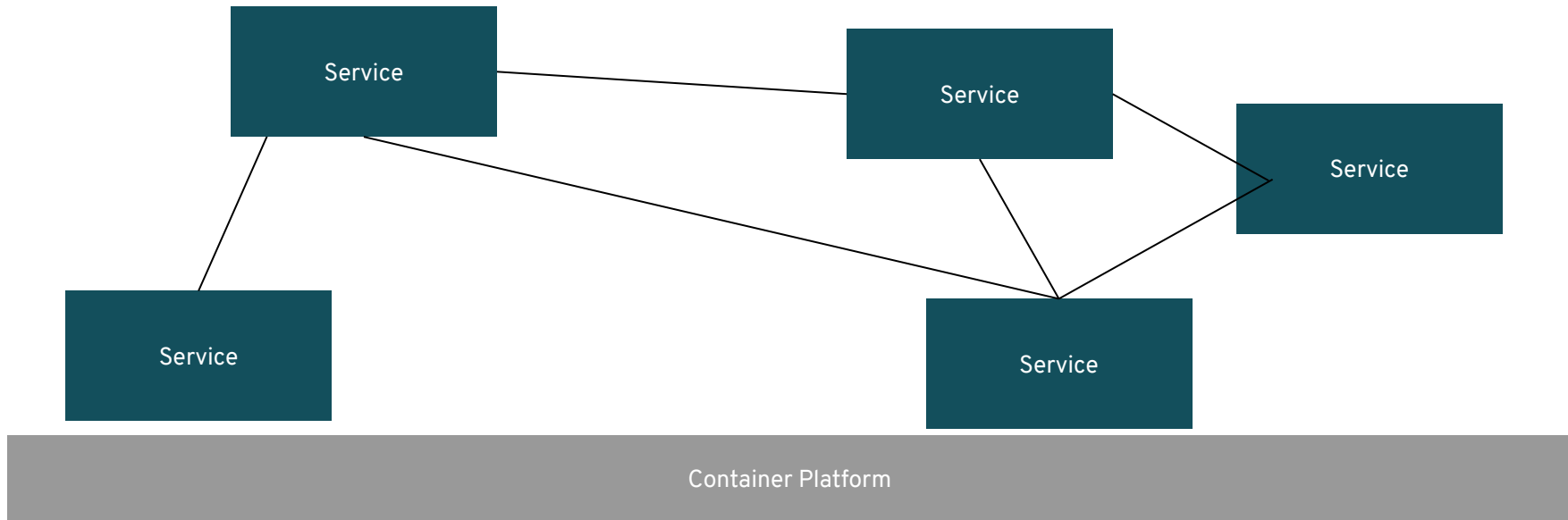
# Building Furniture



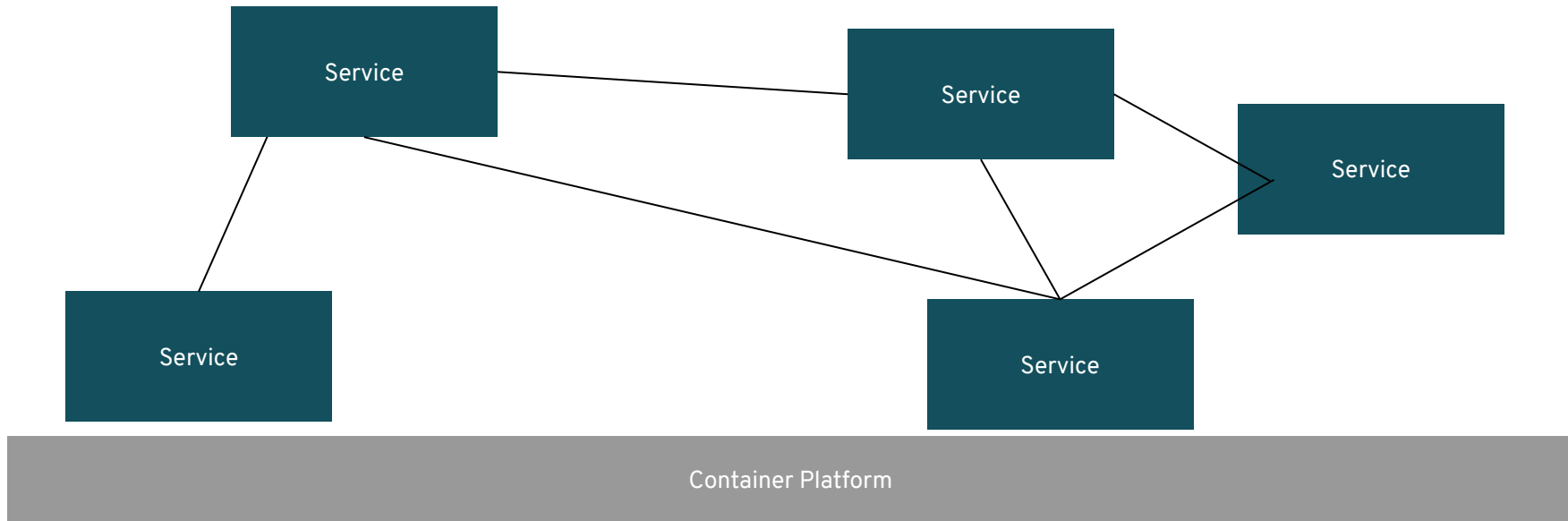
...Remember that one time you thought it would be simple?



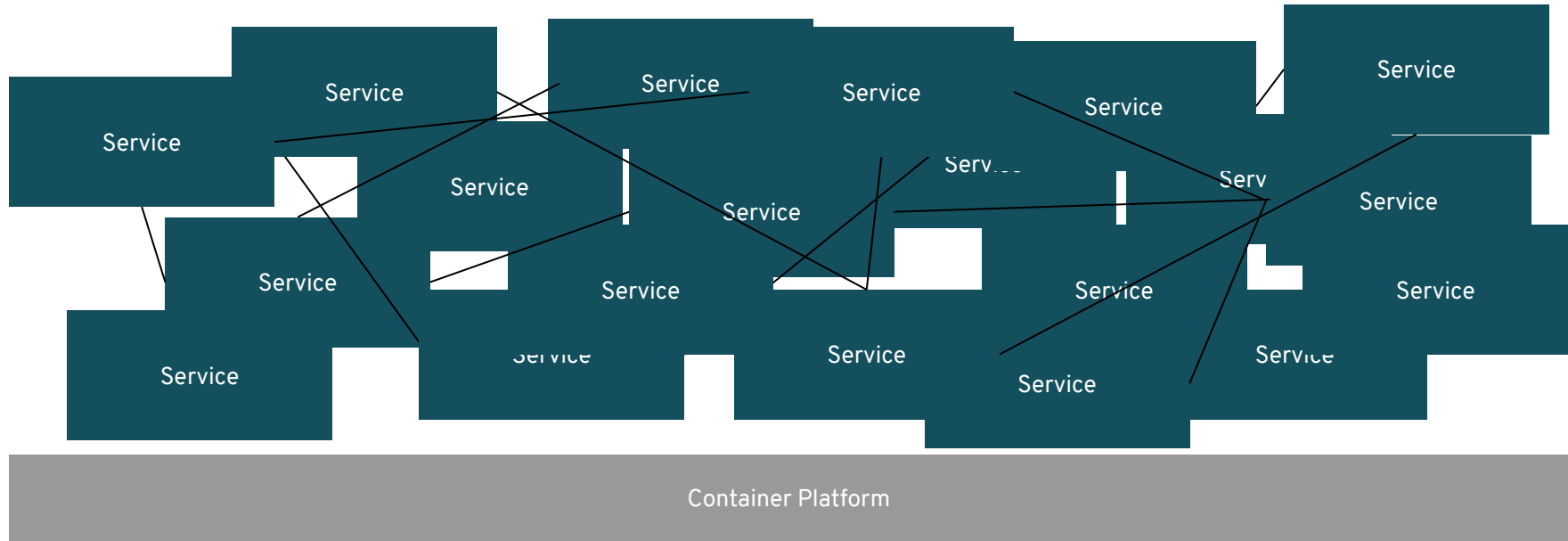
# Microservices!



