Red Hat Summit

World of events

Event Driven Architecture and Serverless with Red Hat

Stefano Linguerri
Solution Architect

Red Hat

Event-Driven Architecture

What is Event-Driven Architecture?

Event-Driven Architecture *(EDA) is a way of designing applications and services to respond to real-time information based on the sending and receiving of event notifications*

Why Event-Driven Architecture or EDA?

Mirrors the real world

The real world is event-driven. Systems generate and respond to events in everyday life, e.g., the human central nervous system.

Reduced coupling

Traditional RPC-style service architecture results in tightly-bound services. Changes to the application flow typically require service code changes. EDA allows new functionality to be added by adding services that consume existing event streams.

Encapsulation

Microservices concepts have grown in popularity due to the ability for service teams to develop services in isolation. EDA means that service designers need not be aware of how events are consumed.

Fine-grained scaling

Services can be independently scaled up and down to meet the event volume.

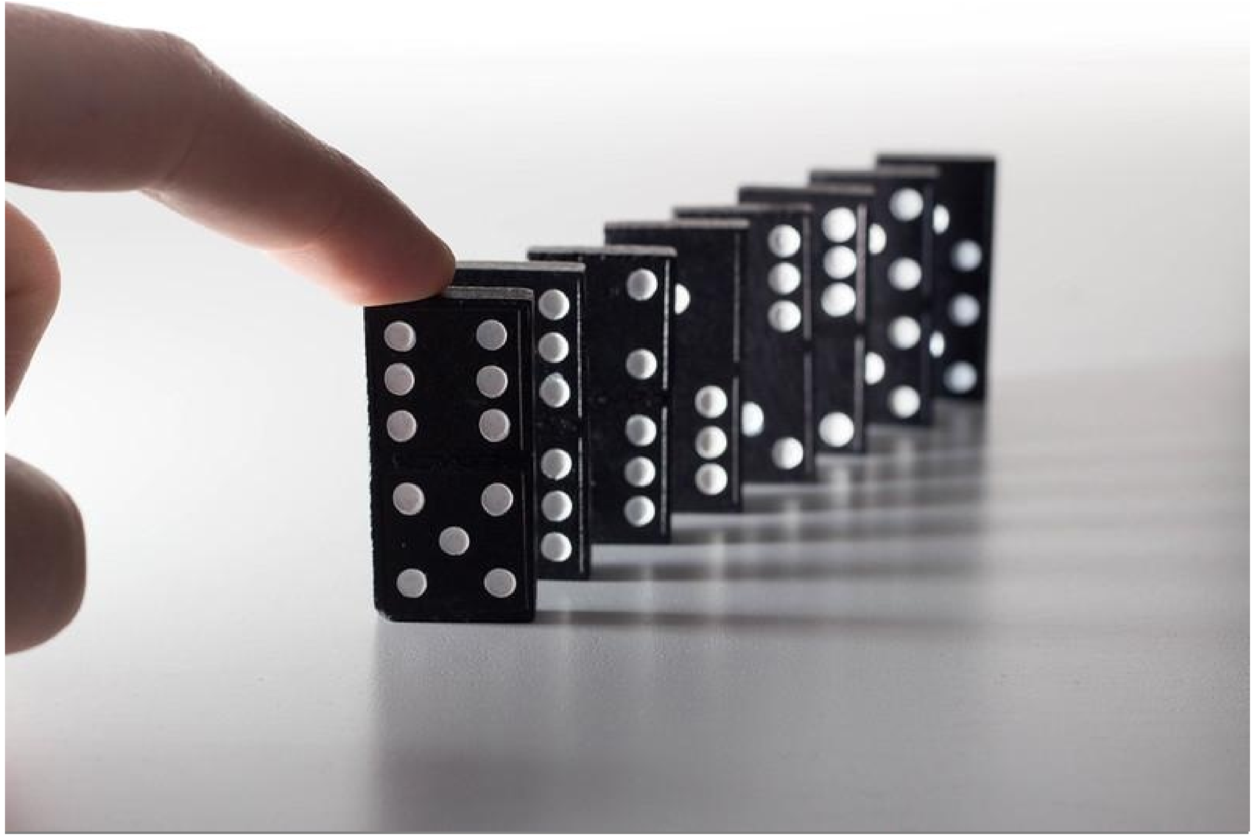
Near real-time latency

Customers increasingly expect a near real-time experience. Polling on APIs is a delicate trade-off between responsiveness and load. EDA allow apps to react in near real-time without compromise.

What is an event?

Event

An action or occurrence recognized by software, often originating asynchronously from the external environment, that may be handled by the software



What is an event?

Event

Immutable state and value of a particular entity, which occurred during operation among services.

Command

Async form of Remote Procedure Call, contains instructions telling recipient what to do, may cause a change of state.

Query

Similar to commands, queries expect a response returning the results, but do not cause any change in state.

Types of event consumption patterns

Volatile

The event needs to be disseminated to all consumers online at time of publication. Not persisted.

