RED HAT SUMMIT

APACHE CAMEL K

Bringing serverless workloads to the enterprise

Luca Burgazzoli
Principal Software Engineer

Nicola Ferraro
Senior Software Engineer

07/05/2019

Luca Burgazzoli
Principal Software Engineer at Red Hat
Twitter: **@lburgazzoli**

Nicola Ferraro
Senior Software Engineer at Red Hat
Twitter: **@ni_ferraro**

We work on:

- Apache Camel and Camel K
- Syndesis
- Red Hat Fuse

Agenda

1. Brief Introduction about Apache Camel
2. Camel K
    a. Demo
3. Camel K and Knative
    a. Demo
4. What’s next

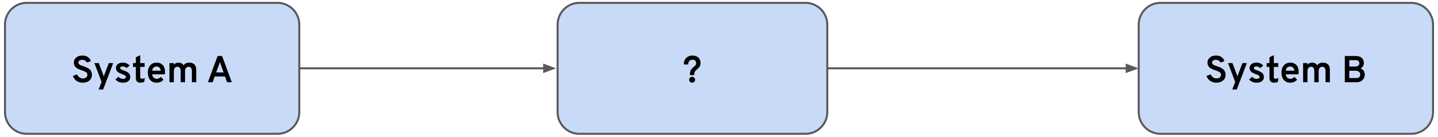
APACHE CAMEL

What is Apache Camel?

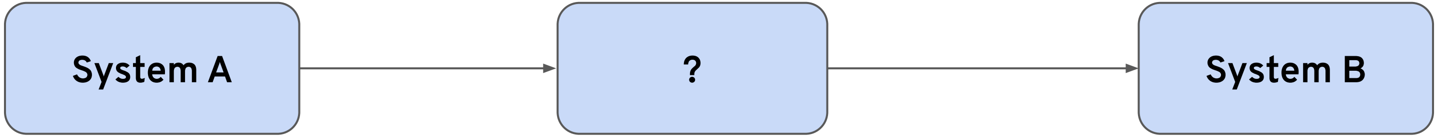
- The swiss knife of integration
- **>10 years of development - still one of the most active Apache projects**
- Based on Enterprise Integration Patterns (EIP)
- Uses a powerful Domain Specific Language (DSL)
- Can integrate **anything**
- Supports 300+ components
- The cornerstone of Red Hat Fuse

http://camel.apache.org/

System Integration

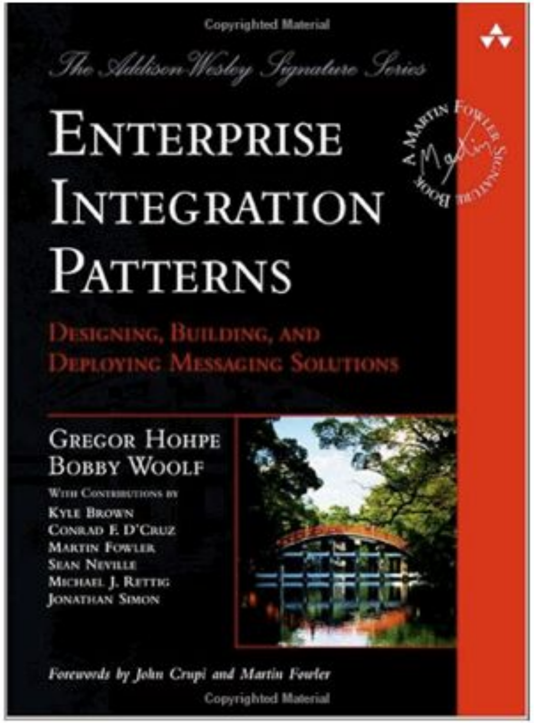


System Integration



- Different transports
- Different Data Model
- Asynchrony
- Failures

Enterprise Integration Patterns



Copyrighted Material

The Addison-Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS

A MARTIN FOWLER SIGNATURE BOOK

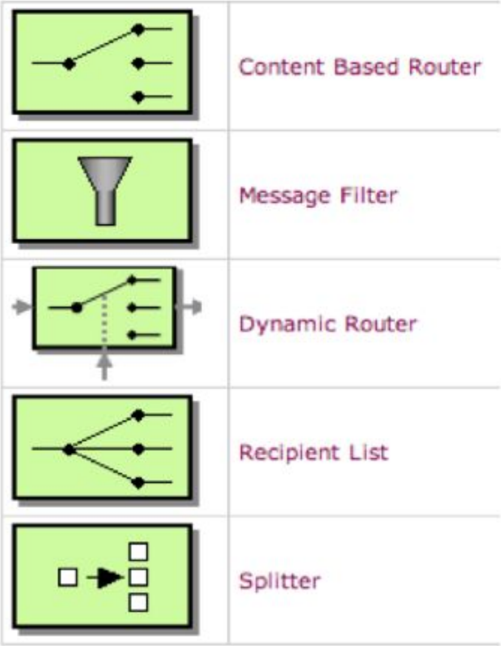
GREGOR HOHPE
BOBBY WOOLF

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

Enterprise Integration Patterns



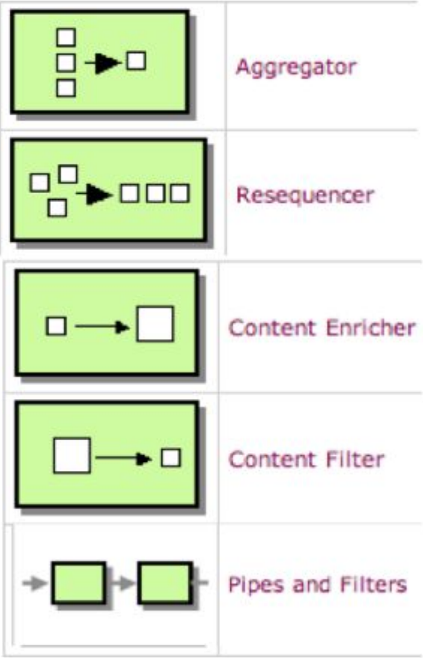
Content Based Router

Message Filter

Dynamic Router

Recipient List

Splitter



Aggregator

Resequencer

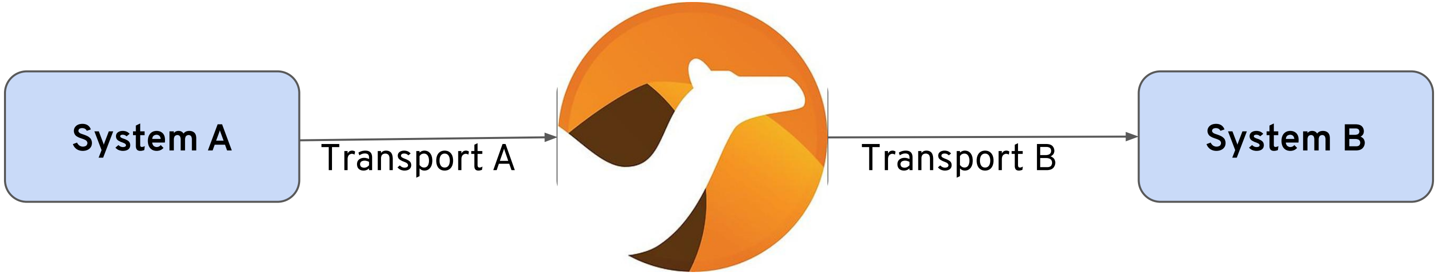
Content Enricher

Content Filter

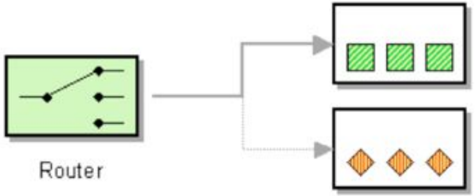
Pipes and Filters

Integration Framework

System Integration



Camel Routes



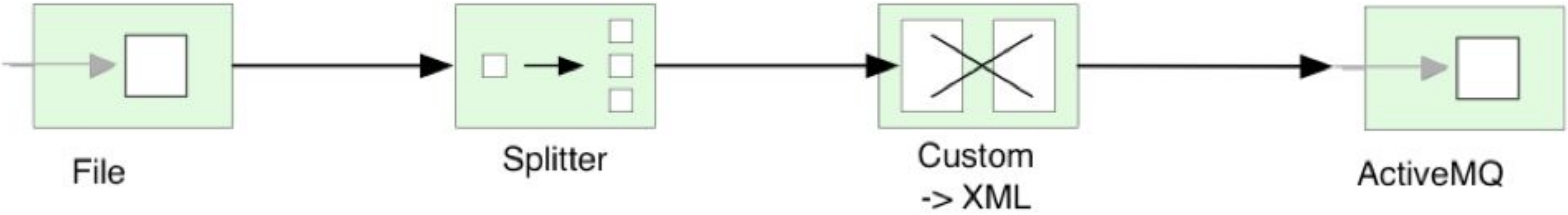
```
from("file:data/inbox")
  .to("jms:queue:order");
```

