Serverless Integration Anatomy

With Camel K, KNative, Kafka and Kubernetes

Christina Lin
Technical Evangelist

APACHECON @Home
Sep 29 - Oct 1, 2020

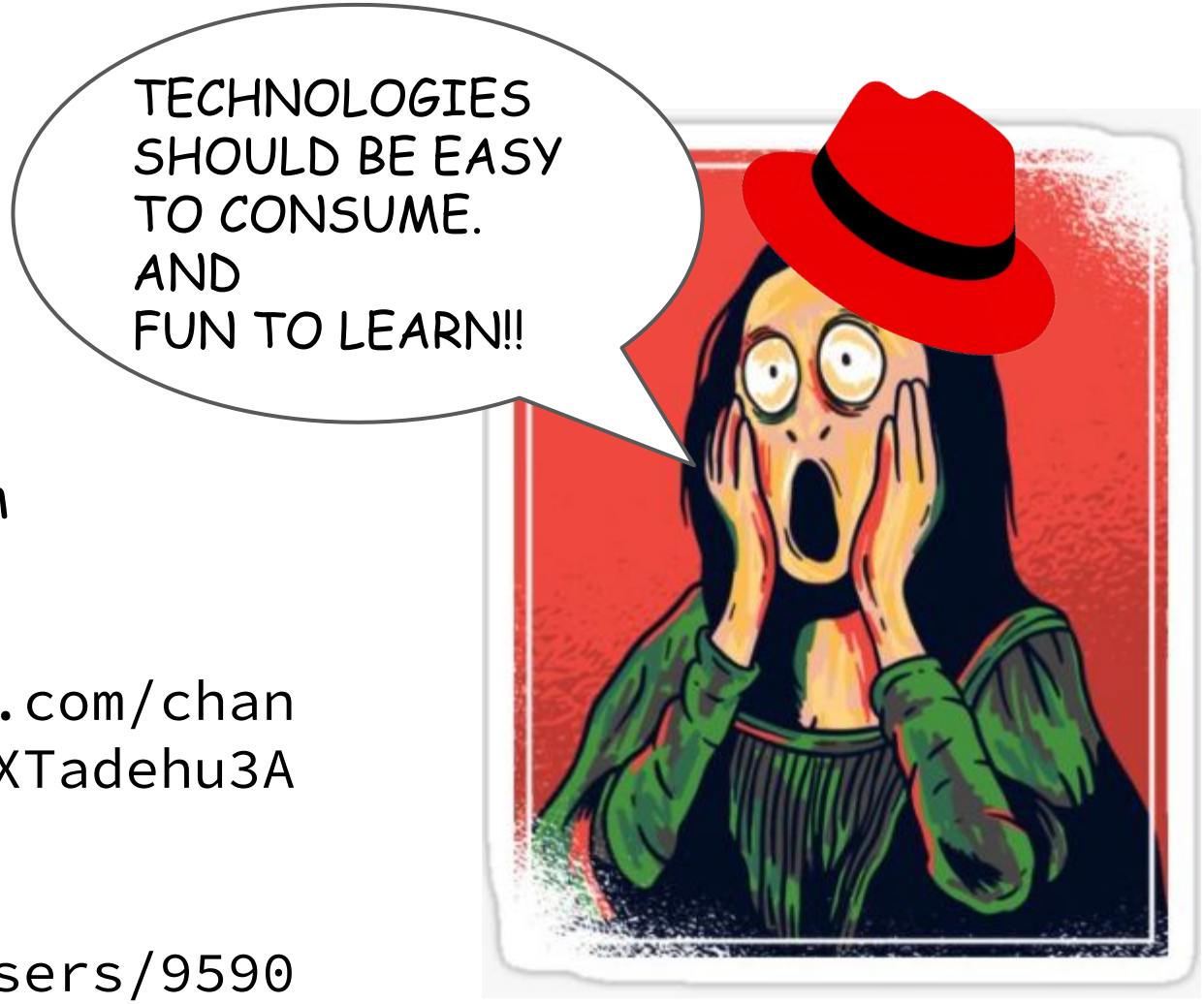
Christina Lin

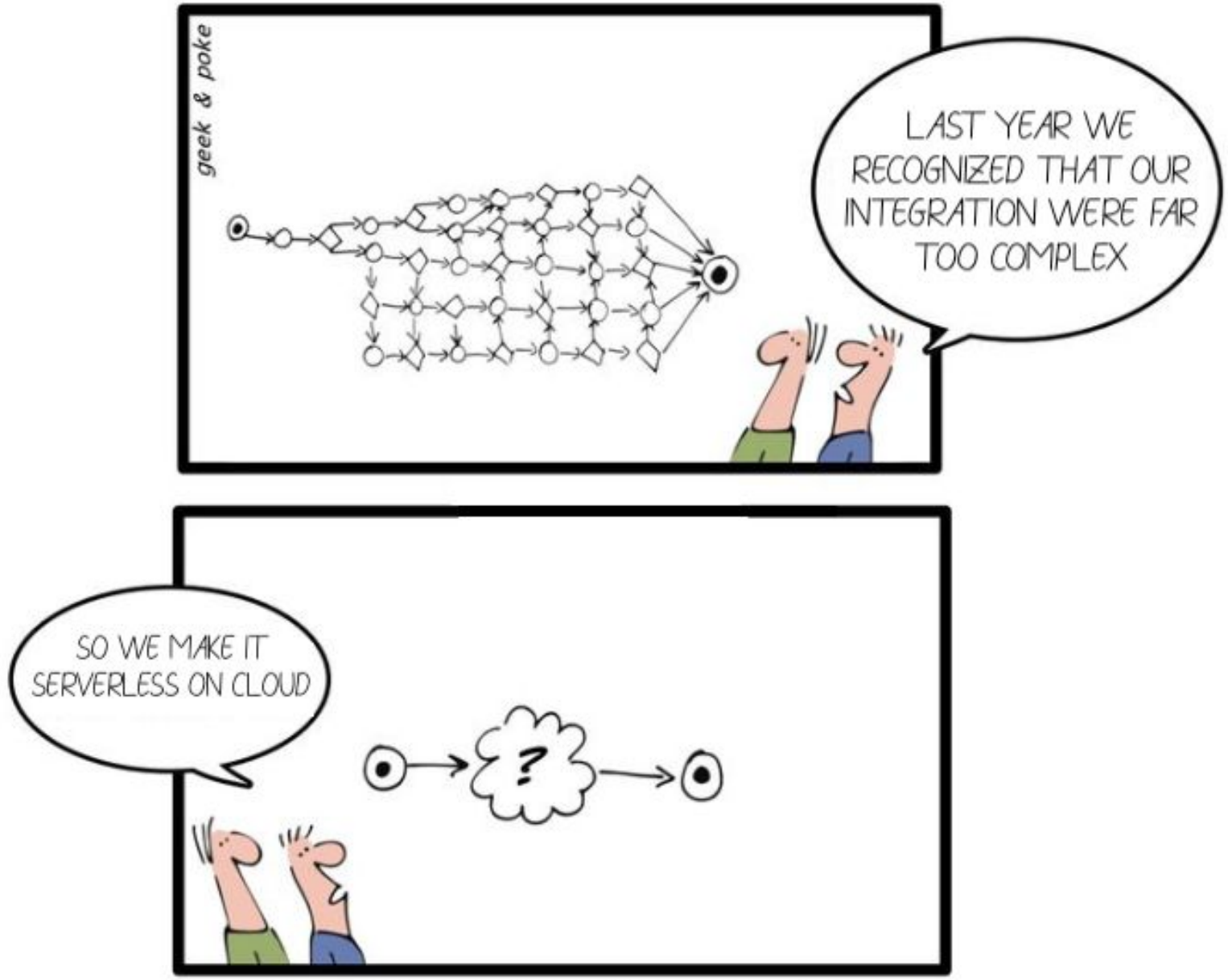
FOLLOW ME on

TWITTER: Christina_wm

YOUTUBE:
https://www.youtube.com/channel/UCY_o1j8mhNqb3hXTadehu3A

DZONE:
https://dzone.com/users/959031/weimeilin.html





LET THE CLOUDS MAKE YOUR LIFE EASIER

Beyond PaaS,
DEVOPS != DEV

Scale by Demand

Near real-time latency

Optimize
Resource usage

SERVERLESS

System Admin

CODE and RUN

Developer

Architect

Faster time to market

Avoid random, arbitrary workload prediction

Lower operational cost

Reduced packaging and deployment complexity

Flexible Scalability

THE WONDERFUL WORLD OF SERVERLESS

You are overcomplicate

COSTLY!!