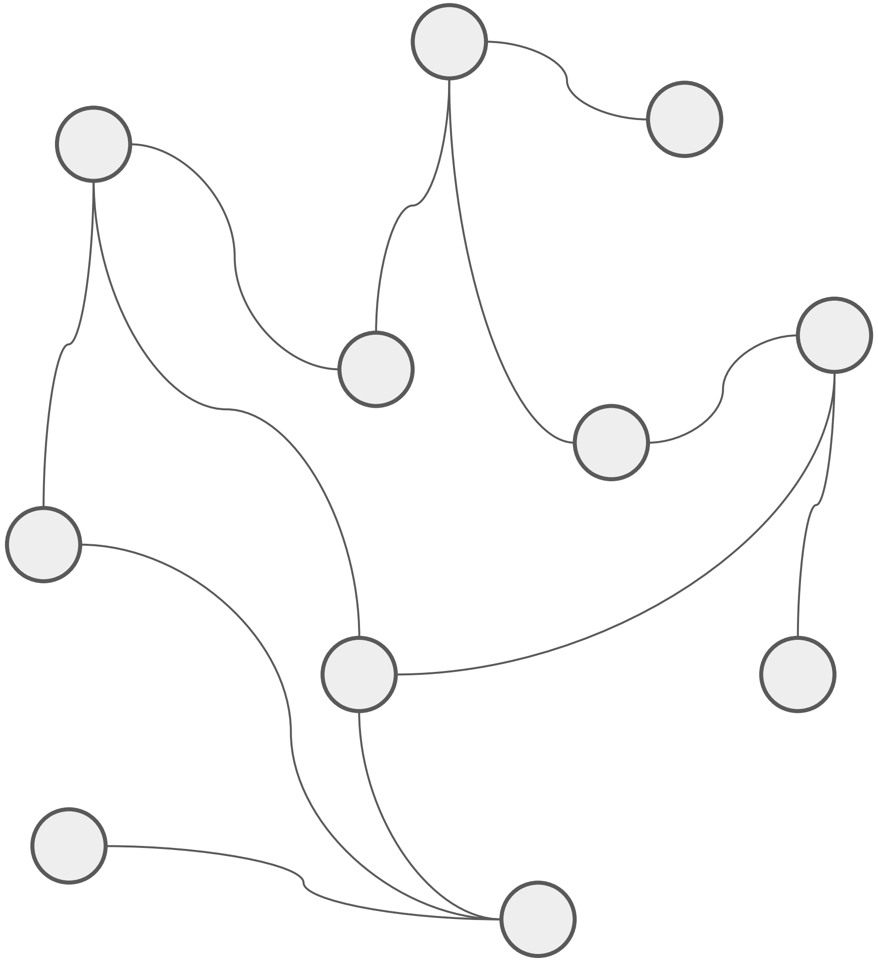
Durable

Events stored durably until read by all registered consumers. Traditional store-and-forward brokers.