# Microservices! Simple!



# Do microservices they said! It'll be easy they said!



# The Microservices Challenge

- Architecture Complexity
- Distributed Computing Problems Multiply
- Polyglot Architecture
- Diagnostic Mess



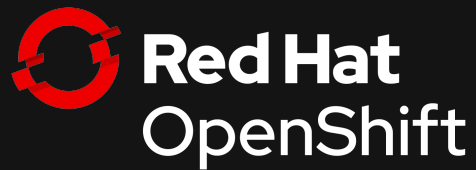


# Addressing the Microservices Challenge

- Defining the problem: distributed computing challenges
- Service Mesh Architecture
- The OpenShift Service Mesh
- Feature deep dive
- Extending Security to the API layer



# Distributed Computing Challenges



# DISTRIBUTED COMPUTING CHALLENGES

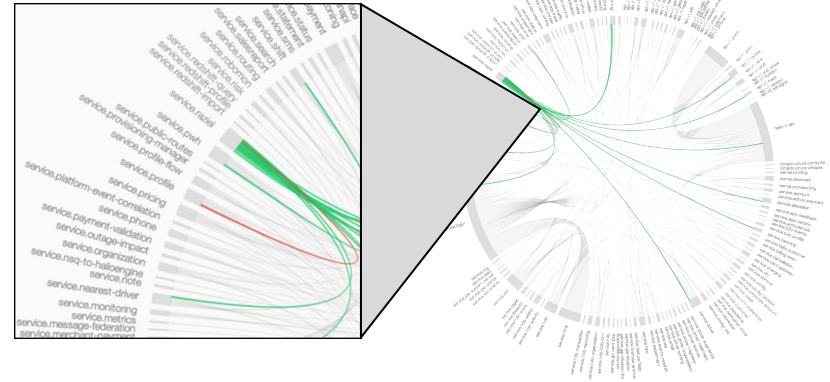
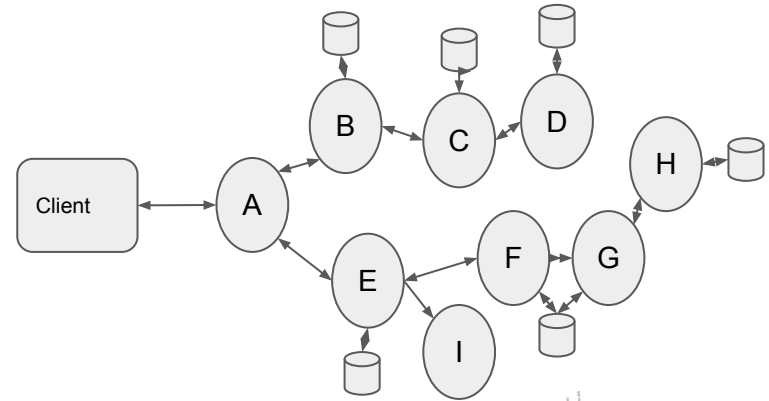
## Fallacies of Distributed Computing

- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.
- Topology doesn't change.
- There is one administrator.
- Transport cost is zero.
- The network is homogeneous.

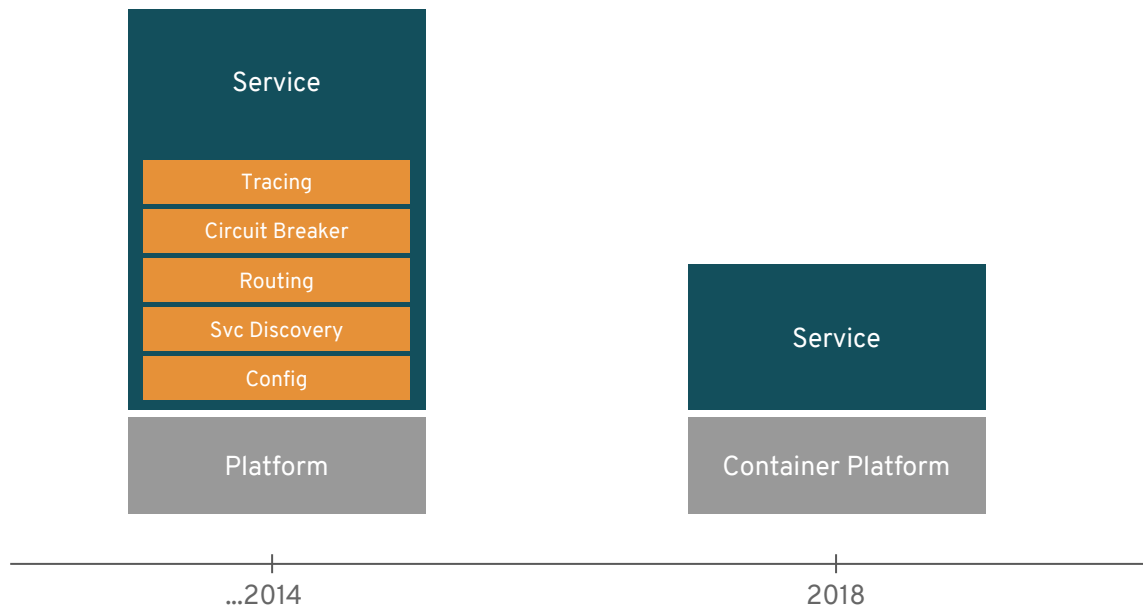
# MICROSERVICES ARE HARD

Because applications must deal with...

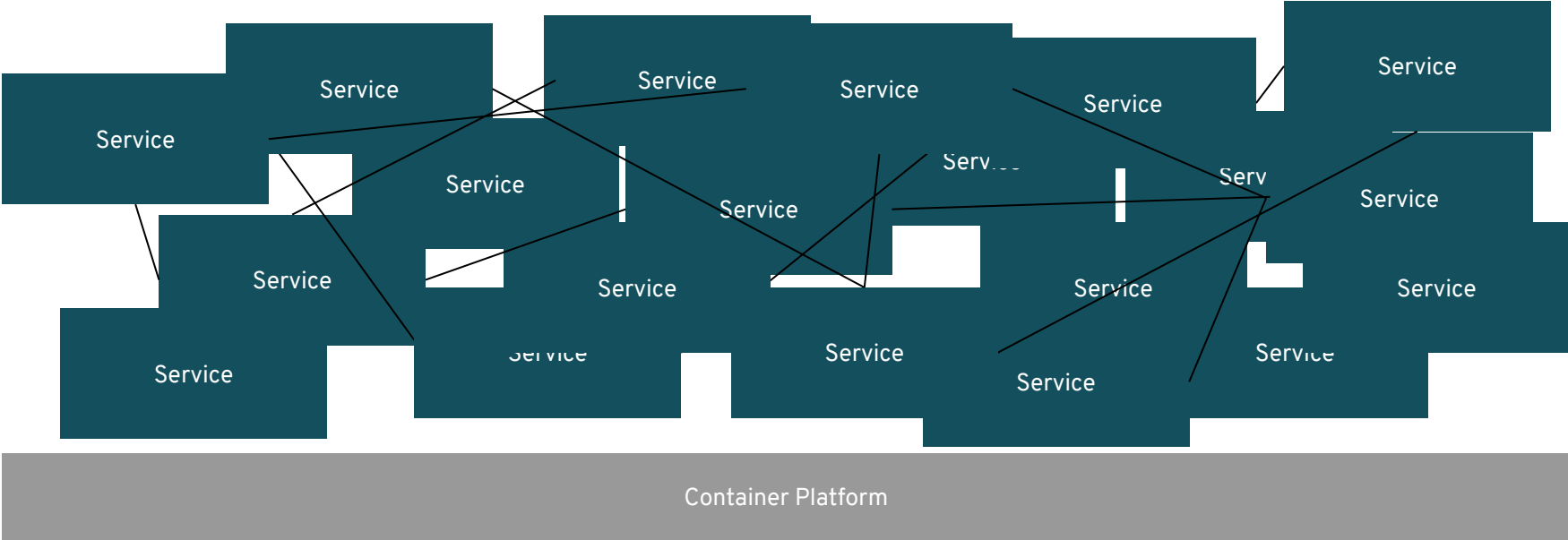
- Unpredictable failures
- End-to-end application correctness
- System degradation
- Topology changes
- Elastic/ephemeral/transient resources
- Distributed logs
- The fallacies of distributed computing



# Microservices Evolution



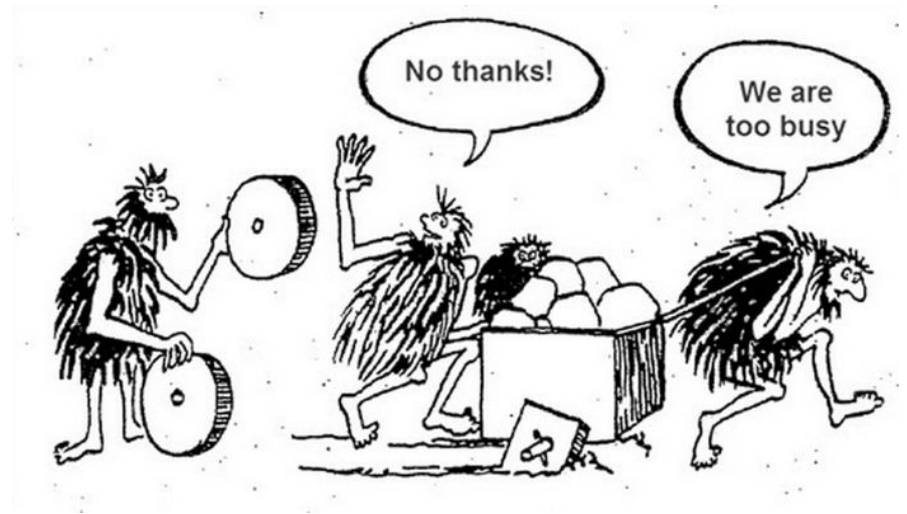
# SUPER Simple...



# POSSIBLE SOLUTIONS

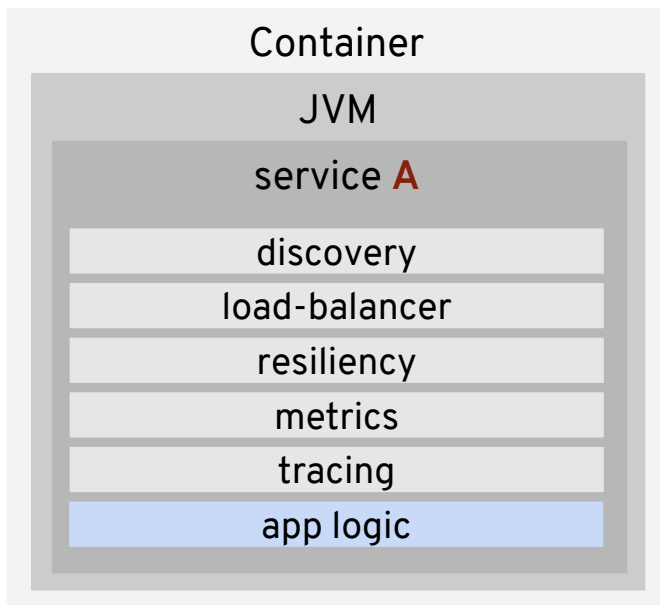
Have your developers do this:

- Circuit Breaking
- Bulkheading
- Timeouts/Retries
- Service Discovery
- Load Balancing
- Traffic Control



# NETFLIX

# OSS

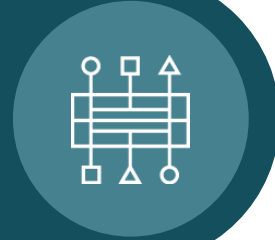


***Need a library to support each language/framework combination***



BUT WHAT ABOUT...?