Java DSL

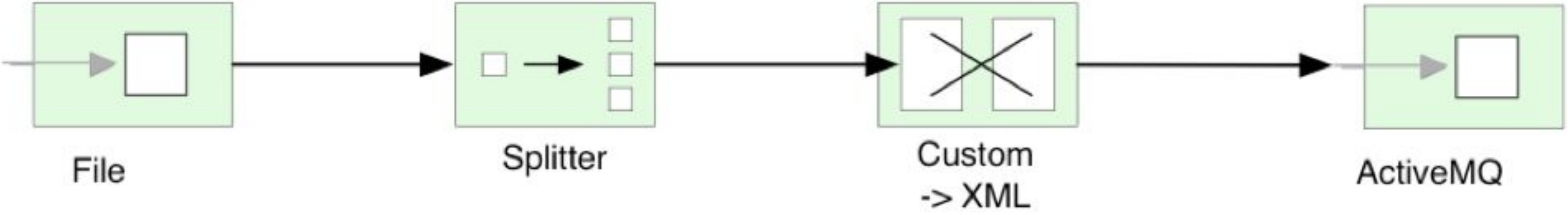
```
<route>
  <from uri="file:data/inbox"/>
  <to uri="jms:queue:order"/>
</route>
```

XML DSL

Camel Routes with Splitter

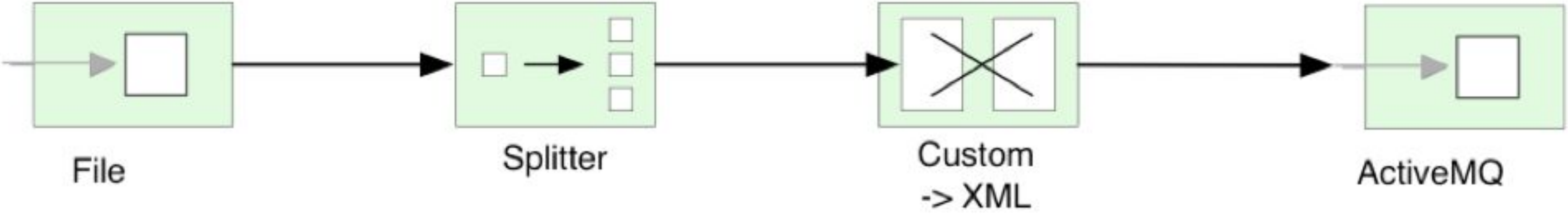


Camel Routes with Splitter



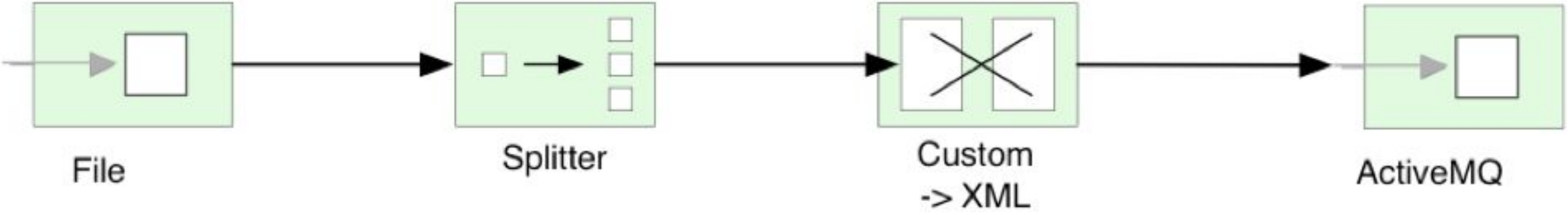
```
from("file:inbox")
```

Camel Routes with Splitter



```
from("file:inbox")
    .split(body().tokenize("\n"))
```

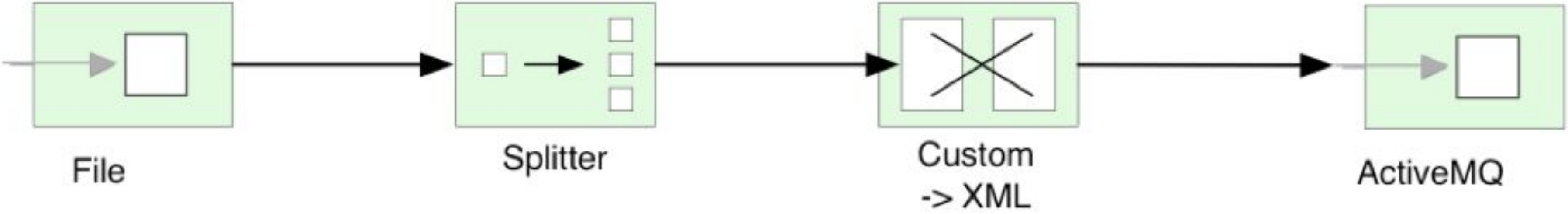
Camel Routes with Splitter



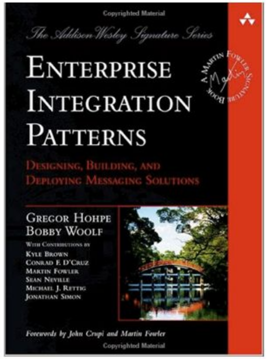
```
from("file:inbox")
     .split(body().tokenize("\n"))
          .marshal(customToXml)
```

Custom data transformation

Camel Routes with Splitter



```
from("file:inbox")
     .split(body().tokenize("\n"))
          .marshal(customToXml)
          .to("activemq:line");
```



Copyrighted Material

The Addison-Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS

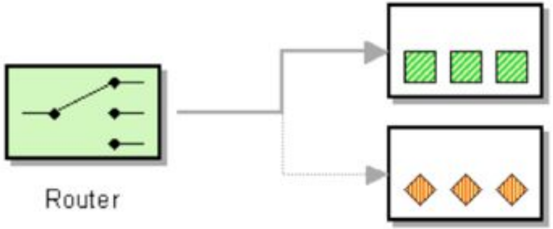
GREGOR HOHPE
BOBBY WOOLF

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

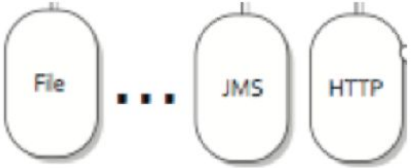
Copyrighted Material

+



+

+



=

Runs on popular Java Runtimes

QUARKUS
WildFly
Karaf
MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

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

How ?

Camel DSL, based on EIPs...