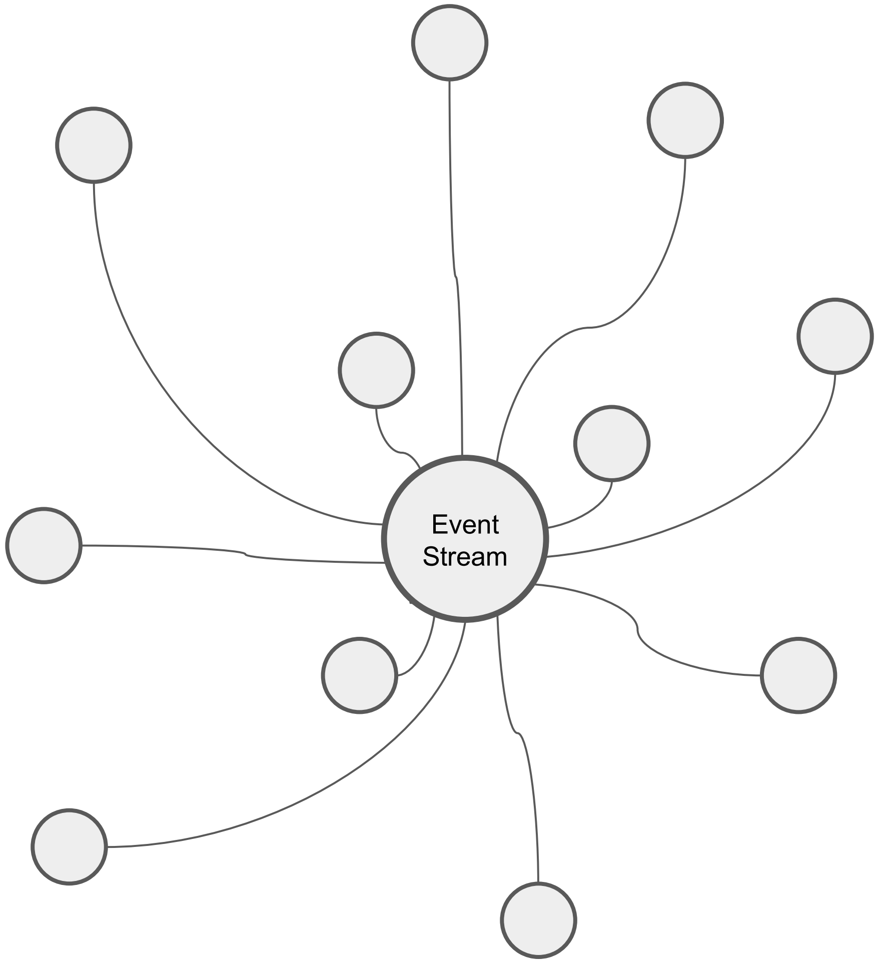
Replayable

Events stored durably for specific period of time or storage capacity. Consumers can move back and forth of the stream.