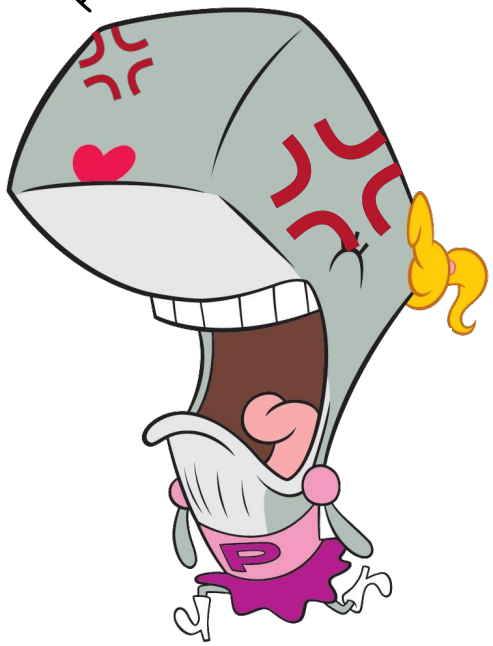
You are rigid and slow

SERVERLESS

Vendor Lock in

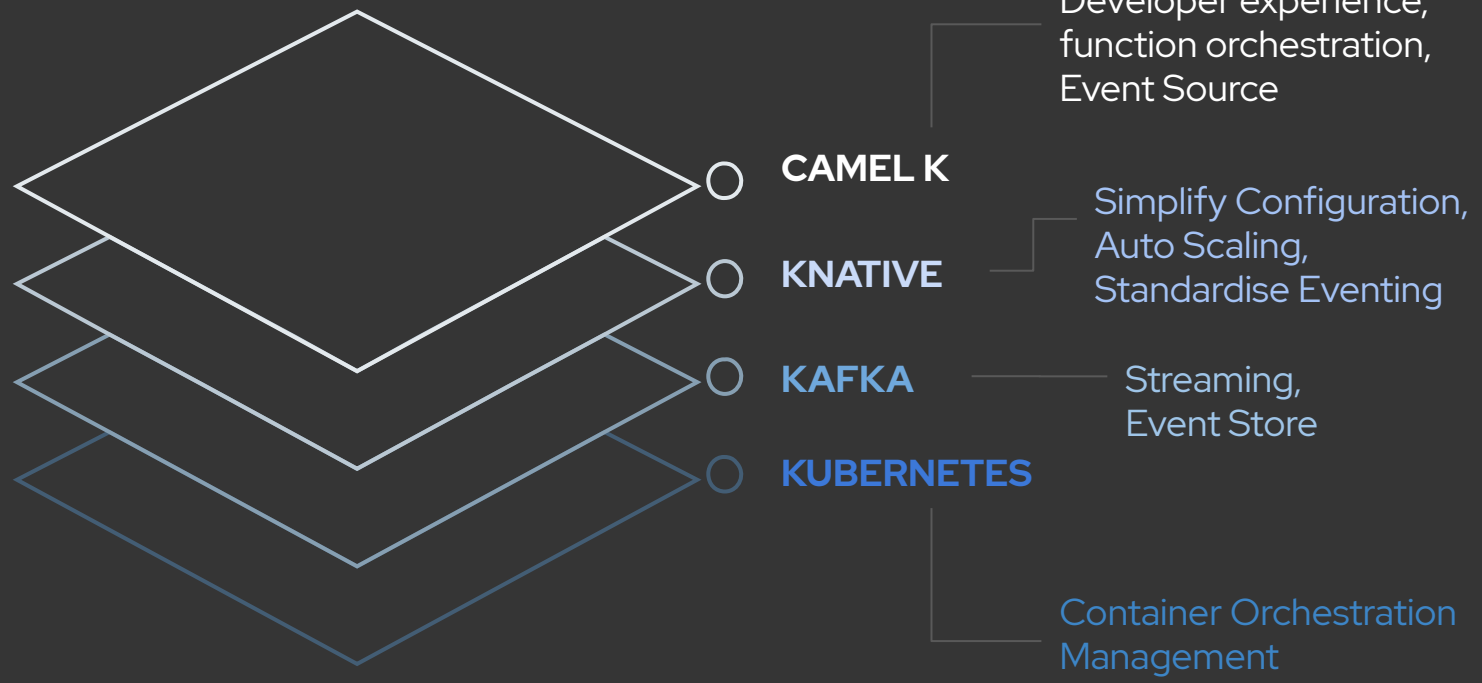
Proper Enterprise Standards?

Can you handle states?



CONTAINER

BEST OF BOTH WORLDS

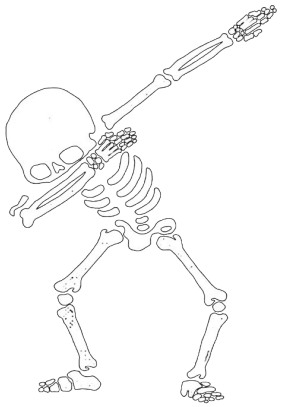


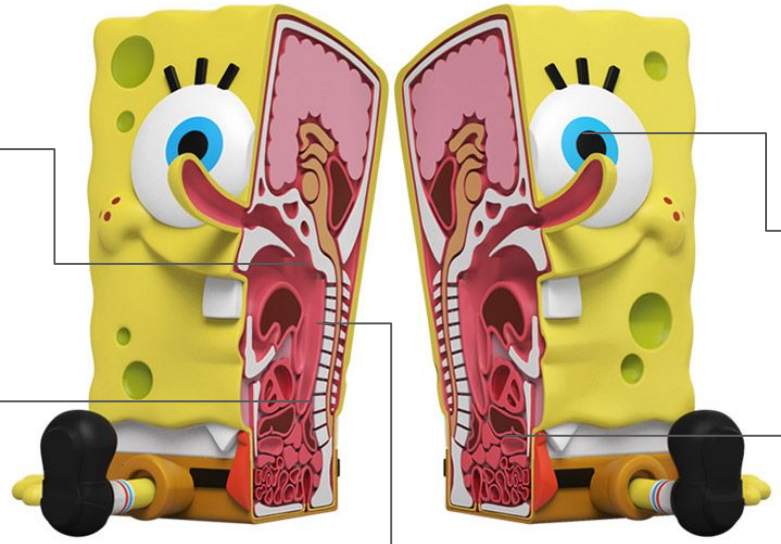
HOW DOES IT WORK?

Deploying, running, and managing serverless application on Kubernetes. With proper event mesh architecture and best developer experience.

Serverless Integration Anatomy

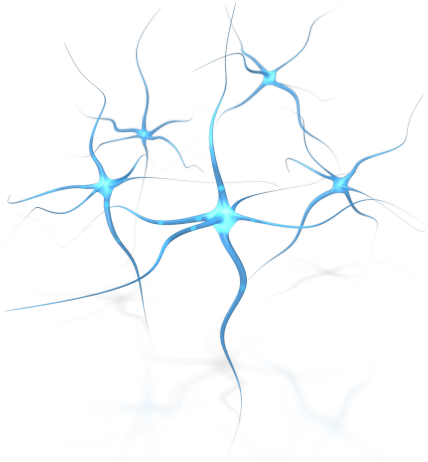
KUBERNETES



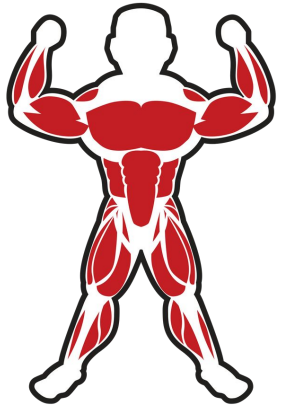


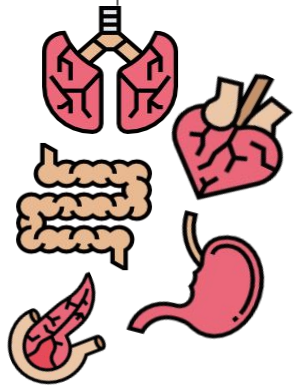
CAMEL K

KAFKA



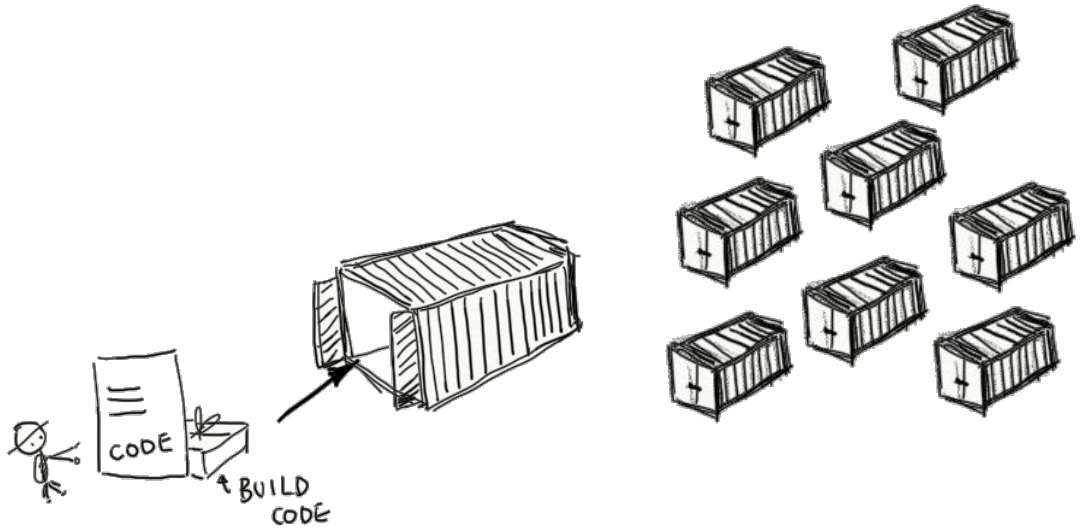
KNATIVE





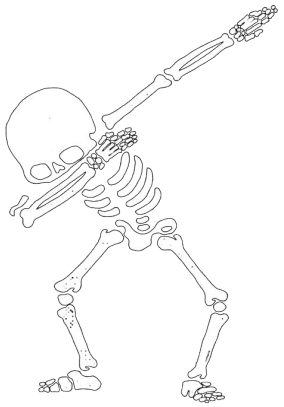
KUBERNETES/OPENSHIFT

OVERVIEW



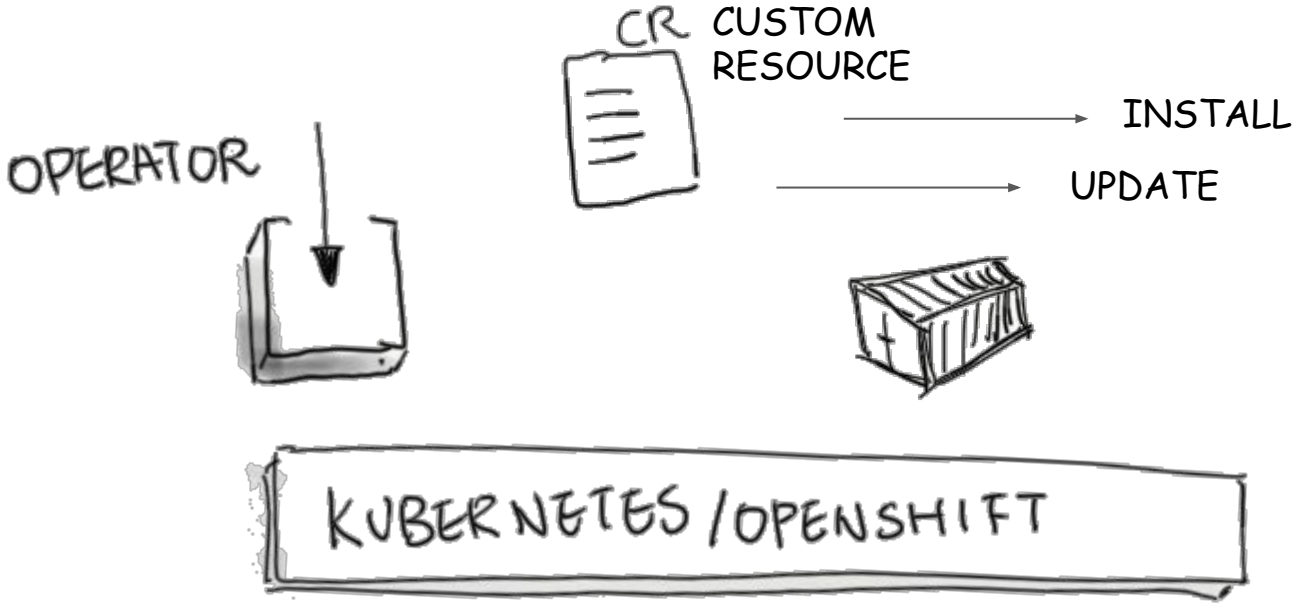
CONTAINER MANAGEMENT (LIFE CYCLE)
MONITORING, METERING
IMAGE REGISTERY
SELF HEALING
SERVICE DISCOVERY
CONFIGURATION
LOGGING
AUTOMATION