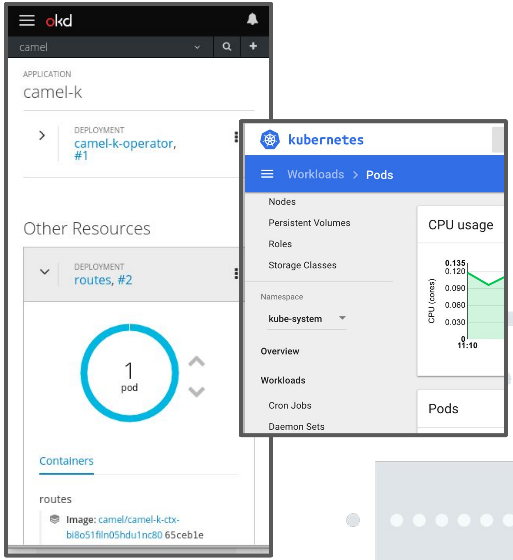
1. Create a integration file (Java, Groovy, Kotlin, JS, XML...)

```
// 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’);
```

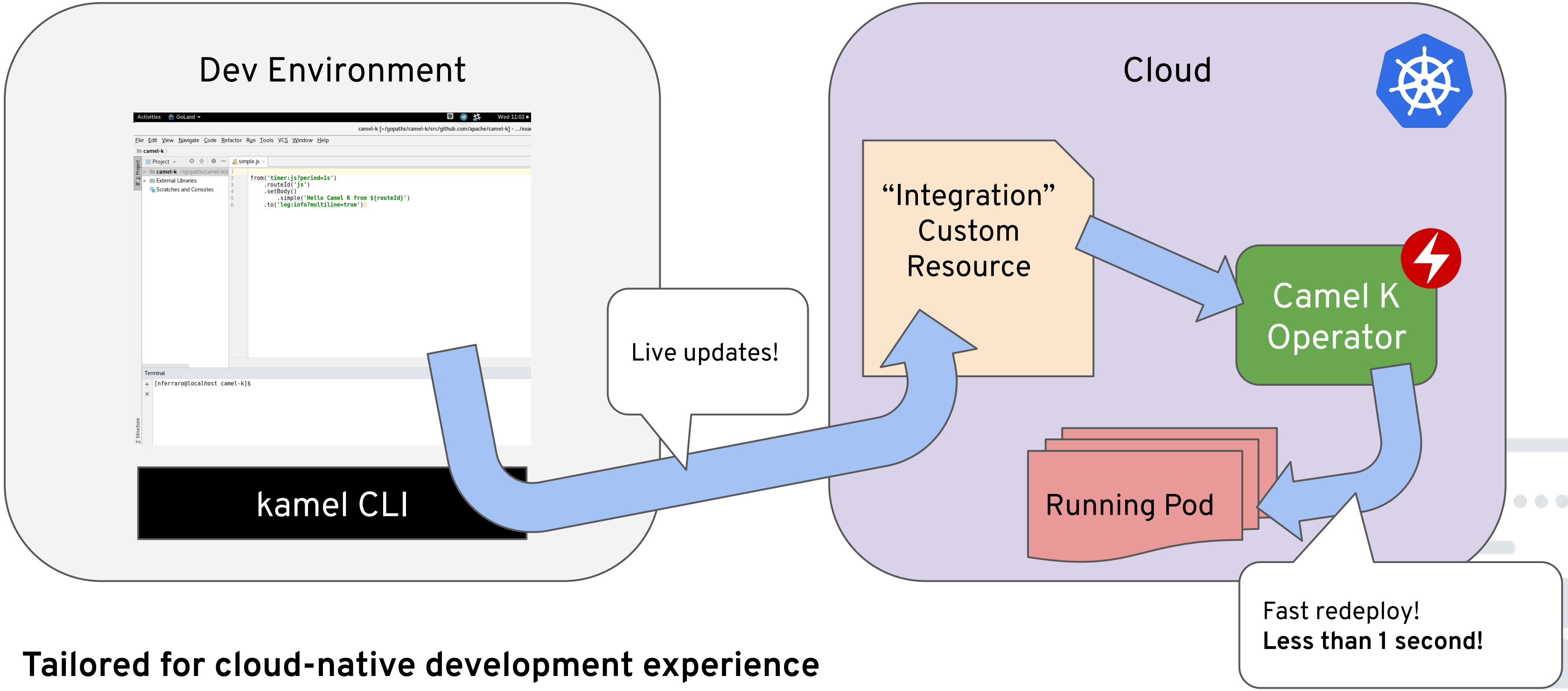
2. Run it

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

3. It runs on Kubernetes



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

CAMEL K PERFORMANCE

Performance

The operator understands the code so it can:

- can **choose the lightest runtime** for the integration (e.g. **Quarkus**)
- can **package a minimal set of components** that are **required by the code**
- can **fine tune Camel and JVM parameters** based on user the code
- can **optimize creation of container images**

Compared to a traditional spring-boot stack

- it requires less memory and cpu
- does not require to generate a uber jar thus improve deployment speed

SYNDESIS

3scale
by Red Hat®

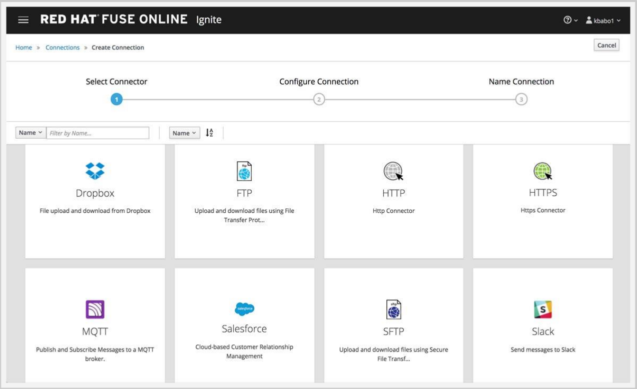
A web-based integration platform (Syndesis).

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!



Dropbox
File upload and download from Dropbox

FTP
Upload and download files using File Transfer Prot...

HTTP
Http Connector

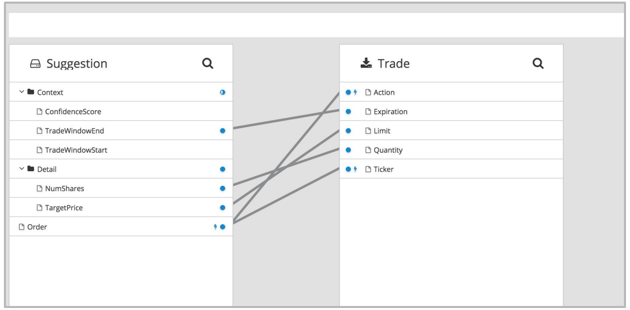
HTTPS
Https Connector

MQTT
Publish and Subscribe Messages to a MQTT broker.

Salesforce
Cloud-based Customer Relationship Management

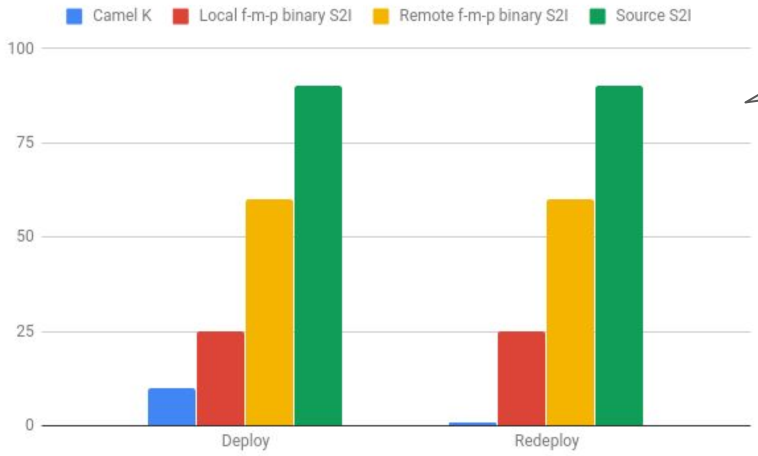
SFTP
Upload and download files using Secure File Transf...