Orchestration vs Choreography

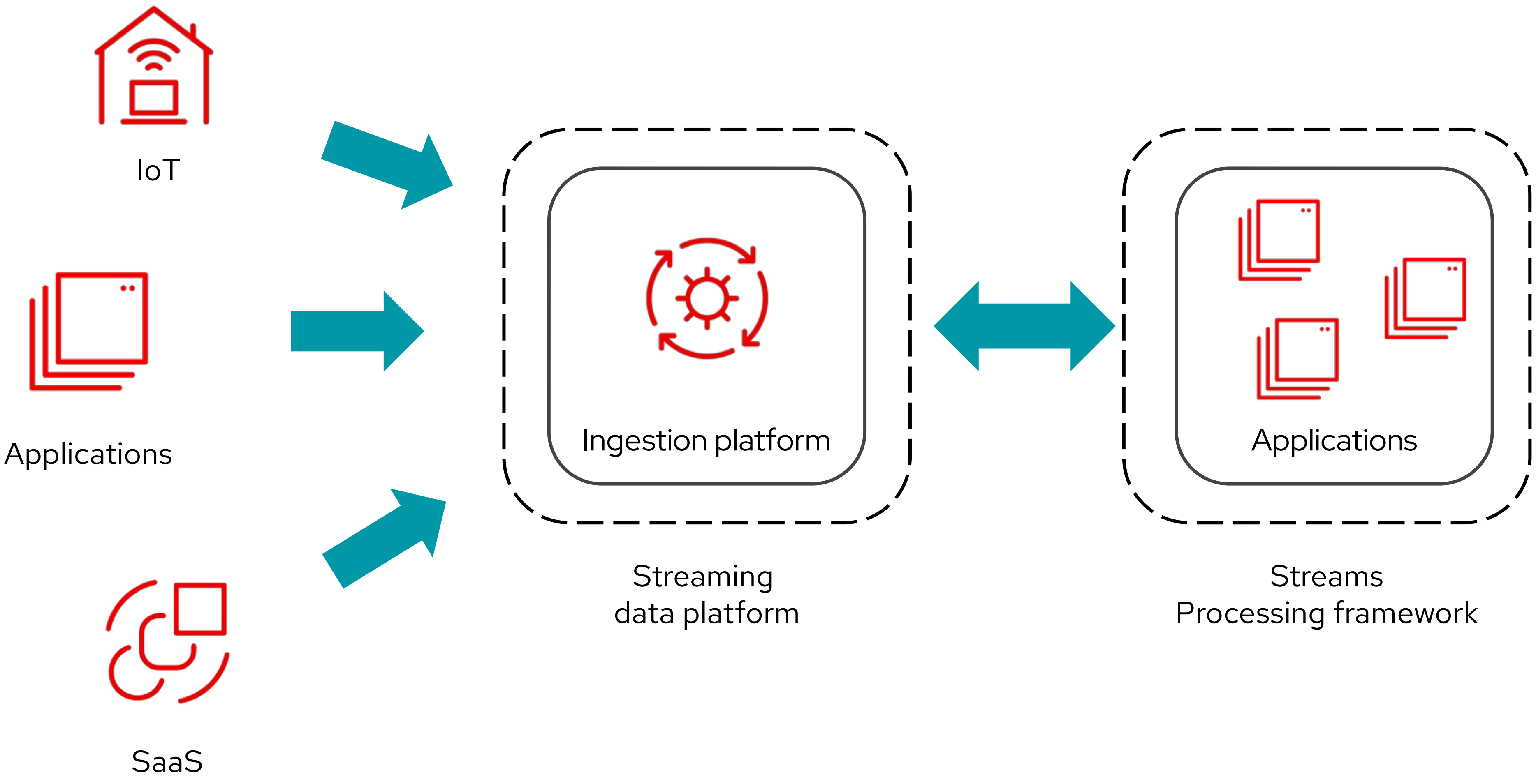


Orchestration vs Choreography

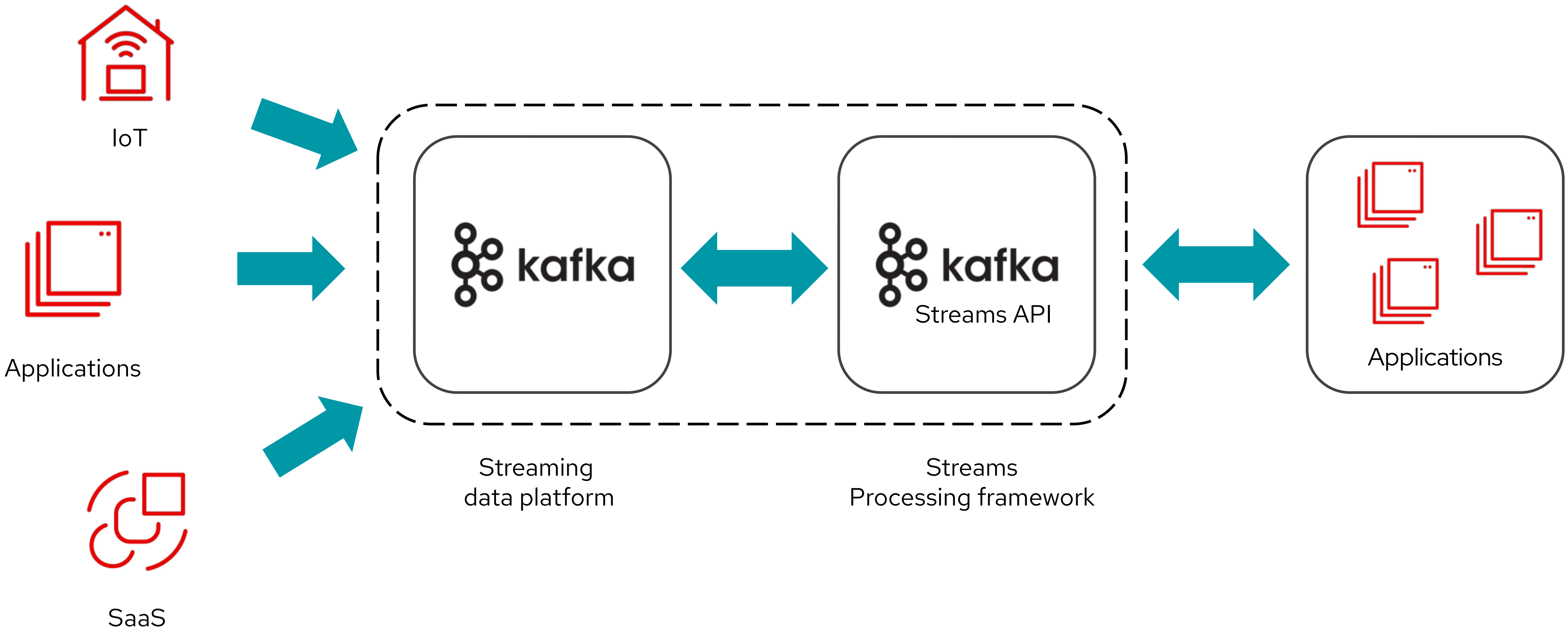


Events management: AMQ Streams

Stream Processing



Let’s use just one



IoT

Applications

SaaS

kafka

kafka

Streams API

Applications

Streaming
data platform

Streams
Processing framework

Red Hat AMQ Streams

Part of the Red Hat AMQ Suite

AMQ Streams on OCP

- Running Apache Kafka on OpenShift Container Platform
- Based on the upstream Strimzi project

AMQ Streams on RHEL

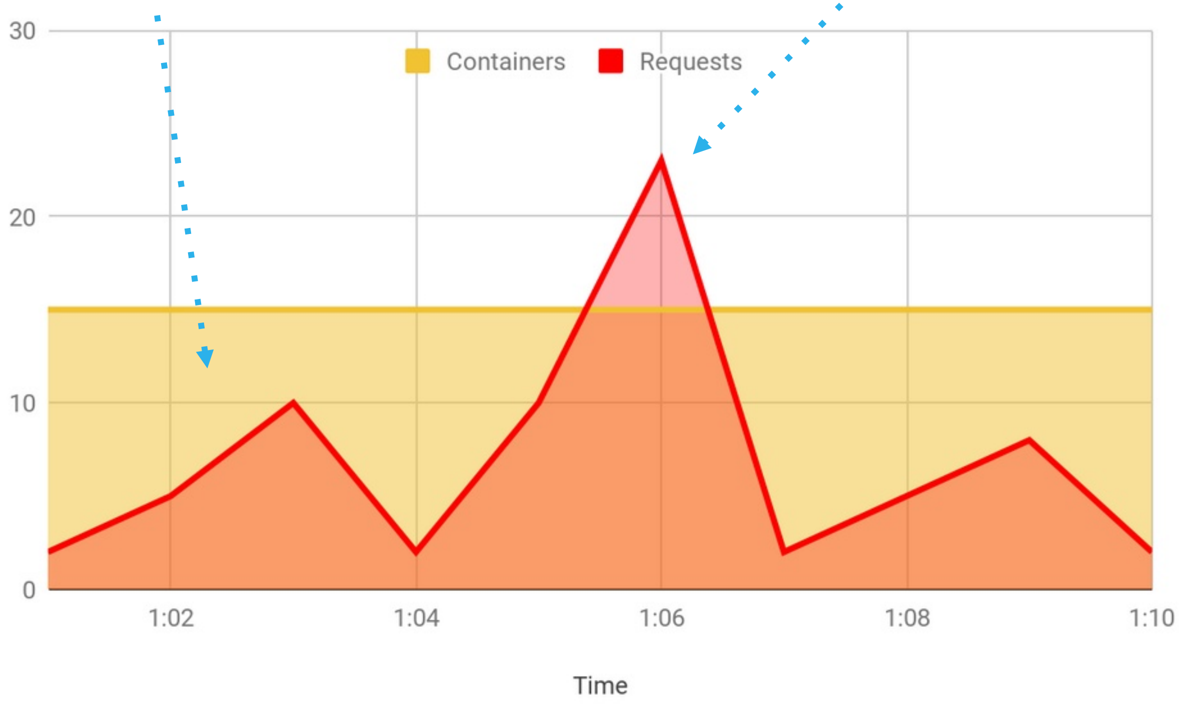
- Running Apache Kafka on “bare metal”