HYBRID



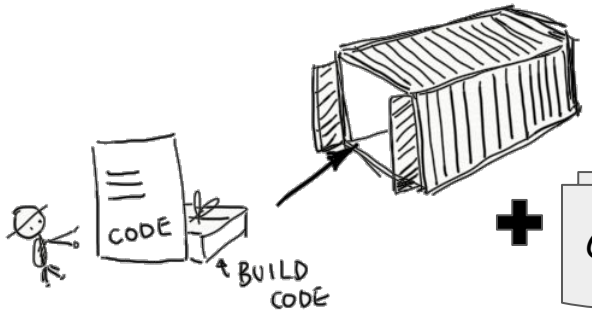
KUBERNETES/OPENSHIFT

OPERATOR PATTERN



KNATIVE

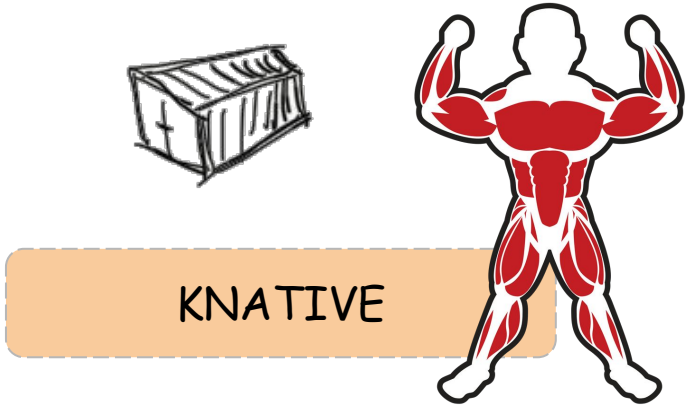
OVERVIEW

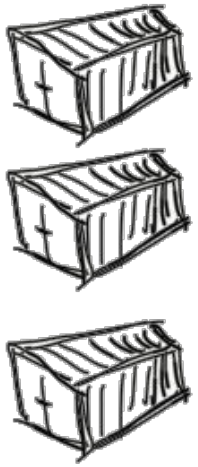


+ ROUTE
+ CONFIG
+ SECRET
+ SERVICE
+ AUTO SCALING

DEVOPS

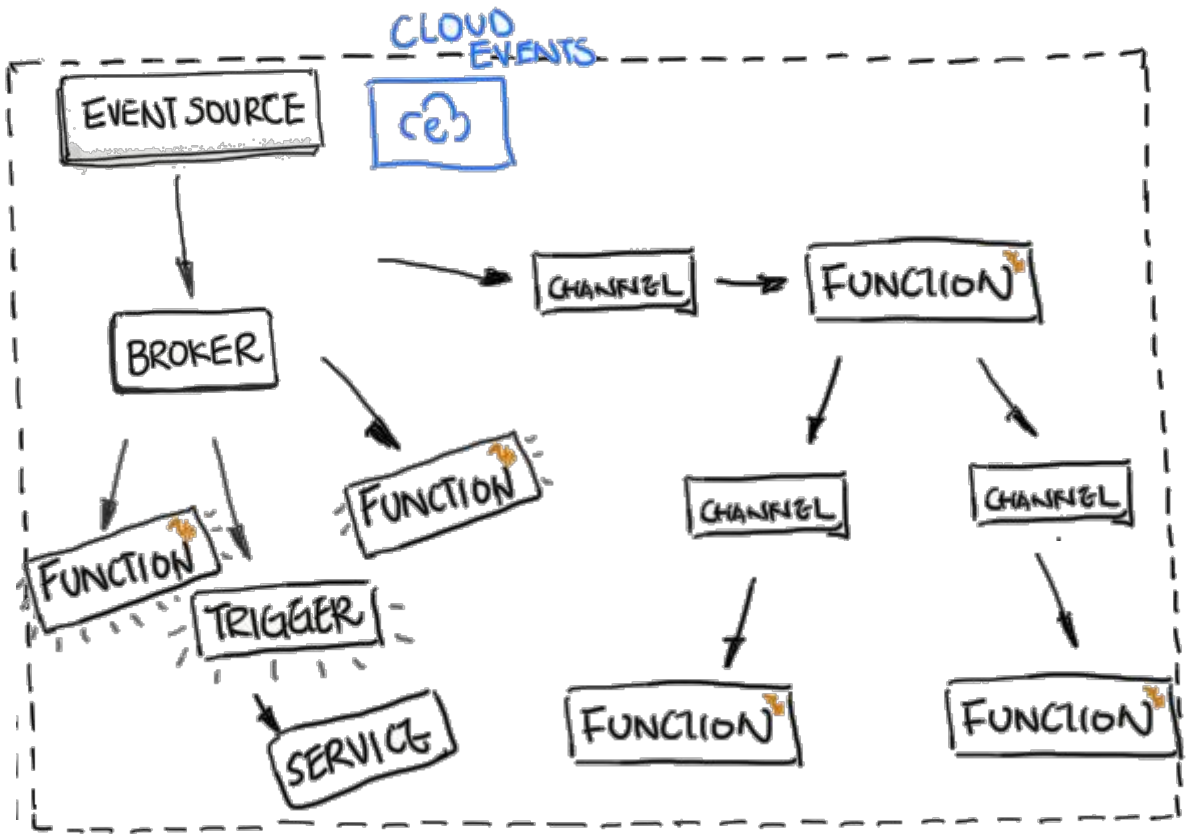
DEV

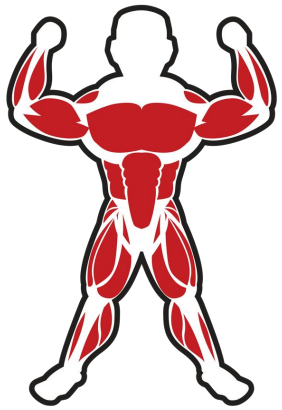




KNATIVE

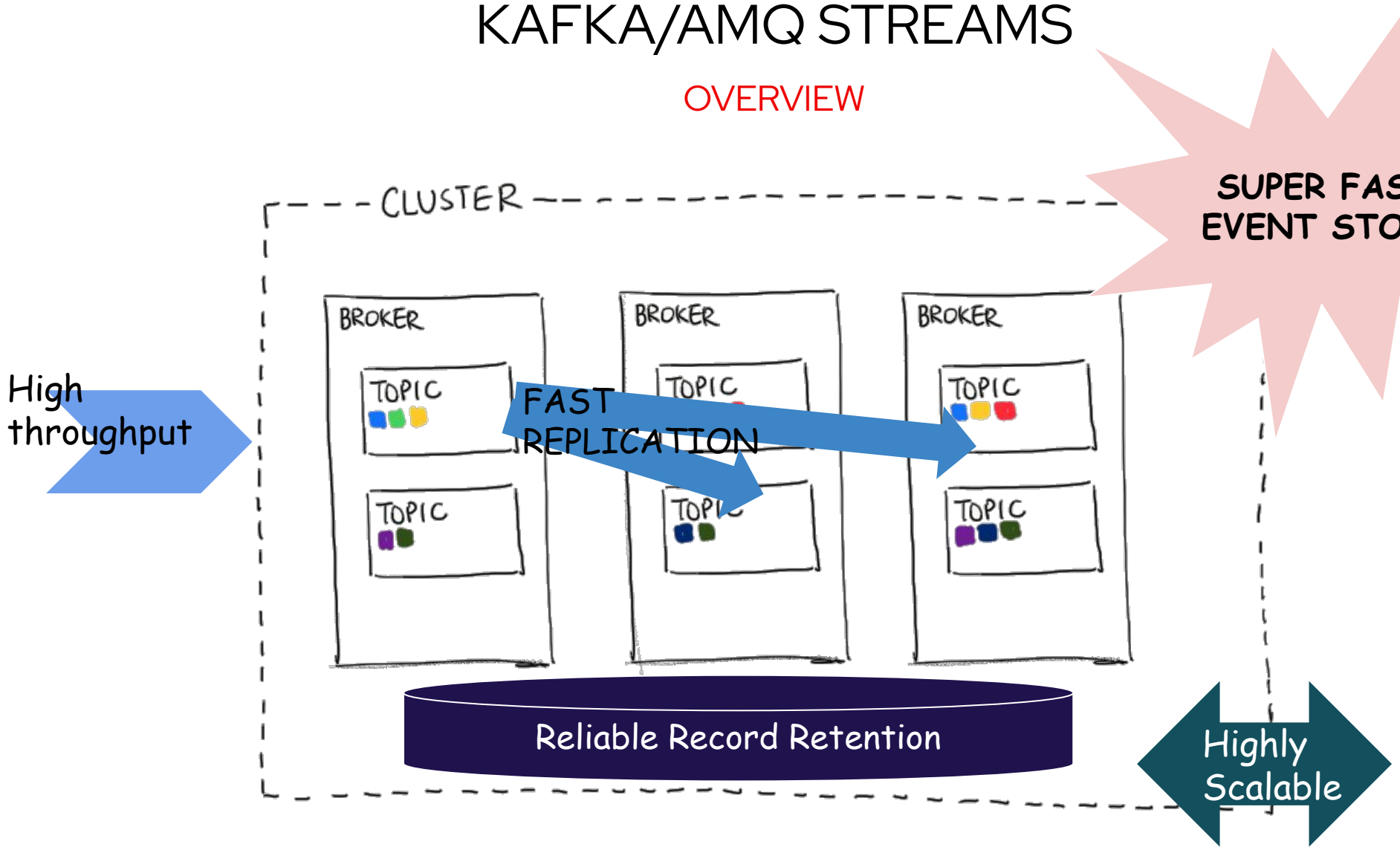
EVENTING





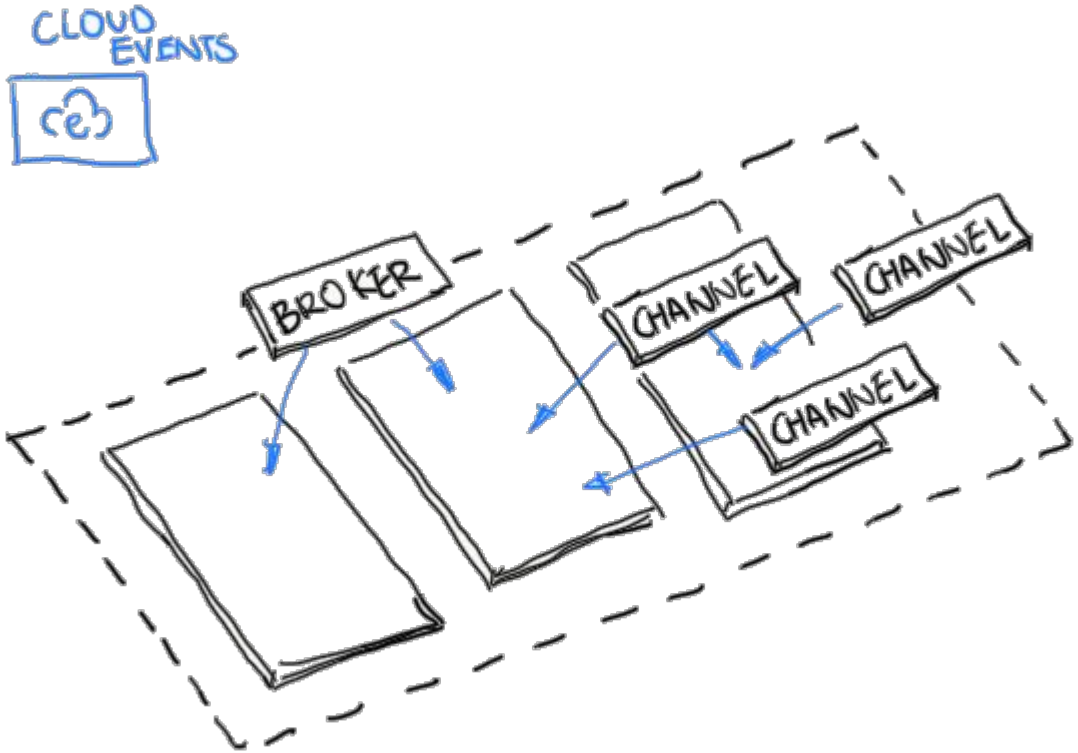
KAFKA/AMQ STREAMS

OVERVIEW



KAFKA/AMQ STREAMS

OVERVIEW



KAFKA/AMQ STREAMS

CONFIG

```yaml
apiVersion: eventing.knative.dev/v1
kind: Broker
metadata:
  annotations:
    # case-sensitive
    eventing.knative.dev/broker.class: Kafka
  name: default
  namespace: default
spec:
  # Configuration specific to this broker.
  config:
    apiVersion: v1
    kind: ConfigMap
    name: kafka-broker-config
    namespace: knative-eventing
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: kafka-broker-config
  namespace: knative-eventing
data:
  # Number of topic partitions
  default.topic.partitions: "10"