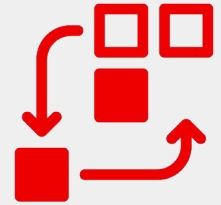
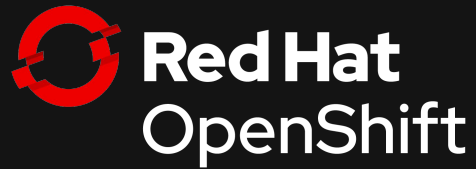
POLYGLOT  
APPS



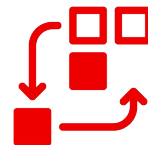
EXISTING  
APPS



# Service Mesh Architecture



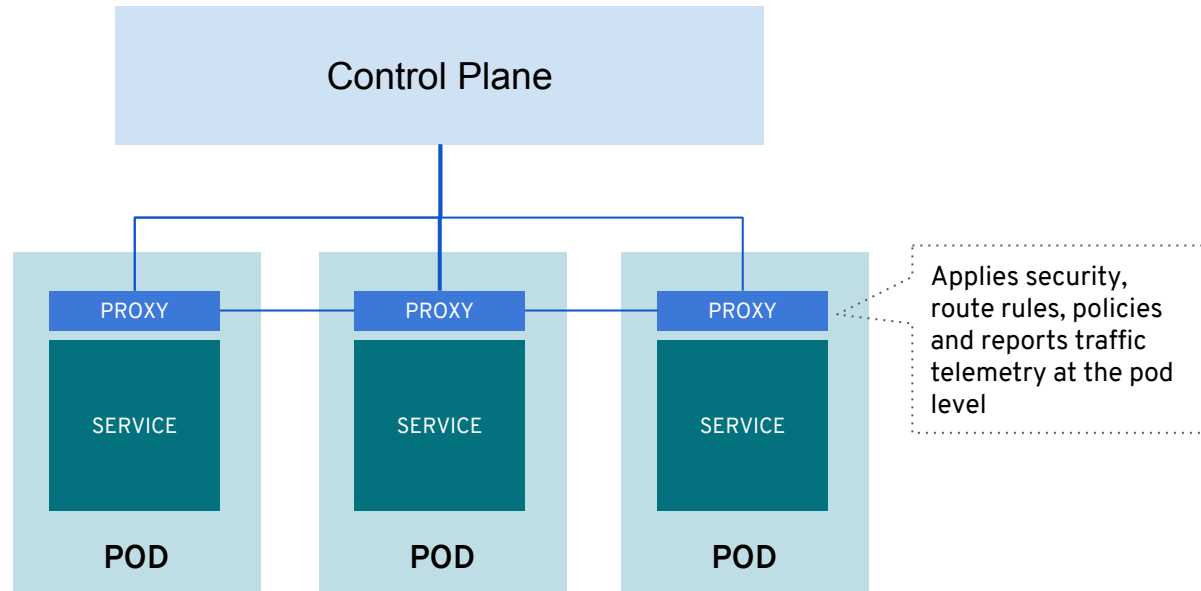
# Enter the Service Mesh



Infrastructure layer to help manage service-to-service communication, delivering enhanced security and traffic monitoring for microservices applications.

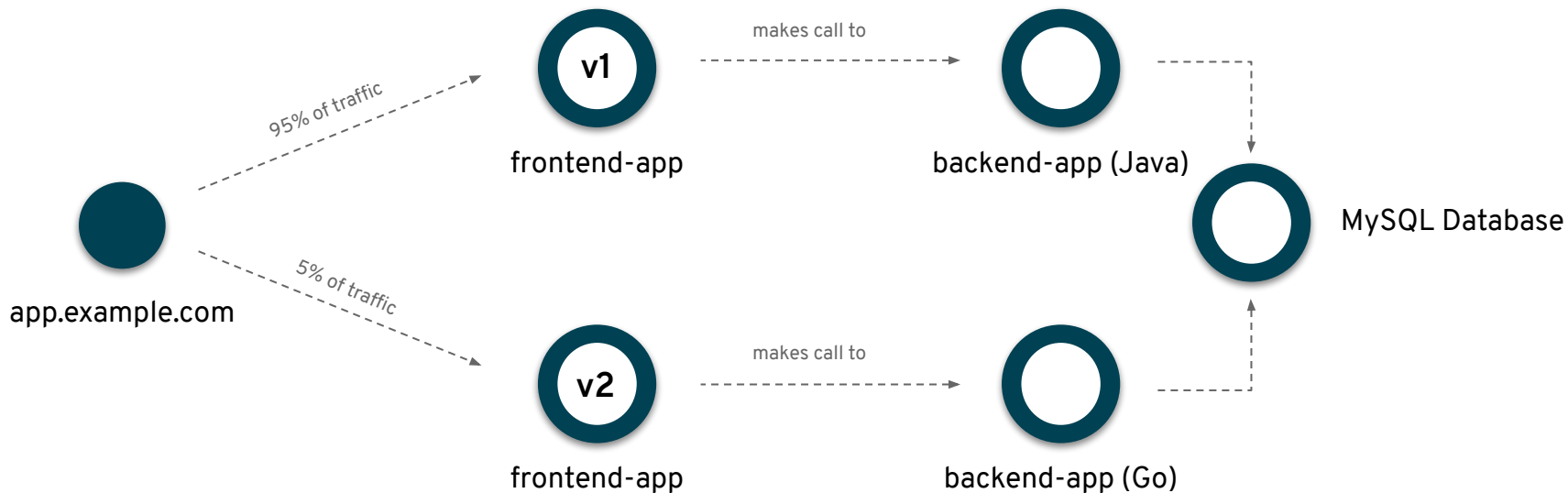
- Load balancing
- Routing rules
- Service monitoring and logging
- Secure cross-service communications

# What does the Service Mesh look like?



# *Simplify* with a Service Mesh

Control flow of traffic between application components

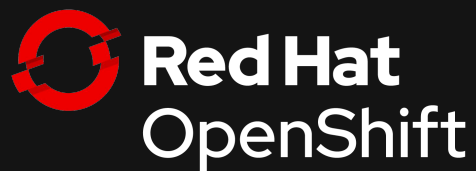


# Limitations of a Service Mesh

On its own, the Service Mesh is just the **communication** layer

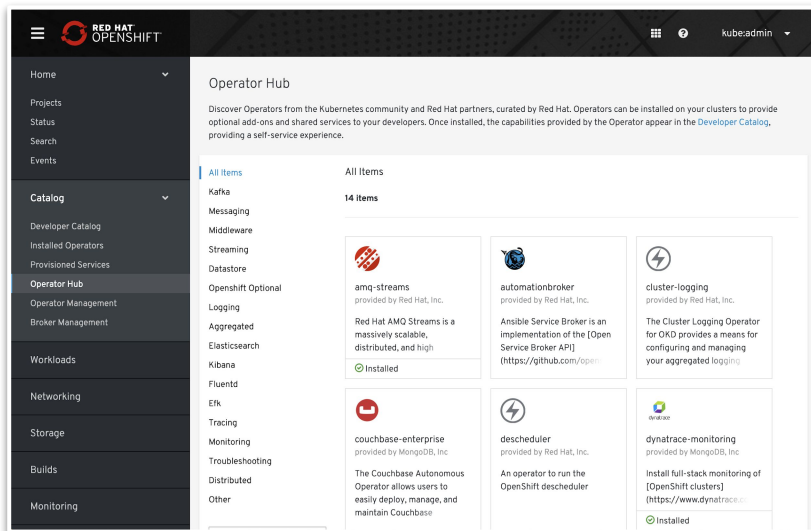
- Limited measurement functionality
- Limited observation capabilities
- Not a complete set of tools developers need to build and deploy microservices