Slack
Send messages to Slack



Time to run an integration in Syndesis

… compared to the traditional spring-boot backend

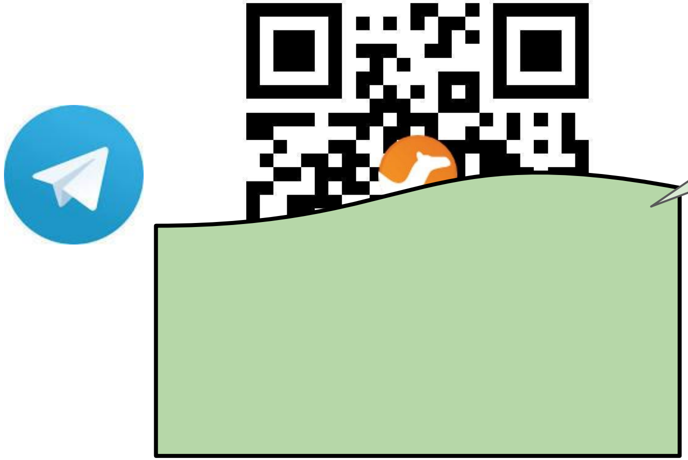


Lower is better :)

DEMO

Camel K Demo

Let’s build a Telegram Chat Bot in few minutes.



You’ll need **shortly on your phone**:

- **Telegram App**
- **A QR Code reader**

Camel K Demo

https://youtu.be/-O-6dwgxH1E

```
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.model.dataformat.JsonLibrary;

public class Routes extends RouteBuilder {
    public void configure() {

        from("telegram:bots/{{token}}")
                .convertBodyTo(String.class)
                .choice()
                    .when(simple("${body.toLowerCase()} contains 'chuck'"))

                        .to("https4://api.icndb.com/jokes/random")
                        .unmarshal().json(JsonLibrary.Jackson)
                        .transform(simple("${body[value][joke]}"))
                        .log("${body}")
```

Camel K Demo

Let’s build a Telegram Chat Bot in few minutes.

With circuit breakers and retries!

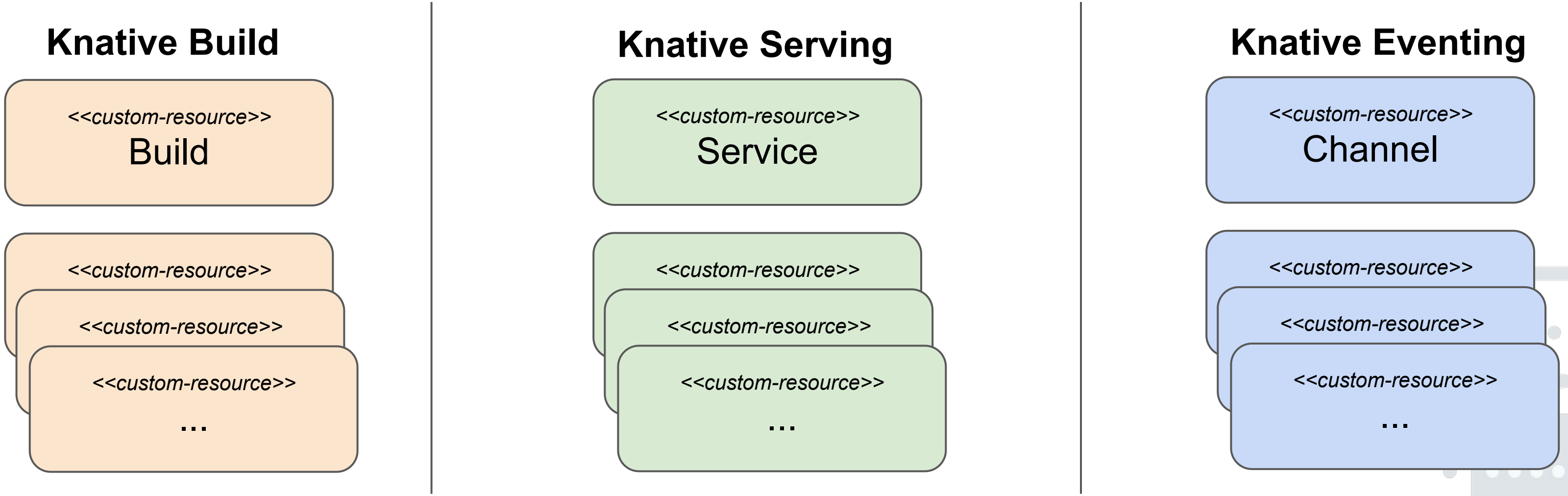
Code:
https://github.com/nicolaferraro/camel-k-chuck-bot

@camelkbot

CAMEL K
&
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 Build

<<custom-resource>>
Build

<<custom-resource>>
<<custom-resource>>
<<custom-resource>>
...

Standardize **building container images**

Knative Serving

<<custom-resource>>
Service

<<custom-resource>>
<<custom-resource>>
<<custom-resource>>
...

Auto-scaling and scale-to-zero

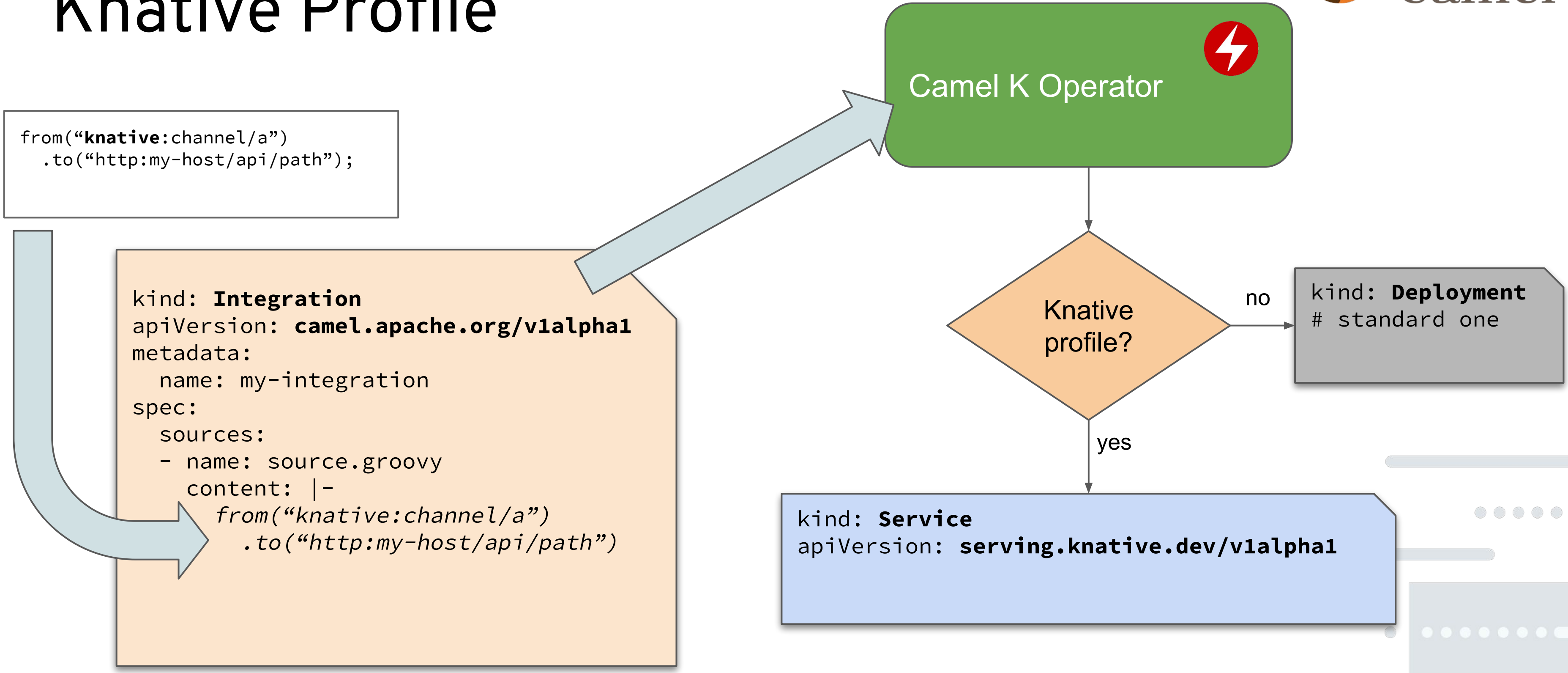
Knative Eventing

<<custom-resource>>
Channel

<<custom-resource>>
<<custom-resource>>
<<custom-resource>>
...