  # Replication factor of topic messages.
  default.topic.replication.factor: "1"
    bootstrap.servers: "my-cluster-kafka-bootstrap.streams:9092"
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: kafka-channel
  namespace: knative-eventing
data:
  channelTemplateSpec: |
    apiVersion: messaging.knative.dev/v1beta1
    kind: KafkaChannel
    spec:
      numPartitions: 3
      replicationFactor: 1
```

CAMEL K

DEVELOPER JOY

```
from("knative:channel/xxxx")
  .transform()...
  .to("kafka:topic")

from("kafka:topic")
  .to("http:my-host/api/path")
```

1

Create
integration file

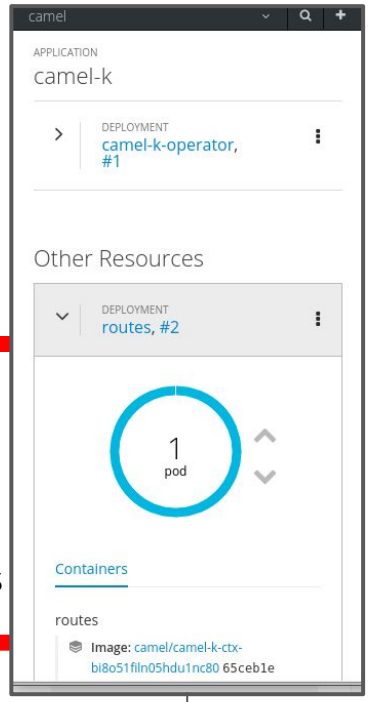
2

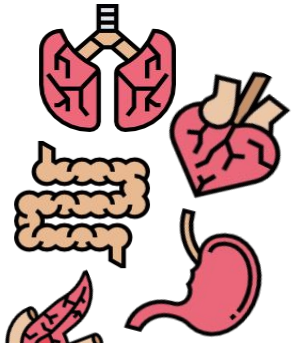
EXECUTE
CLI Tools

```
$ kamel run integration.java
```

3

RUNNING
Serverless on
OpenShift/Kubernetes



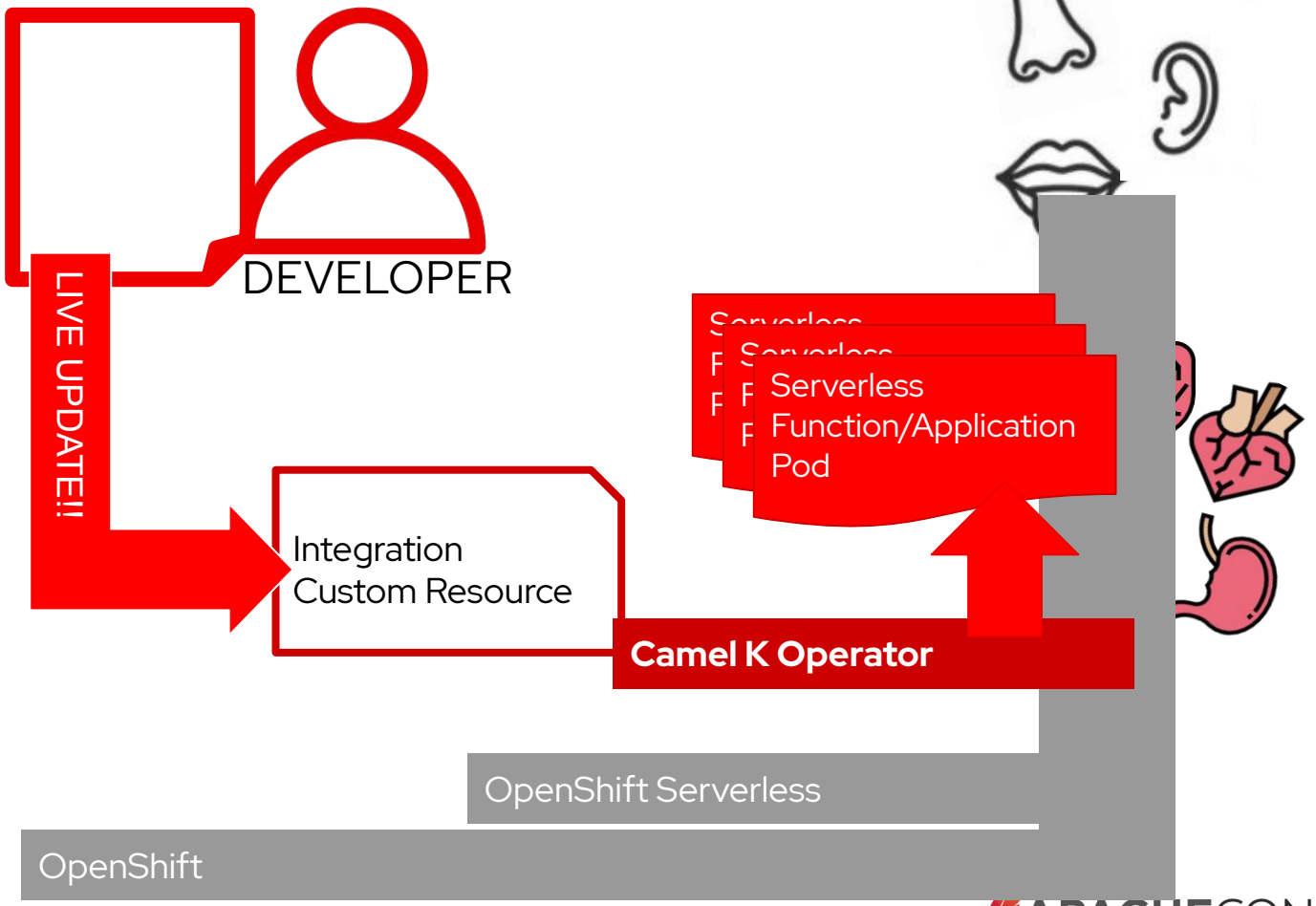


18

EVENT SOURCE & FUNCTIONS

Tailored for ***cloud-native development*** experience.

- Live coding on cloud
- Built-in dependency management
- Rapid deployment, incremental updates
- Automate cloud resource generation
- Highly customizable