Serverless

Serverless Operational Benefits

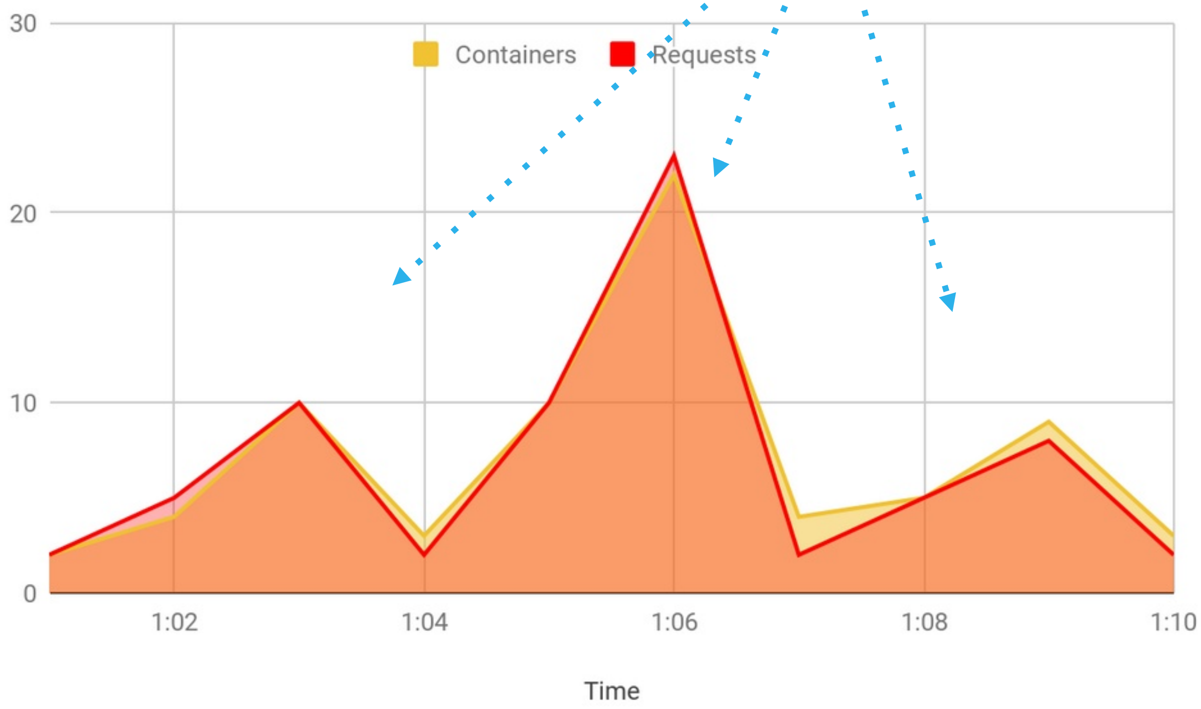
Over provisioning
Time in capacity planning
IT cost of idle resources