Messaging for **event-based applications**

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")
```

Camel K Operator

Knative profile?

no

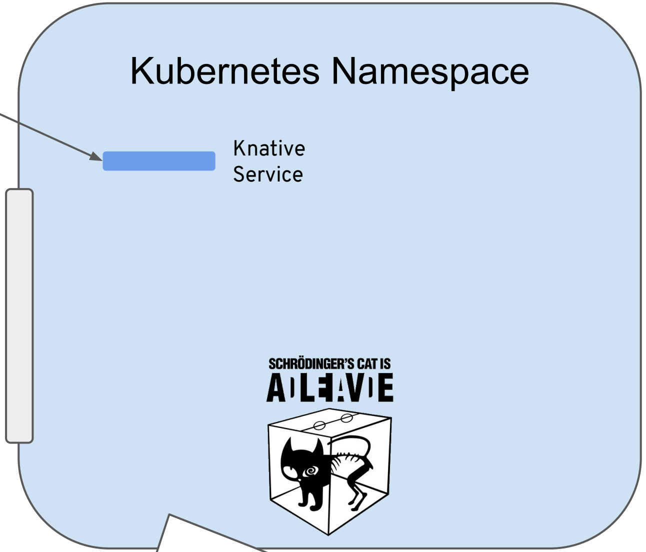
```
kind: Deployment
# standard one
```

yes

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

What does it mean?

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



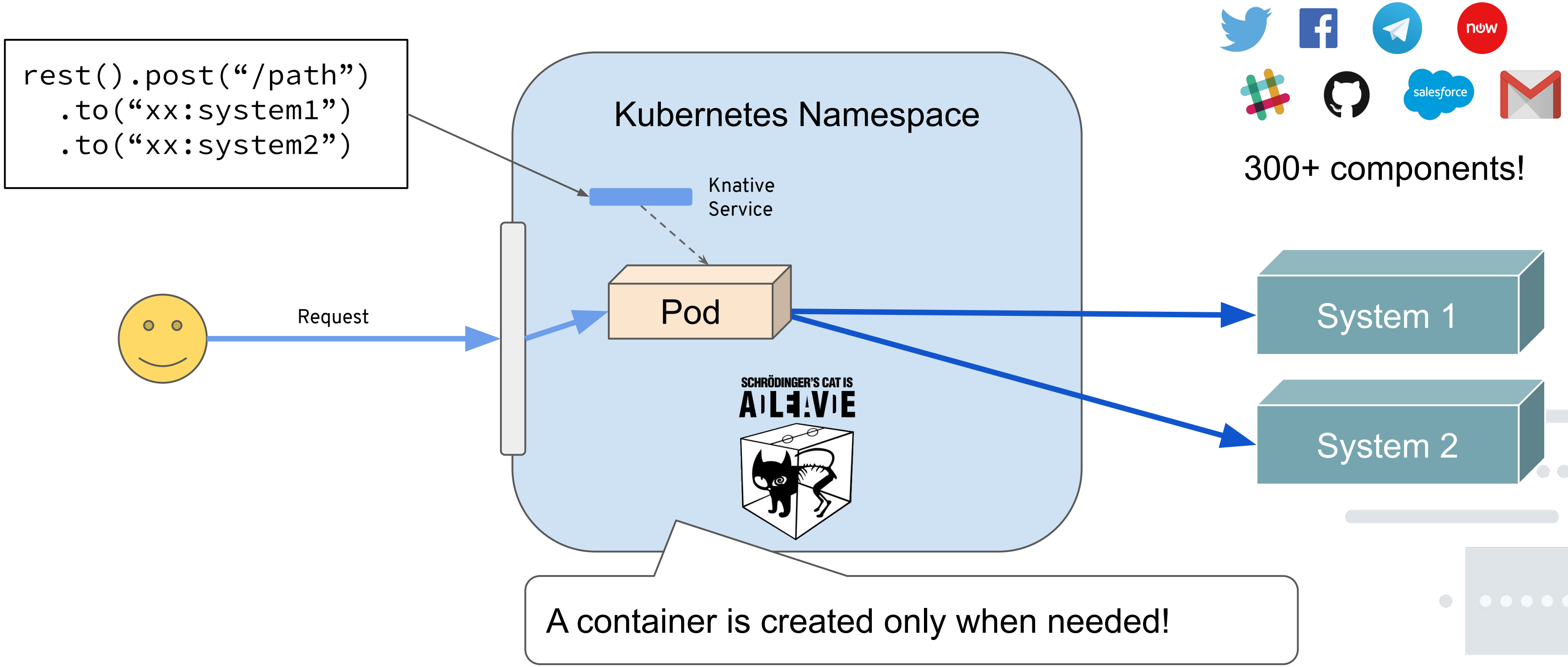
There's no container if no one needs it!

300+ components!

System 1

System 2

What does it mean?

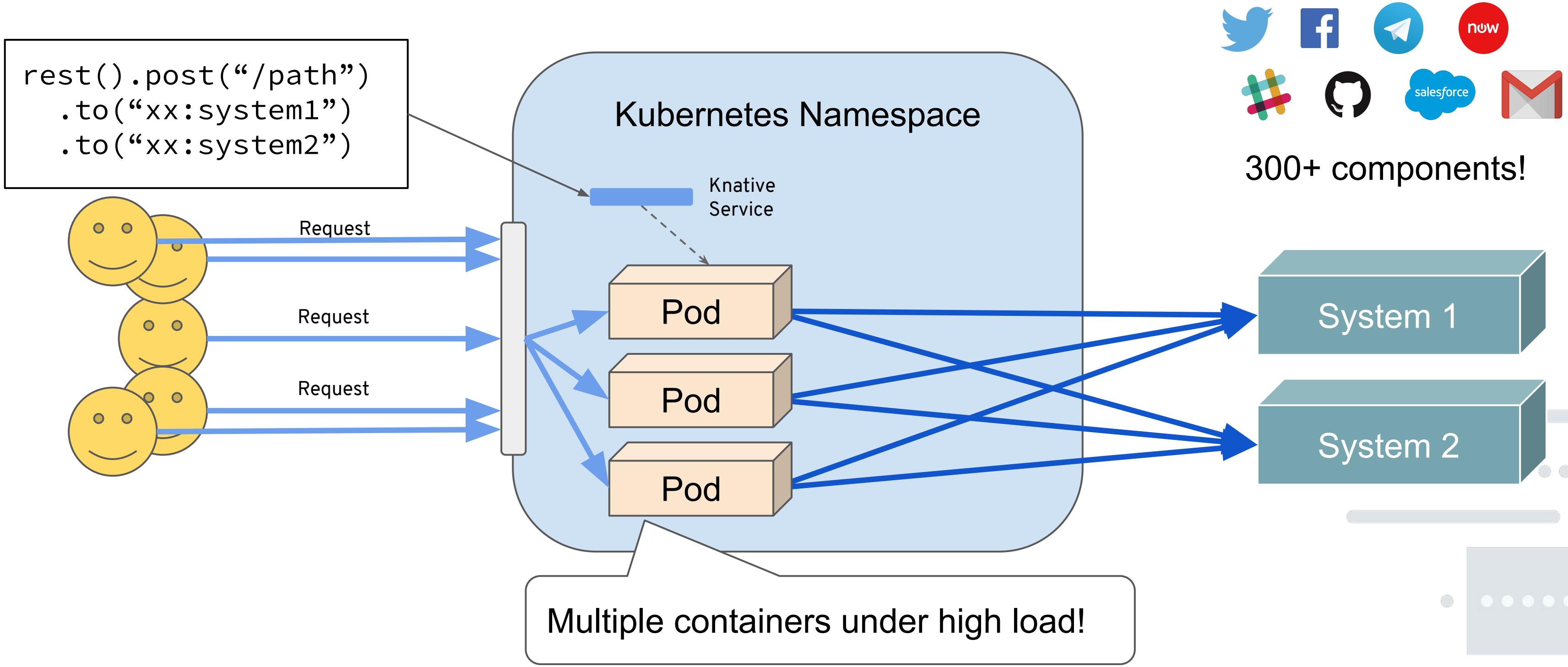