SERVERLESS INTEGRATION

PERSONA RESPONSIBILITY

OPERATION

- MUST BE SECURE
- NO PUBLIC ARE THE SAME
- INCONSISTENCY DRIVE UP COST AND COMPLEXITY
- UNIFY OPERATION

KUBERNETES/OPENSHIFT

PUBLIC

PRIVATE

DEVOPS

- CONTAINER ORCHESTRATION
- CI/CD
- DEPLOYMENT STRATEGY
- SCHEDULING

APPLICATION

KNATIVE

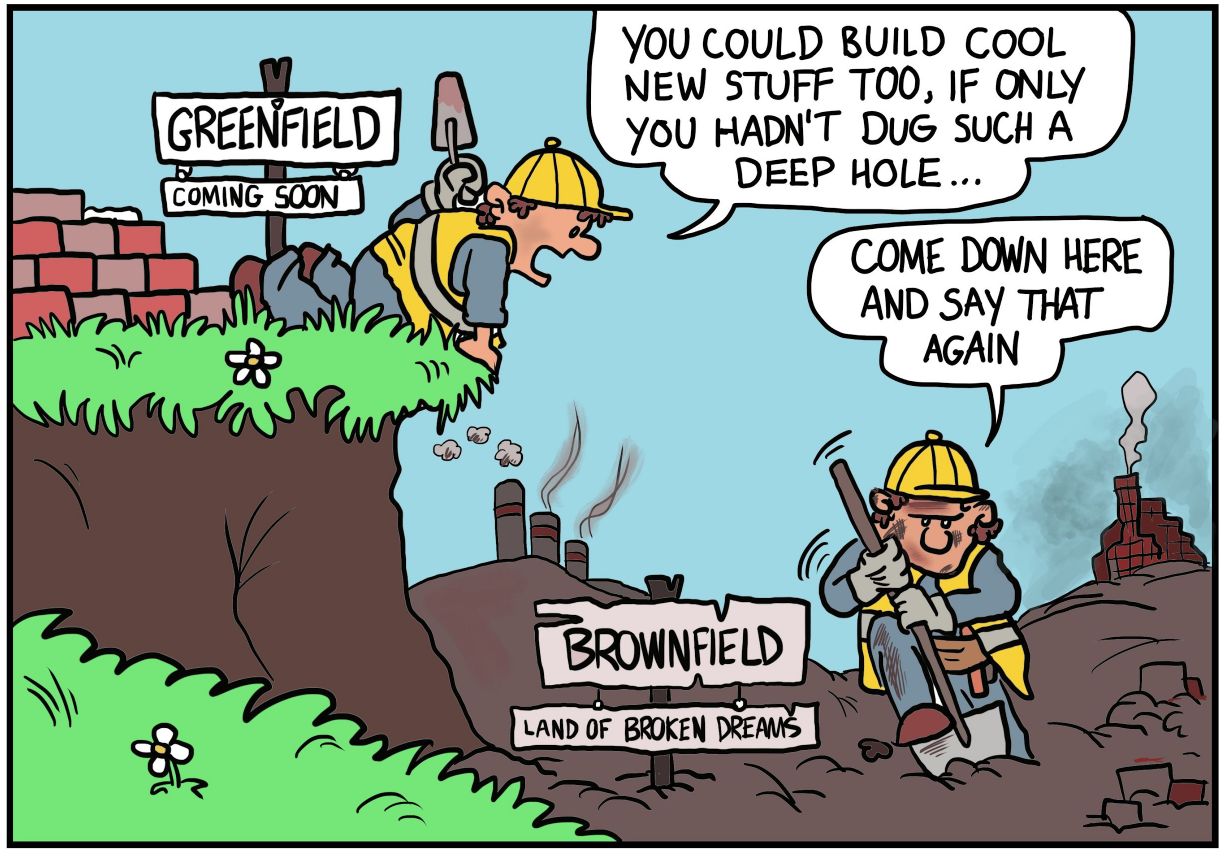
KUBERNETES/OPENSHIFT

PUBLIC

PRIVATE

SERVERLESS INTEGRATION

PERSONA RESPONSIBILITY



SERVERLESS INTEGRATION

PERSONA RESPONSIBILITY

- CONNECTIVITY
- ORCHESTRATION
- DATA TRANSFORM
- APPLICATION LOGIC

BROWNFIELD DEVELOPER

SOURCE

TRANSFORM

SINK

ANTI-CORRUPTION

APPLICATIONS

KUBERNETES/ OPENSHIFT

LEGACY APPLICATIONS

LEGACY APPLICATIONS

DEVELOPERS

SOURCE

SINK

COMPOSITION

ORCHESTRATION

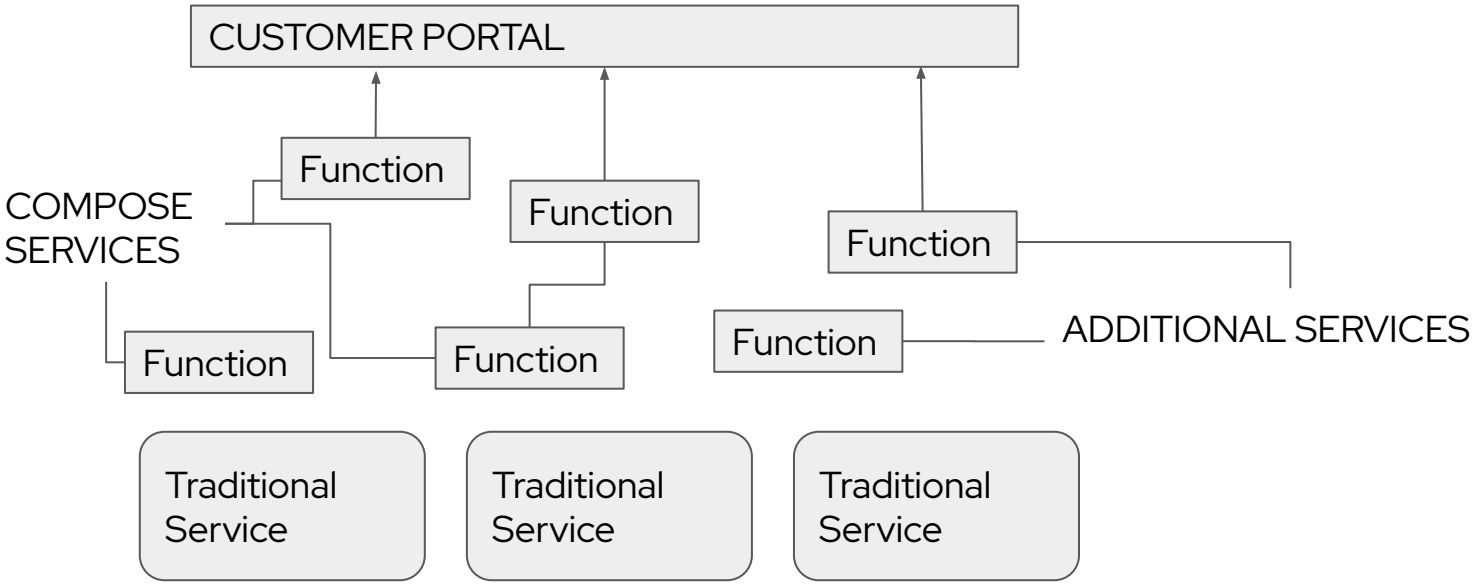
EVENTS

CALLS