Under provisioning
Lost business revenue
Poor quality of service



Without Serverless

More applications
Direct line between IT
costs & business revenue



With Serverless

Red Hat Serverless

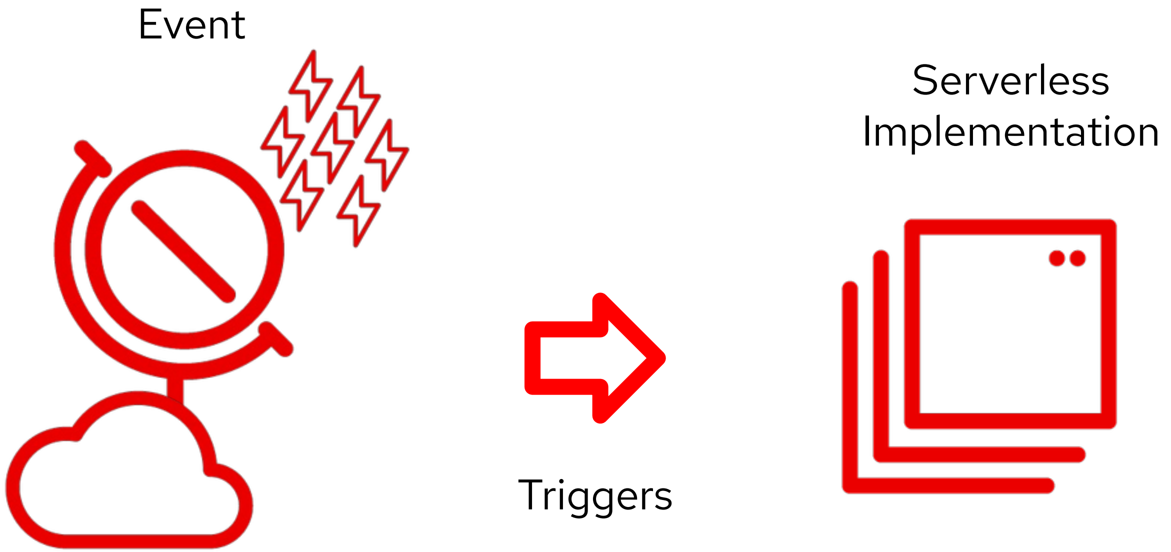
Knative Serving

Auto-scaling and scale-to-zero

Knative Eventing

Messaging for event-based applications

Serverless Behaviour



Event

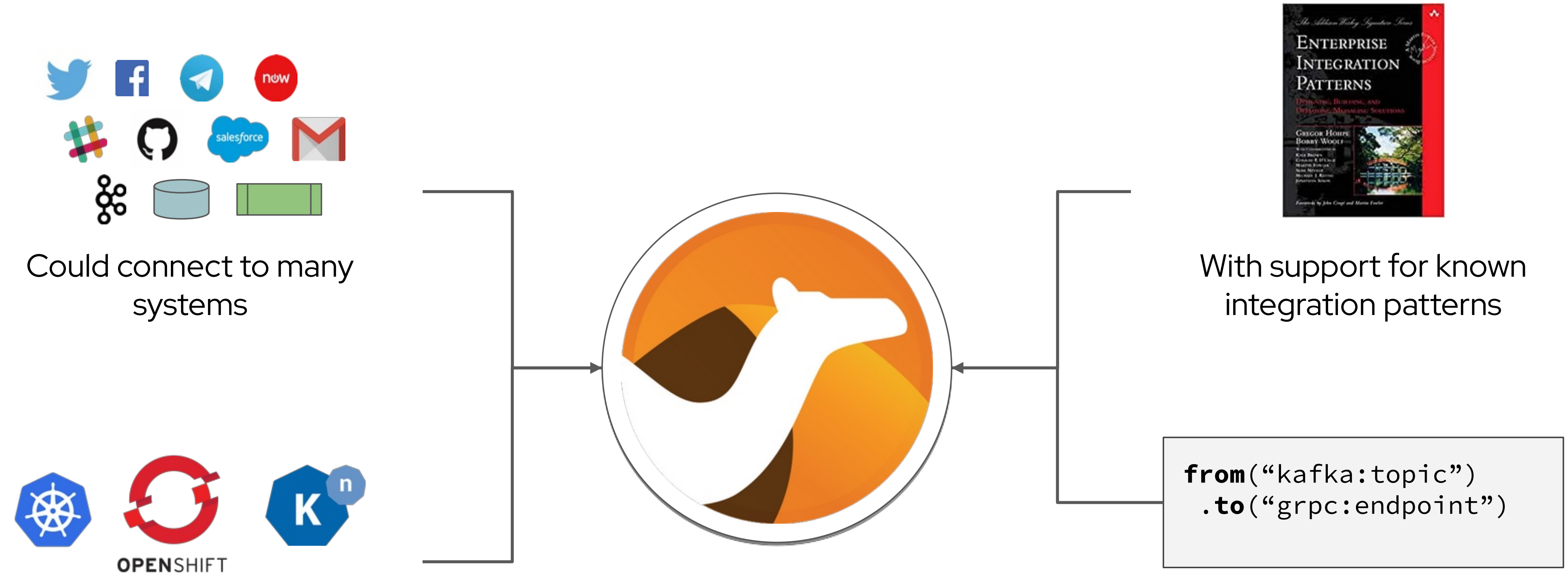
Triggers

Serverless Implementation

- Fast boot up time
- Stateless
- Natively consuming cloud events

Serverless Integration: Camel K

Apache Camel



Could connect to many systems

Works on and off the cloud

With support for known integration patterns

```
from(“kafka:topic”)
 .to(“grpc:endpoint”)
```

Write integrations with a simple language such as XML, Java and YAML

What is Camel K?