```
rest().post(“/path”)
  .to(“xx:system1”)
  .to(“xx:system2”)
```

300+ components!

A container is created only when needed!

What does it mean?



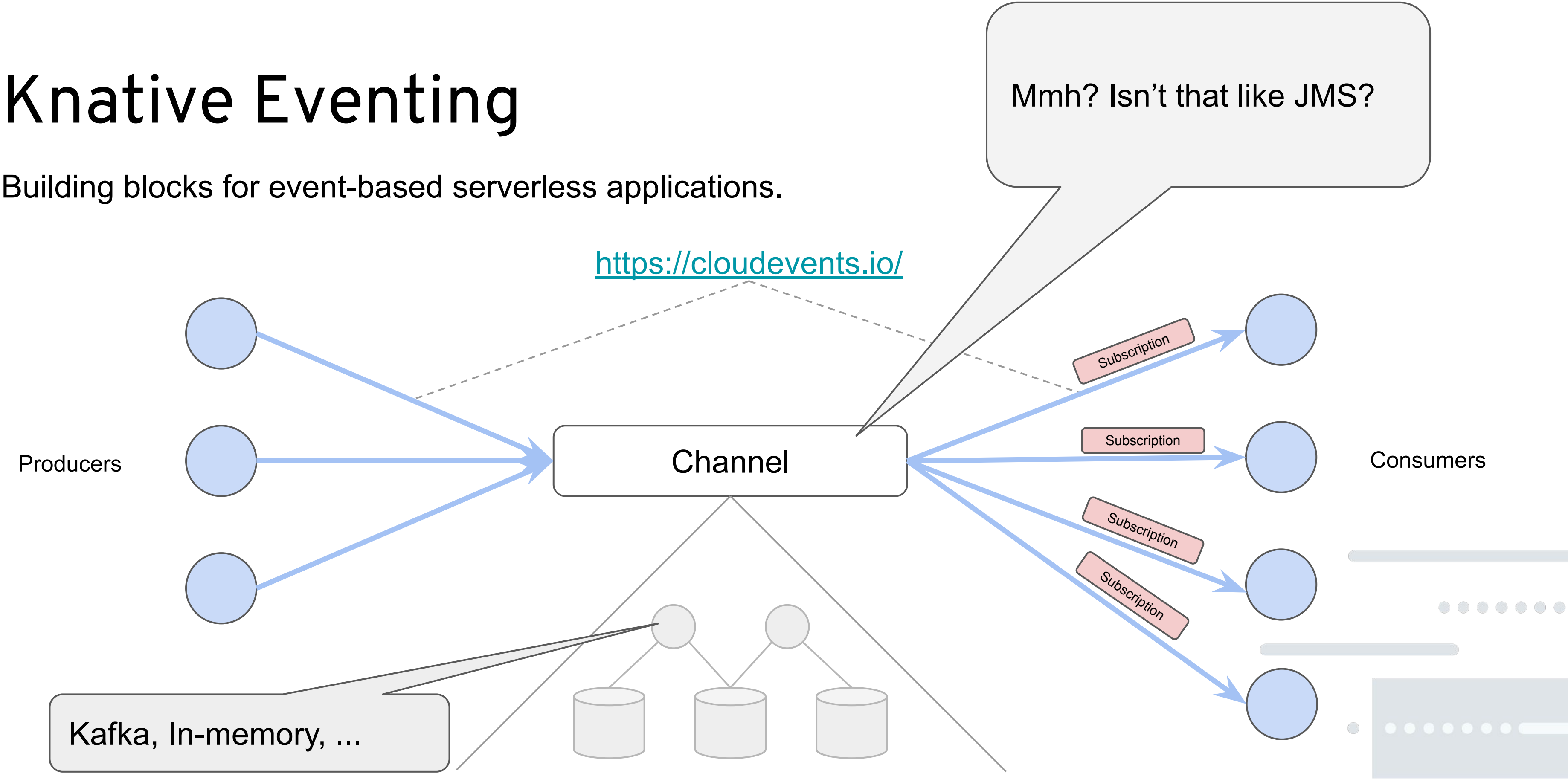
```
rest().post("/path")
  .to("xx:system1")
  .to("xx:system2")
```

300+ components!

Multiple containers under high load!

Knative Eventing

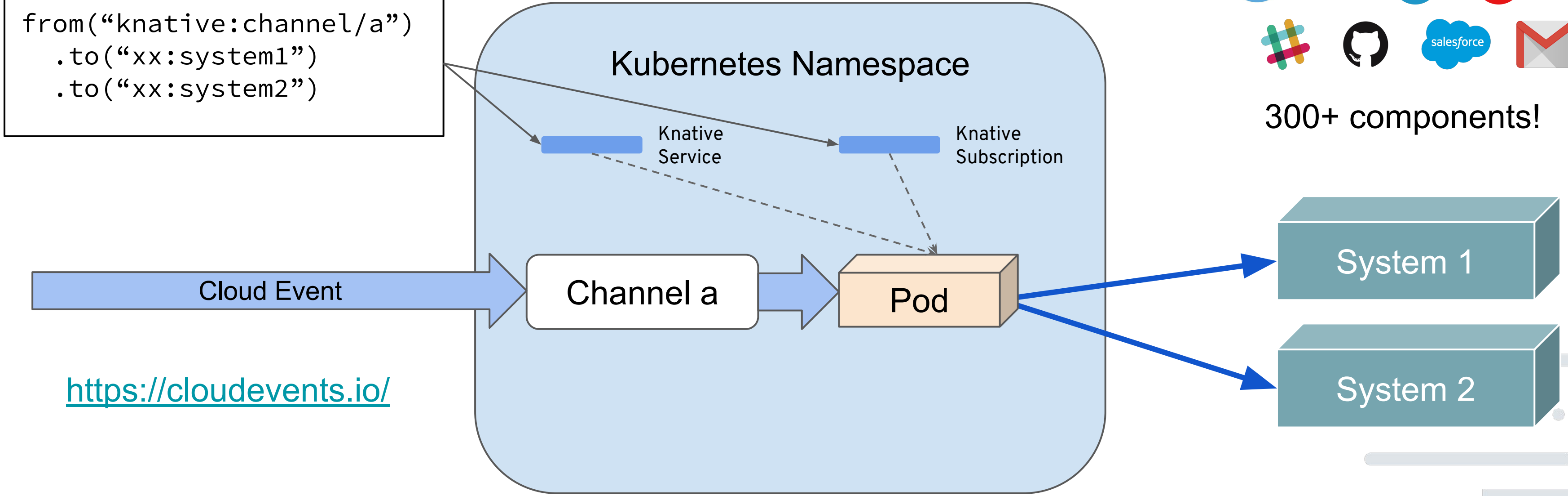
Building blocks for event-based serverless applications.



Knative Eventing

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

300+ components!

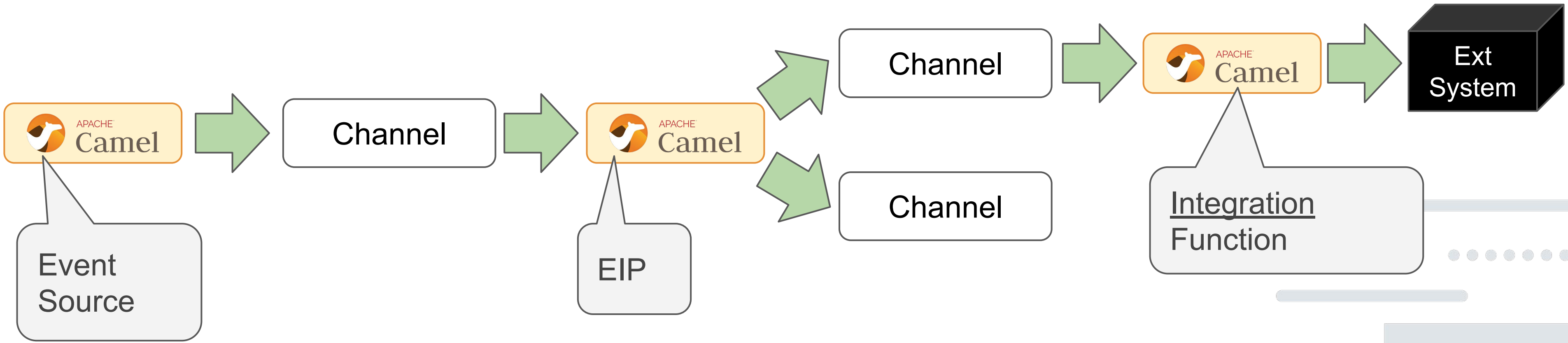


https://cloudevents.io/

Camel K in Knative Eventing

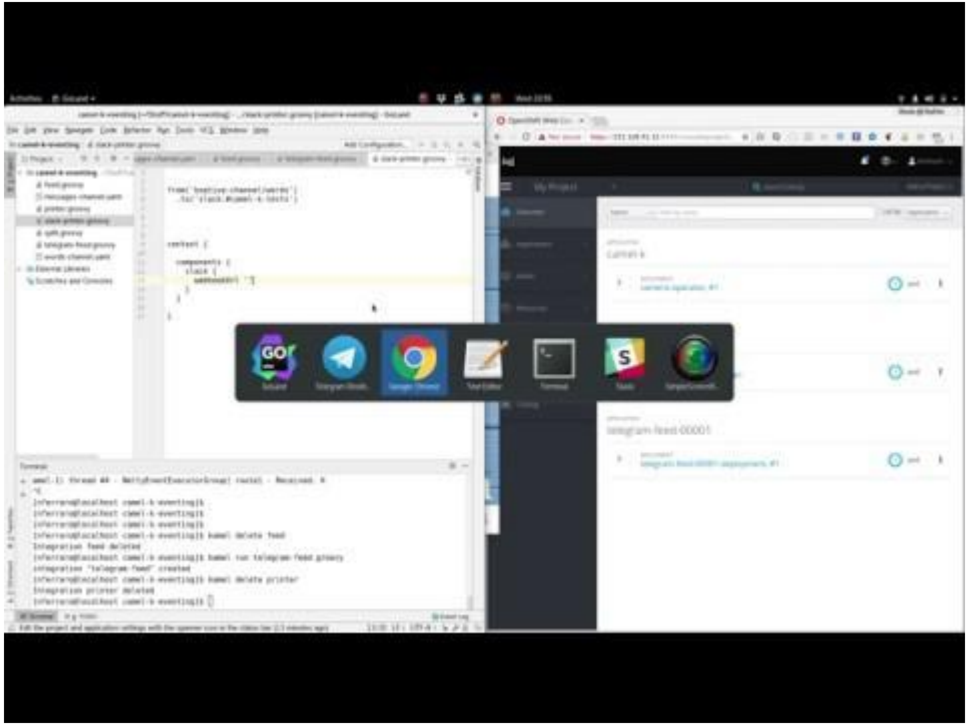
300+ components!

Same model for different purposes



Knative Demo

https://youtu.be/btf_e2GniXM



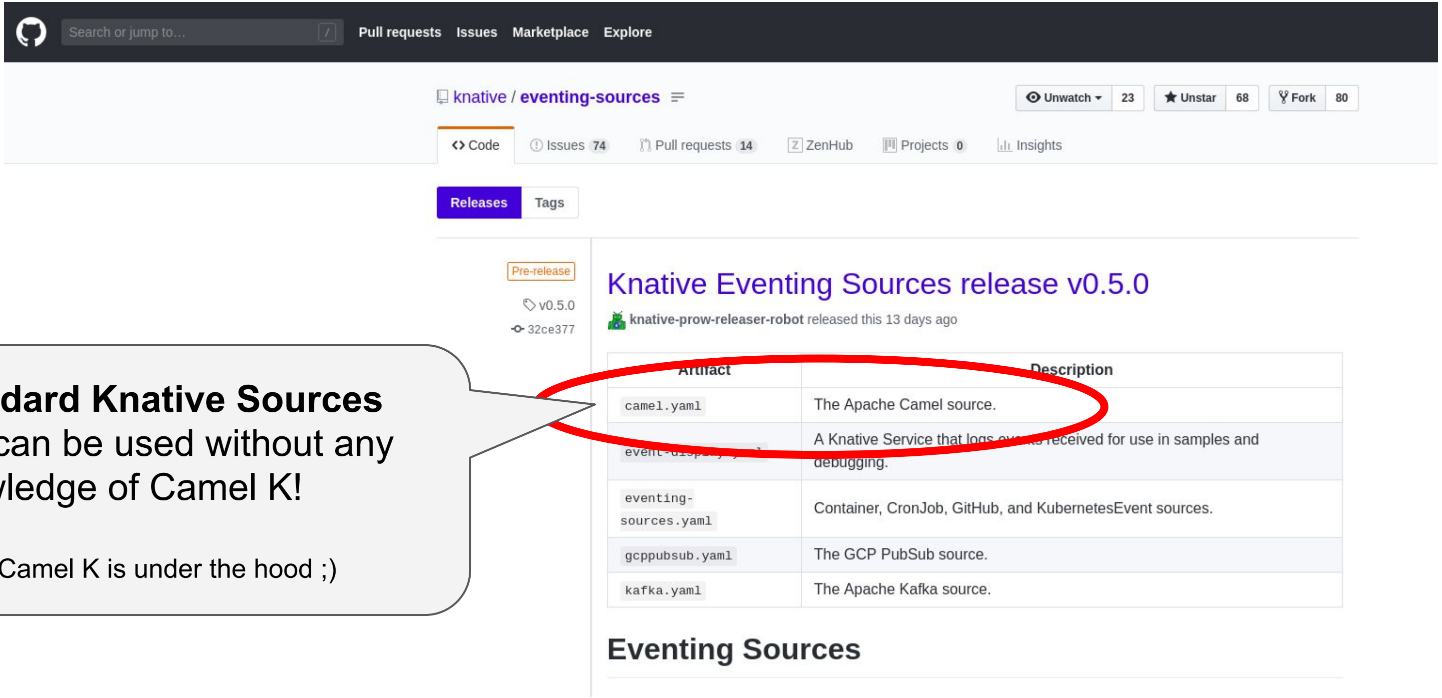
```
from('knative:channel/words')
  .to('slack:#camel-k-tests')

context {
  components {
    slack {
      webhookUrl '']
    }
  }
}
```

```
amel-1) thread #4 - NettyEventExecutorGroup] route1 - Received: ...
^C
[nferrari@localhost camel-k-eventing]$
[nferrari@localhost camel-k-eventing]$
[nferrari@localhost camel-k-eventing]$
[nferrari@localhost camel-k-eventing]$ kamel delete feed
Integration feed deleted
[nferrari@localhost camel-k-eventing]$ kamel run telegram-feed.groovy
integration "telegram-feed" created
[nferrari@localhost camel-k-eventing]$ kamel delete printer
Integration printer deleted
[nferrari@localhost camel-k-eventing]$
```

KNATIVE CAMEL SOURCES

Knative Camel Sources



Knative Eventing Sources release v0.5.0

knative-prow-releaser-robot released this 13 days ago

Artifact	Description
`camel.yaml`	The Apache Camel source.
`event-display.yaml`	A Knative Service that logs events received for use in samples and debugging.
`eventing-sources.yaml`	Container, CronJob, GitHub, and KubernetesEvent sources.
`gcppubsub.yaml`	The GCP PubSub source.
`kafka.yaml`	The Apache Kafka source.

Eventing Sources

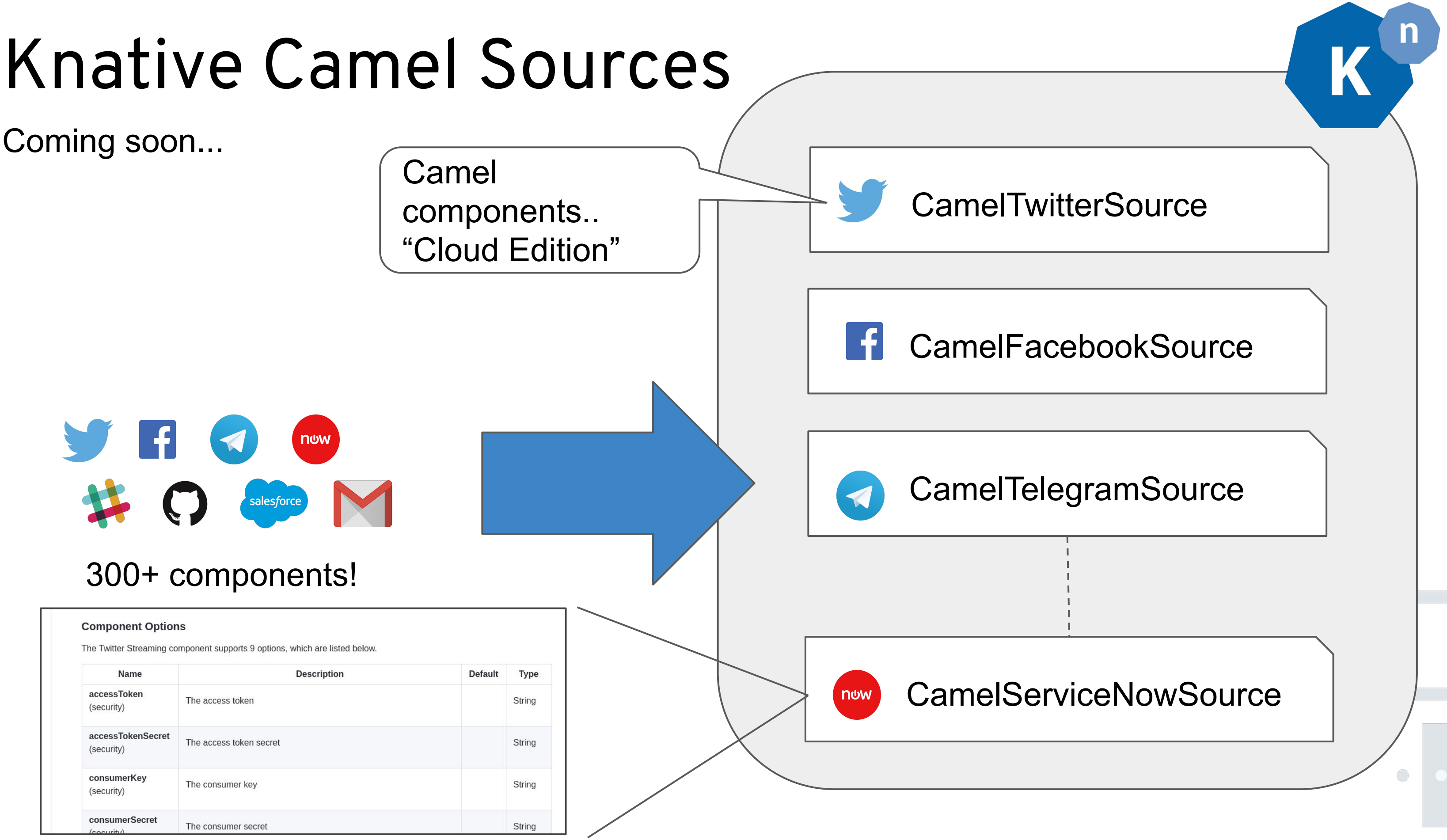
Standard Knative Sources that can be used without any knowledge of Camel K!

… but Camel K is under the hood ;)

https://github.com/knative/eventing-sources/releases

Knative Camel Sources

Coming soon...



Camel components..
“Cloud Edition”

CamelTwitterSource

CamelFacebookSource

CamelTelegramSource

CamelServiceNowSource

300+ components!

Component Options

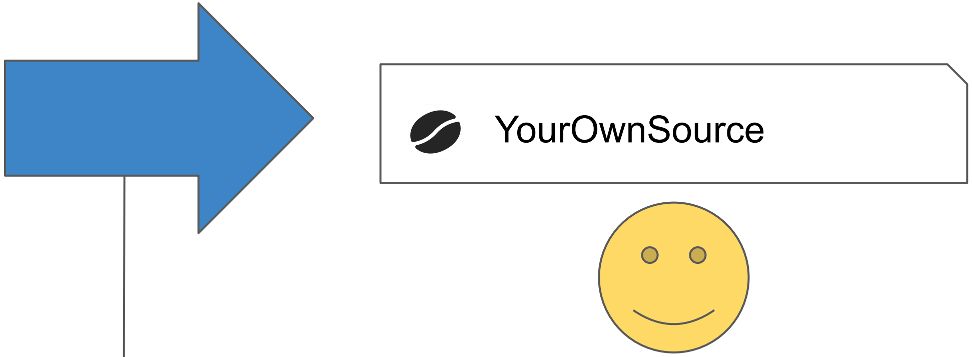
The Twitter Streaming component supports 9 options, which are listed below.

Name	Description	Default	Type
accessToken (security)	The access token		String
accessTokenSecret (security)	The access token secret		String
consumerKey (security)	The consumer key		String
consumerSecret (security)	The consumer secret		String

Knative Camel Sources

Coming soon...

```
from(“timer:tick?period={{period}}”)
  .to(“xx:my-system”)
  .unmarshal().json()
  .transform(“custom”)
  .to(“knative:endpoint/sink”)
```



A **Dev** writes Camel K code.

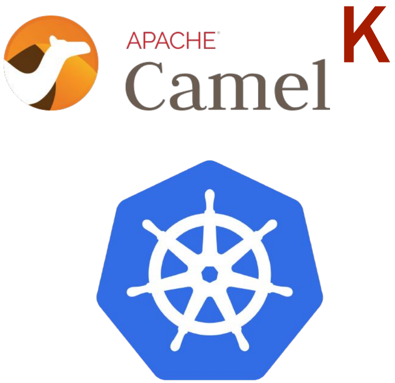
Users can instantiate the source to do their job.

RECAP AND WHAT'S NEXT...

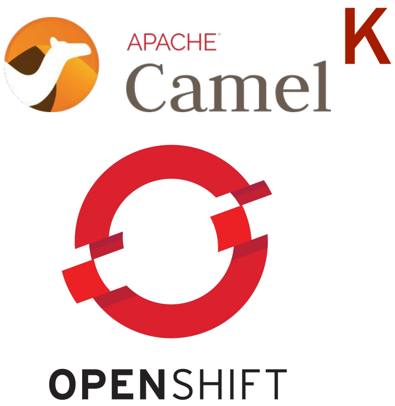
What is Camel K?

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)!

What’s next?

Knative features:

- **Camel Components as Knative Sources**
- **Use Camel K to build your own Knative Source**

Most notable features:

- **Support for Camel 3 (Now M2, GA in September 2019)**
  - **Lightweight modules to reduce footprint**
  - **Webhook meta-component (more components that can scale down to 0)**
- **Integration with Operator Hub**
- **Full integrated with Syndesis**
- **Better IDE support**
- **Better integration with CI tools**
- **Testing**
- **Camel K UI**

What can you do next?

If you like Camel K:

1) Star it on github! ★ Star
2) Use it!
3) Contribute!

We love contributions!

APACHE Camel

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

Follow us on Twitter:

@ni_ferraro

@lburgazzoli

RED HAT
SUMMIT

THANK YOU

- plus.google.com/+RedHat
- linkedin.com/company/red-hat
- youtube.com/user/RedHatVideos
- facebook.com/redhatinc
- twitter.com/redhat