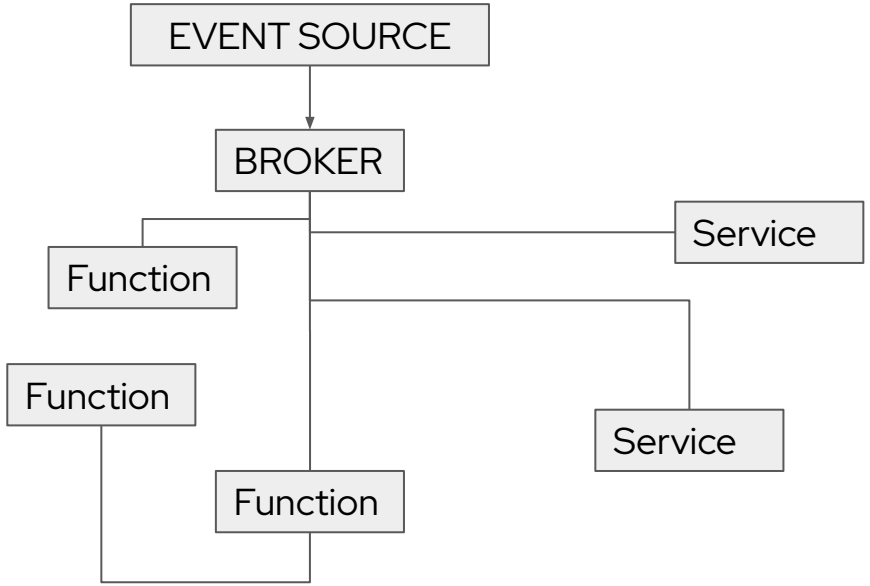
SERVERLESS INTEGRATION

USE CASE

FINANCIAL
BETTER CUSTOMER EXPERIENCE
OMNI CHANNEL



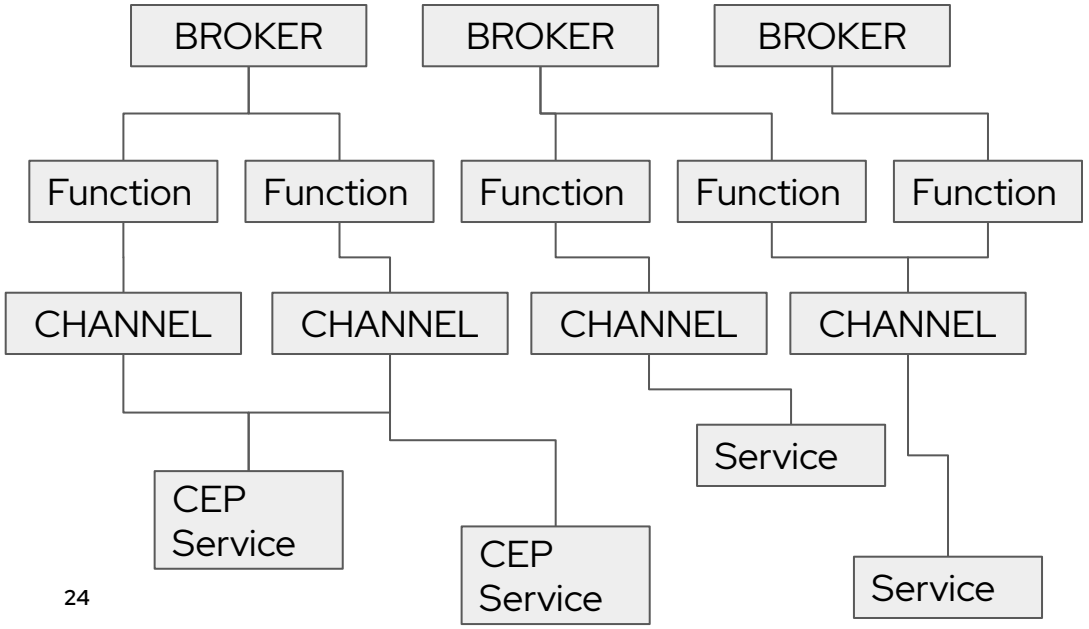
TRAVEL
INSTANT BROADCAST



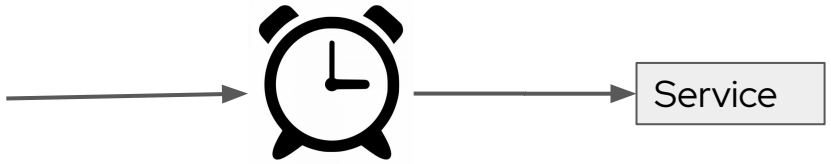
SERVERLESS INTEGRATION

USE CASE

TELCOM
IOT STREAMING PROCESSING



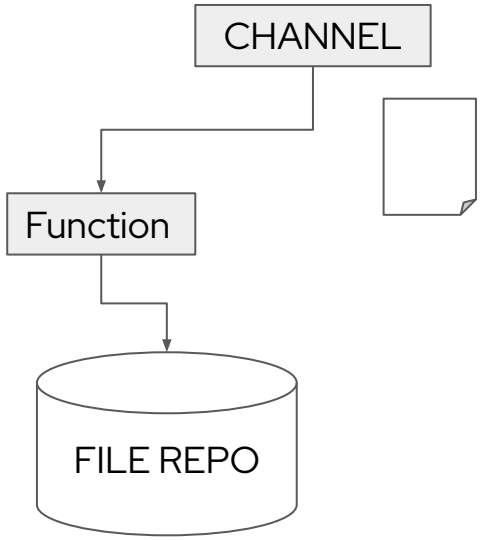
MANUFACTURE
BATCH CRON



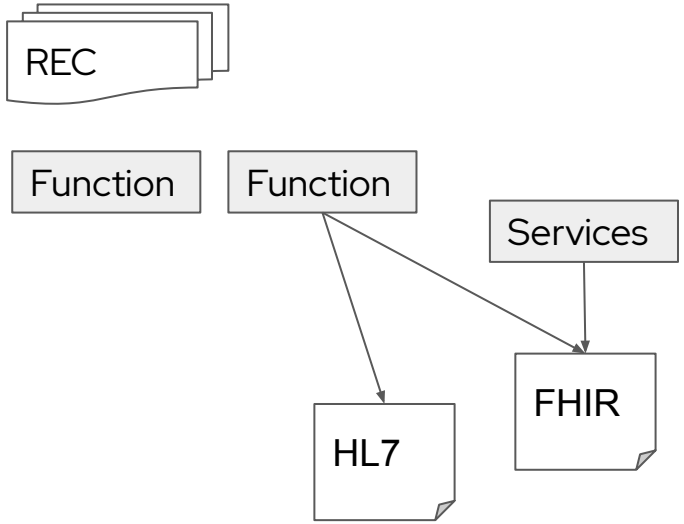
SERVERLESS INTEGRATION

USE CASE

LABORATORY
MANAGED FILE TRANSFER



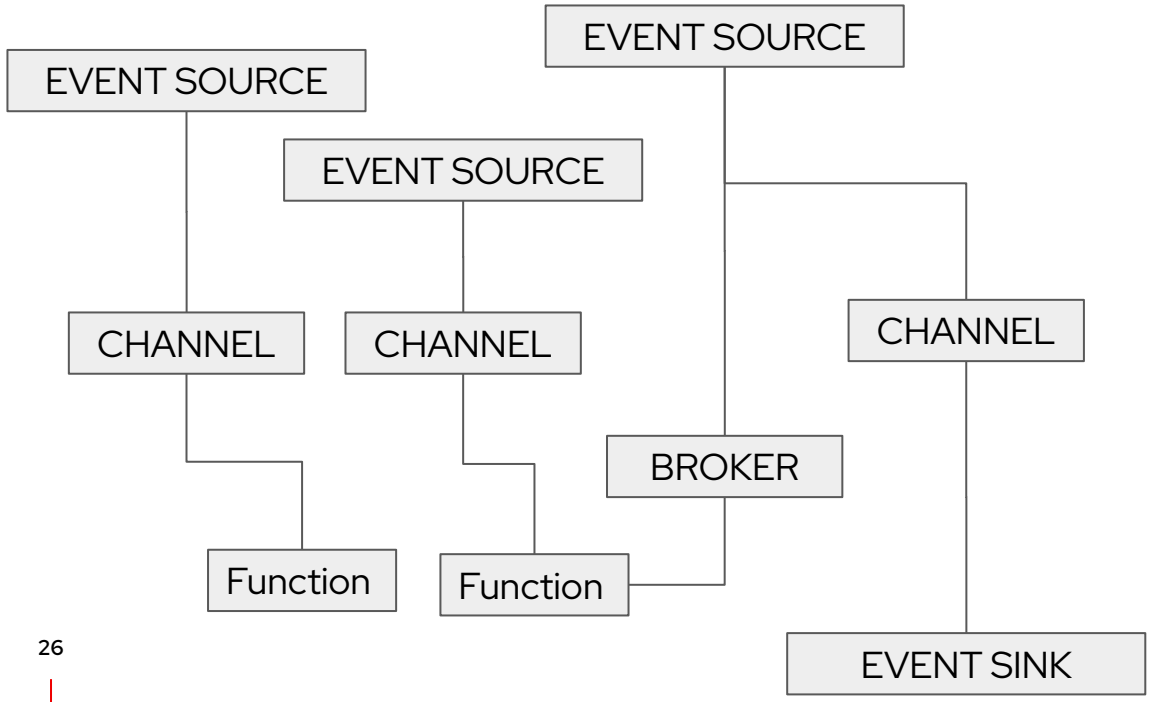
HEALTHCARE
INDUSTRY STANDARD



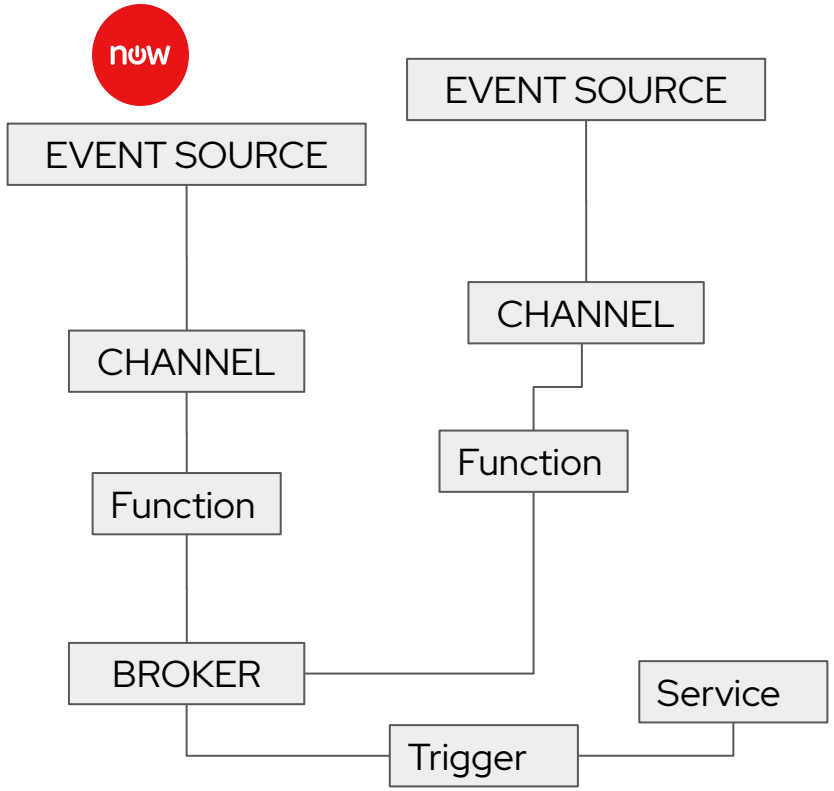
SERVERLESS INTEGRATION

USE CASE

RETAIL
VENDOR

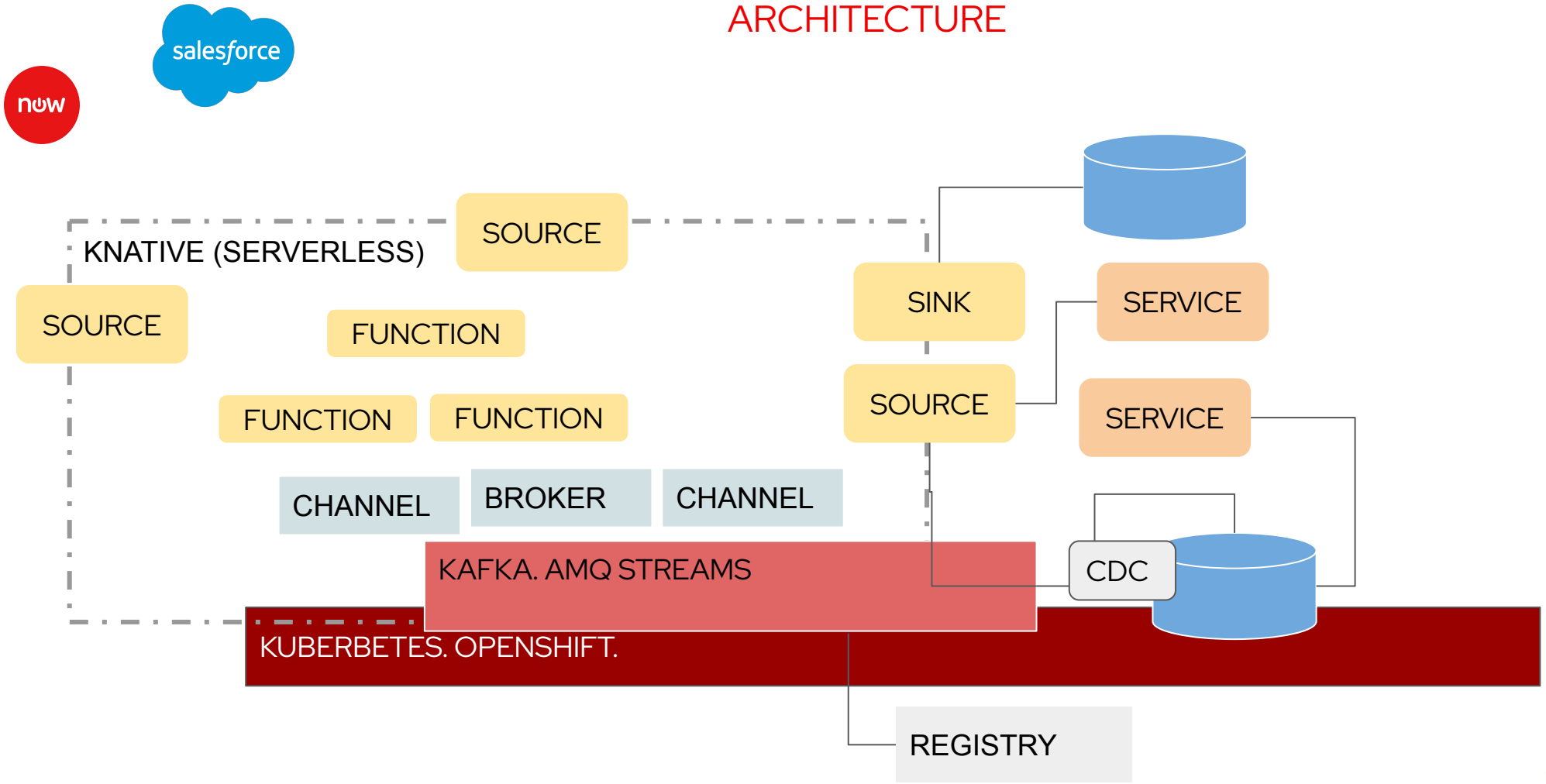


COMMERCE
SALESFORCE/SERVICE NOW

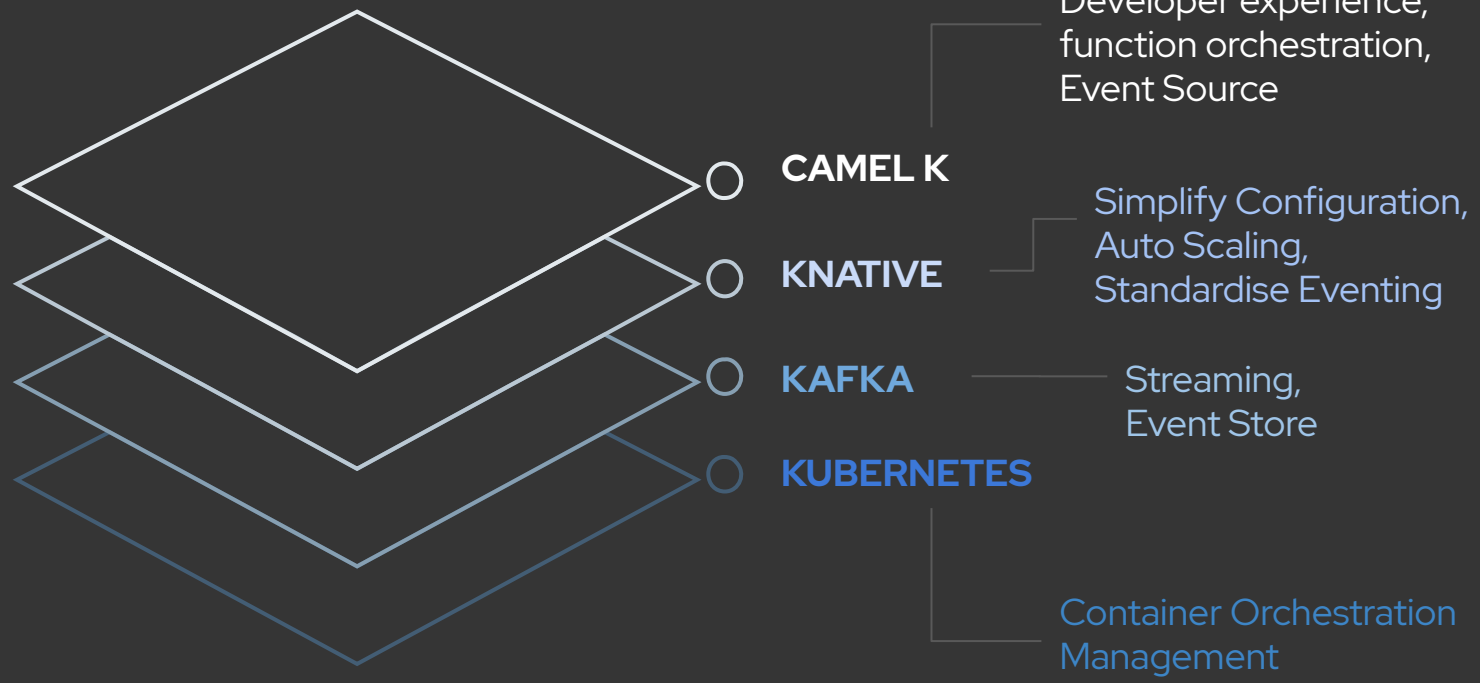


SERVERLESS INTEGRATION

ARCHITECTURE



SUMMARY