# OpenShift Service Mesh



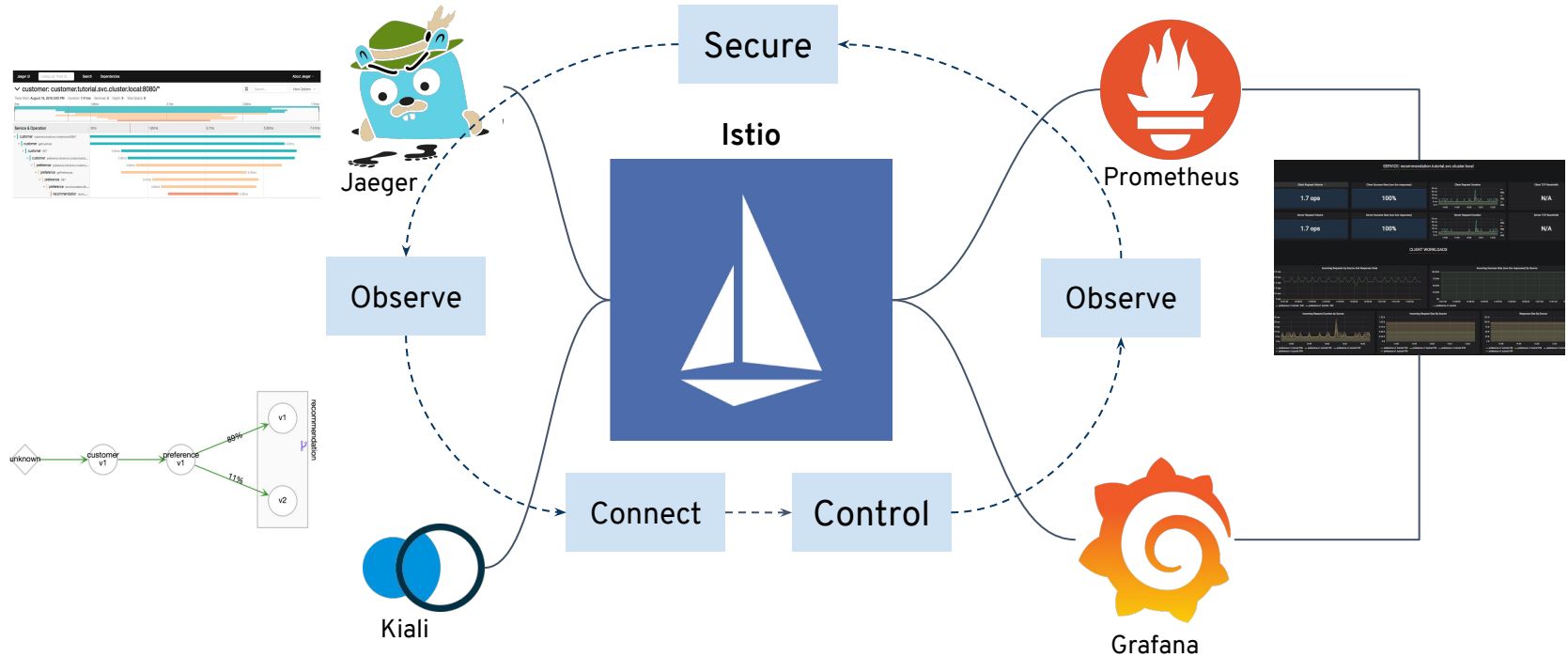
# OpenShift Service Mesh

- OpenShift Service Mesh is available and supported at *no additional cost* with OpenShift 4
- OpenShift Service Mesh Operator is found in the OperatorHub menu





# OpenShift Service Mesh



# OpenShift Service Mesh

## BENEFITS

- Complete service mesh packaged for ease of use
- Built with key open source projects and integrations
- Extend security through the service mesh into the API layer with with 3scale API management integration

## USE CASES

- Adaptive traffic management
- Monitoring and alerting
- Service performance tracing
- Secure communications and API access
- Foundation for Serverless Knative functionality

# Enhanced Visualization of Cluster Traffic with Kiali

Visualization of what matters most:

- Application Topology
- Traffic throughput
- Error Rates
- Service Latency
- Service Versioning



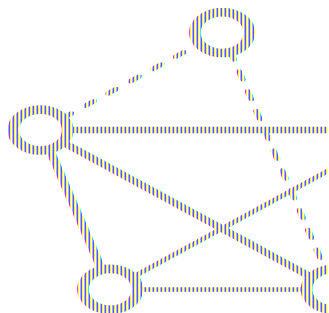
# OpenShift Service Mesh Uses

1. Service Mesh visualization
2. Tracing with Jaeger
3. Routing
4. Service versioning
5. Mutual TLS
6. Authentication (e.g. RBAC with JWT)
7. Whitelisting/Blacklisting Auth
8. Retries, timeouts, circuit breakers
9. Rate limiting
10. Fault injection