- **A platform directly running integrations on Openshift and Kubernetes for Serverless loads**
- Architectured by Kubernetes CRDs and Operators
- A community-driven project
- Part of Apache Camel

Fast development with Camel K

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

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

1

Create
Integration file

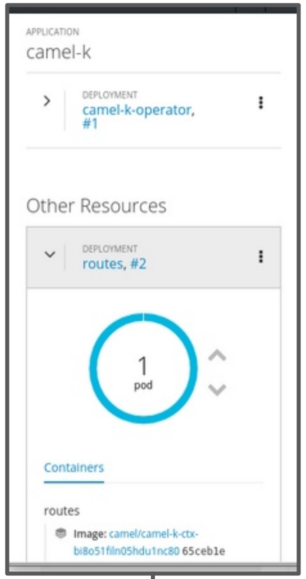
2

EXECUT
E CLI Tools

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

3

RUNNING
Serverless
on OpenShift/Kubernetes



APPLICATION
camel-k

DEPLOYMENT
camel-k-operator, #1

Other Resources

DEPLOYMENT
routes, #2

1
pod

Containers

routes

Image: camel/camel-k-ctx-bi8o51filn05hdu1nc80 65ceble

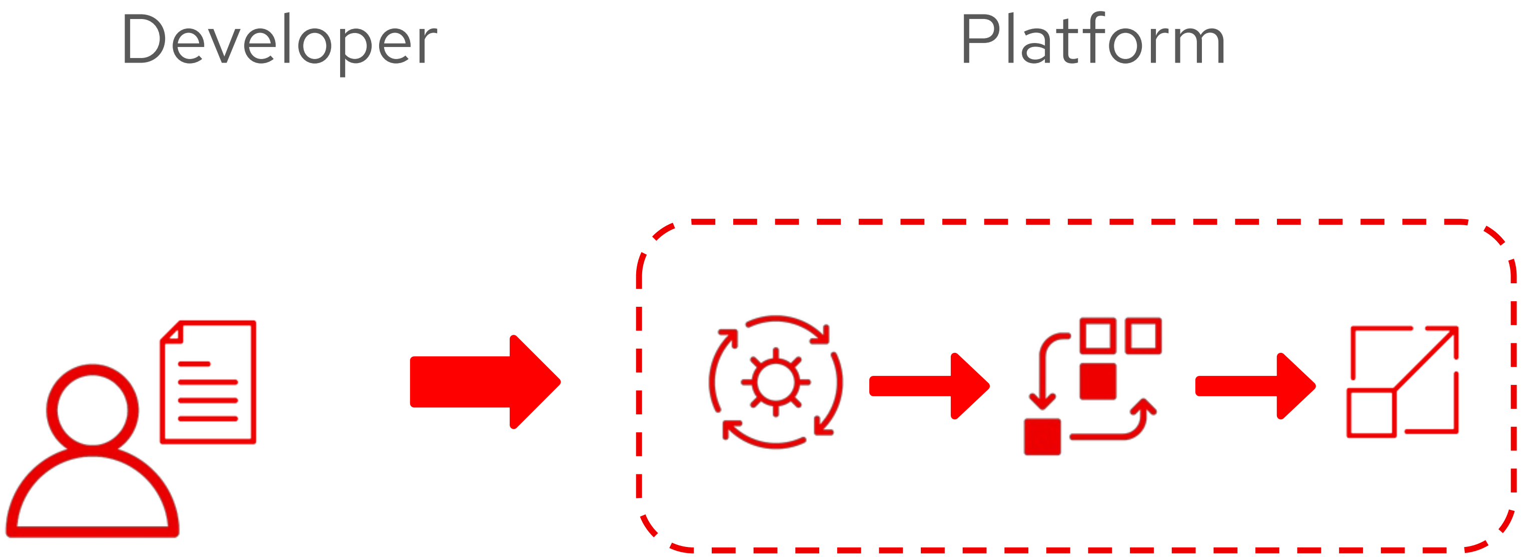
Red Hat

Developer and Serverless with Camel K

"Seamless Developer Focus Experience"