SERVERLESS INTEGRATION

Getting Started Resources

- CAMEL K and KNATIVE tutorial videos
  - GETTING STARTED https://www.youtube.com/watch?v=LaBvBonUC6g&list=PLcdUgeoWxosRqfavxpmGES5JjUjGqKLob
- Hands-on tutorial
  - BASICS https://learn.openshift.com/middleware/courses/middleware-camelk/camel-k-basic
  - SERVING https://learn.openshift.com/middleware/courses/middleware-camelk/camel-k-serving
- Today's Demo Repo
  - https://github.com/nicolaferraro/camel-k-example-knative
  - https://github.com/weimeilin79/camel-k-example-api
  - https://github.com/weimeilin79/coronavirus

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

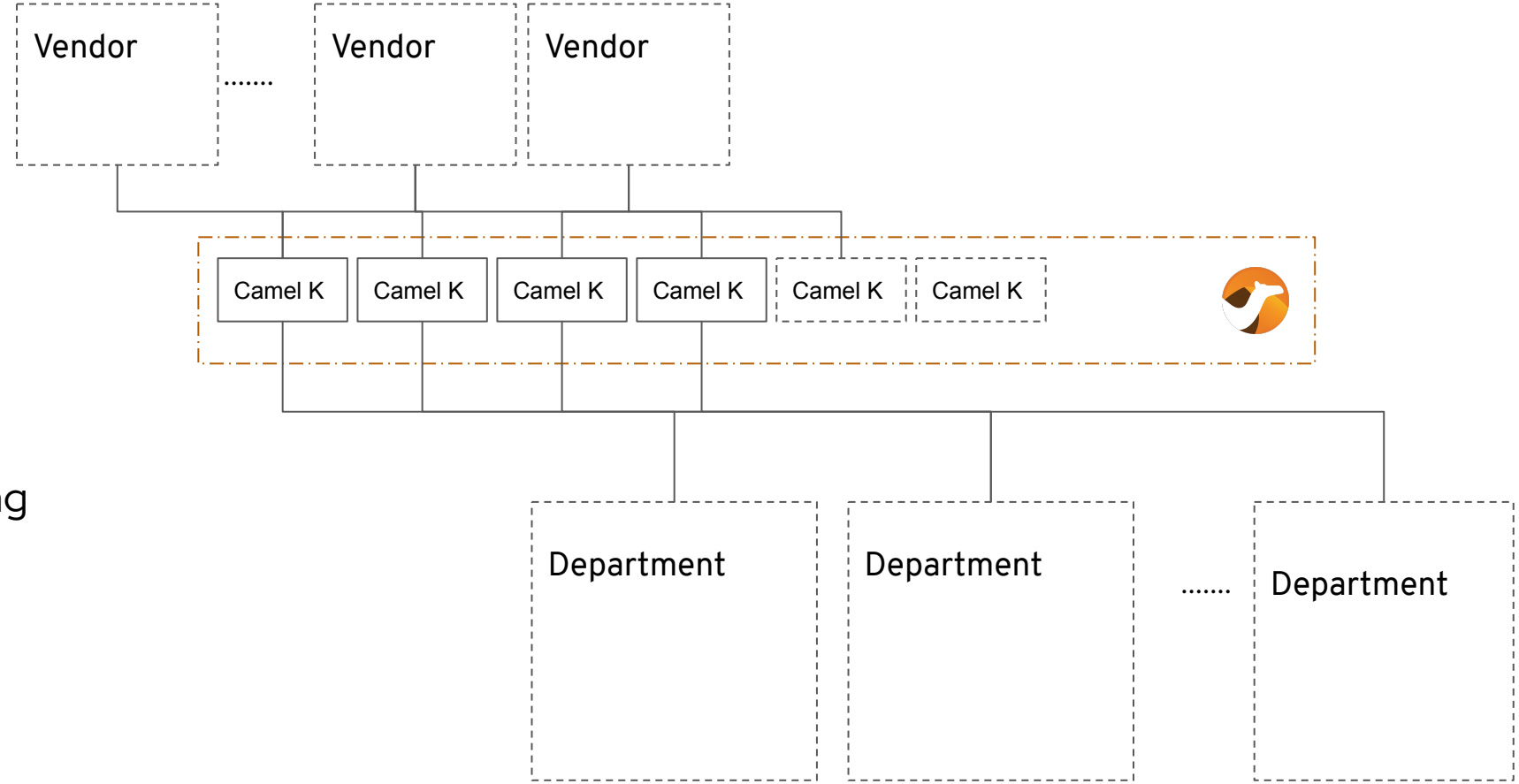
facebook.com/redhatinc

twitter.com/RedHat

Red Hat

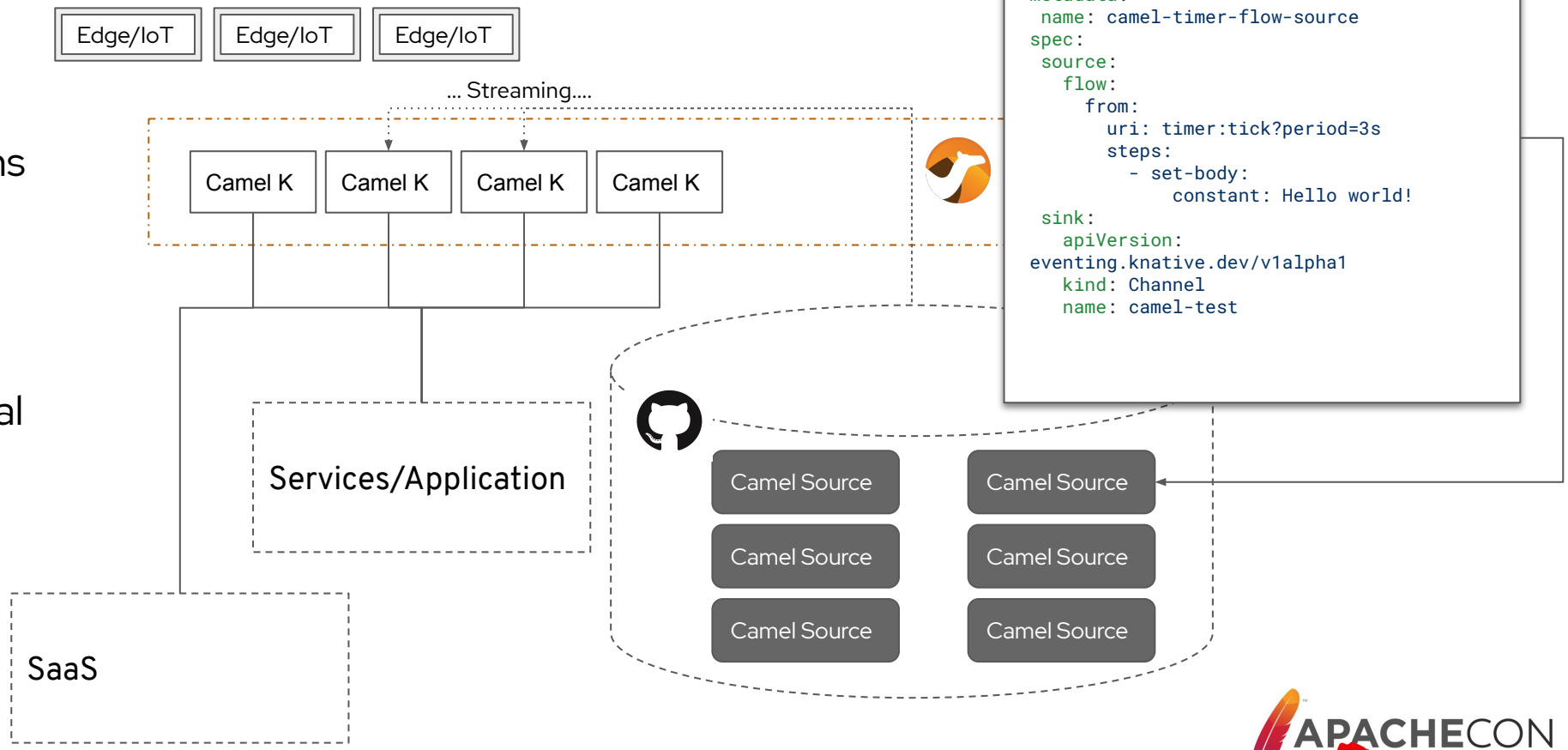
Use Case - Rapid growth/changing organizations

- Fast Deployment Cycle
- Microservice Native
- Smart Orchestrating
- Data Format Customizing
- Data Normalization
- Enable Event Driven
- Mild learning curve



Use Case - Near Real time, Agile processing

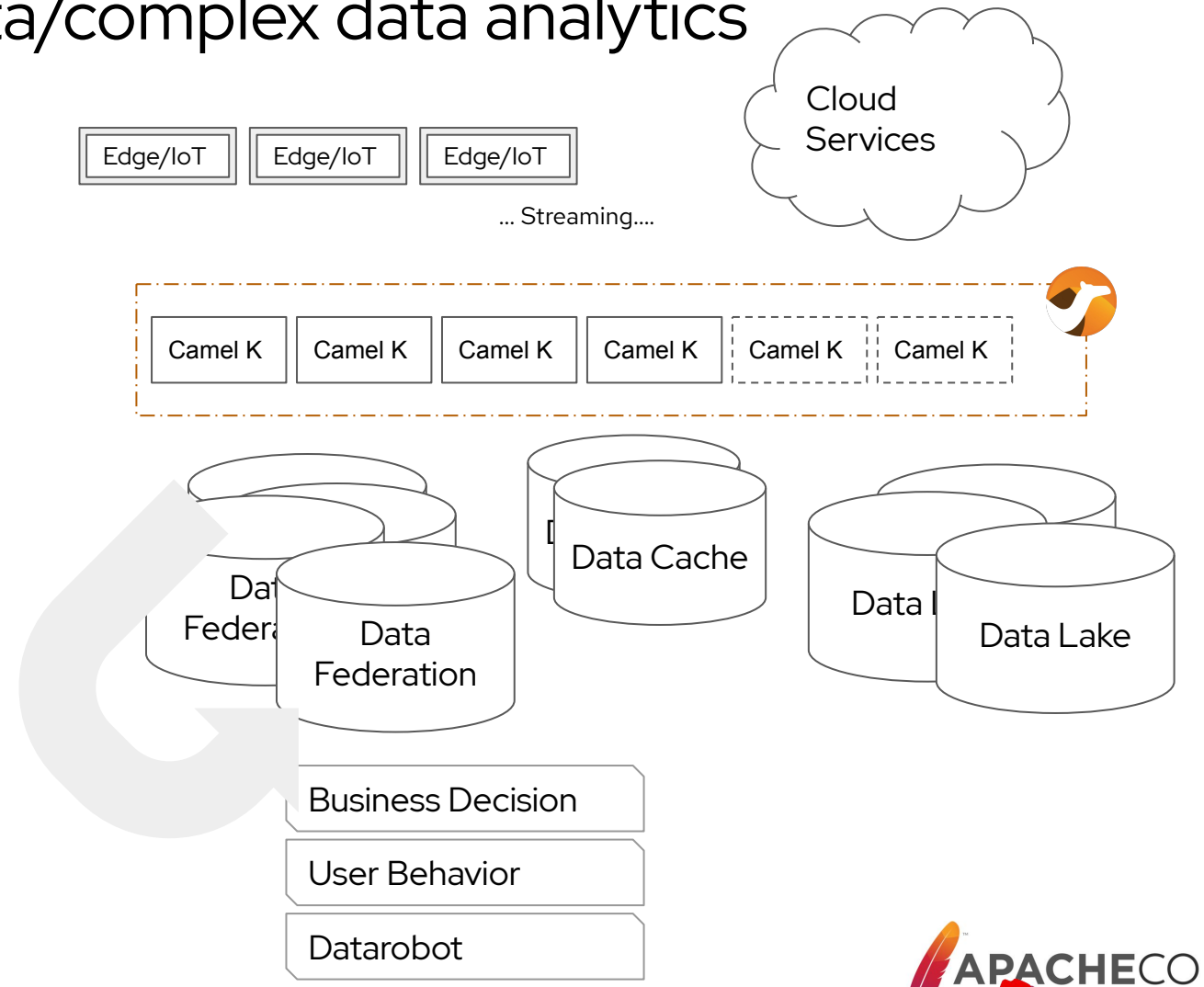
- Integration repository
- Flexible under unpredictable conditions
- Process and filter streaming data
- Gather real time external information




```yaml
apiVersion:
sources.eventing.knative.dev/v1alpha1
kind: CamelSource
metadata:
 name: camel-timer-flow-source
spec:
 source:
   flow:
     from:
       uri: timer:tick?period=3s
       steps:
         - set-body:
             constant: Hello world!
 sink:
   apiVersion:
eventing.knative.dev/v1alpha1
   kind: Channel
   name: camel-test
```

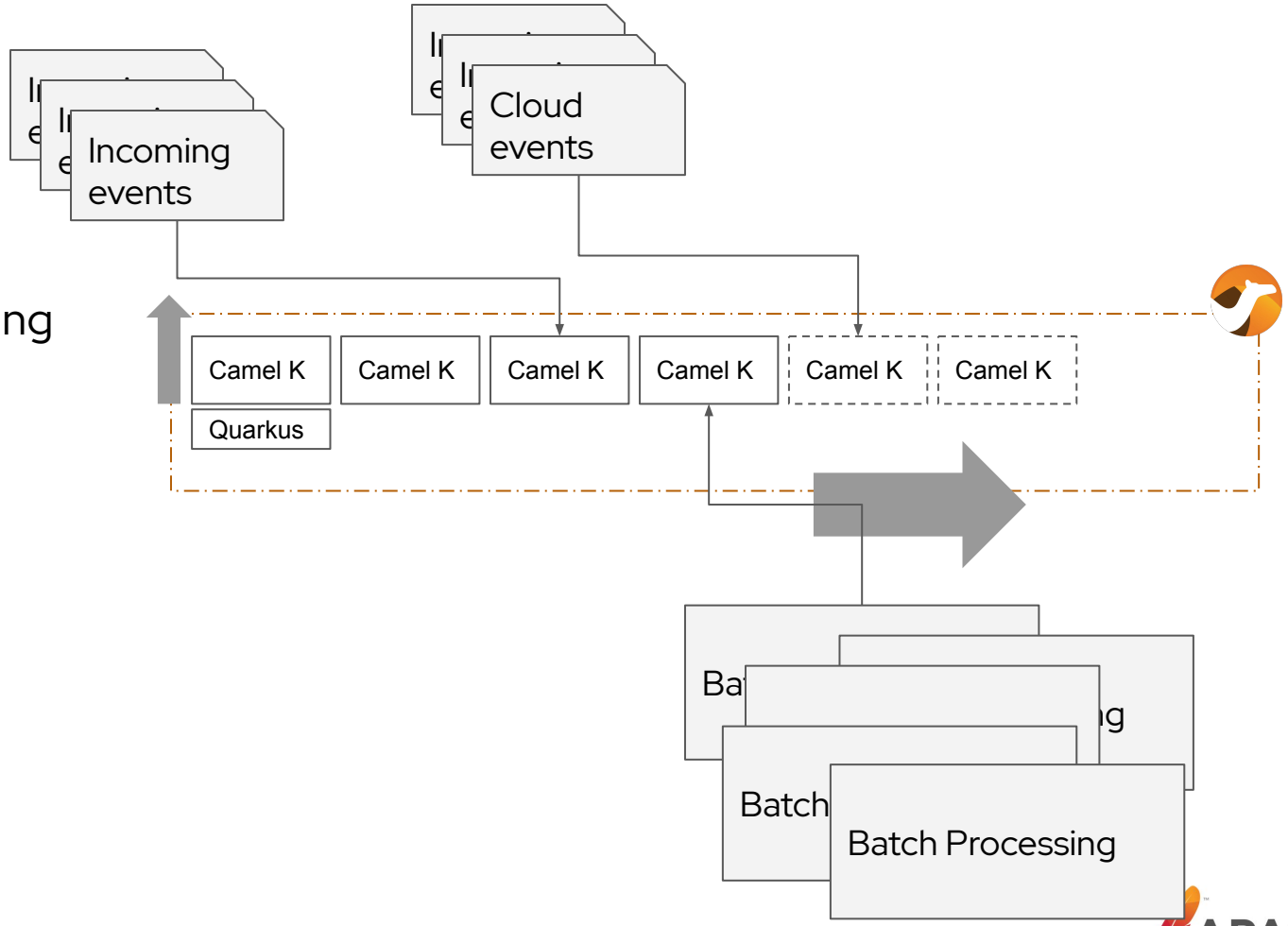
Use Case - Big data/complex data analytics

- Integrate complex multidimensional data sources
- Process real time data feeds
- Synchronizing, aggregating updating data state.
- Feeding quality info to AI/ML applications

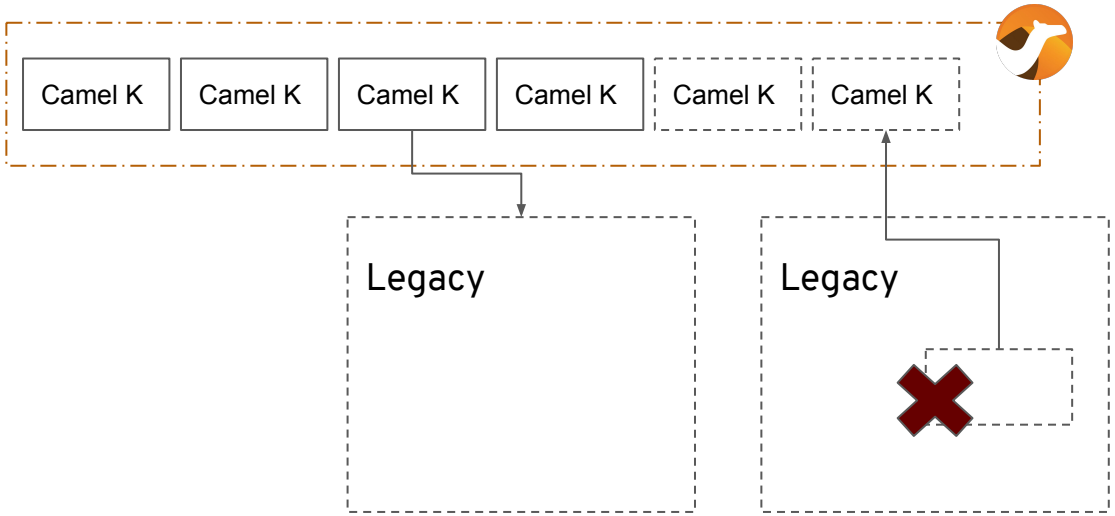


Use Case - Horizontal Scaling/Resource Optimization

- Fast boot up time
- Serverless capability supports scaling to zero
- Scaling out with ease on cloud
- Deployment is transparent to developers



Use Case - Legacy application modernization



- Anti-corruption layer to legacy
- Can use to apply strangler pattern
- Painless migration effort