# OpenShift Service Mesh

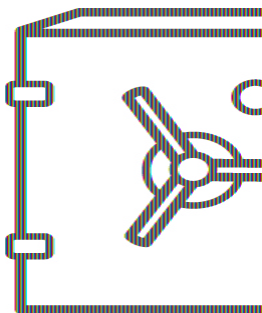
## Connect

Control the flow of traffic between services



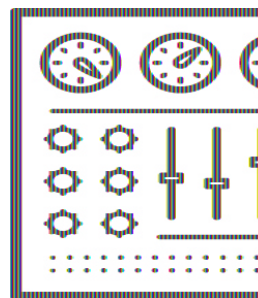
## Secure

Application independent security



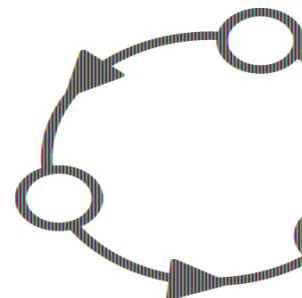
## Control

Uniform abstraction for policy control

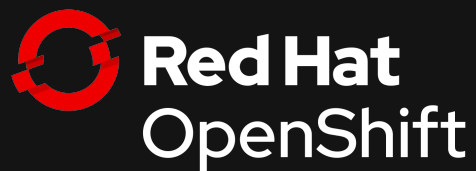


## Observe

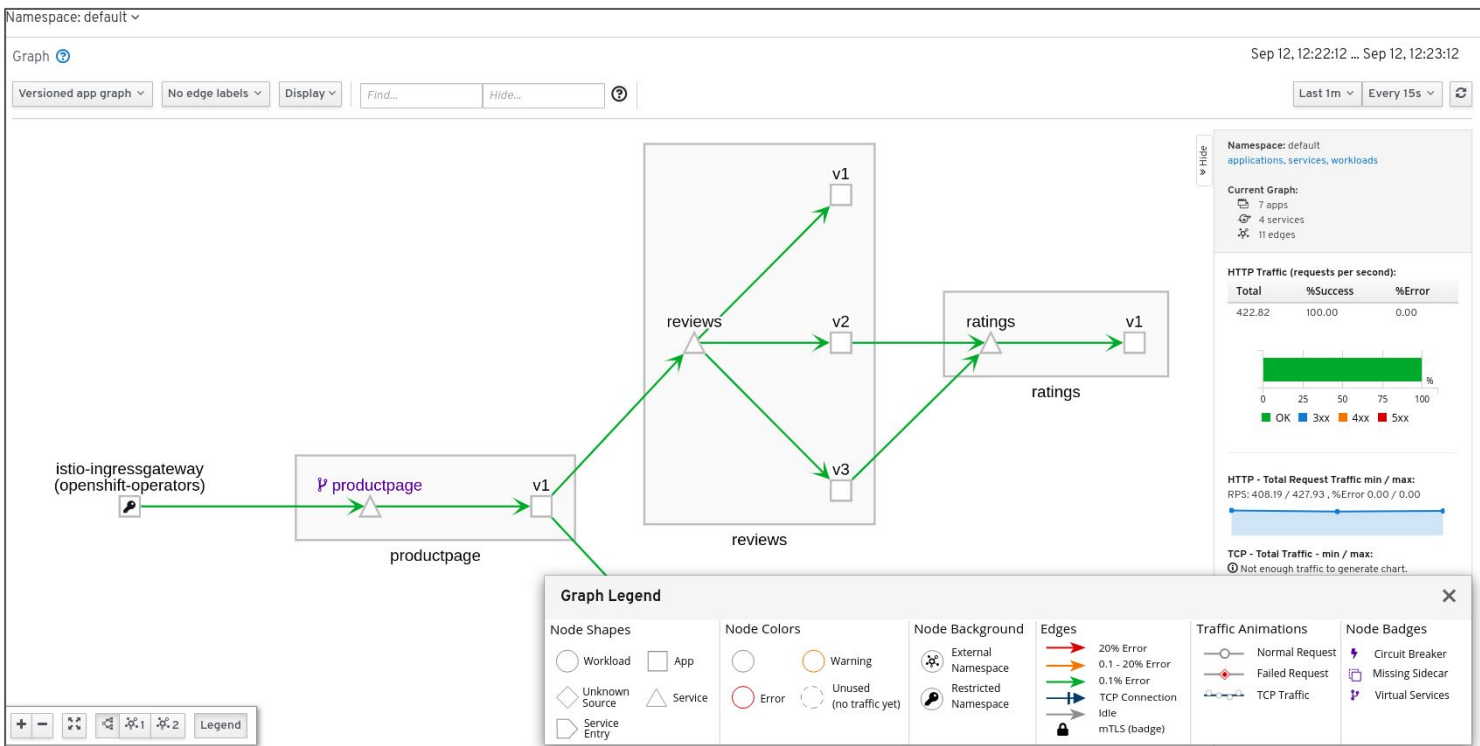
Visibility into application deployments



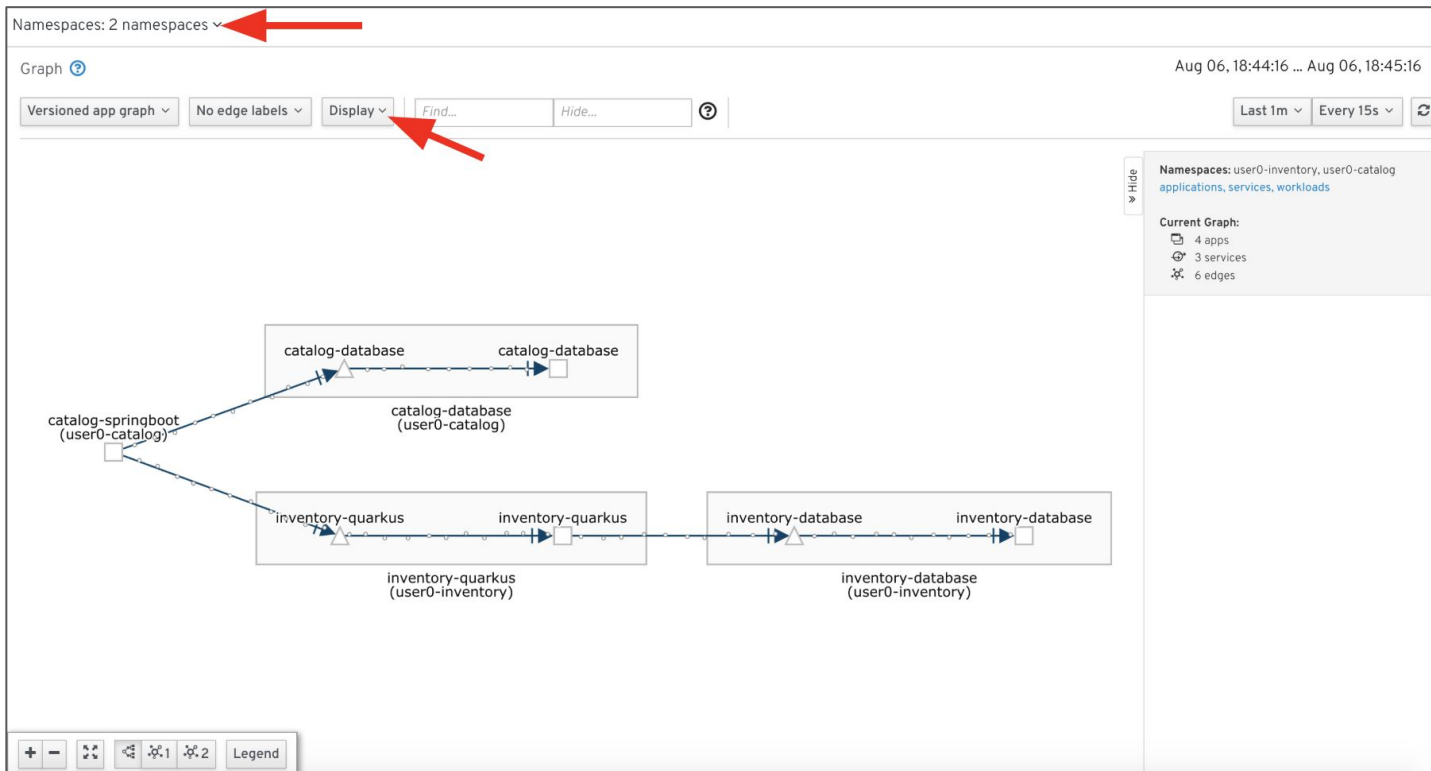
# Visualization Deep Dive



# Application Topology & Service Versioning



# Cross-Application Views





# Traffic Throughput

Namespace: user0-bookinfo

Graph

Versioned app graph | No edge labels | Display | Hide... | Last 1m | Every 15s

- Node Names
- Service Nodes
- Traffic Animation
- Unused Nodes

Badges:

- Circuit Breakers
- Virtual Services
- Missing Sidecars
- Security

```
graph LR; ingress[istio-ingressgateway (istio-system)] --> productpage[v1]; productpage --> details[v1]; productpage --> reviews; reviews --> details; reviews --> ratings; reviews --> reviews; reviews --> v3[v3]; reviews --> v2[v2]; v2 --> ratings; v3 --> ratings; ratings --> ratings_v1[v1];
```

istio-ingressgateway (istio-system)

productpage

reviews

details

ratings

Namespace: user0-bookinfo  
applications, services, workloads

Current Graph:  
7 apps  
4 services  
11 edges

HTTP Traffic (requests per second):

Total	%Success	%Error
9.49	100.00	0.00

0 25 50 75 100 %

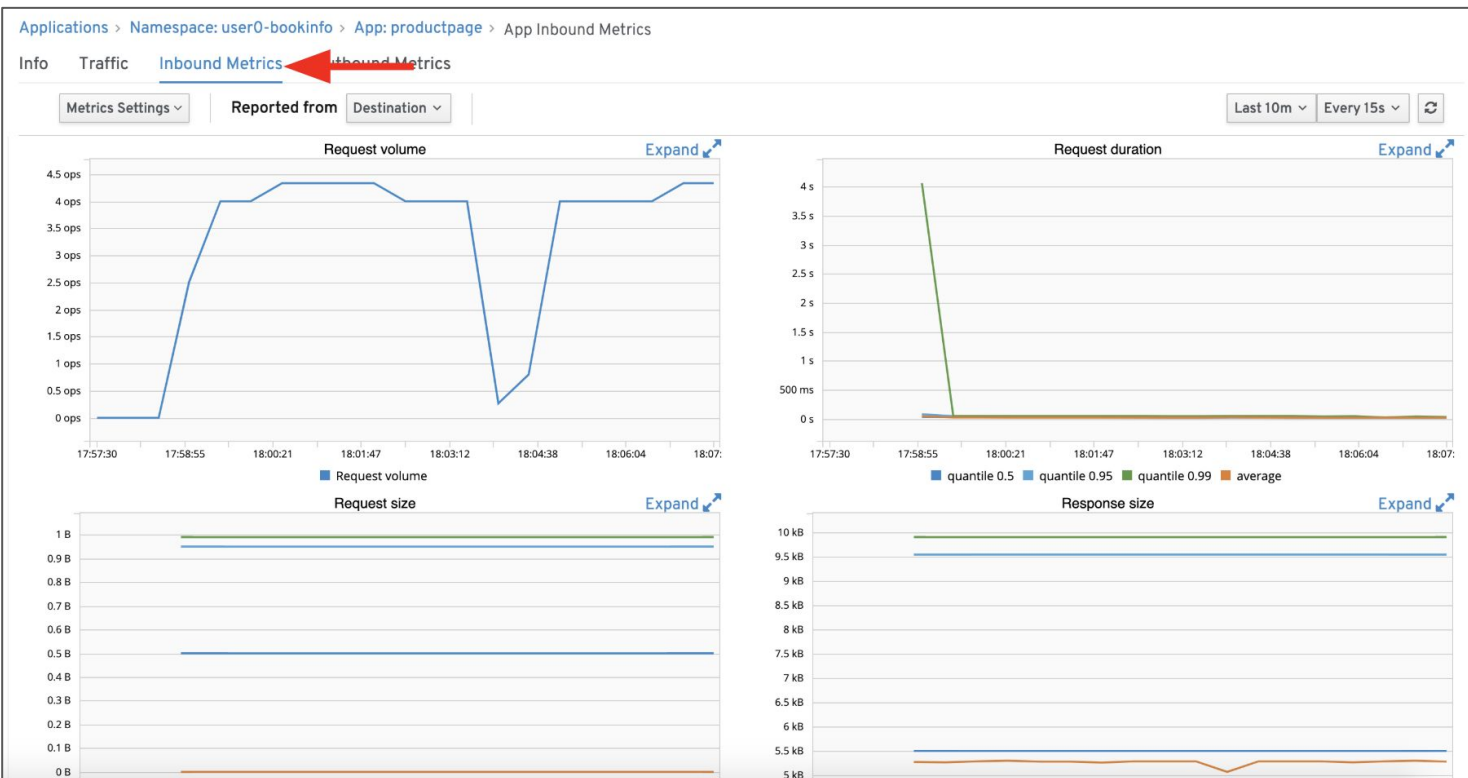
OK 3xx 4xx 5xx

HTTP - Total Request Traffic min / max:  
RPS: 0.93 / 15.07, %Error 0.00 / 0.00

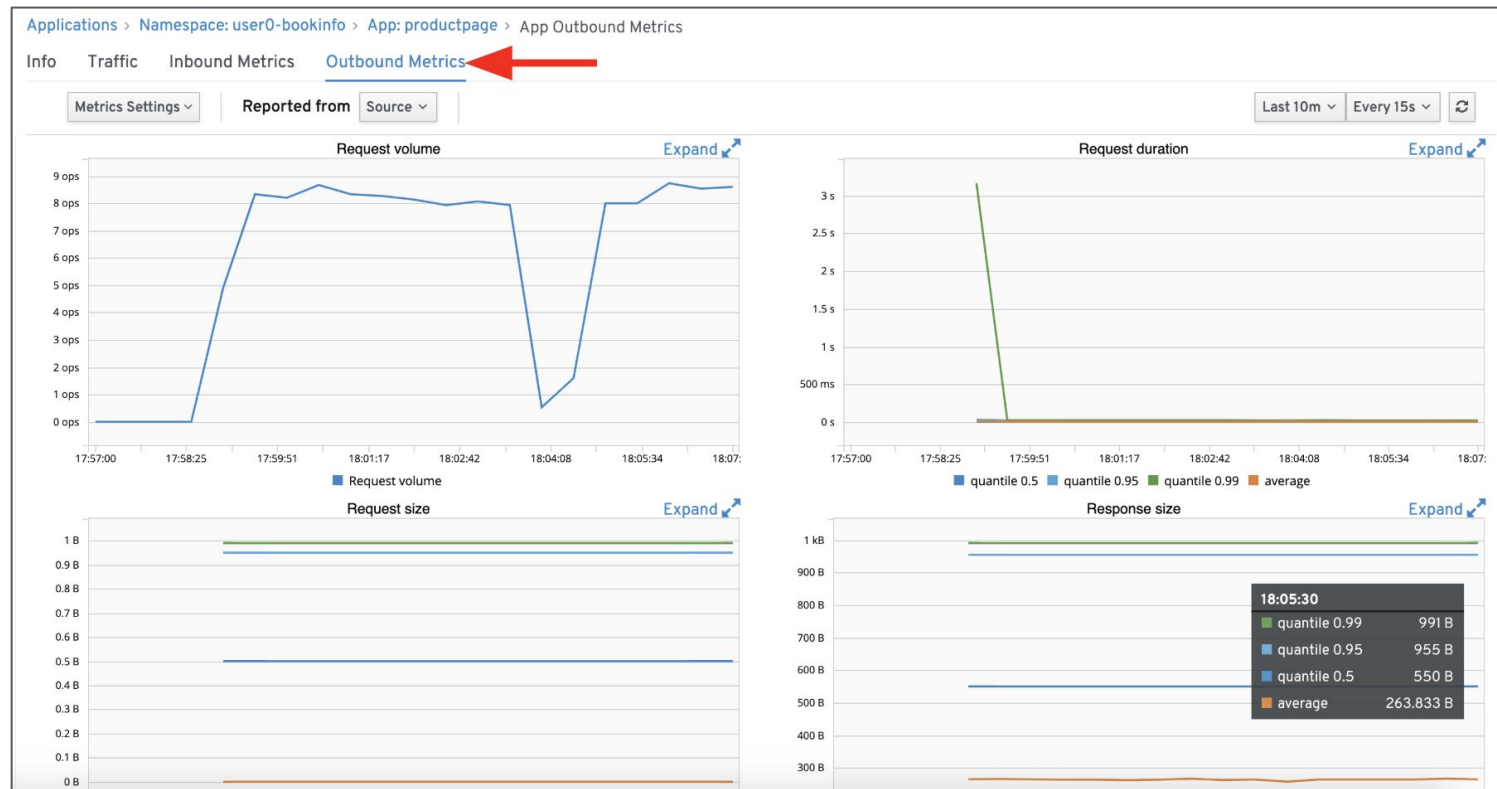
TCP - Total Traffic - min / max:  
⊘ Not enough traffic to generate chart.

+ - [Refresh] [Back] [Forward] [Zoom 1] [Zoom 2] Legend

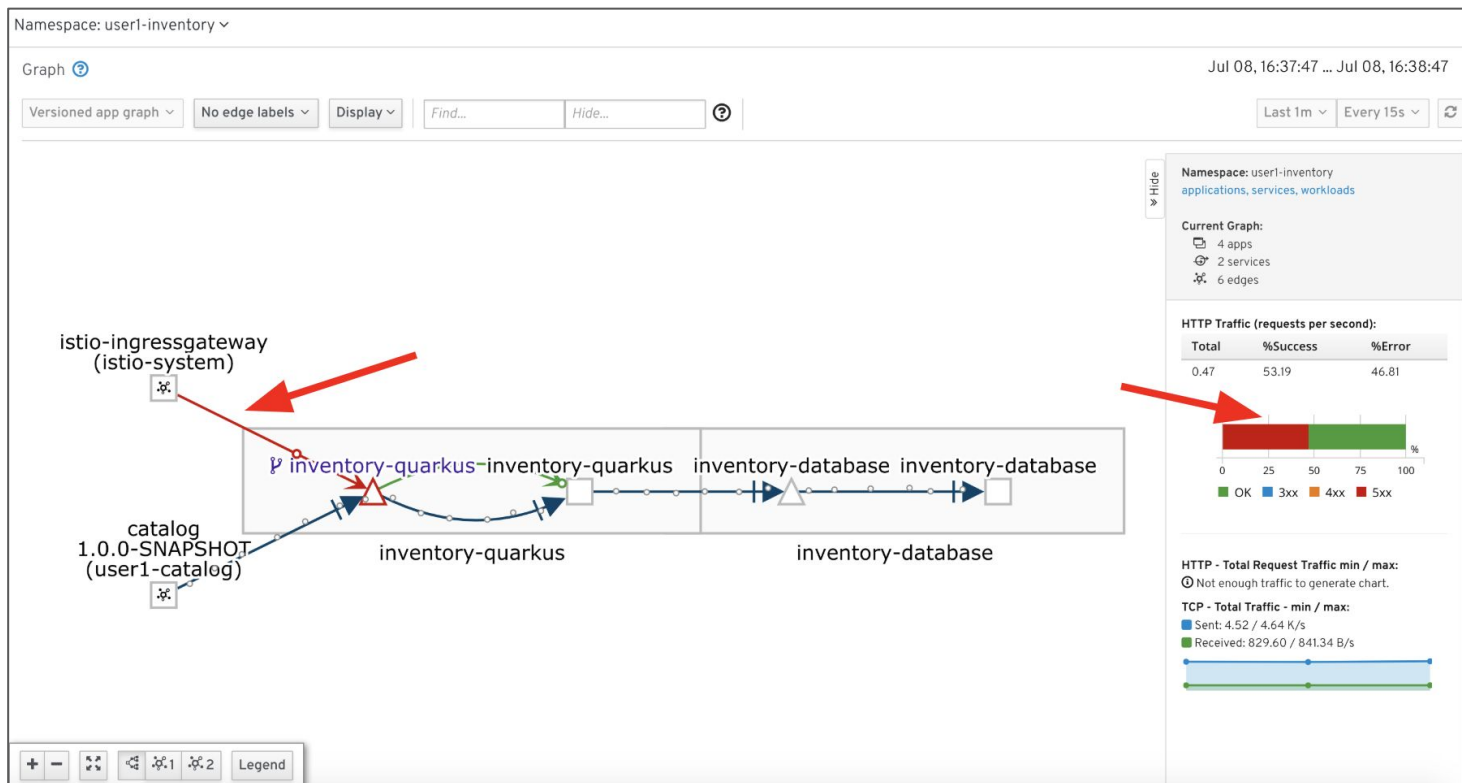
# Traffic Throughput Details: Inbound



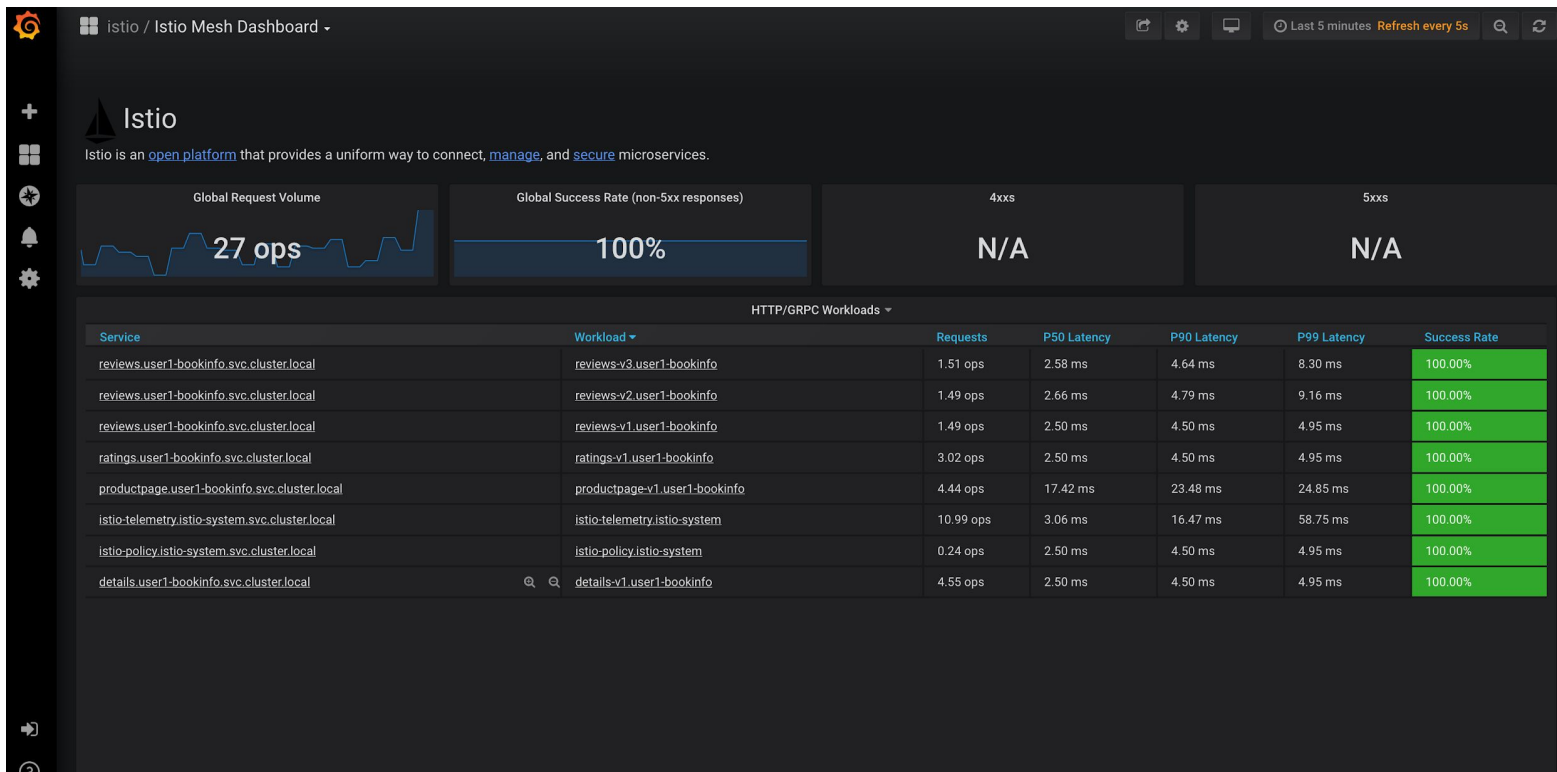
# Traffic Throughput Details: Outbound



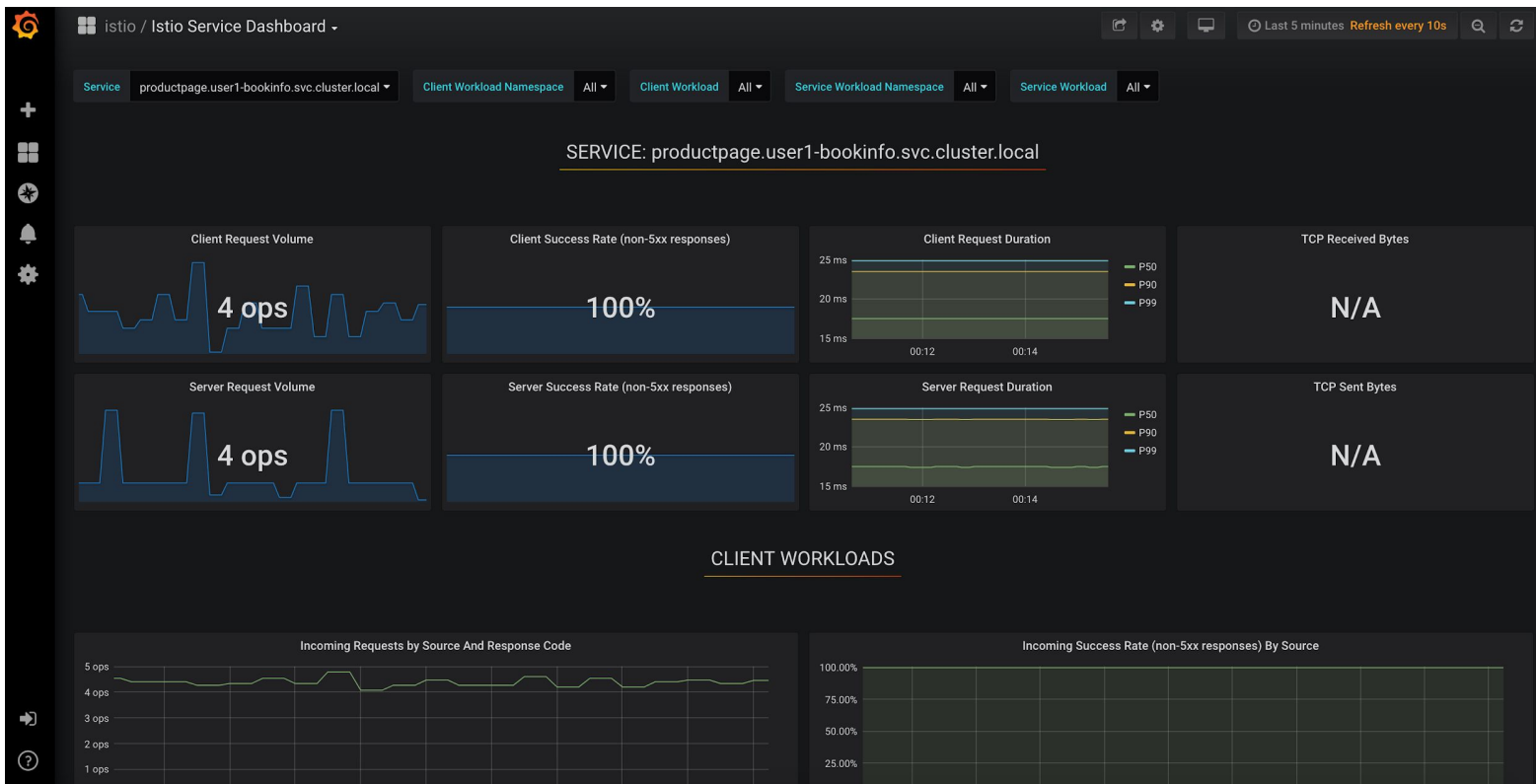
# Errors and Error Rates



# Global Dashboarding



# Global Dashboarding



# Distributed Tracing with Jaeger



- Discover service relationships and process times, transparent to the services
- Visualize the service execution times across the application
- Identify potential latency issues in each service



# Service Mesh Use Cases

Service mesh enables a number of capabilities and key use cases:

- Traffic Management
- Production Support: Observability
- Production Support: Application Tracing
- Role and auth checks between applications/Lines of Business
- “Zero-trust” Network



# Reducing Installation and Management Overhead



## Kubernetes Operator model

- Single package Install
- Service Mesh Operator reduces complexity to get running quickly
- Installation, configuration and updates of all components in one place
- Best practices and human operational knowledge baked-in for installation, configuration and upgrades

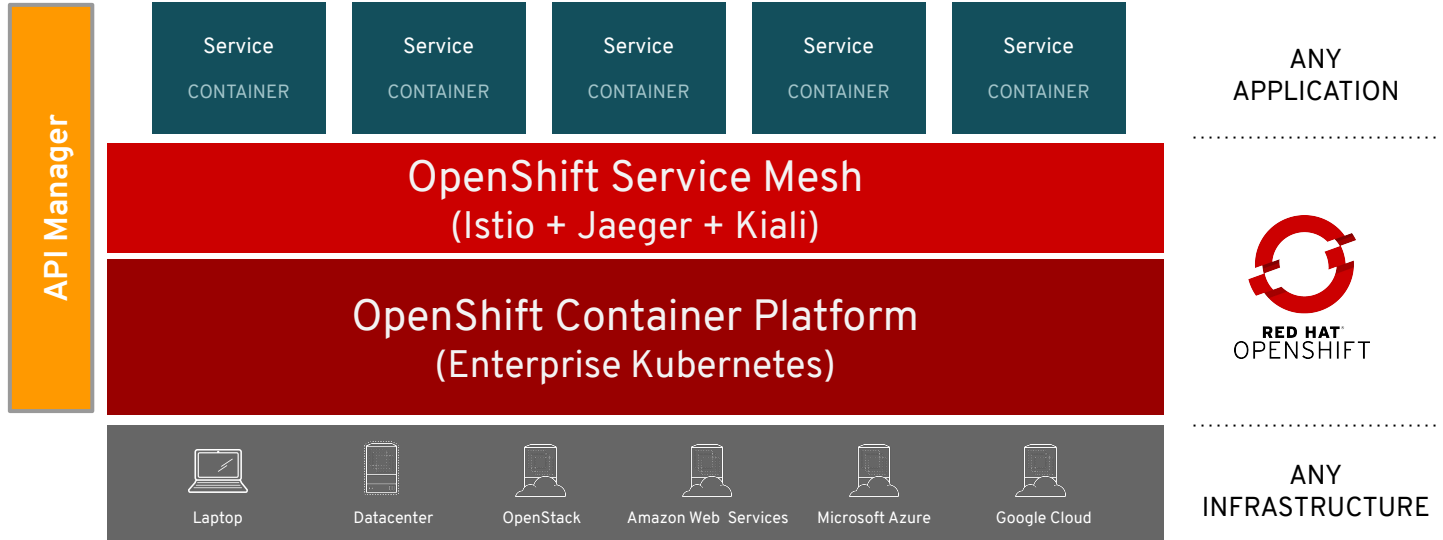
# API Management with Service Mesh

- Do you have tens / hundreds of services / APIs?
- Are applications consuming your APIs internal services?
- Need to package those services into consumable API products?
- Are there different types of customers for the API's? E.g. internal applications, partner applications, etc.
- Do you need a portal where customers can explore available API products and get immediate access to them?

# API Management with Service Mesh

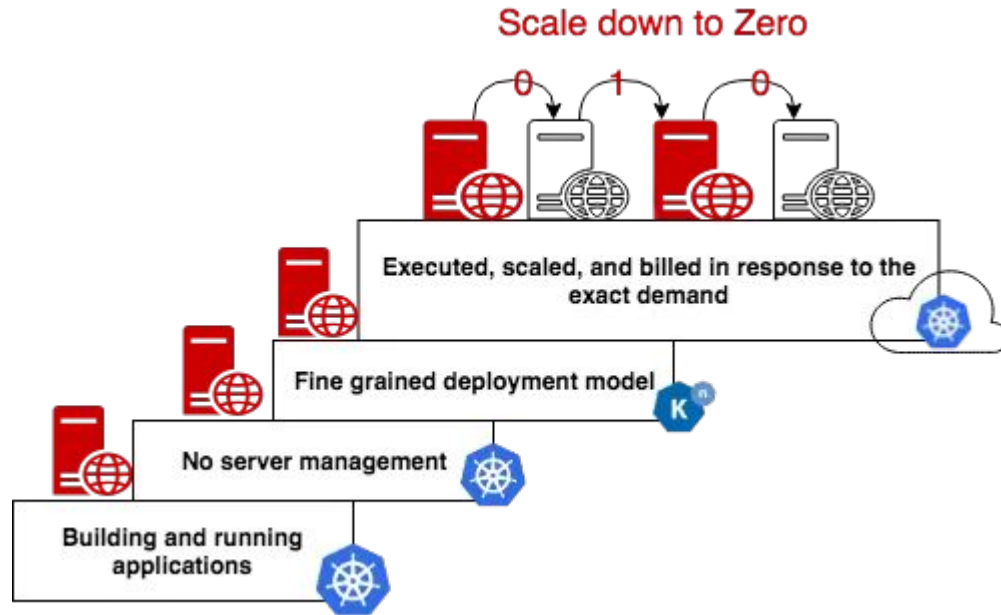
- The 3scale Istio Mixer Adapter gives services exposed within the service mesh API management capabilities.
- Developer access via developer portal and documentation, configuring different types of access for different type of developers, usage analytics, billing and invoicing.
- Quota enforcement, caching, and analytics are available at the 'API product' level.

# Distributed Services Platform





# Serverless Architecture: AutoScaling, Event-driven Architecture





# Serverless Architecture: AutoScaling, Event-driven Architecture

**Overview** Resources

4  
scaling to 10

<b>Name</b>	spring-petclinic-bchpw-deployment	<b>Update Strategy</b>	RollingUpdate
<b>Namespace</b>	markito-rhte	<b>Max Unavailable</b>	25% of 10 pods
<b>Labels</b>	app=spring-petclinic-bchpw app.kubernetes.io/... =springBoo... app.kubernetes.io/... =spring-pe... servicing.knative... =spring-petcli... servicing.knative.dev/configurati... =1 servicing.knat... =b8c3da91-d4cb-1... servicing.knative.dev... =spring-pe... servicing.knative.dev/... =spring-p...	<b>Max Surge</b>	25% greater than 10 pods
		<b>Progress Deadline</b>	2m 0s
		<b>Min Ready Seconds</b>	Not Configured



# OpenShift Serverless: Built on Service Mesh

## Key Features

- Familiar to Kubernetes users, native to K8s
- Scale to 0 and autoscale to N based on demand
- Applications *and* functions. Any container workload.
- Powerful eventing model with multiple event sources.
- Operator available via OperatorHub
- Knative v0.7.1 (v1beta1 APIs)
- No vendor lock in

## Learn more

<https://openshift.com/learn/topics/serverless>

<https://redhat-developer-demos.github.io/knative-tutorial>

The screenshot displays the OpenShift Serverless console interface. The top navigation bar shows the user is logged in as a temporary administrative user. The main content area is divided into two panels. The left panel shows a cluster overview with several pods, including 'wild-west-front...', 'wild-west-back...', and 'kiok-encoder-y...'. The right panel shows the details for the 'spring-petclinic-bchpw-deployment', including a large circular gauge indicating 4 pods scaling to 10. The deployment details table is as follows:

Name	Update Strategy
spring-petclinic-bchpw-deployment	RollingUpdate

Namespace	Max Unavailable
markito-rhte	25% of 10 pods

Labels	Max Surge
app=spring-petclinic-bchpw	25% greater than 10 pods
app.kubernetes.io/...=spring-bo...	
app.kubernetes.io/...=spring-pe...	
servicing.knative.dev/...=spring-pe...	Progress Deadline
servicing.knative.dev/configurati...	2m 0s
servicing.knative.dev/...=1	
servicing.knative.dev/...=b8c3d971-d4cb-1...	
servicing.knative.dev/...=spring-pe...	Min Ready Seconds
servicing.knative.dev/...=spring.g...	Not Configured

# Growth in Application Architecture Choices



Monolith



Cloud Native



Microservices



**Serverless**



**Event-Driven  
Architecture**



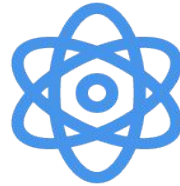
# Complete Platform for Your Architecture Choices



Monolith



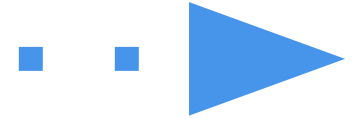
Cloud Native



Microservices



**Serverless**



**Event-Driven  
Architecture**



kubernetes



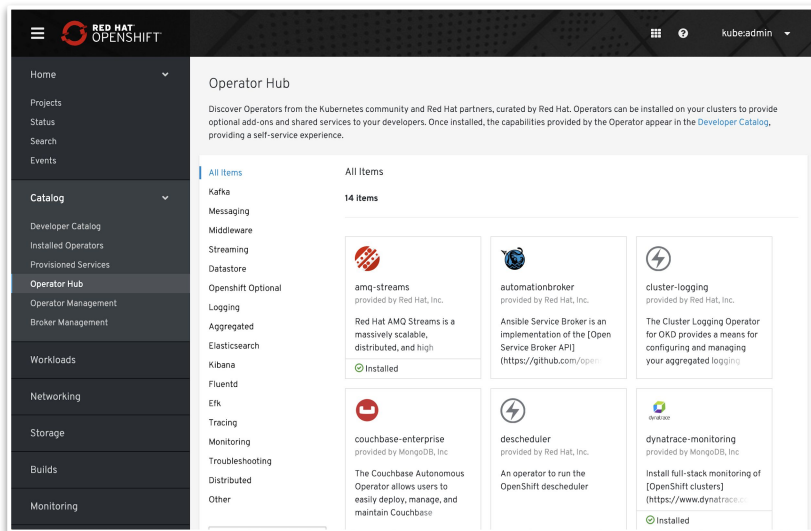
Istio



Knative

# OpenShift Service Mesh

- OpenShift Service Mesh is available and supported at no additional cost with OpenShift 4
- OpenShift Service Mesh Operator is found in the OperatorHub menu



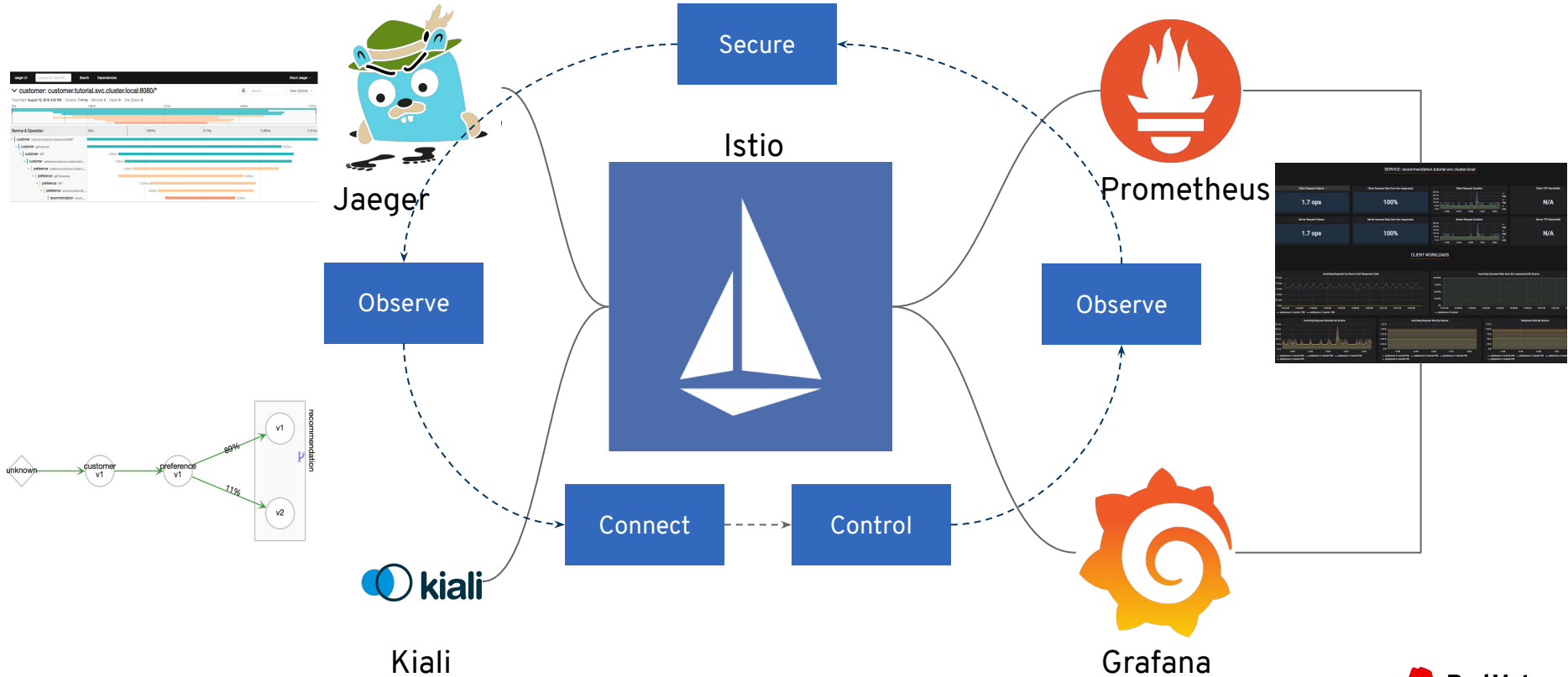
# Next Step: Free Hands-On

## Training

Online Labs: <https://learn.openshift.com/service-mesh/>

- Istio Architecture
- Microservices deployment into a Service Mesh
- Monitoring, tracing
- Traffic routing
- Fault injection
- Circuit breakers
- Egress security
- Mutual TLS security

# Demo and Q&A



# Next Step: Free Hands-On

## Training

Online Labs: <https://learn.openshift.com/servicemesh/>

- Istio Architecture
- Microservices deployment into a Service Mesh
- Monitoring, tracing
- Traffic routing
- Fault injection
- Circuit breakers
- Egress security
- Mutual TLS security

# Thank You!



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



[twitter.com/RedHat](https://twitter.com/RedHat)

## Reference Links

- [Product Overview](#)
- [Service Mesh Release Announcement](#)
- [Product Documentation](#)
- [Service Mesh E-book](#)
- [Online Service Mesh Training](#)