Red Hat

Red Hat OpenShift Integration

Istio, Kafka and Camel

Robert Sedor
Senior Cloud Platform Architect

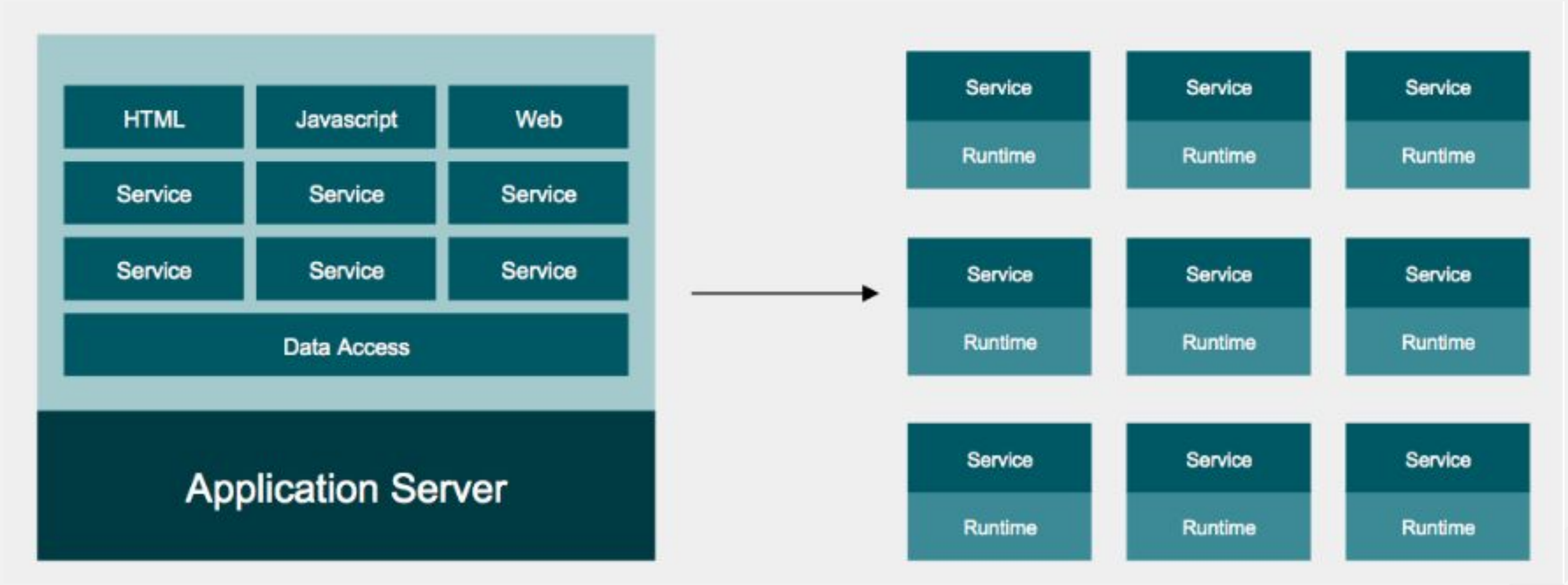
Amritpal Jhajj
Senior Solutions Architect

AGENDA

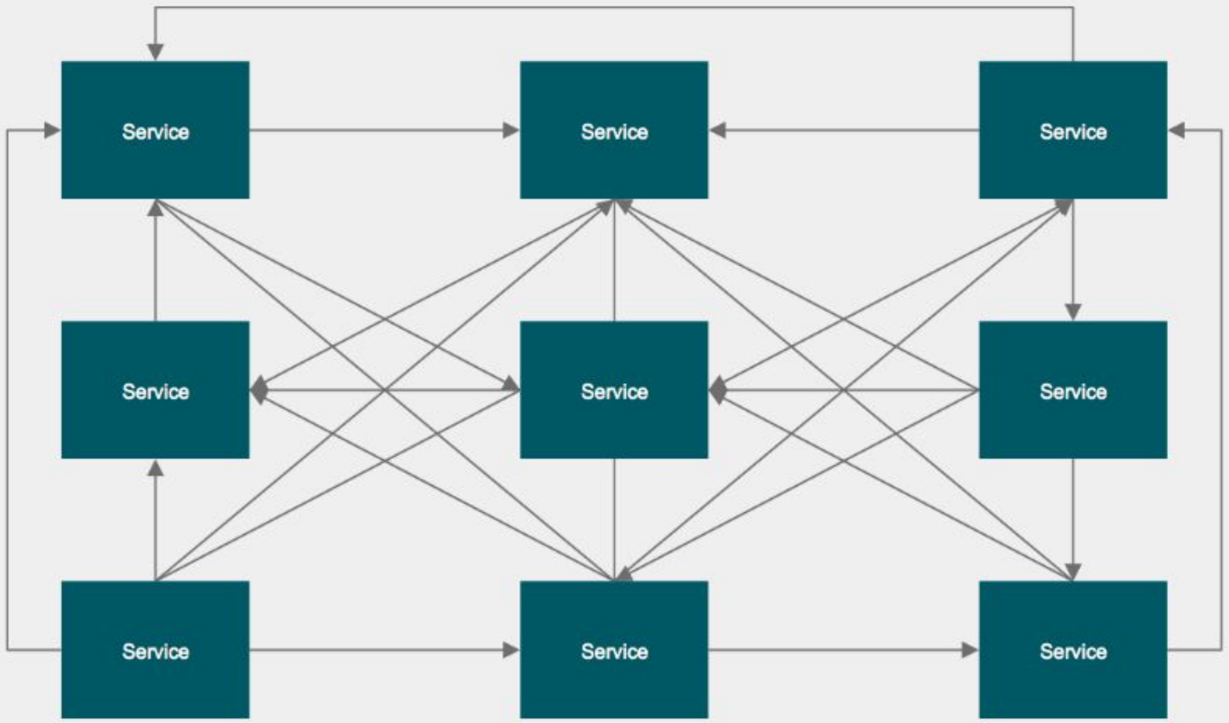
- Complexities of Microservices Architecture
- Service Mesh (Istio)
- AMQ Streaming (Kafka)
- Cloud Native Camel (Camel K)

Microservices Architecture

~~Microservices~~ Distributed Architecture

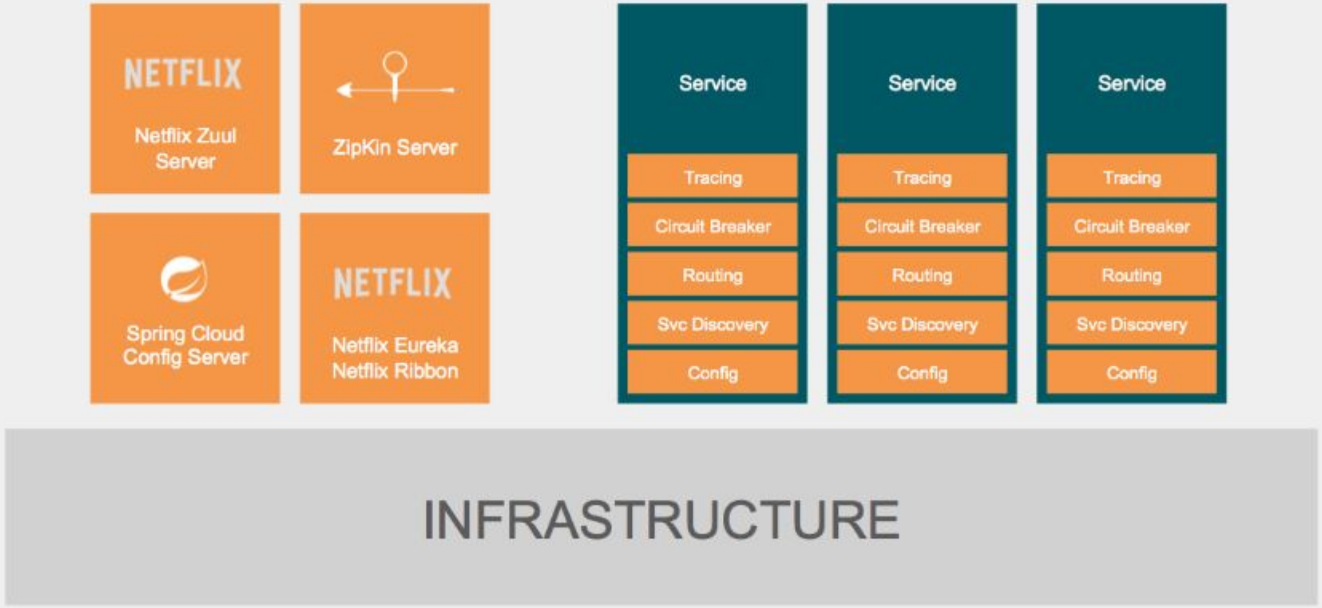


Distributed Architecture



Dealing with Complexity

Netflix OSS And Spring Cloud



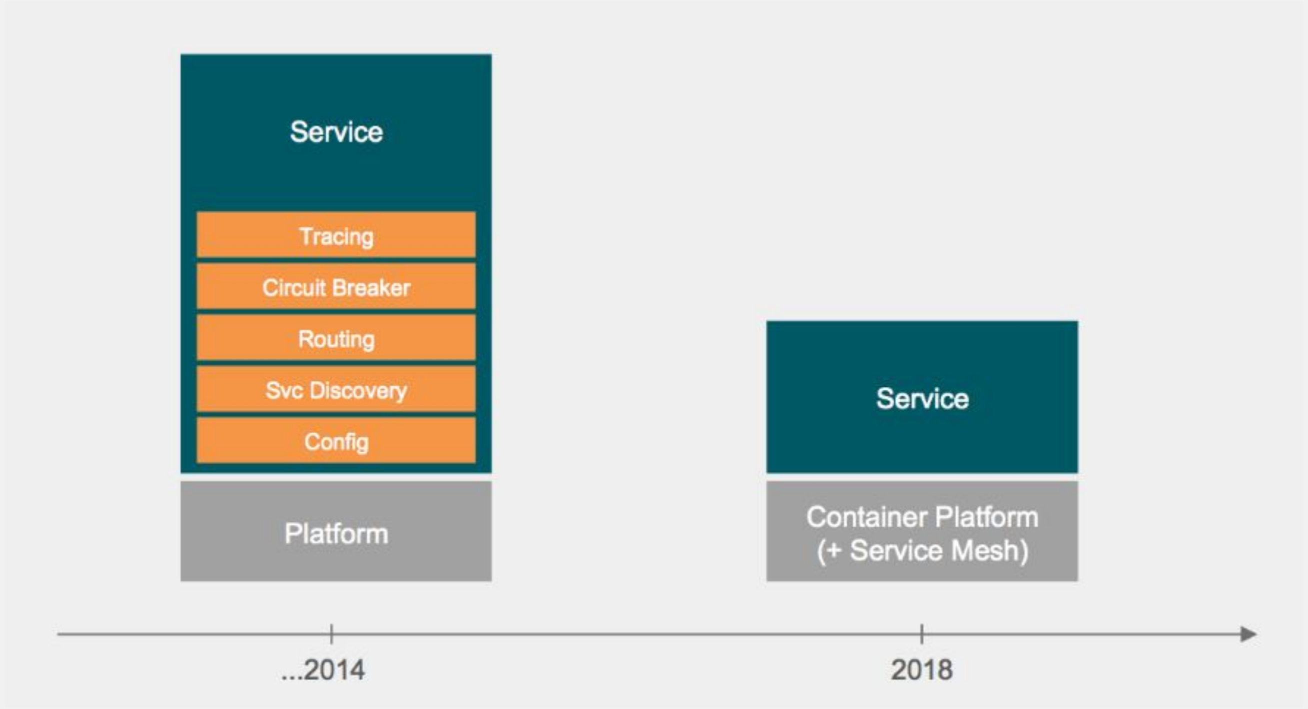
What About ... ?

There Should Be A Better Way

Service Mesh

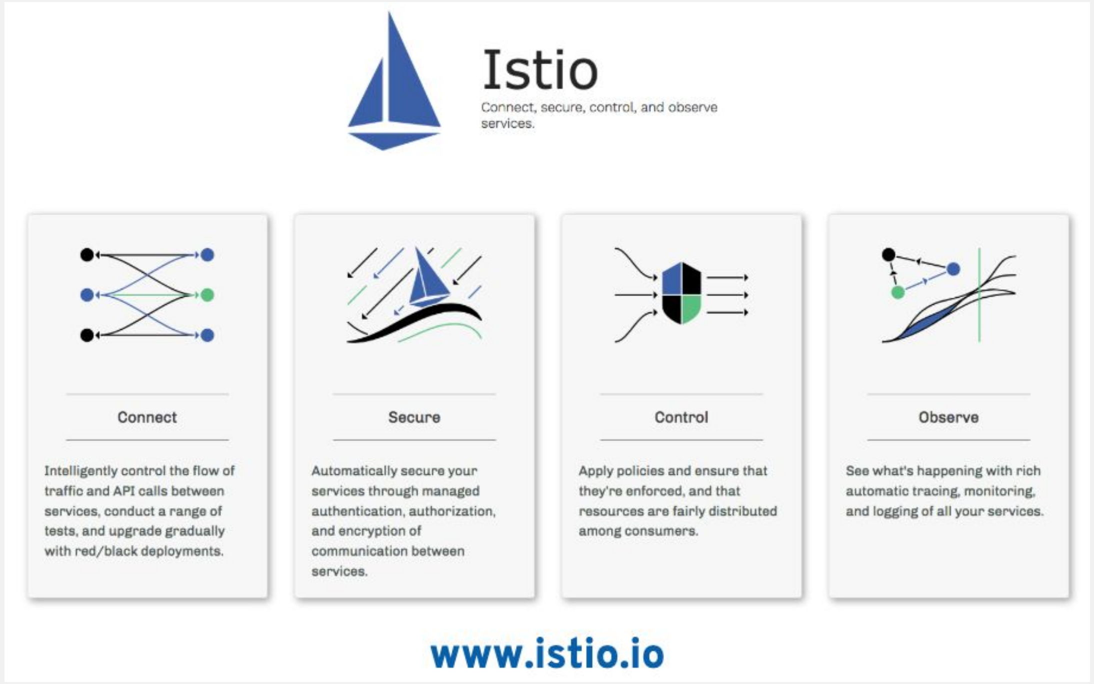
A dedicated network for
Service-to-service communications

Microservices Evolution



Istio

- Istio is an open-source service mesh



Istio

Connect, secure, control, and observe services.

Connect

Intelligently control the flow of traffic and API calls between services, conduct a range of tests, and upgrade gradually with red/black deployments.

Secure

Automatically secure your services through managed authentication, authorization, and encryption of communication between services.

Control

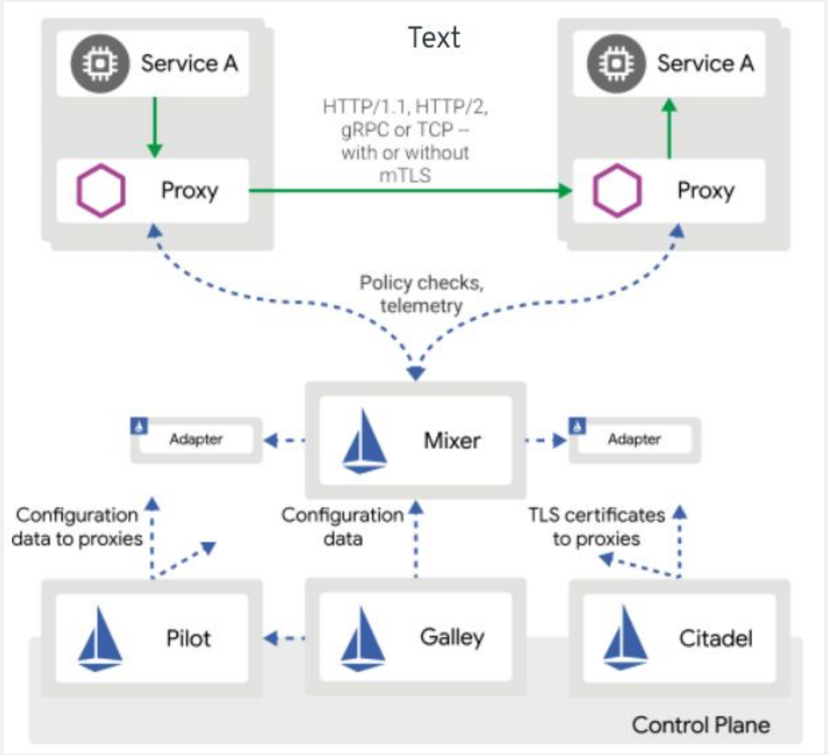
Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.

Observe

See what's happening with rich automatic tracing, monitoring, and logging of all your services.

www.istio.io

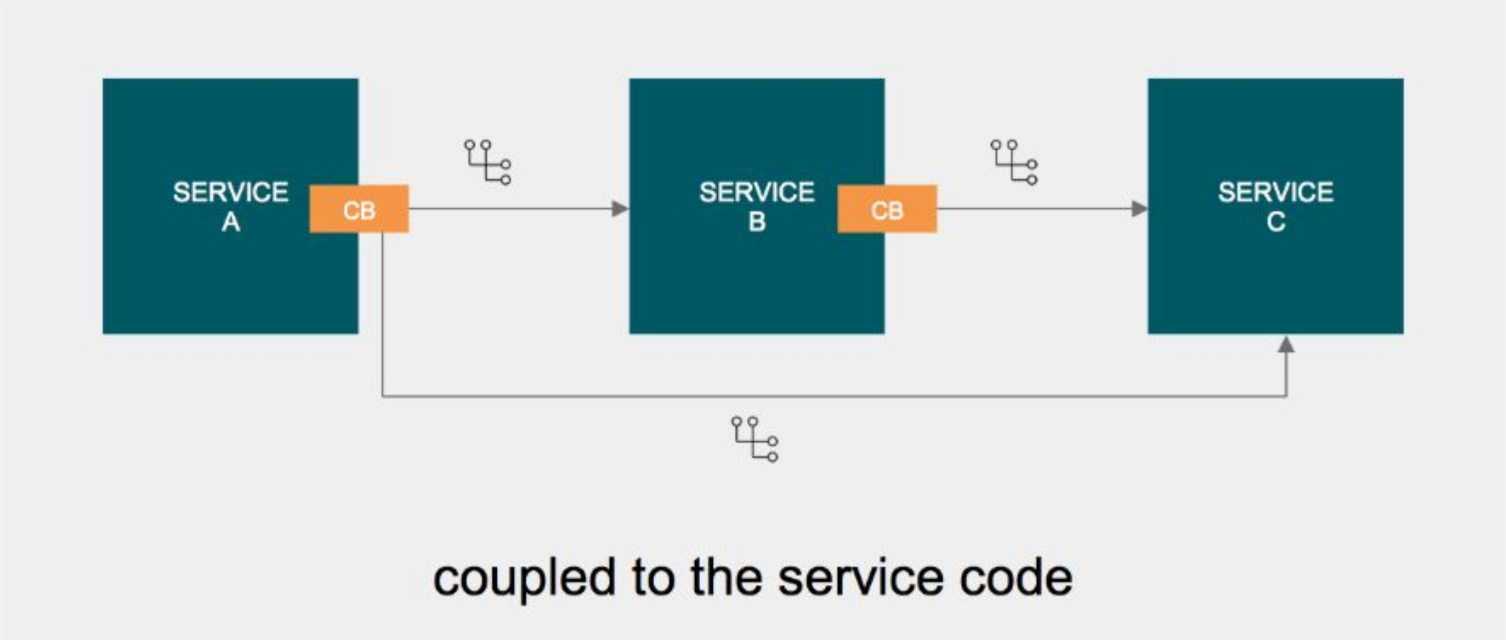
Istio - Service Mesh Architecture



Istio Features

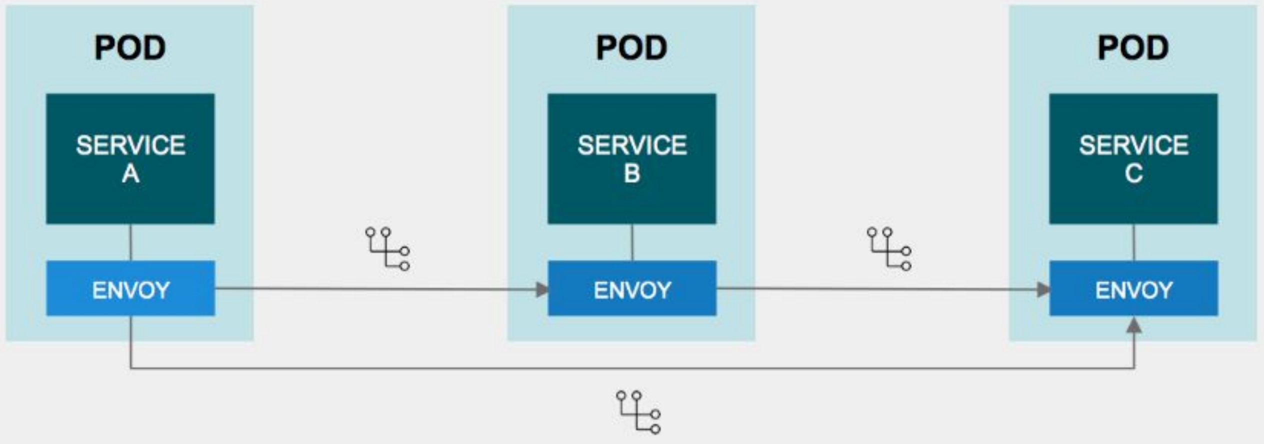
Istio - Fault Tolerance

Circuit Breakers - Without Istio



Istio - Fault Tolerance

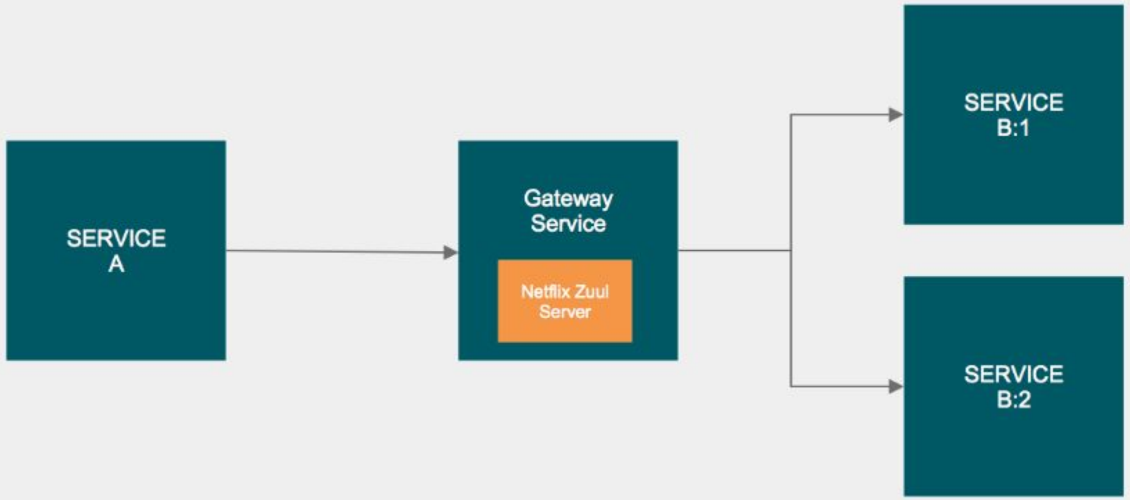
Circuit Breakers - With Istio



transparent to the services

Istio - Dynamic Routing

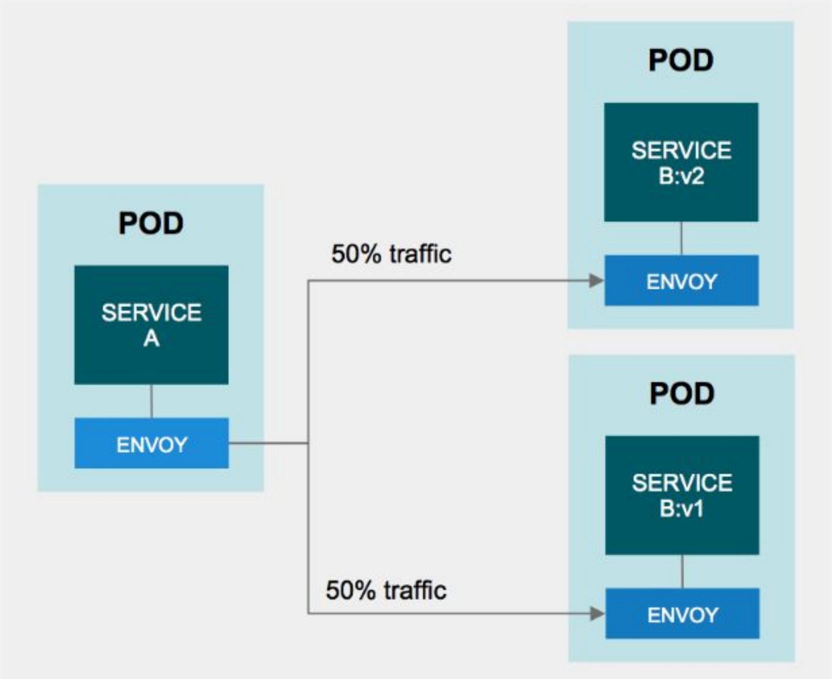
Routing - Without Istio



custom code to enable dynamic routing

Istio - Dynamic Routing

With Istio [A/B Deployment]



Istio - Security

The Istio security features provide:

- Strong identity
- Powerful policy
- Transparent TLS encryption and
  - Authentication, Authorization and Audit (AAA) tools to protect your services and data

Istio - Security

Kubernetes Scenario:

1. SPIFFE certificate (X.509) and key pair for each of the existing and new service accounts
2. Kubernetes mounts the certificate and key pair to a pod according to its service account via Kubernetes secret volume
3. Auto rotation of the certificates by rewriting the Kubernetes secrets
4. Secure naming generation and distribution to proxies

Istio - Security

Mutual Authentication (mTLS) Scenario:

1. Client’s outbound traffic goes to its Envoy Proxy
2. The client side Envoy starts a mutual TLS handshake with the server side Envoy, does a secure naming check to verify that the service account is authorized
3. The client side Envoy and the server side Envoy establish a mutual TLS connection, and Istio forwards the traffic from the client side Envoy to the server side Envoy
4. After authorization, the server side Envoy forwards the traffic to the server service through local TCP connections

Istio - Security

Authorization Scenario:

1. ClusterRbacConfig enables/disables (including selectively) Authorization in the mesh
2. ServiceRole and ServiceRoleBinding define the overall authorization
3. ServiceRole defines permissions to access services
4. ServiceRoleBinding bind the ServiceRole to Subjects. Subjects can be Users (service accounts or properties)

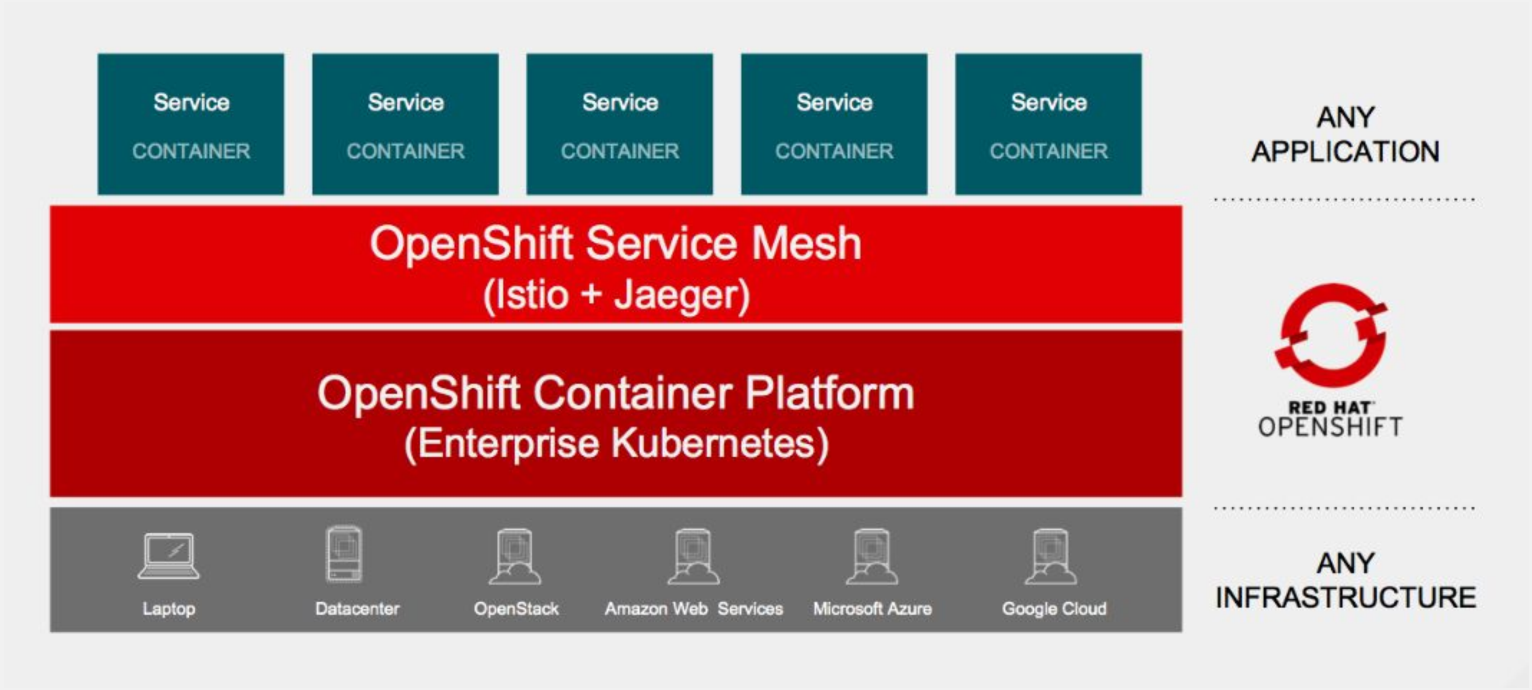
More information here

Istio - Other Features

- Distributed Tracing
- Chaos Engineering
- Dark Launch
- *More ...*

Pulling It All Together

OpenShift - Service Mesh



Red Hat AMQ Streams

Red Hat AMQ Streams

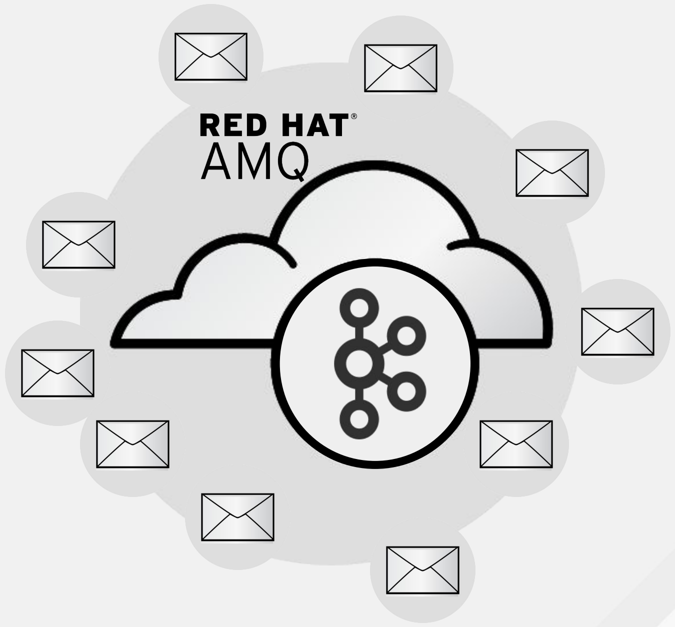
Overview

Data streaming platform based on Apache Kafka.

Available standalone (RHEL) or on OpenShift (OCP).

RED HAT® ENTERPRISE LINUX® 7

RED HAT® OPENSHIFT



Red Hat AMQ Streams

What is Apache Kafka?

Apache Kafka is a distributed system designed for streams. It is built to be an horizontally-scalable, fault-tolerant, commit log, and allows distributed data streams and stream processing applications.

Red Hat AMQ Streams

What is Apache Kafka?

- Developed at Linkedin back in 2010, open sourced in 2011
- Designed to be fast, scalable, durable and available
- Distributed by nature
- Data partitioning (sharding)
- High throughput / low latency
- Ability to handle huge number of consumers

Red Hat AMQ Streams

What is AMQ Streaming for? Main Use Cases

Messaging

Replacement of traditional message broker, has better throughput, built-in partitioning, replication, and fault-tolerance. Provides strong durability.

Website Activity Tracker

Rebuild user activity tracking pipeline as a set of real-time publish-subscribe feeds. Activity is published to central topics with one topic per activity type.

Red Hat AMQ Streams

What is AMQ Streaming for? Main Use Cases

Metrics

Aggregation of statistics from distributed applications to produce centralized feeds of operational data.

Log Aggregation

Abstracts details of files an gives event data as stream of messages. Offers good performance, stronger durability guarantees due to replication.

Red Hat AMQ Streams

What is AMQ Streaming for? Main Use Cases

Stream Processing

Enables continuous, real-time applications built to react to, process, or transform streams.

Data Integration

Captures streams of events or data changes and feeds these to other data systems.

Red Hat AMQ Streams

Why should you use it?

- Scalability and Performance
    - Designed for horizontal scalability
    - Scaling has minimal impact on throughput and latency
    - Adding nodes to running cluster is easy
    - Different approaches: One big cluster vs. Several small clusters
- Message ordering guarantee
    - Messages are written to disk in the same order as received by the broker
    - Messages are read from disk from the requested offset
    - Kafka protocol makes sure that one consumer can read only one partition
    - Order is guaranteed only within a single partition!
    - No synchronization between producers!

Red Hat

Red Hat AMQ Streams

Why should you use it?

- Message rewind/replay
    - Limited only by available disk space
        - Amount of stored messages has no impact on performance
    - Topic/Partition size has no direct impact on performance
    - Allows to reconstruct application state by replaying the messages
    - Combined with compacted topics allows to use Kafka as key-value store
    - Event source and/or Parallel Running patterns

Red Hat AMQ Streams

What’s the catch?

- Kafka protocol cannot be proxied
  - Clients need access to all brokers in the cluster
  - Producers/consumers might need to maintain large number of TCP connections
  - Proxying via HTTP REST or AMQP could be a solution
- Dumb broker, smart clients
  - Carefully decide the “right” number of partitions for each topic
  - Adding partitions can change destination partition for “keyed” messages
  - Removing partitions is not possible

Red Hat AMQ Streams On OCP

AMQ Streams on OCP

Supported Configurations

- Based on OSS project called Strimzi
- OpenShift Container Platform 3.9 and newer
- Apache Kafka Ecosystem
  - Only the java components released directly from Apache Software Foundation
  - Apache Kafka Broker
  - Apache Kafka Connect
  - Apache Kafka Streaming API
  - Apache Kafka Java Producer, Consumer and Management Clients
  - Apache Zookeeper (ONLY as an implementation detail of Apache Kafka)
- Source: Red Hat AMQ 7 Supported Configurations - AMQ Streams 1.0

AMQ Streams on OCP

What is Strimzi?

Strimzi is a set of enabling services that allow Kafka to work in OpenShift as a first class citizen, be installed easily and configured and managed simply.

AMQ Streams on OCP

What is Strimzi?

- Provides:
    - Docker images for running Apache Kafka and Zookeeper
    - Tooling for managing and configuring Apache Kafka clusters, topics and users
- Follows the Kubernetes Operator Model

AMQ Streams on OCP

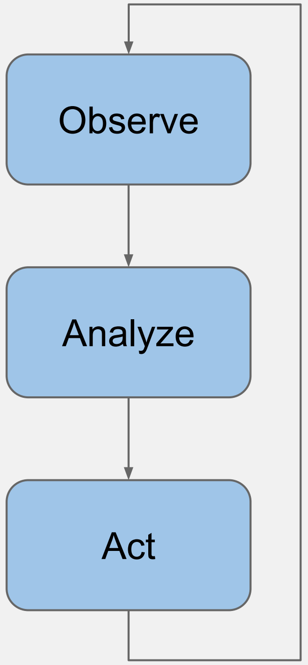
Goals

- Simplifying the Apache Kafka deployment on OpenShift
- Using the OpenShift native mechanisms for
  - Provisioning the cluster
  - Managing the topics
- Removing the need to use Kafka command-line tools
- Providing a better integration with applications running on OpenShift
  - Microservices, data streaming and even-sourcing

AMQ Streams on OCP

Kubernetes Operators

- Observe: Monitor the current state of the application
- Analyze: Compare the actual state to the desire state
- Act: Resolve any differences between actual and desired state



AMQ Streams on OCP

Kubernetes Operators

- Application-specific controller is used to create, configure and manage other complex application:
  - The controller contains specific domain/application knowledge
  - Usually used for stateful applications (databases, …) which are non-trivial to operate on Kubernetes/OpenShift
- Controller operates based on input from Config Maps or Custom Resource Definitions
  - User describes the desired state
  - Controller applies this state to the application

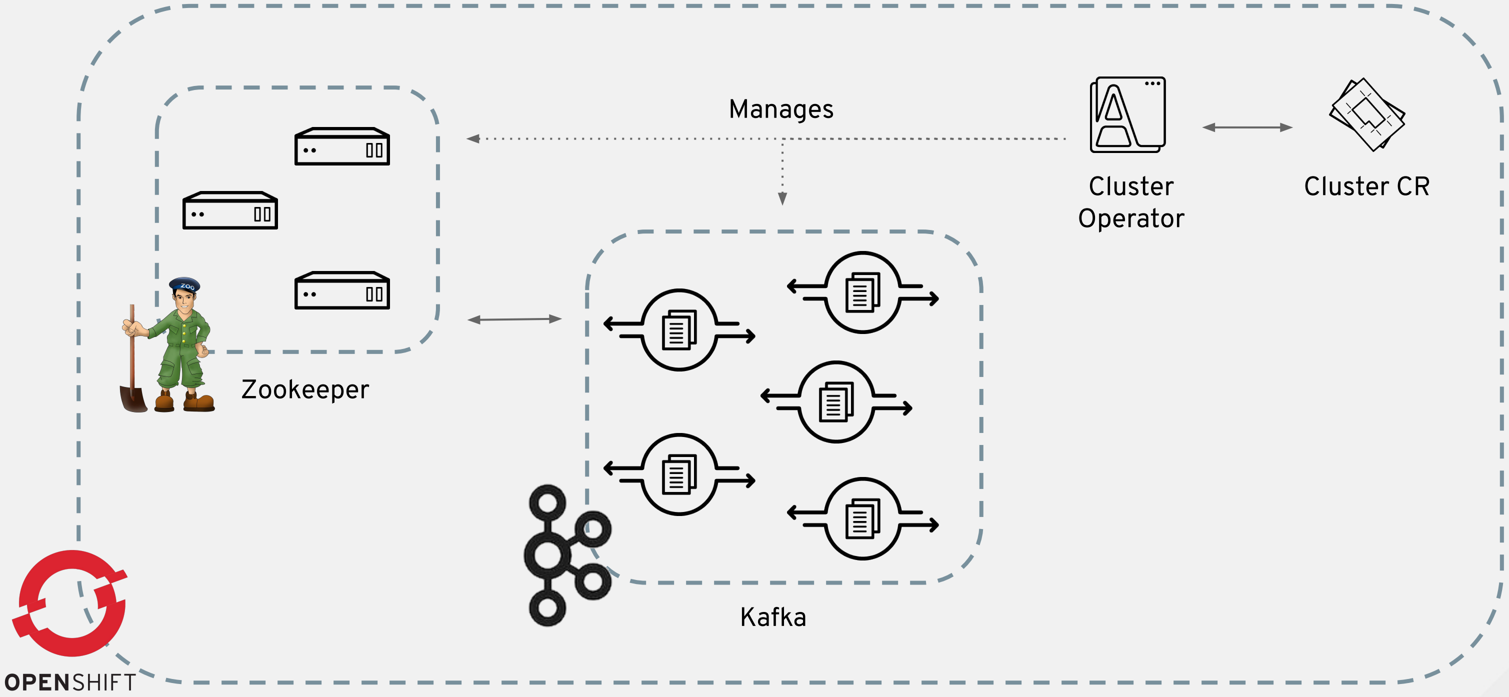
AMQ Streams on OCP

AMQ Streams Operators

- Cluster Operator
- Topic Operator
- User Operator

AMQ Streams on OCP

Cluster Operator - Topology



AMQ Streams on OCP

Cluster Operator - Main Features

- Creating and managing Apache Kafka clusters
- Two kinds of cluster
  - Kafka Cluster
    - Cluster of Kafka Brokers
    - Includes Zookeeper deployment
    - Using Stateful Sets for managing Kafka and Zookeeper
  - Kafka Connect Cluster
    - Distributed Kafka Connect cluster
    - Using Deployment
    - S2I support for adding additional plugins

AMQ Streams on OCP

Cluster Operator - Supported Features

- Number of Zookeeper, Kafka and Kafka Connect nodes
- Configuration of Kafka and Kafka Connect
- Storage:
  - Persistent versus Ephemeral
  - Storage size
- Metrics exports for Prometheus
- Healthchecks

AMQ Streams on OCP

Cluster Operator - Workflow

- One Cluster Operator can manage several clusters in parallel
  - Can cover one or more projects
- To deploy new cluster
  - [Deploy the Cluster Operator]
  - Create a Custom Resource Definitions describing the cluster
  - Cluster Operator will see the Custom Resource Definitions and start deploying the cluster
  - The cluster will be deployed and ready to use

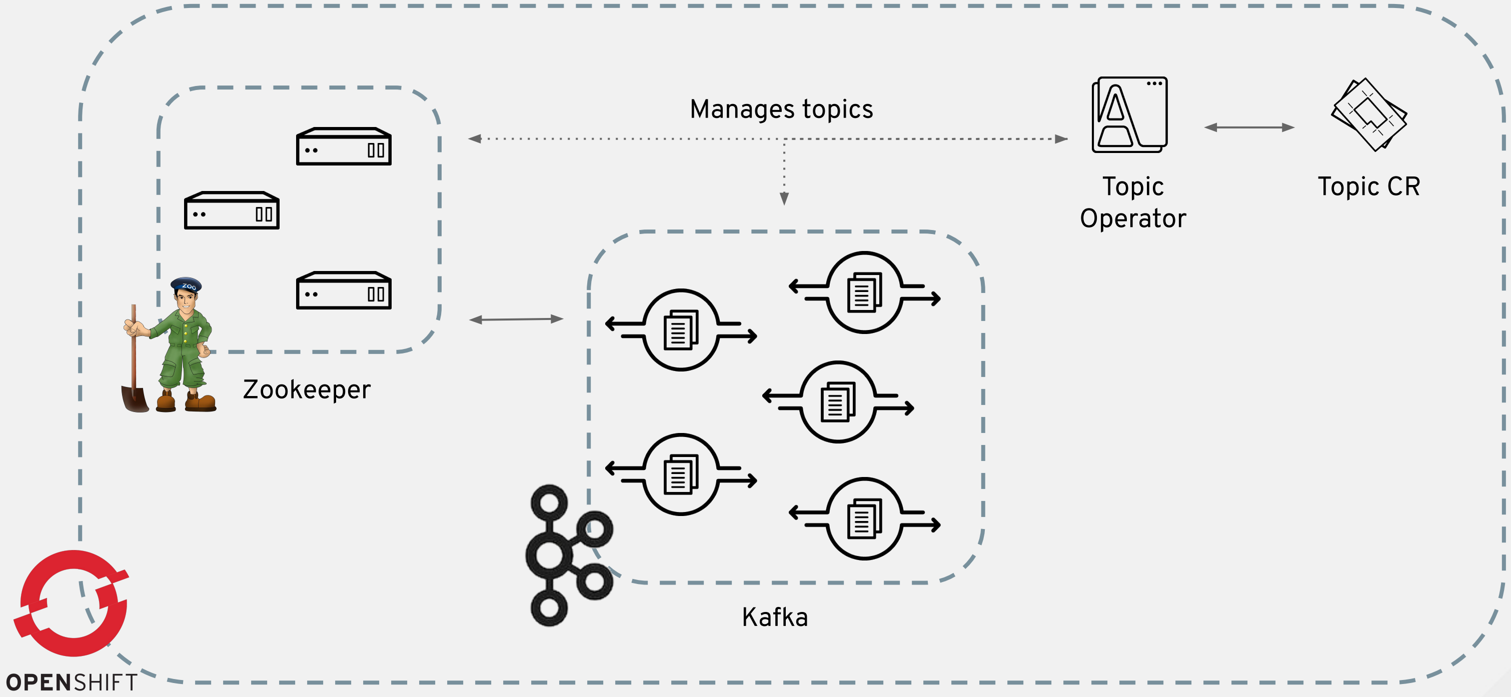
AMQ Streams on OCP

Cluster Operator - Managing Cluster

- Clusters can be modified by modifying the Custom Resource Definition
  - Scale up / Scale down
  - Kafka Configuration
- Cluster Operator will update the cluster to match the desired state described in Custom Resource Definition
- Update does not allow to change storage configuration
  - Such operation cannot be done without losing data
- Clusters can be deleted by deleting the Custom Resource Definition
  - All cluster resources will be removed
  - PVC will be deleted according to user configuration
- Deleting the Custom Resource Definition is irreversible

AMQ Streams on OCP

Topic Operator - Topology



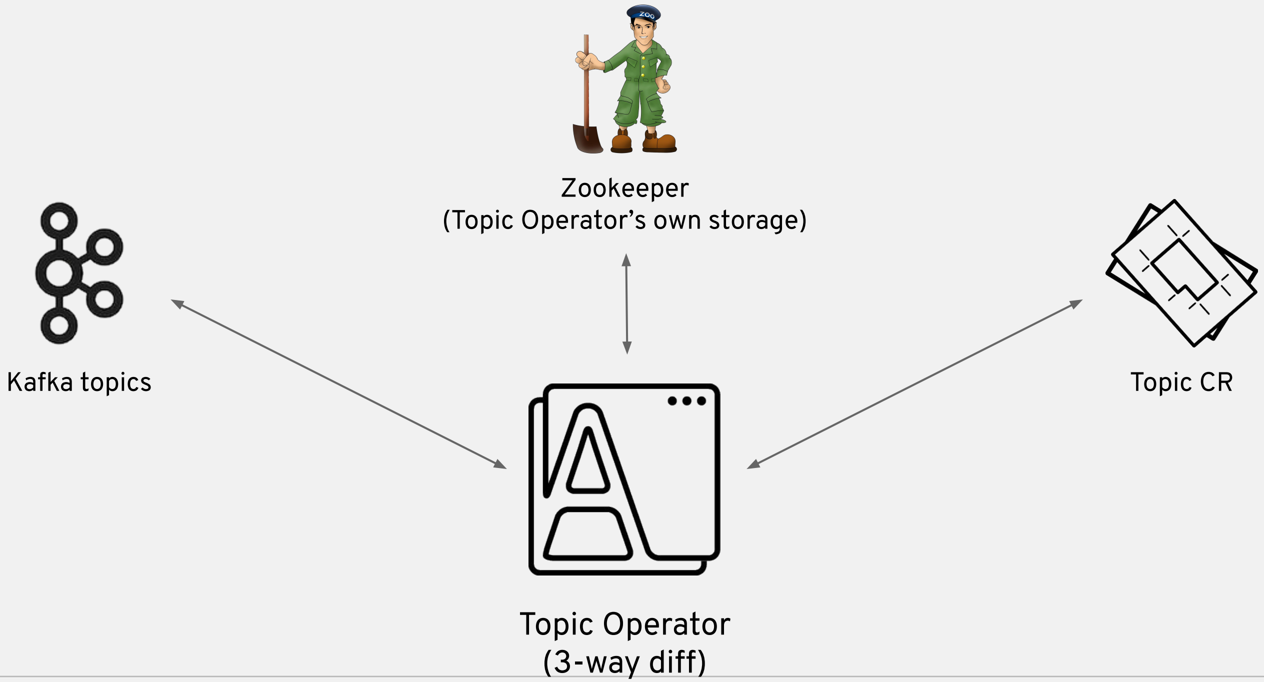
AMQ Streams on OCP

Topic Operator - Main Features

- Creating and managing Kafka topics
- Some Kafka components (Streams, Connect) often create their own topics
  - Bi-directional synchronization
  - Changes done directly in Kafka/Zookeeper are applied to Custom Resource Definitions
  - Changes done in Custom Resource Definitions are applied to Kafka topics
- Topic Operator solves this by using 3-way diff
  - Our own Zookeeper-based store
  - Apache Kafka / Zookeeper
  - Custom Resource Definitions

AMQ Streams on OCP

Topic Operator - Managing Kafka Topics



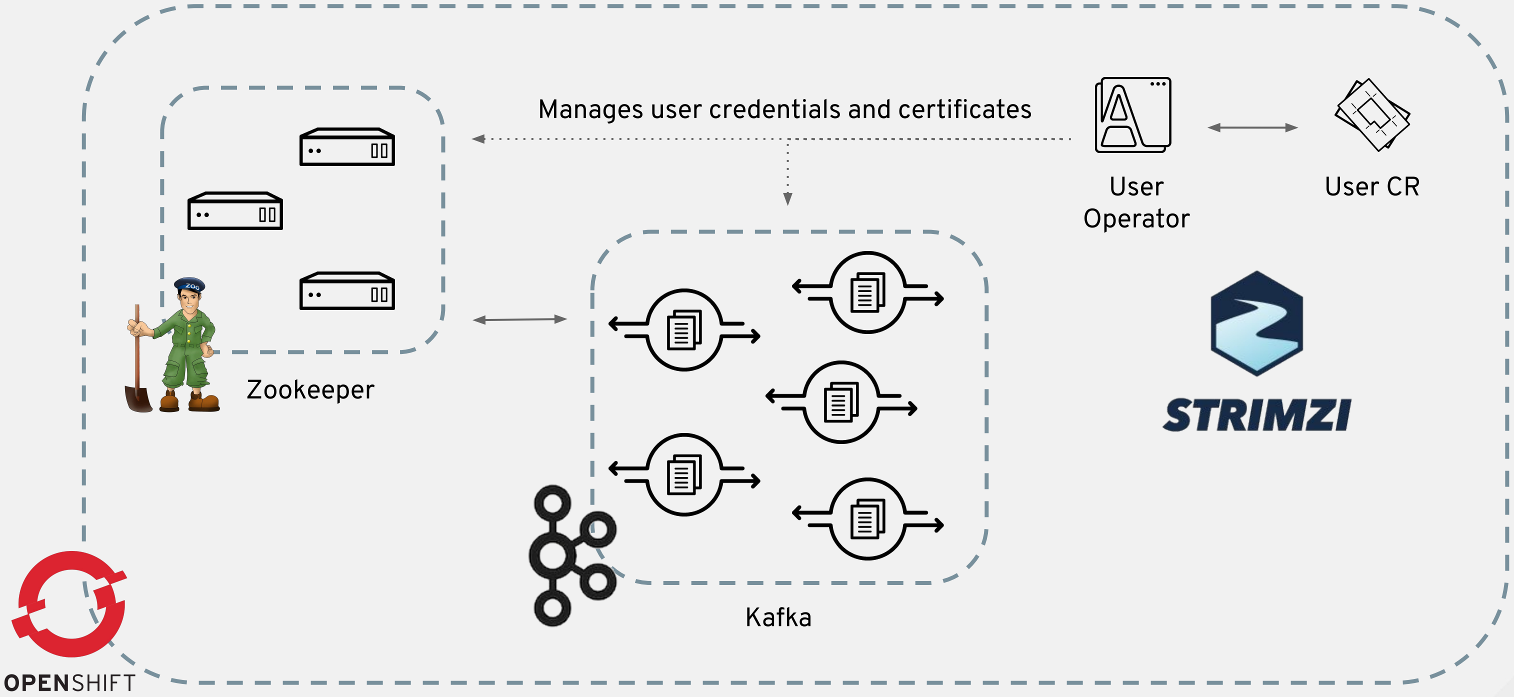
AMQ Streams on OCP

Topic Operator - Naming

- Kafka gives user more freedom for naming than OpenShift
  - ConfigMaps configuring topics contain “name” field to define the name which is not allowed for OpenShift resource
  - If such topic is created by Kafka, it will be mapped to a Config Map named after specially encoded name
- Recommended to use only topic names which are allowed as OpenShift resource names

AMQ Streams on OCP

User Operator - Topology



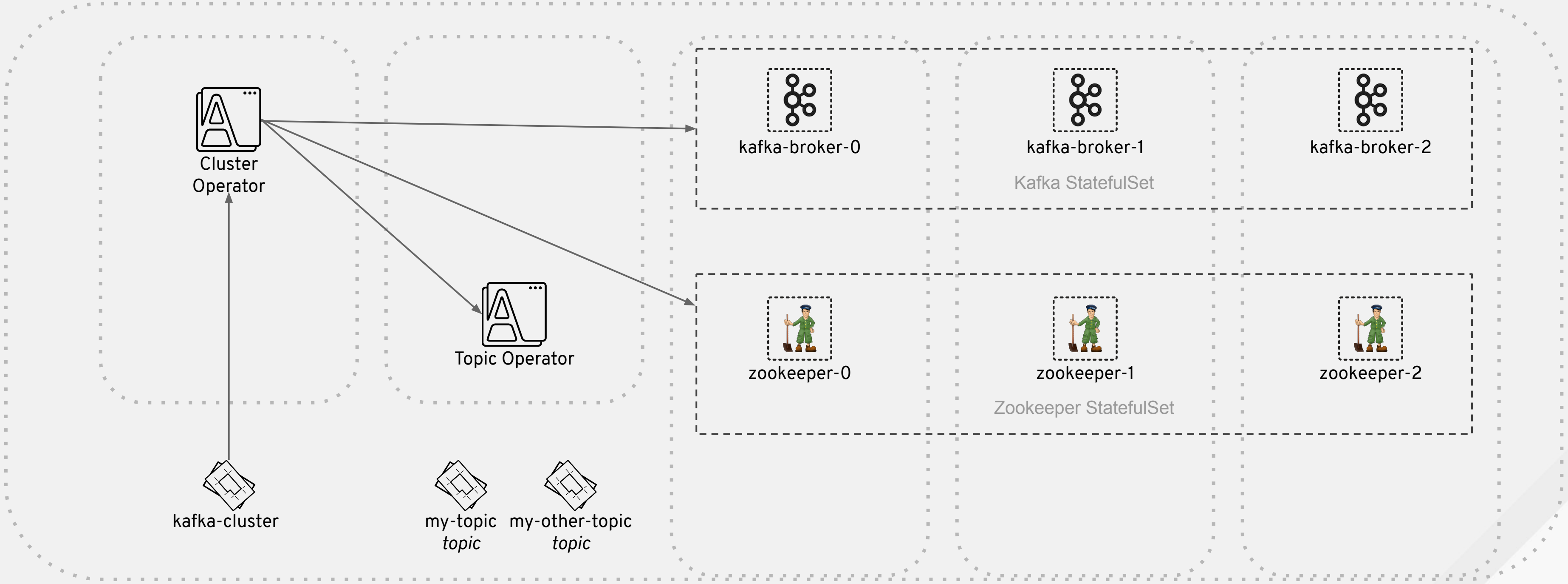
AMQ Streams on OCP

User Operator

- Creating and managing Kafka users
- Unlike Topic Operator, does not sync any changes from the Kafka cluster with OCP
- It is not expected that the users will be managed directly in Kafka cluster in parallel with the User operator
- User Credentials managed in a Secret
- Manages authorization rules

AMQ Streams on OCP

Sample Kafka Cluster



AMQ Streams on OCP

Images Registry

- Images imported in Registry:
    - amqstreams-1/amqstreams10-kafka-openshift
    - amqstreams-1/amqstreams10-zookeeper-openshift
    - amqstreams-1/amqstreams10-topicoperator-openshift
    - amqstreams-1/amqstreams10-useroperator-openshift
    - amqstreams-1/amqstreams10-clusteroperator-openshift
    - amqstreams-1/amqstreams10-zookeeperstunnel-openshift
    - amqstreams-1/amqstreams10-kafkastunnel-openshift
    - amqstreams-1/amqstreams10-entityoperatorstunnel-openshift

AMQ Streams on OCP

OCP Description

- AMQ Streams Operators installed as cluster-admin

```
$ sed -i s/myproject/amq-streams/g *.yaml \
    install/cluster-operator -n amq-streams
$ oc apply -f install/cluster-operator -n amq-streams
```

AMQ Streams on OCP

Pods Resourcing

- Default values used for *-Xms* and *-Xmx* depends on where there is a memory request limit configured for the container
    - JVM's minimum and maximum memory will be set to a value corresponding to the limit.
    - If there is not memory limit, JVM's minimum memory will be set to 128M and JVM's maximum memory will not be defined.
- Setting *-Xmx*, it is recommended to:
    - Set memory request and memory limit to the same value
    - Use a memory request that is at least 4 x the -Xmx value
    - Consider setting -Xms to the same value as -Xmx
- Containers doing lots of disk I/O will need to leave some memory available for use as operating system page cache. On such containers, the requested memory should be significantly higher than the memory used by the JVM

AMQ Streams on OCP

Sample Kafka Cluster Definition

```
apiVersion: kafka.strimzi.io/v1alpha1
kind: Kafka
metadata:
  name: amq-streams-cluster
spec:
  kafka:
    replicas: 2
    # listeners, config, metrics, jvmOptions, resources
    storage:
      type: persistent-claim
      size: 12Gi
  zookeeper:
    replicas: 3
    # jvmOptions, resources
    storage:
      type: persistent-claim
      size: 5Gi
  entityOperator:
    # topicOperator, userOperator, tlsSidecar
```

Cloud Native Camel

System Integration

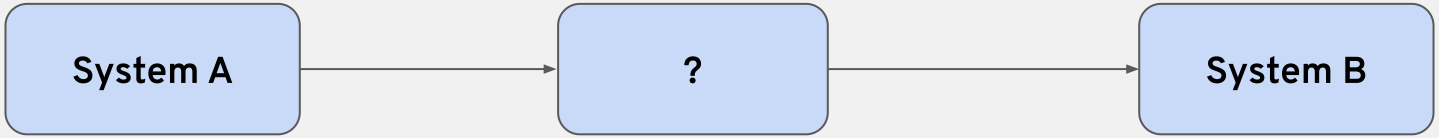
Camel is the glue between disparate systems

- The swiss knife of integration
- **>10 years of development - still one of the most active Apache projects**
- Based on Enterprise Integration Patterns (EIP)
- Uses a powerful Domain Specific Language (DSL)
- Can integrate **anything**
- Supports 300+ components
- The cornerstone of Red Hat Fuse

http://camel.apache.org/

System Integration

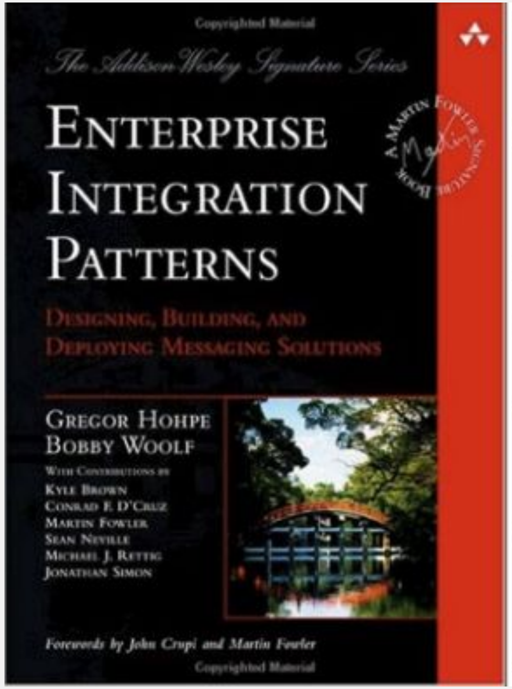
Camel is the glue between disparate systems



- Different transports
- Different Data Model
- Asynchrony
- Failures

Enterprise Integration Patterns

Camel is an integration framework based on Enterprise Integration Patterns


Copyrighted Material

The Addison-Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS

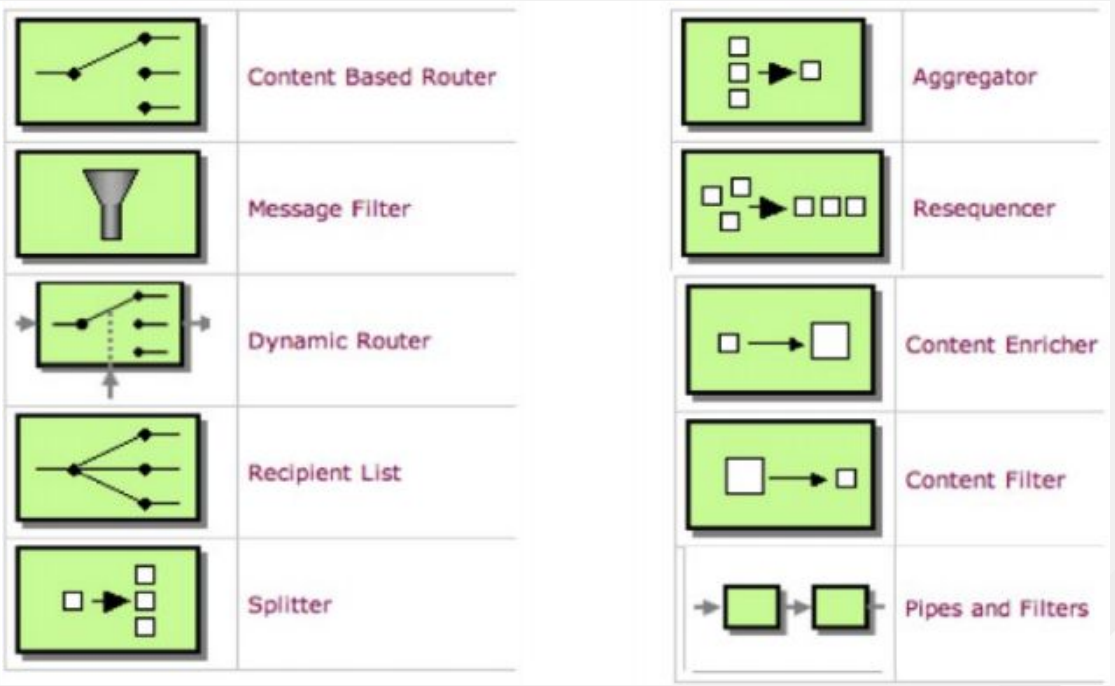
A MARTIN FOWLER SIGNATURE BOOK

GREGOR HOHPE
BOBBY WOOLF

WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F. D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON

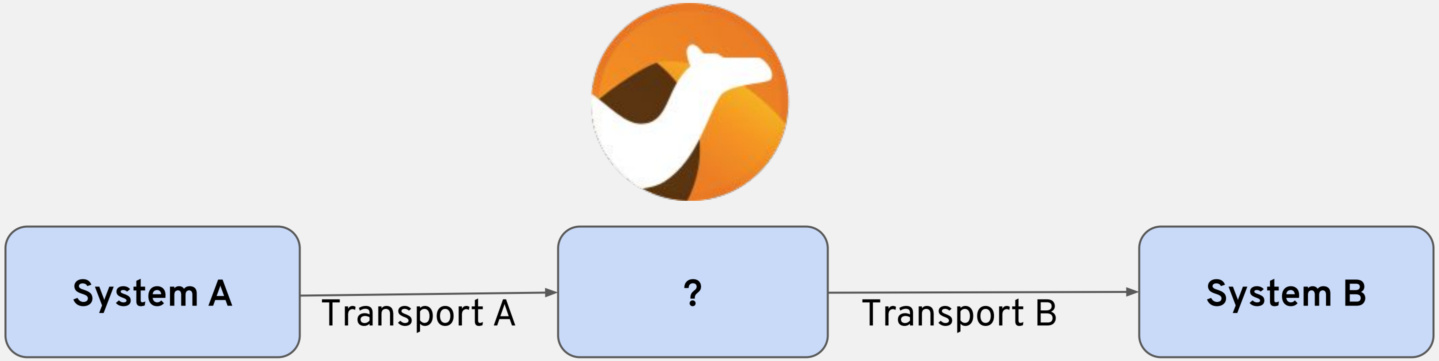
Forewords by John Crupi and Martin Fowler

Copyrighted Material



System Integration

Camel is the glue between disparate systems



Camel Runs Everywhere

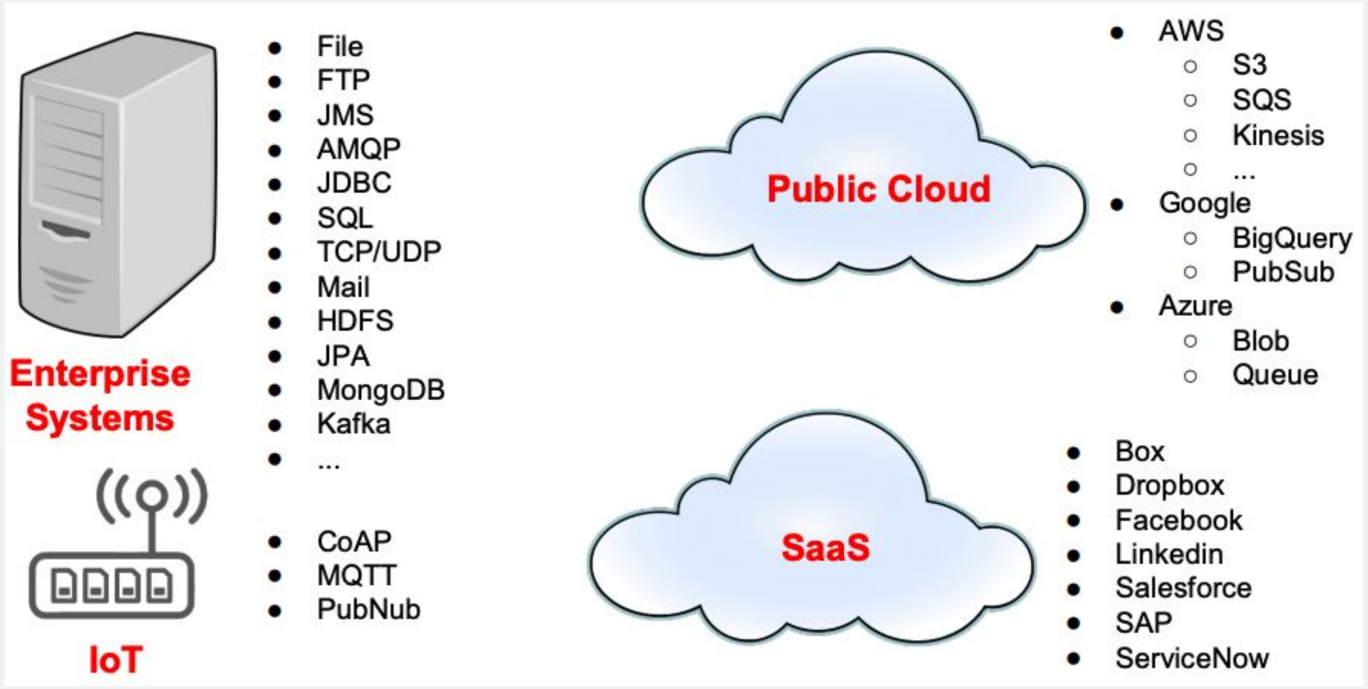
QUARKUS

WildFly

Karaf

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

Camel Connects Everything



- File
- FTP
- JMS
- AMQP
- JDBC
- SQL
- TCP/UDP
- Mail
- HDFS
- JPA
- MongoDB
- Kafka
- ...

- CoAP
- MQTT
- PubNub

- AWS
  - S3
  - SQS
  - Kinesis
  - ...
- Google
  - BigQuery
  - PubSub
- Azure
  - Blob
  - Queue

- Box
- Dropbox
- Facebook
- Linkedin
- Salesforce
- SAP
- ServiceNow

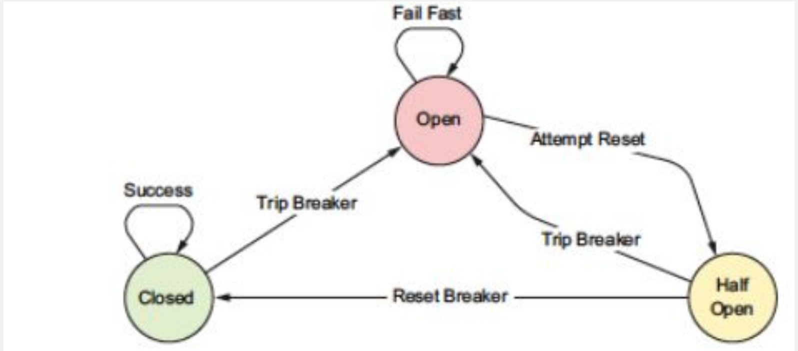
Best Practices

- Camel is light-weight
  - Camel-core is 4mb
  - Camel can be added to a single fat-jar via
    - Spring Boot
    - Wildfly Swarm
    - Vert.x
- Microservices favor stateless applications but
  - If state is needed
    - Camel-infinispan
    - Camel-hazelcast
    - Camel-ignite
    - Camel-jpa
    - Camel-sql
    - Camel-kafka
    - Stateful-set (K8s)

- Camel supports
  - Kubernetes ConfigMap
    - Inject via ENV
    - Inject via files
  - Kubernetes Secrets
    - Inject via ENV
    - Inject via files
- Camel supports fault tolerance
  - Camel retry
    - onException
    - errorHandler
  - Circuit Breaker
    - camel-hystrix

EIP Cloud Patterns

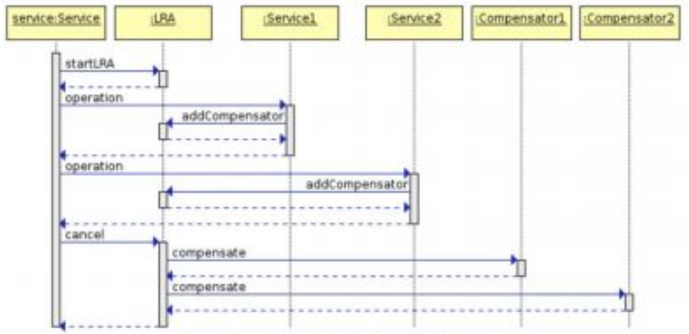
Hystrix EIP



```
from("timer:foo")
   .hystrix()
     .to("http:myservice")
   .onFallback()
     .to("bean:myfallback")
   .end()
```

EIP Cloud Patterns

Saga EIP



Sequence diagram of a failed LRA saga

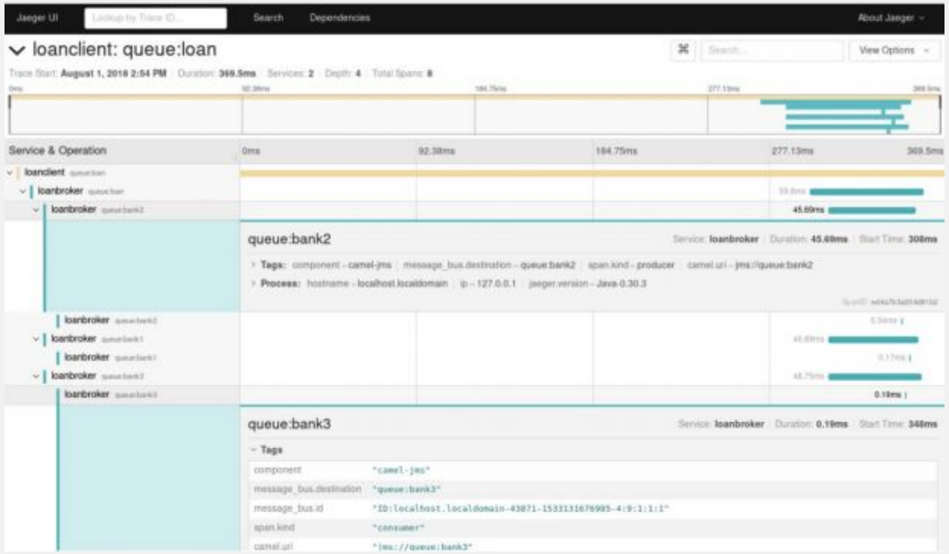
```
rest().post("train/buy/seat")
   .saga()
     .compensation("direct:cancel")
     ...
     .to("http:trainservice/buy")
```

EIP Cloud Patterns

Distributed Tracing

OPENTRACING

ZIPKIN



loanclient: queue:loan

Trace Start: **August 1, 2018 2:54 PM** | Duration: **369.5ms** | Services: **2** | Depth: **4** | Total Spans: **8**

Service & Operation	0ms	92.38ms	184.75ms	277.13ms	369.5ms
loanclient queue:loan					
loanbroker queue:loan				55.8ms	
loanbroker queue:bank2				45.69ms	

queue:bank2 — Service: **loanbroker** | Duration: **45.69ms** | Start Time: **308ms**

Tags: component - camel-jms | message_bus.destination - queue:bank2 | span.kind - producer | camel.uri - jms://queue:bank2

Process: hostname - localhost.localdomain | ip - 127.0.0.1 | jaeger.version - Java-0.30.3

Service & Operation					
loanbroker queue:bank2				0.5ms	
loanbroker queue:bank1				45.69ms	
loanbroker queue:bank1				0.17ms	
loanbroker queue:bank3				48.75ms	
loanbroker queue:bank3				0.19ms	

queue:bank3 — Service: **loanbroker** | Duration: **0.19ms** | Start Time: **348ms**

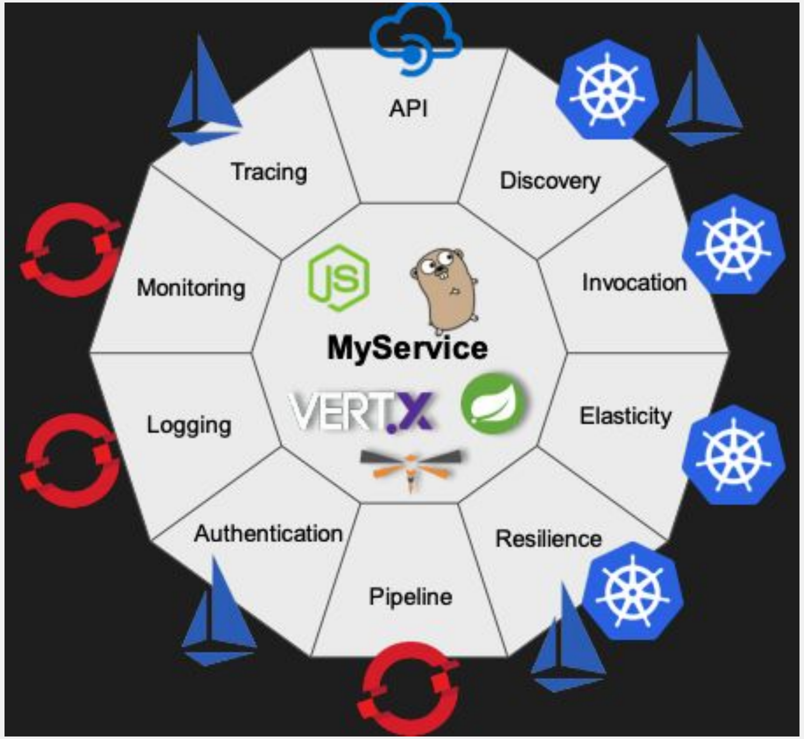
Tags

component	"camel-jms"
message_bus.destination	"queue:bank3"
message_bus.id	"ID:localhost.localdomain-43871-1533131676985-4:9:1:1:1"
span.kind	"consumer"
camel.uri	"jms://queue:bank3"

Jaeger UI

Camel Cloud Capabilities

- Camel Zipkin
- Camel OpenTracing
- Camel Rest
- Camel Components
- Camel Retry
- Camel Hystrix



Camel Cloud Future

- Apache Camel K
  - https://github.com/apache/camel-k
  - Blogs:
    - https://www.nicolaferraro.me/2018/10/15/introducing-camel-k/
    - https://www.nicolaferraro.me/2018/12/10/camel-k-on-knative/

Apache Camel K

What is Apache Camel K

- **A lightweight integration platform based on Apache Camel, born on Kubernetes, with serverless superpowers.**
- Based on operator-sdk
- A community-driven project
- A subproject of Apache Camel started on **August 31st, 2018**

https://github.com/apache/camel-k

What is Apache Camel K

Camel DSL, based on EIPs...

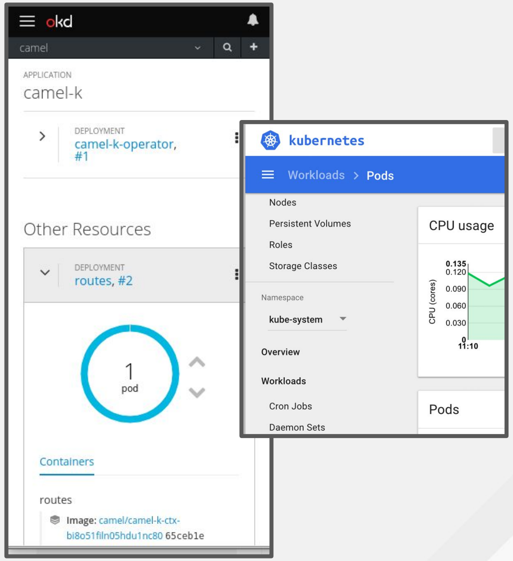
1. Create a integration file (Java, Groovy, Kotlin, JS, XML...)

```
// Lookup every second the 'www.google.com' domain name and log the
output
from(‘timer:dns?period=1s’)
     .routeId(‘dns’)
     .setHeader(‘dns.domain’)
          .constant(‘www.google.com’)
     .to(‘dns:ip’)
     .log(‘log:dns’);
```

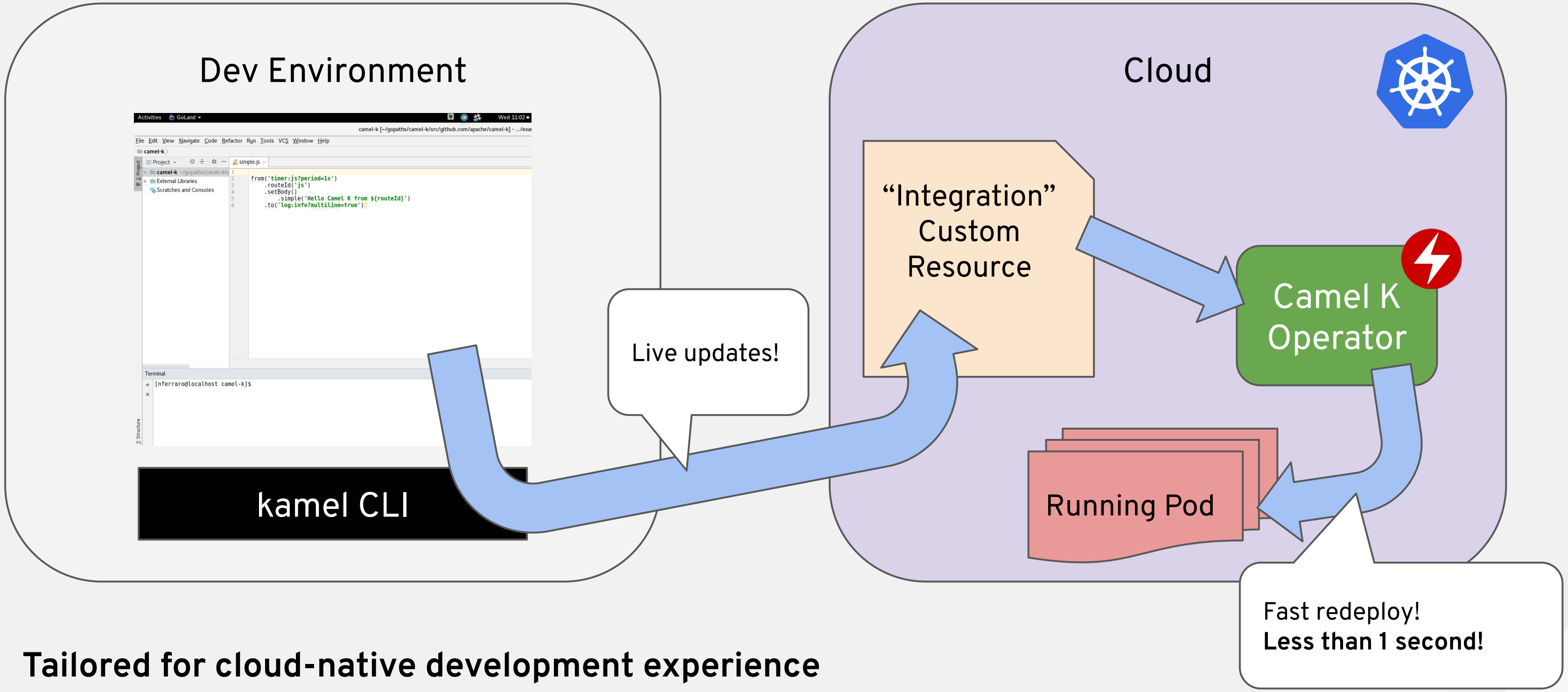
2. Run it

```
$ kamel run integration.groovy
```

3. It runs on Kubernetes



Apache Camel K Architecture



```
from('timer:js?period=1s')
    .routeId('js')
    .setBody()
        .simple('Hello Camel K from ${routeId}')
    .to('log:info?multiline=true')
```

```
[nferraro@localhost camel-k]$
```

Tailored for cloud-native development experience

Apache Camel K Performance

Apache Camel K Performance

The operator understands the code so it can

- **Choose the lightest runtime** for the integration (e.g. **Quarkus**)
- **Package a minimal set of components** that are **required by the code**
- **Fine tune Camel and JVM parameters** based on the code
- **Optimize creation of container images**

Compared to a traditional spring-boot stack

- Requires less memory and cpu
- Does not require to generate a uber jar thus improving deployment speed

Apache Camel K

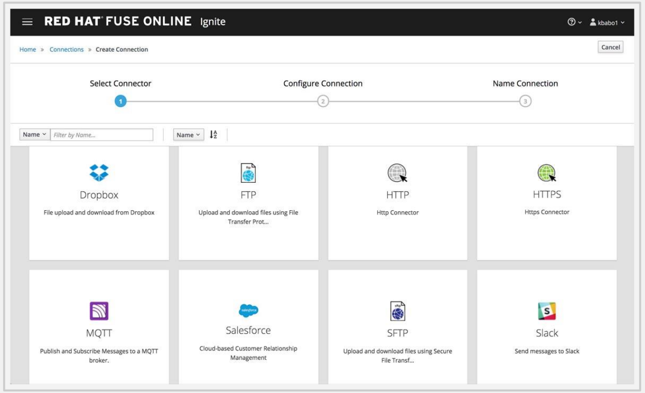
A web-based integration platform (Syndesis).

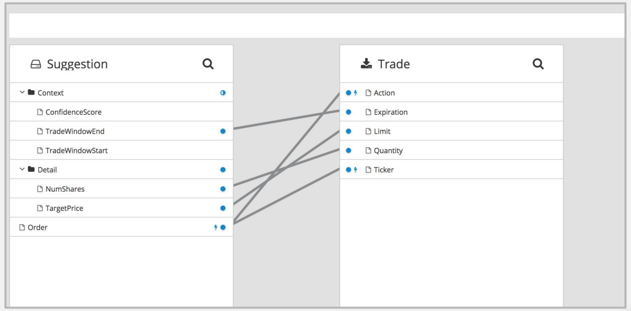
Target:

- **Citizen Integrators**

Features:

- Multiple **connectors** built from Camel components
- Few **clicks** to define a integration
- Graphical **data mapping** capabilities
- Design, expose or consume **REST API**
- Integrated with Apicur.io for API design
- Integrated with **3-scale** for API management
- Now **works with Camel K** as runtime engine for integrations!

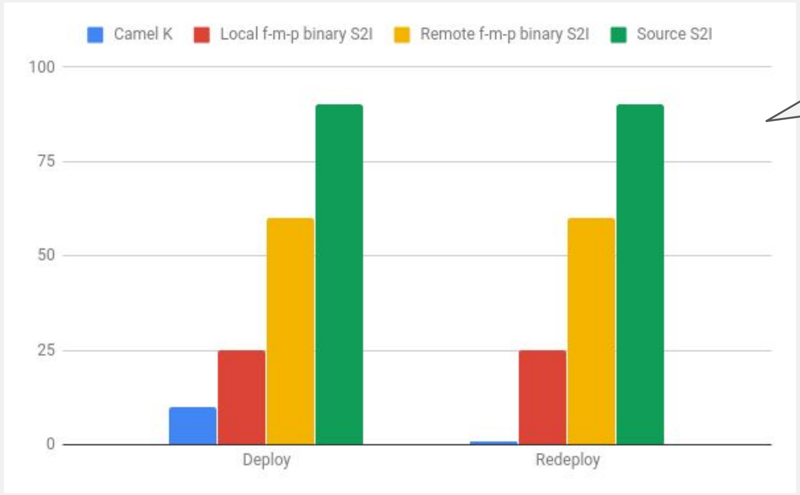

RED HAT FUSE ONLINE Ignite
Home » Connections » Create Connection
Cancel
Select Connector
Configure Connection
Name Connection
Name
Filter by Name...
Dropbox
File upload and download from Dropbox
FTP
Upload and download files using File Transfer Prot...
HTTP
Http Connector
HTTPS
Https Connector
MQTT
Publish and Subscribe Messages to a MQTT broker.
Salesforce
Cloud-based Customer Relationship Management
SFTP
Upload and download files using Secure File Transf...
Slack
Send messages to Slack



Apache Camel K Time to run an integration

... compared to the traditional spring-boot backend

SYNDESIS



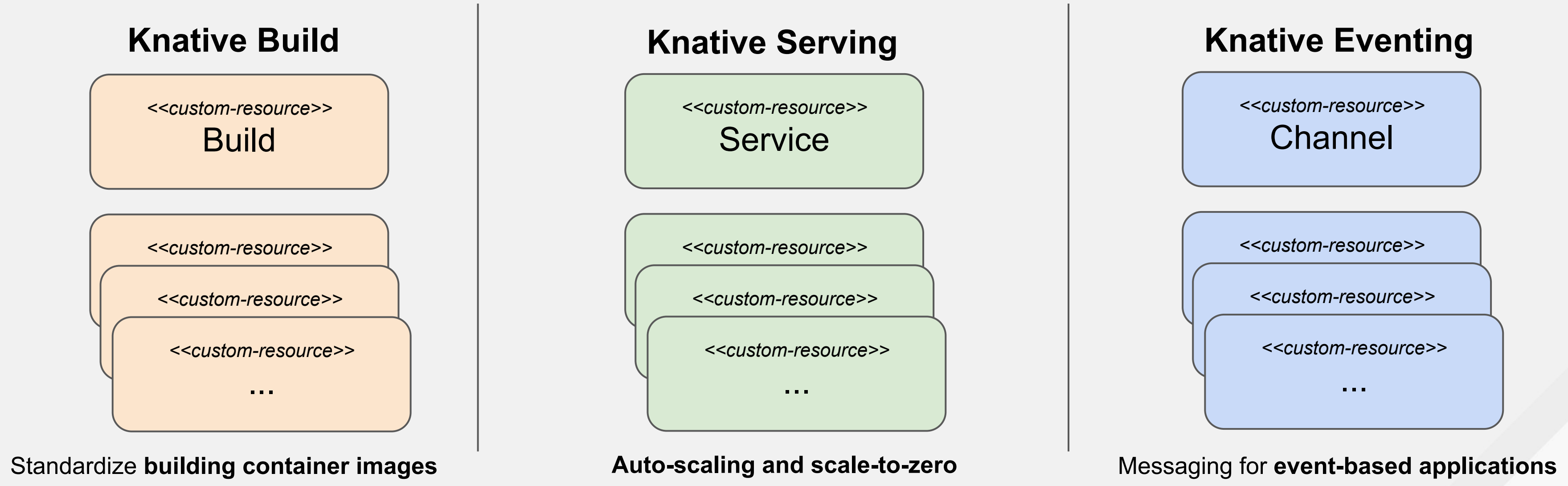
Lower is better :)

Red Hat

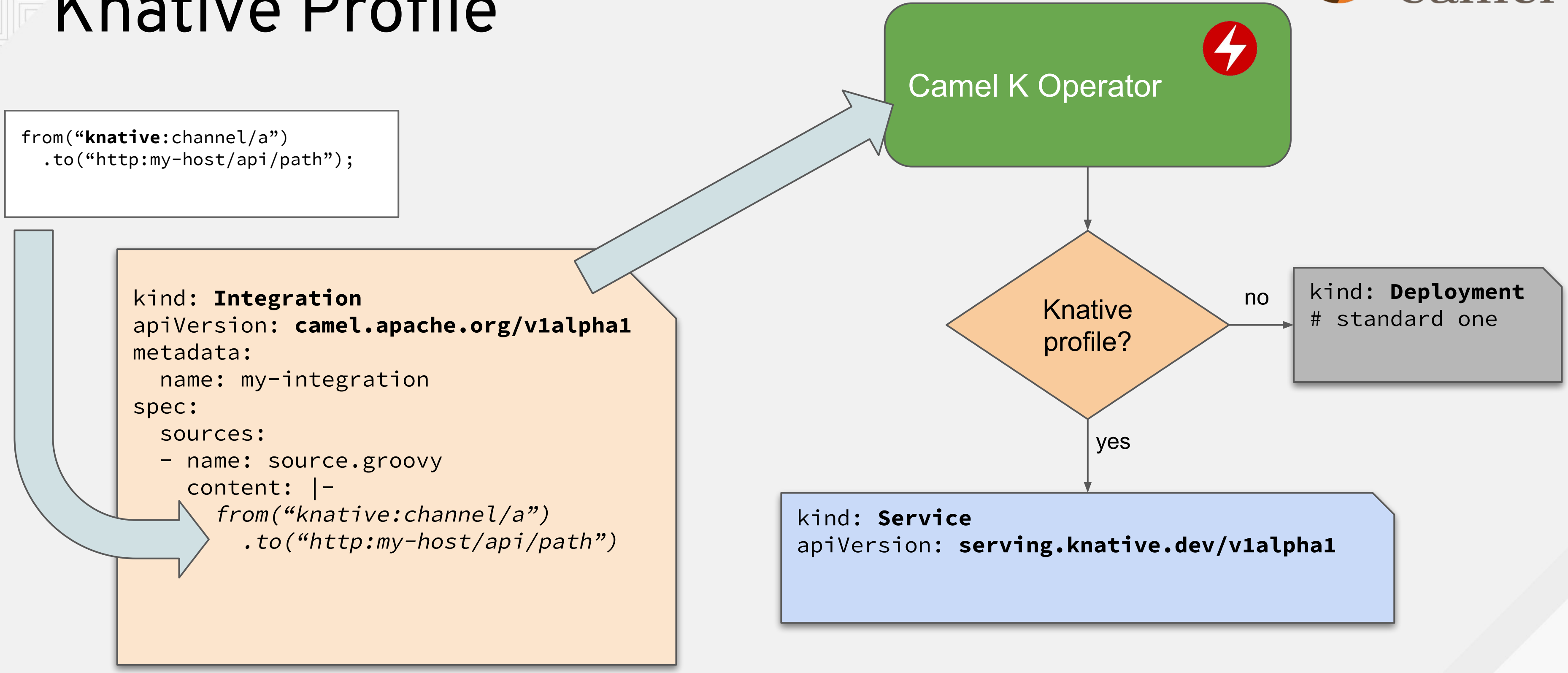
Apache Camel K
And
Knative

Camel K and Knative

Knative defines **building blocks** for "Serverless" applications (https://github.com/knative/).
A **building block is a CRD** with a controller (or "operator") that manages its lifecycle.



Knative Profile



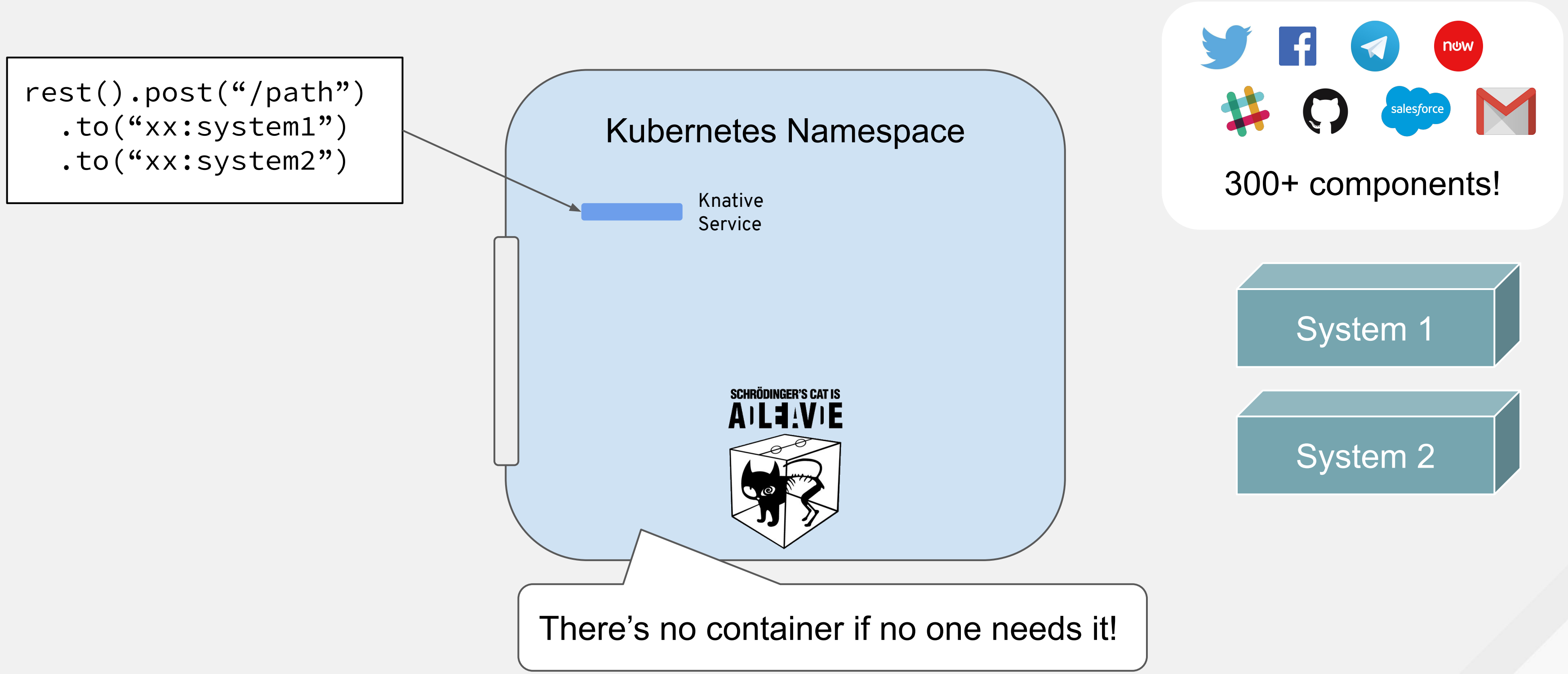
```
from("knative:channel/a")
  .to("http:my-host/api/path");
```

```
kind: Integration
apiVersion: camel.apache.org/v1alpha1
metadata:
  name: my-integration
spec:
  sources:
  - name: source.groovy
    content: |-
      from("knative:channel/a")
        .to("http:my-host/api/path")
```

```
kind: Deployment
# standard one
```

```
kind: Service
apiVersion: serving.knative.dev/v1alpha1
```

What does it mean?

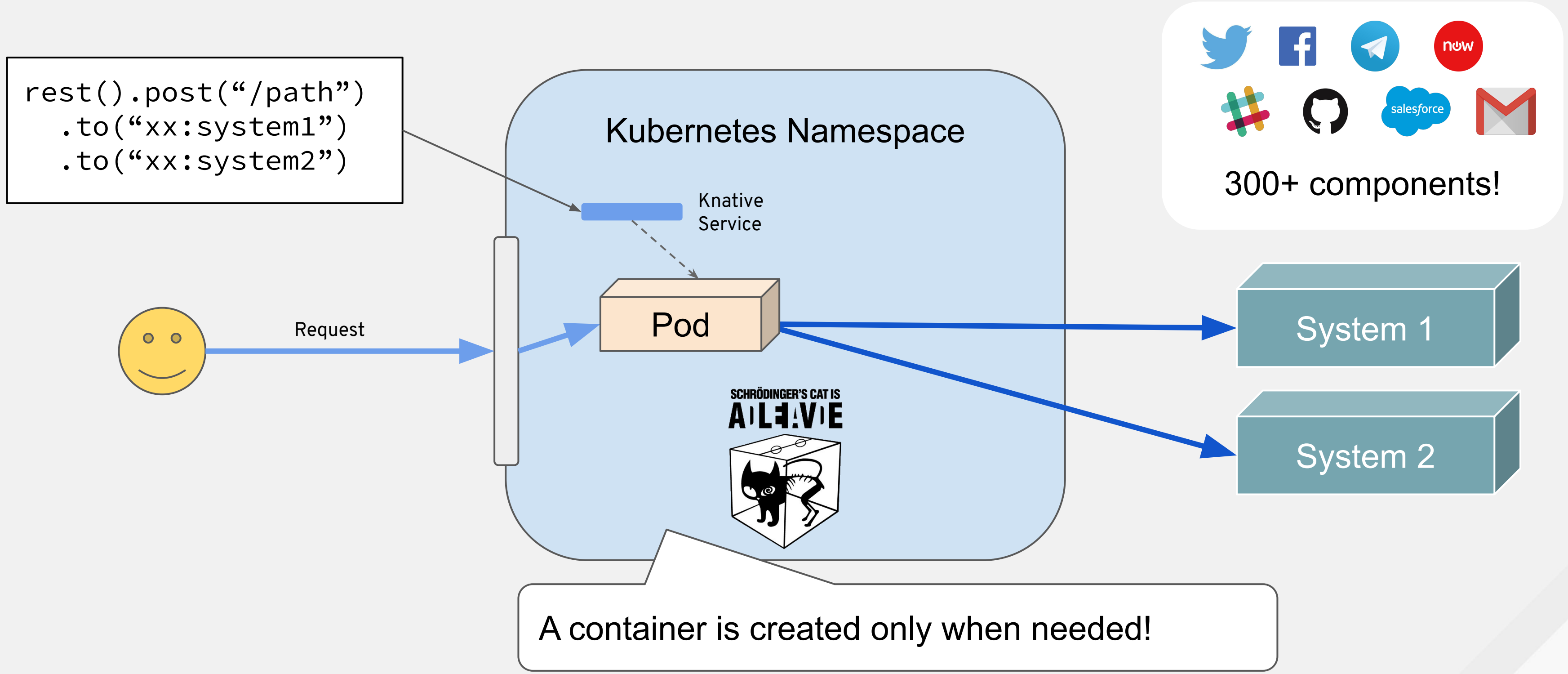


```
rest().post("/path")
  .to("xx:system1")
  .to("xx:system2")
```

There's no container if no one needs it!

What does it mean?

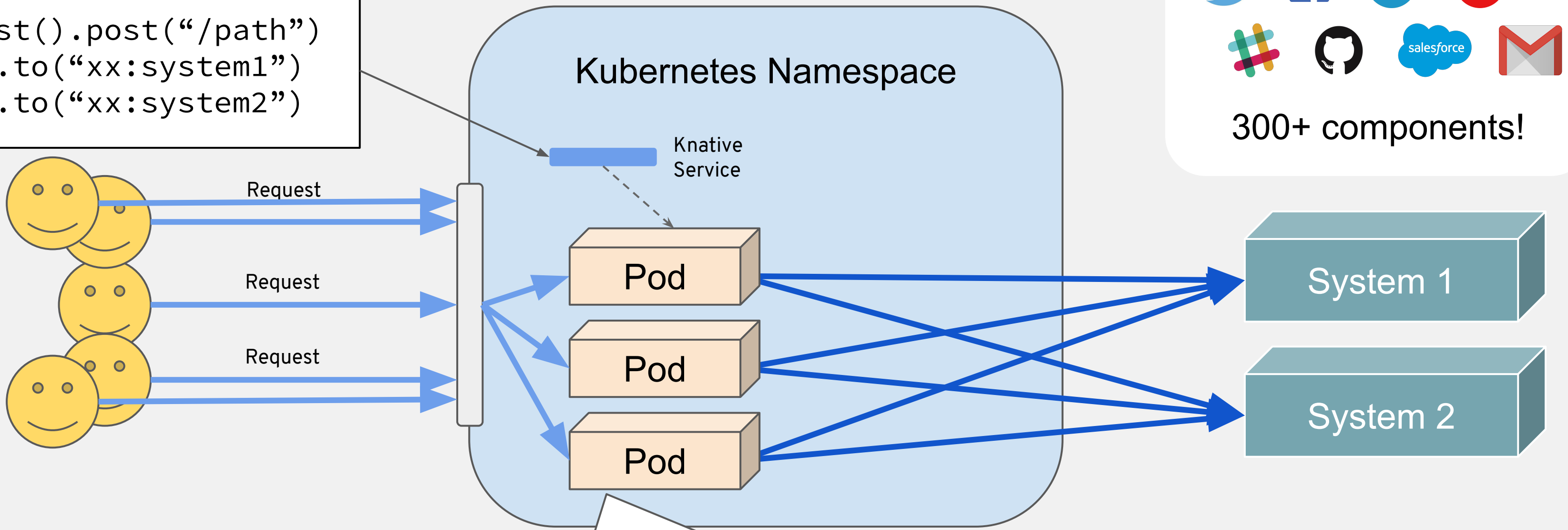
```
rest().post("/path")
  .to("xx:system1")
  .to("xx:system2")
```



A container is created only when needed!

What does it mean?

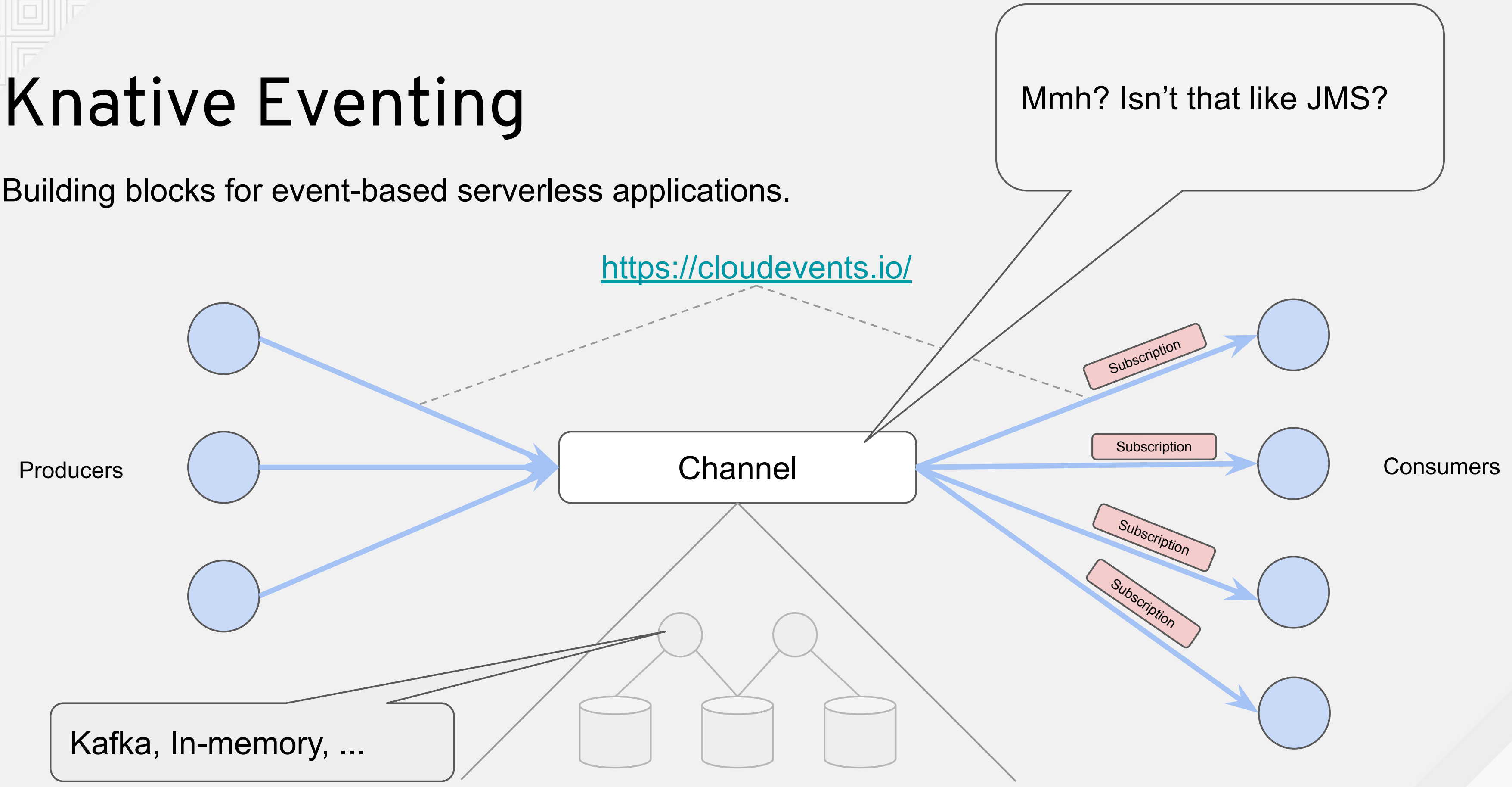
```
rest().post("/path")
  .to("xx:system1")
  .to("xx:system2")
```



Multiple containers under high load!

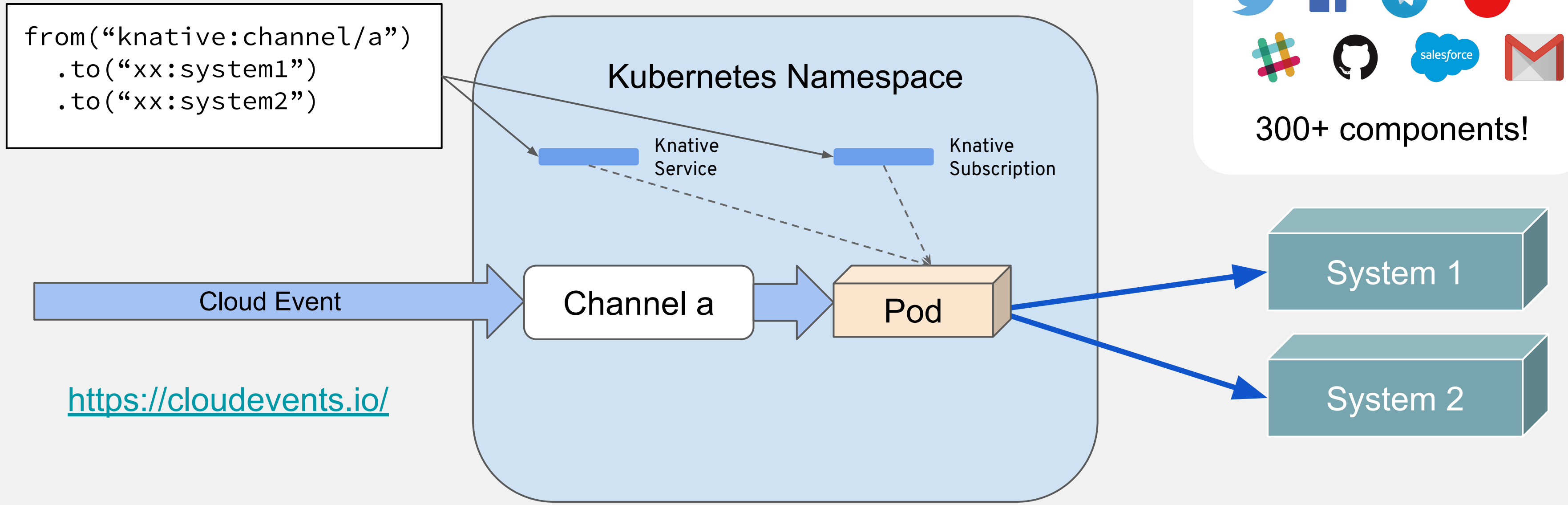
Knative Eventing

Building blocks for event-based serverless applications.

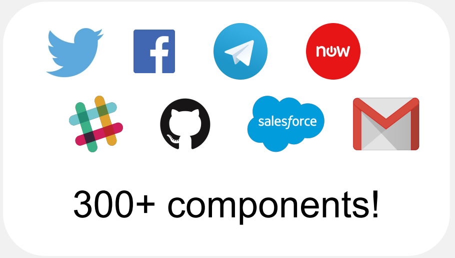


Knative Eventing

```
from("knative:channel/a")
  .to("xx:system1")
  .to("xx:system2")
```



https://cloudevents.io/

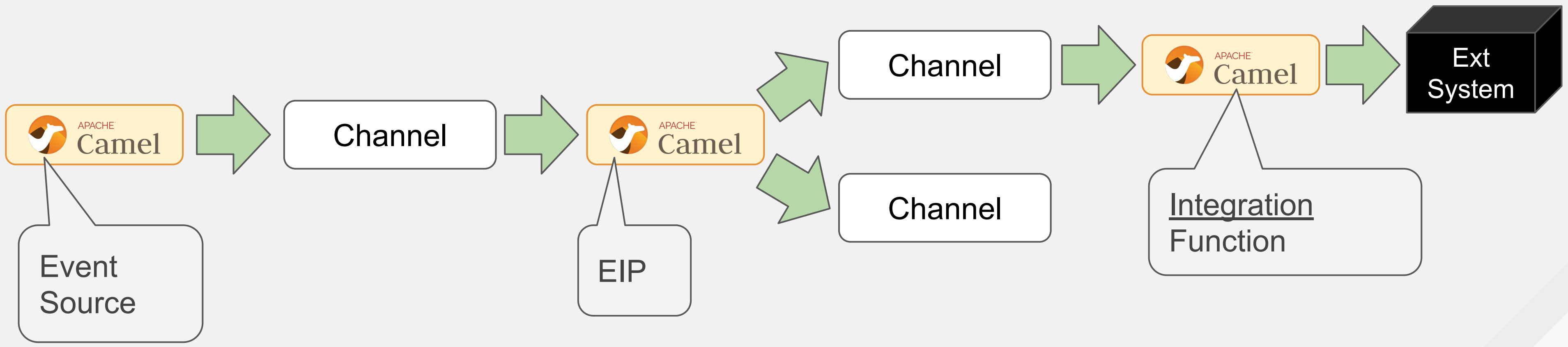


300+ components!

Camel K in Knative Eventing

Same model for different purposes

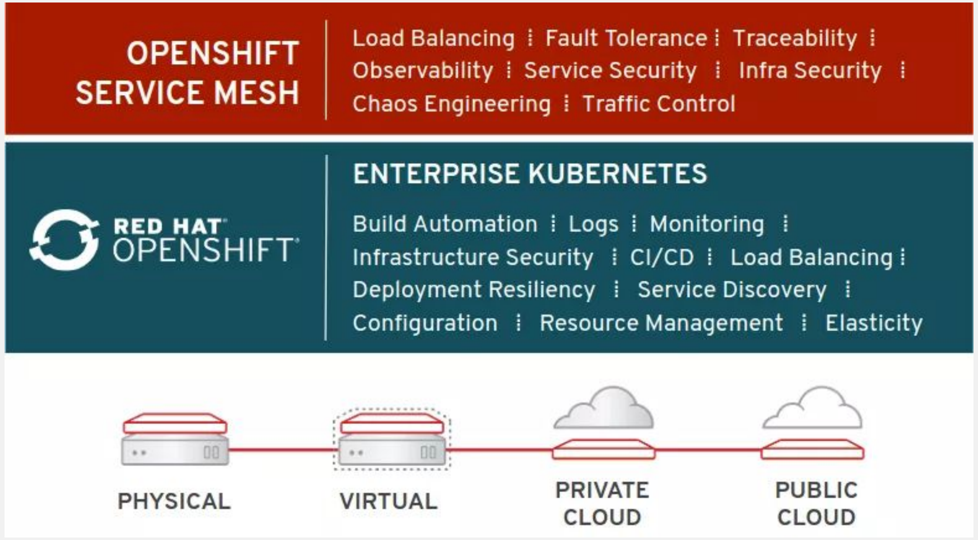
300+ components!



Recap and What’s Next …

What is Istio?

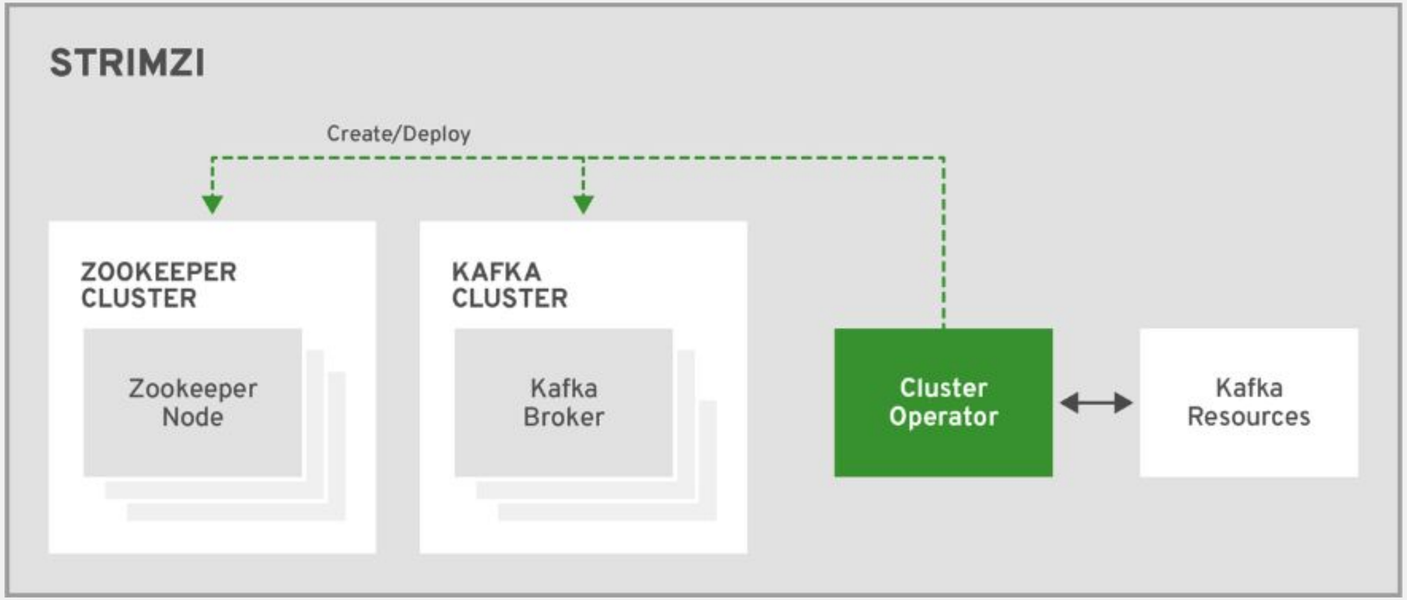
An open source service mesh providing fundamentals we need to run a distributed microservices architecture


OPENSHIFT SERVICE MESH
Load Balancing ⁝ Fault Tolerance ⁝ Traceability ⁝
Observability ⁝ Service Security ⁝ Infra Security ⁝
Chaos Engineering ⁝ Traffic Control
RED HAT OPENSHIFT
ENTERPRISE KUBERNETES
Build Automation ⁝ Logs ⁝ Monitoring ⁝
Infrastructure Security ⁝ CI/CD ⁝ Load Balancing ⁝
Deployment Resiliency ⁝ Service Discovery ⁝
Configuration ⁝ Resource Management ⁝ Elasticity
PHYSICAL
VIRTUAL
PRIVATE CLOUD
PUBLIC CLOUD

https://learn.openshift.com/servicemesh/1-introduction

What is AMQ Streaming?

A high-performance data streaming capability based on Apache Kafka and Strimzi



What is Camel K?

A lightweight integration platform, born on Kubernetes, with serverless superpowers.

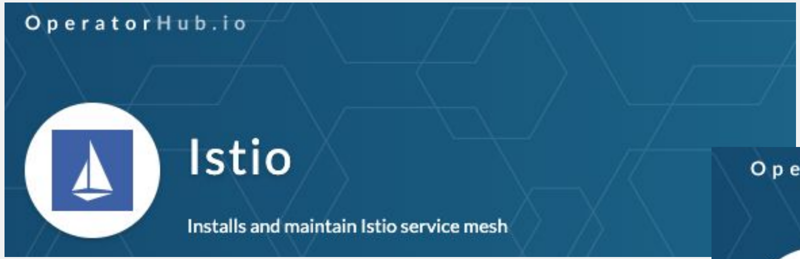
1.

2.

3.

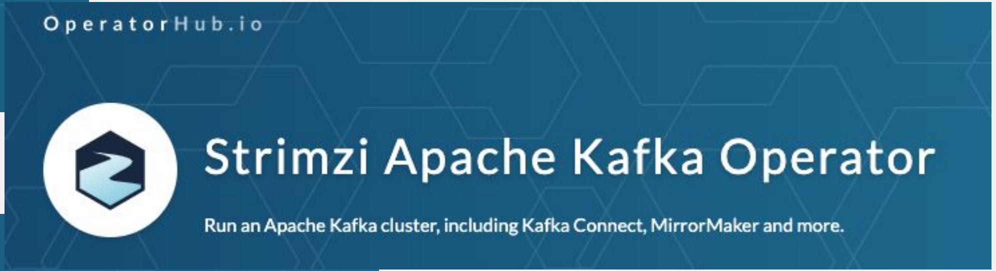
Runs on “vanilla” Kubernetes (1), Openshift (2) and gives its best on a Knative-powered cluster (3)!

OperatorHub.io


OperatorHub.io

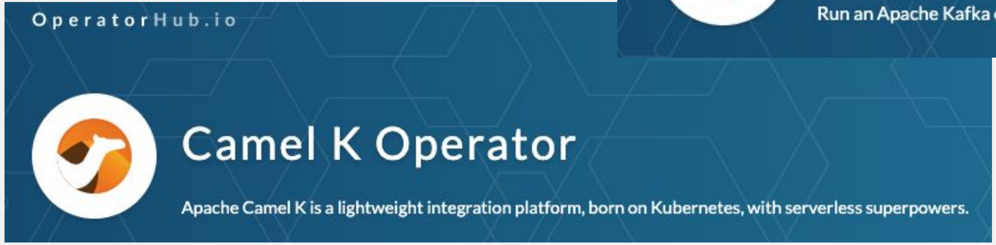
Istio

Installs and maintain Istio service mesh


OperatorHub.io

Strimzi Apache Kafka Operator

Run an Apache Kafka cluster, including Kafka Connect, MirrorMaker and more.


OperatorHub.io

Camel K Operator

Apache Camel K is a lightweight integration platform, born on Kubernetes, with serverless superpowers.

Red Hat

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos

Collateral

- http://developers.redhat.com/promotions/microservices-for-java-developers
- https://developers.redhat.com/books/introducing-istio-service-mesh-microservices/
- https://developers.redhat.com/books/adventures-aboard-kluster-kruise-ship/old/
- https://www.manning.com/books/camel-in-action-second-edition
- https://github.com/davsclaus/camel-riders-in-the-cloud/tree/webinar2019