1. Write simple stateless snippet codes.

2. Pushes code(Text).



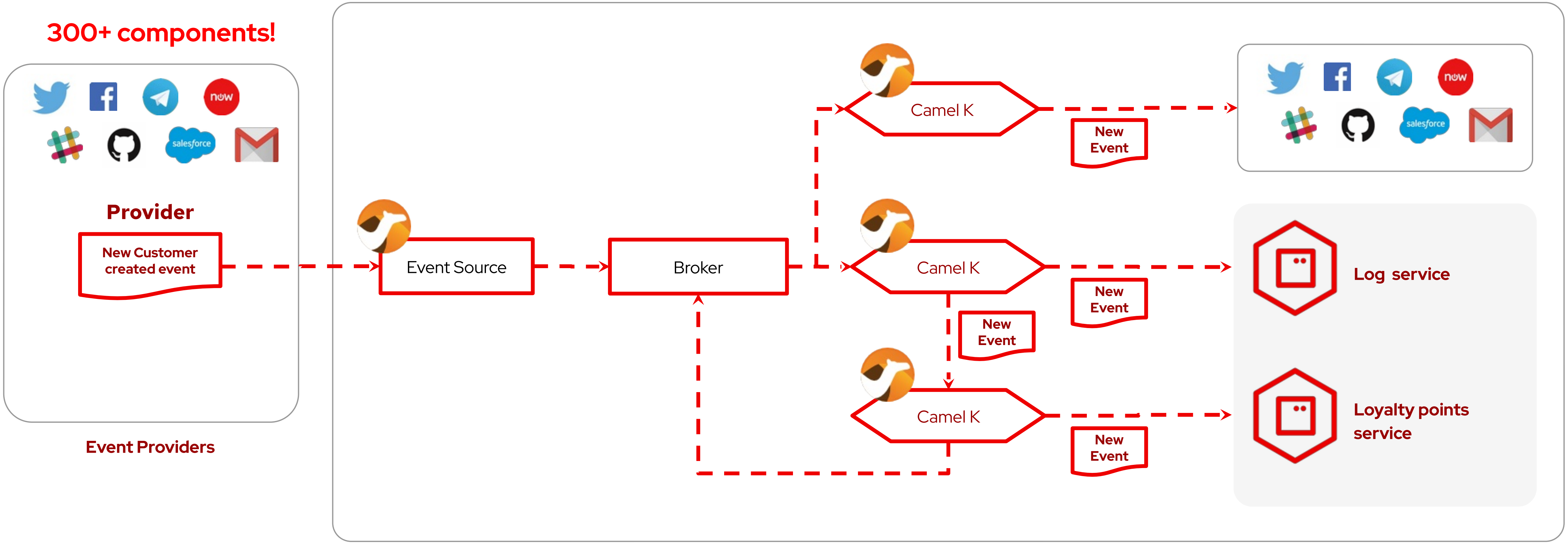
3. Compile, build and package

4. Deploy, create resources/configuratio n

5. Apply scaling policies, connects event mesh

Eventing With Camel K

300+ components!



Provider

New Customer created event

Event Providers

Event Source

Broker

Camel K

New Event

Camel K

New Event

New Event

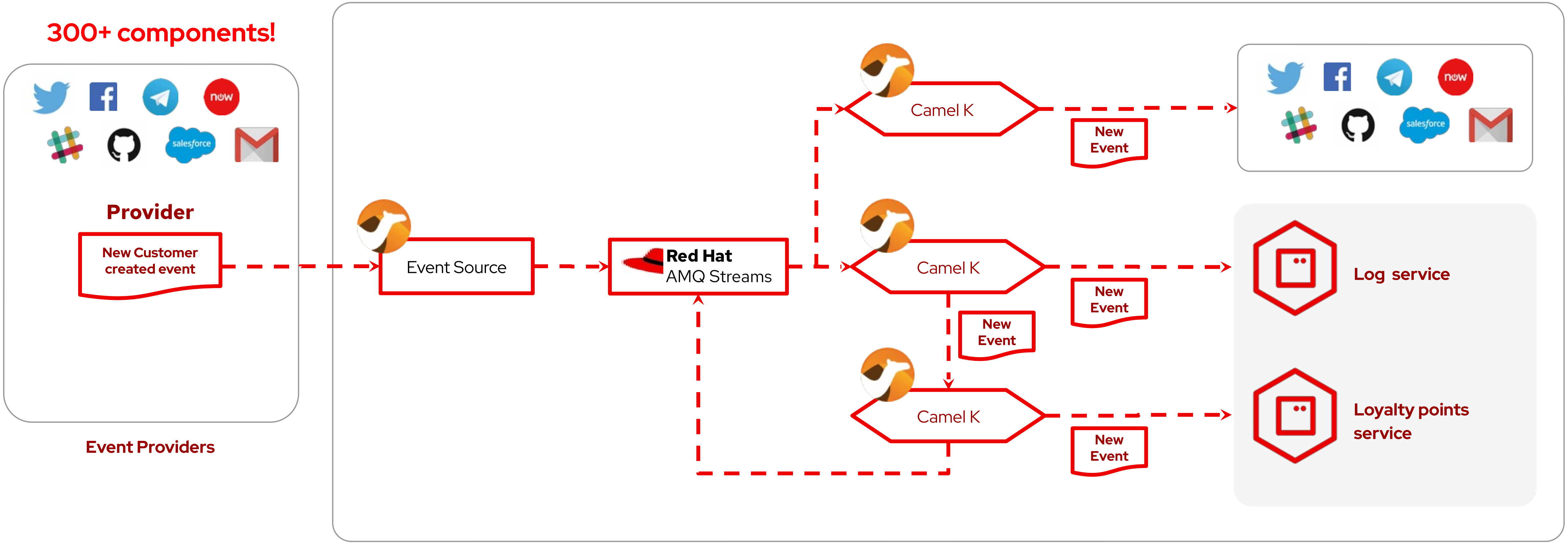
Camel K

New Event

Log service

Loyalty points service

Eventing With Camel K



Red Hat Summit

Thank you

linkedin.com/company/Red-Hat

facebook.com/RedHatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat