



# Riding Apache Camel on Cloud

[willem.jiang@gmail.com](mailto:willem.jiang@gmail.com)

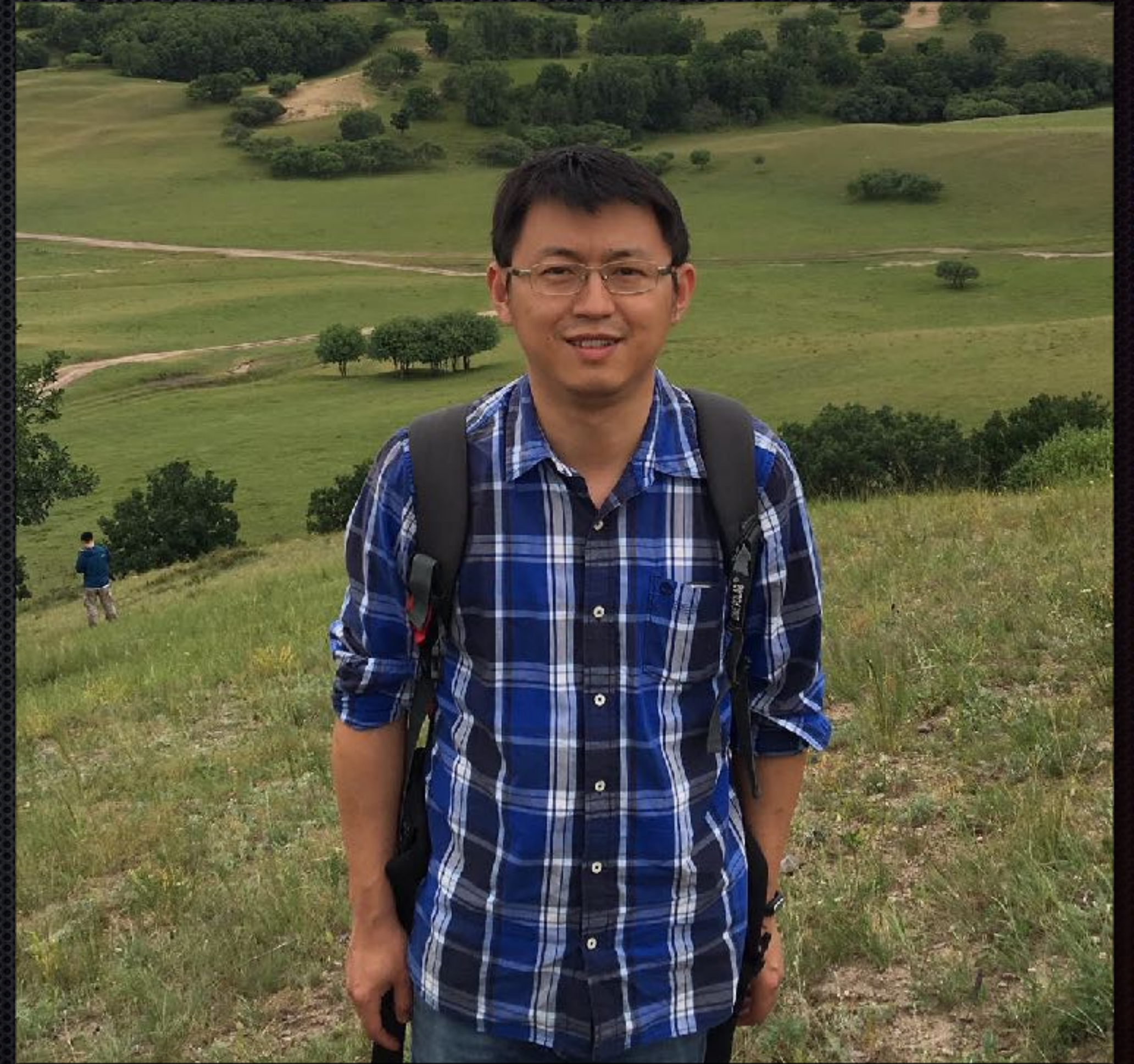
blog: <https://willemjiang.github.io>

weibo: [willemjiang](#)

2019-03

# About Me

- ✦ Open Source Developer in Huawei
- ✦ Apache Camel Committer since 2008
- ✦ Apache Member



# Agenda

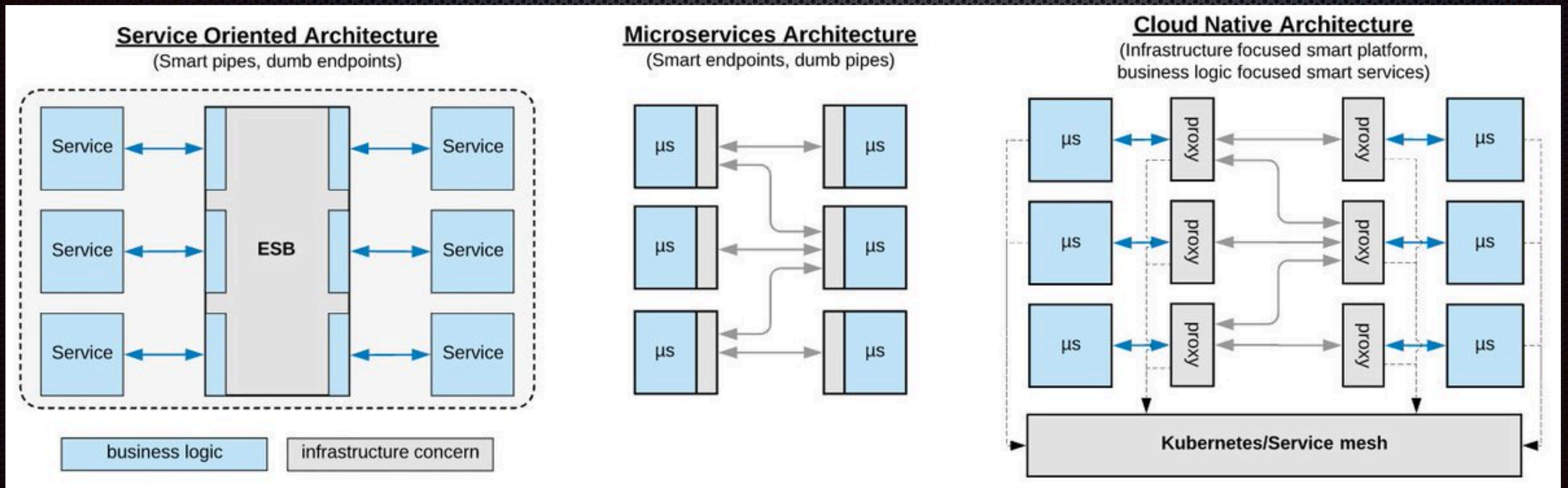
- ✦ Cloud native and Integration
- ✦ Apache Camel Introduction
- ✦ Apache Camel-K
- ✦ Demos



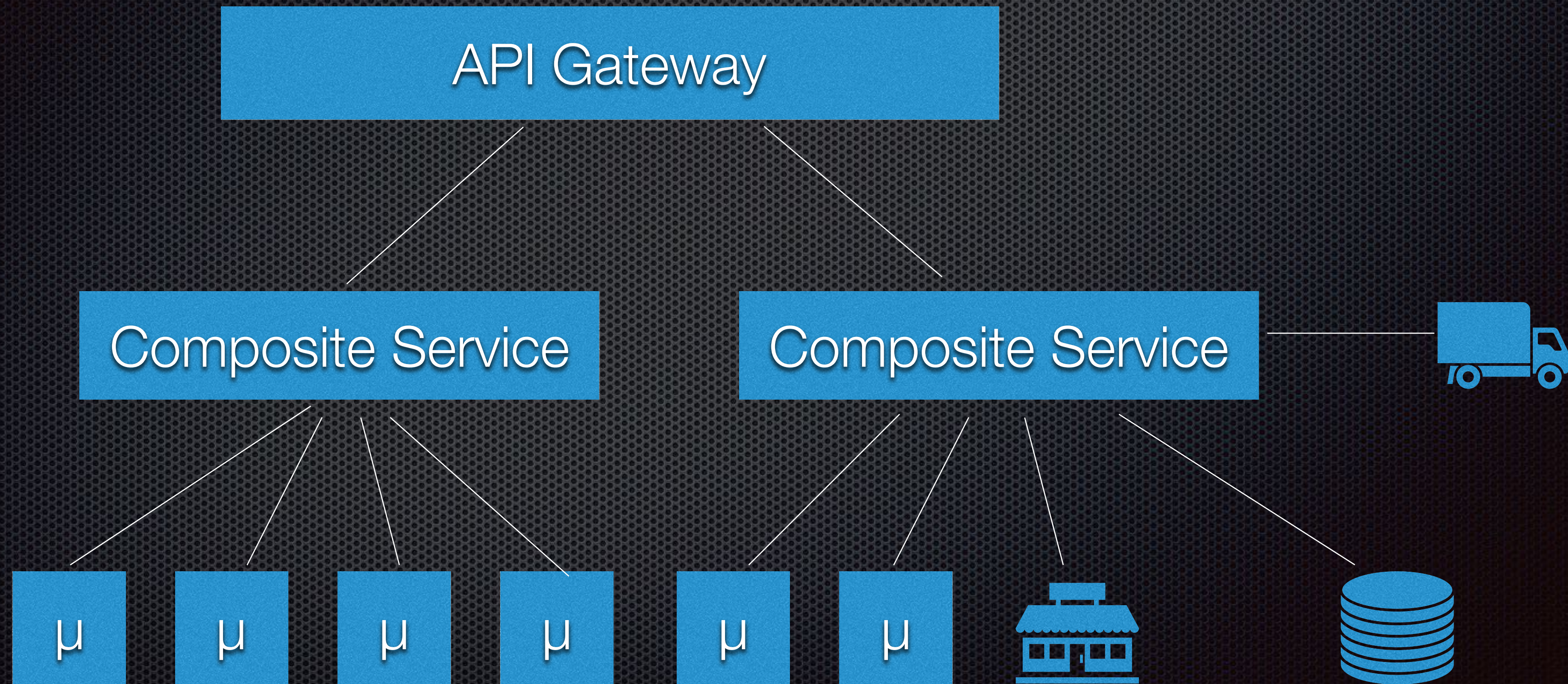
# Digital Transformation



# SOA to Cloud Native



# Composite Services as Integration tool



# Apache Camel

- ✦ Open Source **integration framework** based on known **Enterprise Integration Patterns**
- ✦ Started as a subproject of **ActiveMQ** from **ServiceMix EIP** module
  - ✦ r519901 | jstrachan | 2007-03-19 11:54:57 +0100 (Mon, 19 Mar 2007) | 1 line
- ✦ Became the Top Level Project of Apache in 2009
- ✦ A Camel can carry 4 times as much load as other beasts of burden!

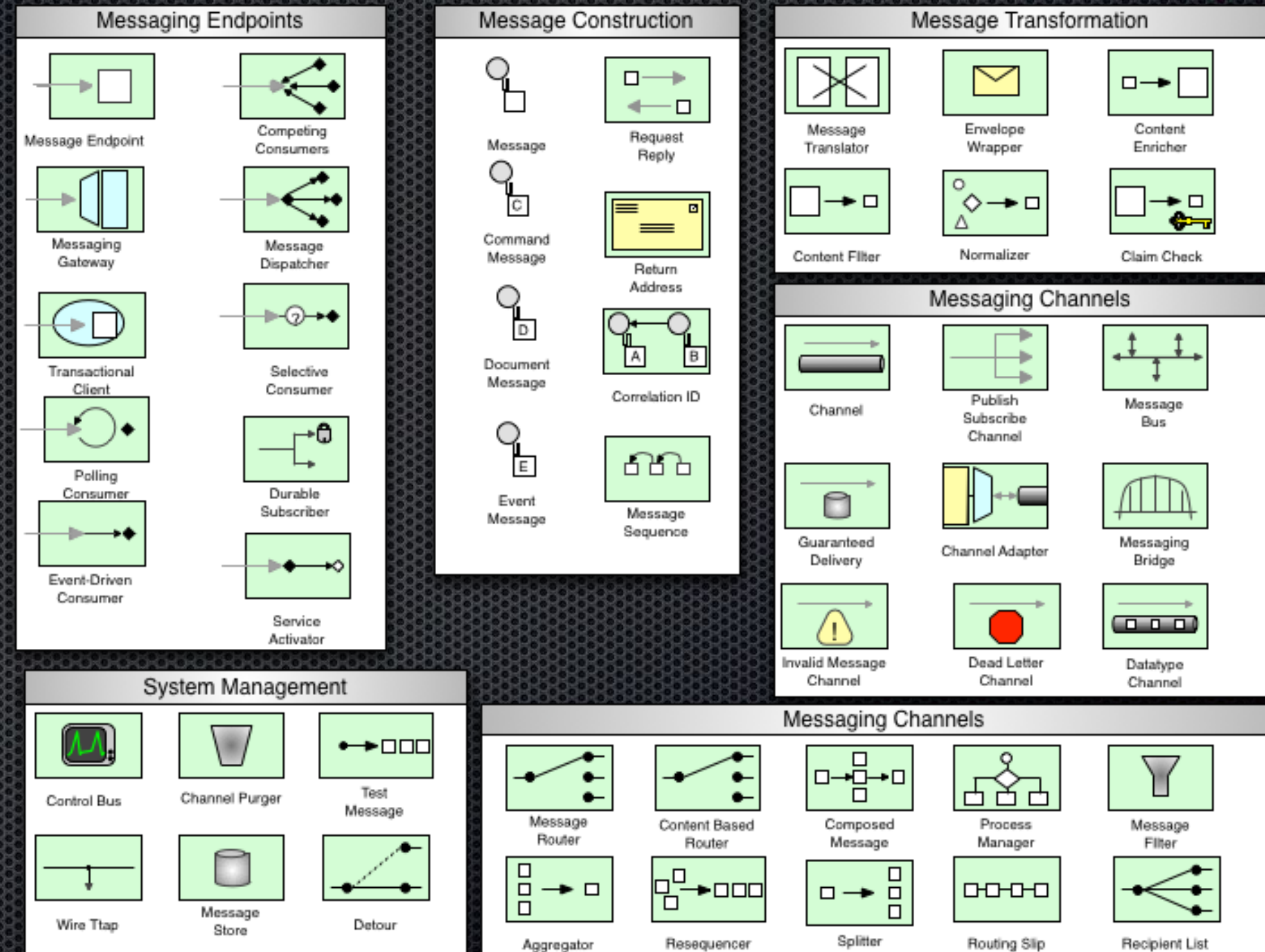
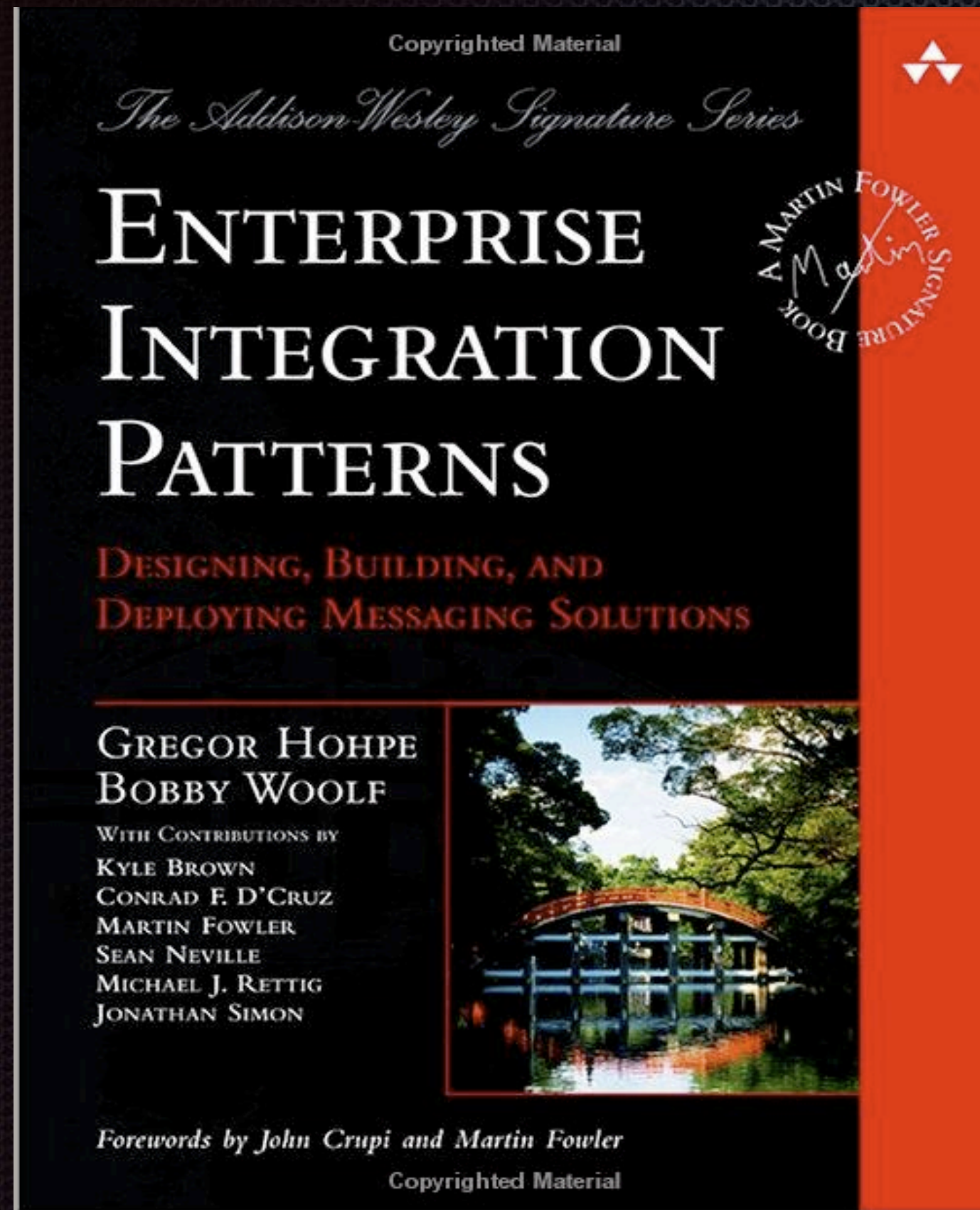
# Why Camel

- ✦ EIP implementation
- ✦ 300+ Components
- ✦ Easy to run
- ✦ Amazing Community

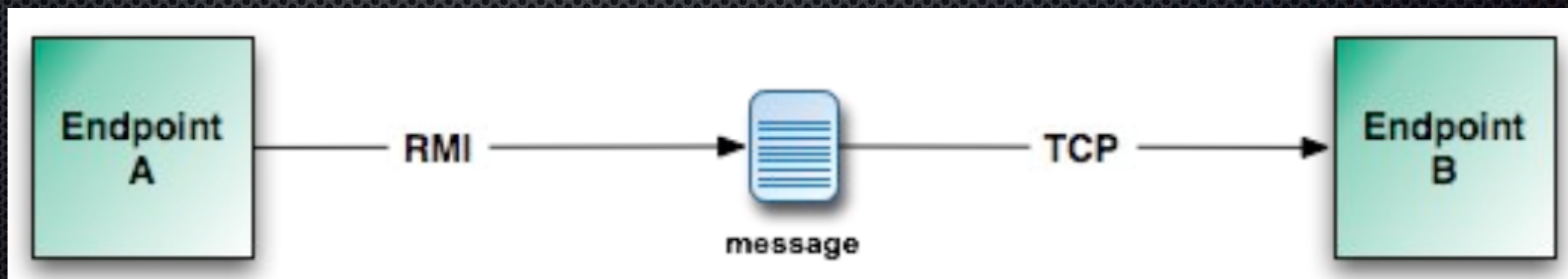




# Enterprise Integration Patterns



# Message Routing



# Simple Routing



```
from("file:src/data?noop=true").  
to("jms:queue:myqueue");
```

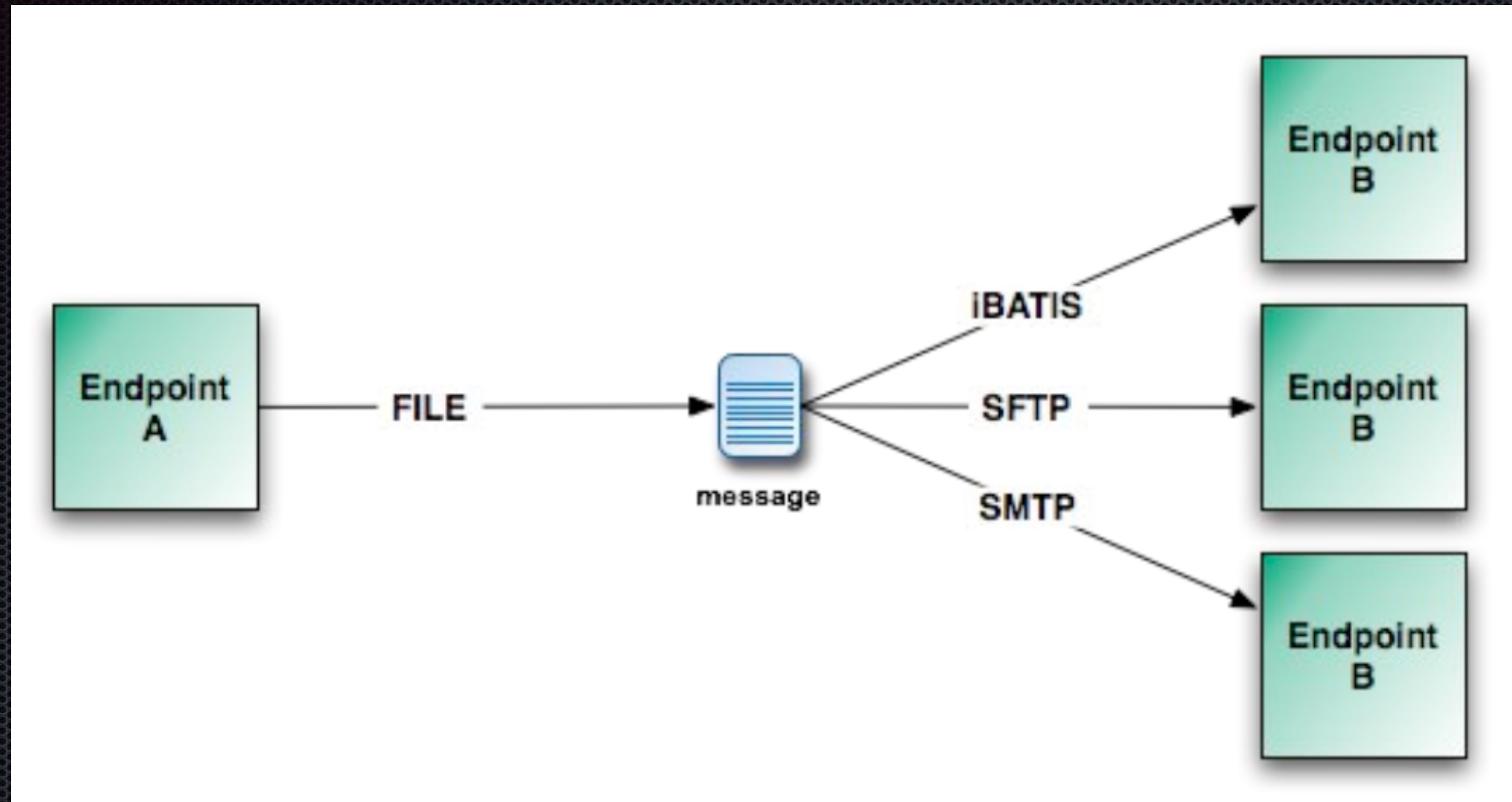
xxx://endpoint/address?accessToken=xxxxx&delay=30

scheme

relation path

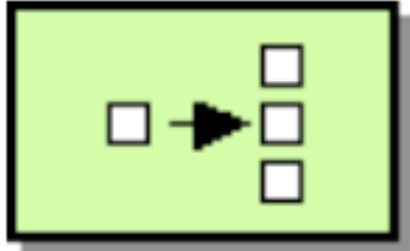
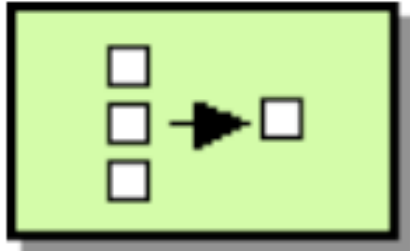
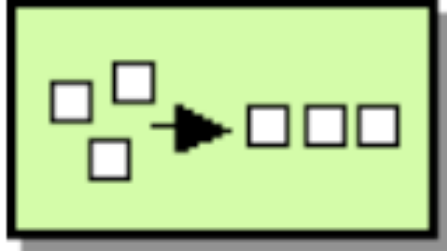
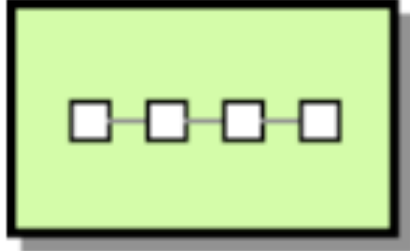
parameters

# Multicast Routing

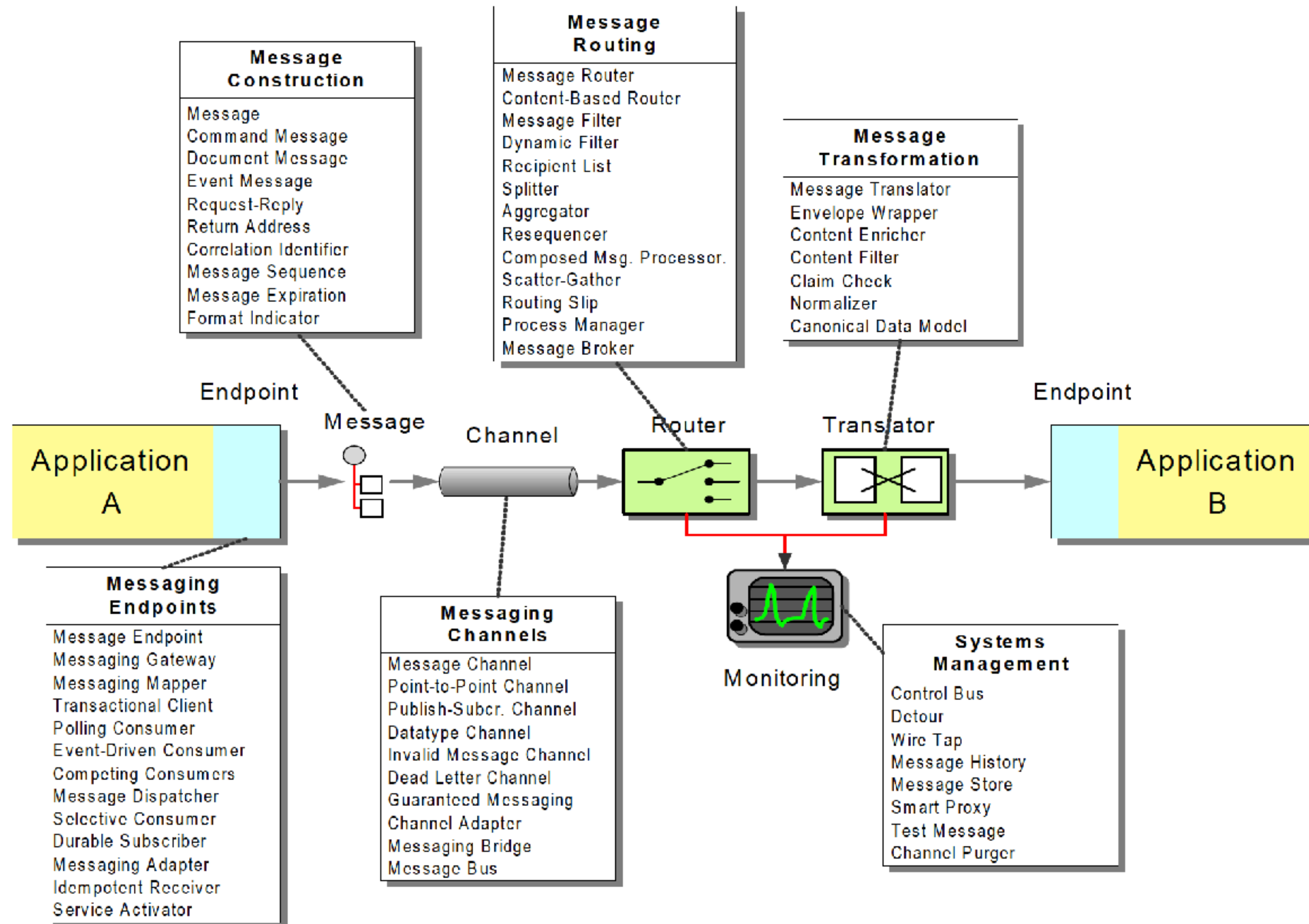


```
from("file:src/data?noop=true").  
multicast("ibatis://xxx", "sftp://xxx", "smtp://xxx");
```

# Message Routing

	<b>Content Based Router</b>	How do we handle a situation where the implementation of a single logical function (e.g., inventory check) is spread across multiple physical systems?
	<b>Message Filter</b>	How can a component avoid receiving uninteresting messages?
	<b>Dynamic Router</b>	How can you avoid the dependency of the router on all possible destinations while maintaining its efficiency?
	<b>Recipient List</b>	How do we route a message to a list of dynamically specified recipients?
	<b>Splitter</b>	How can we process a message if it contains multiple elements, each of which may have to be processed in a different way?
	<b>Aggregator</b>	How do we combine the results of individual, but related messages so that they can be processed as a whole?
	<b>Resequencer</b>	How can we get a stream of related but out-of-sequence messages back into the correct order?
	<b>Routing Slip</b>	How do we route a message consecutively through a series of processing steps when the sequence of steps is not known at design-time and may vary for each message?
	<b>Throttler</b>	How can I throttle messages to ensure that a specific endpoint does not get overloaded, or we don't exceed an agreed SLA with some external service?
	<b>Delayer</b>	How can I delay the sending of a message?
	<b>Load Balancer</b>	How can I balance load across a number of endpoints?
	<b>Multicast</b>	How can I route a message to a number of endpoints at the same time?
	<b>Loop</b>	How can I repeat processing a message in a loop?

# Enterprise Integration Patterns



# Camel Component

XXXComponent

XXXEndpoint

XXXConsumer

XXXProducer

XXXApplication

Camel Processors

XXXApplication

# Camel Components

ActiveMQ	File	JBIC	MINA	RMI	TCP
ActiveMQ Journal	FIX	JCR	Mock	RNC	Test
AMQP	Flatpack	JDBC	MSMQ	RNG	Timer
Atom	FTP	Jetty	MSV	SEDA	UDP
Bean	Hibernate	JMS	Multicast	SFTP	Validation
CXF	HTTP	JPA	POJO	SMTP	Velocity
DataSet	iBATIS	JT/400	POP	Spring Integration	VM
Direct	IMAP	List	Quartz	SQL	XMPP
Esper	IRC	Log	Queue	Stream	XQuery
Event	JavaSpace	Mail	Ref	String Template	XSLT

350+ components



# How to run the camel Application

```
CamelContext context = new DefaultCamelContext();
context.addRoutes(new RouteBuilder() {
    () -> {
        from("direct:start").to("mock:result");
    }
});
context.start();
```

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:start"/>
    <from to="mock:result"/>
  </route>
</camelContext>
```

- Start a JVM which holds right dependencies

# Community

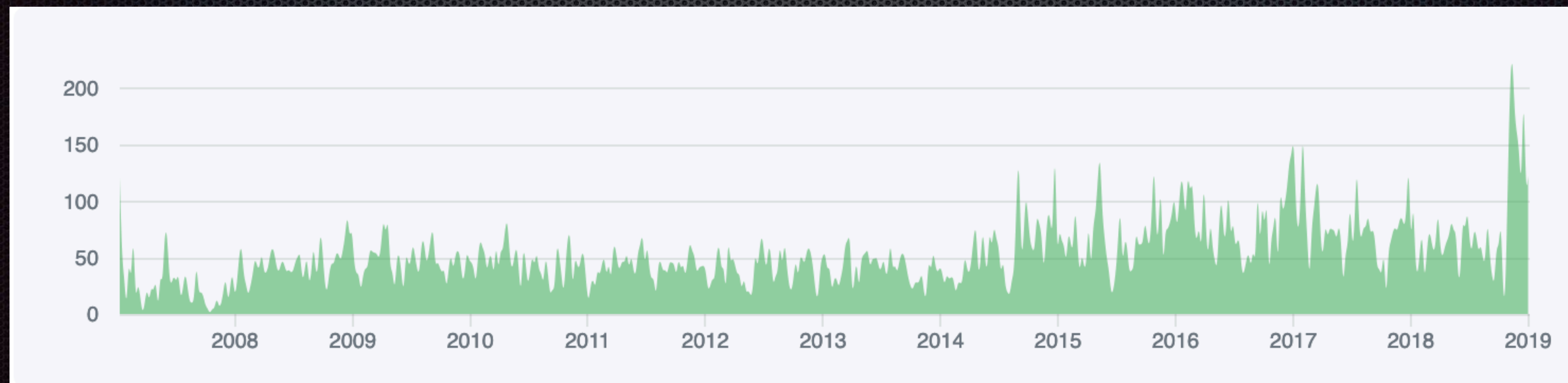
apache / camel

View Repository   Unwatch 283   Unstar 2,454   Fork 3,292

Code   Pull requests 4   Actions   Projects 0   Insights

Apache Camel <https://camel.apache.org/>

36,095 commits   58 branches   136 releases   466 contributors   Apache-2.0



# Spring boot

- ✦ <https://start.spring.io/> -> choose camel
- ✦ Write your own builder
- ✦ Add the components you want
- ✦ Running the application

# Demo Time

<https://github.com/WillemJiang/camel-elasticsearch-demo>

# Apache Camel-K

- ✦ Running Camel Application natively on Kubernetes and Openshift.
- ✦ Designed for **serverless** and **microservice architectures**.
- ✦ Leveraging the **Operator SDK**
- ✦ Building with **Knative Eventing** and **Knative Serving**

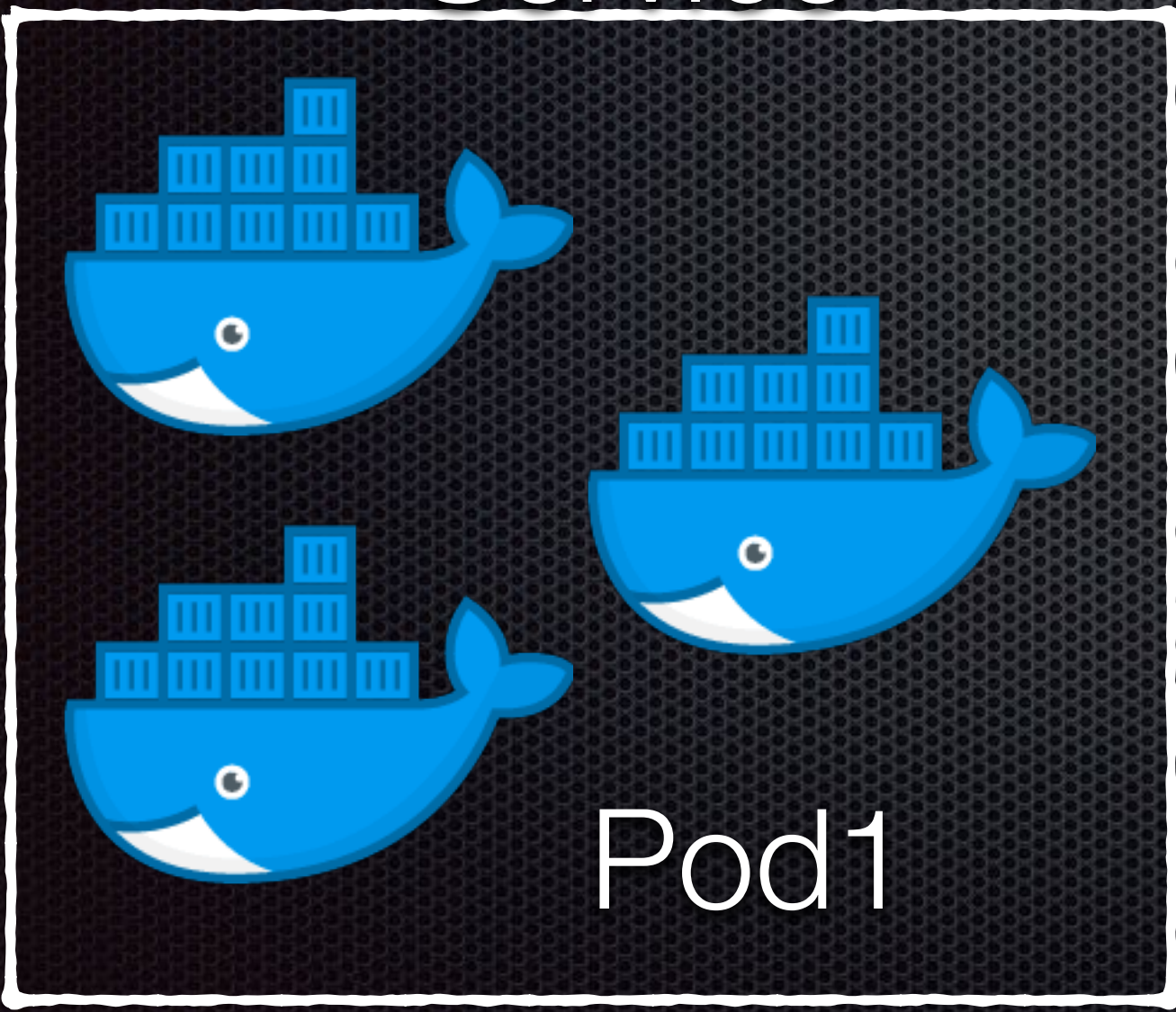
# Running Camel on Kubernetes



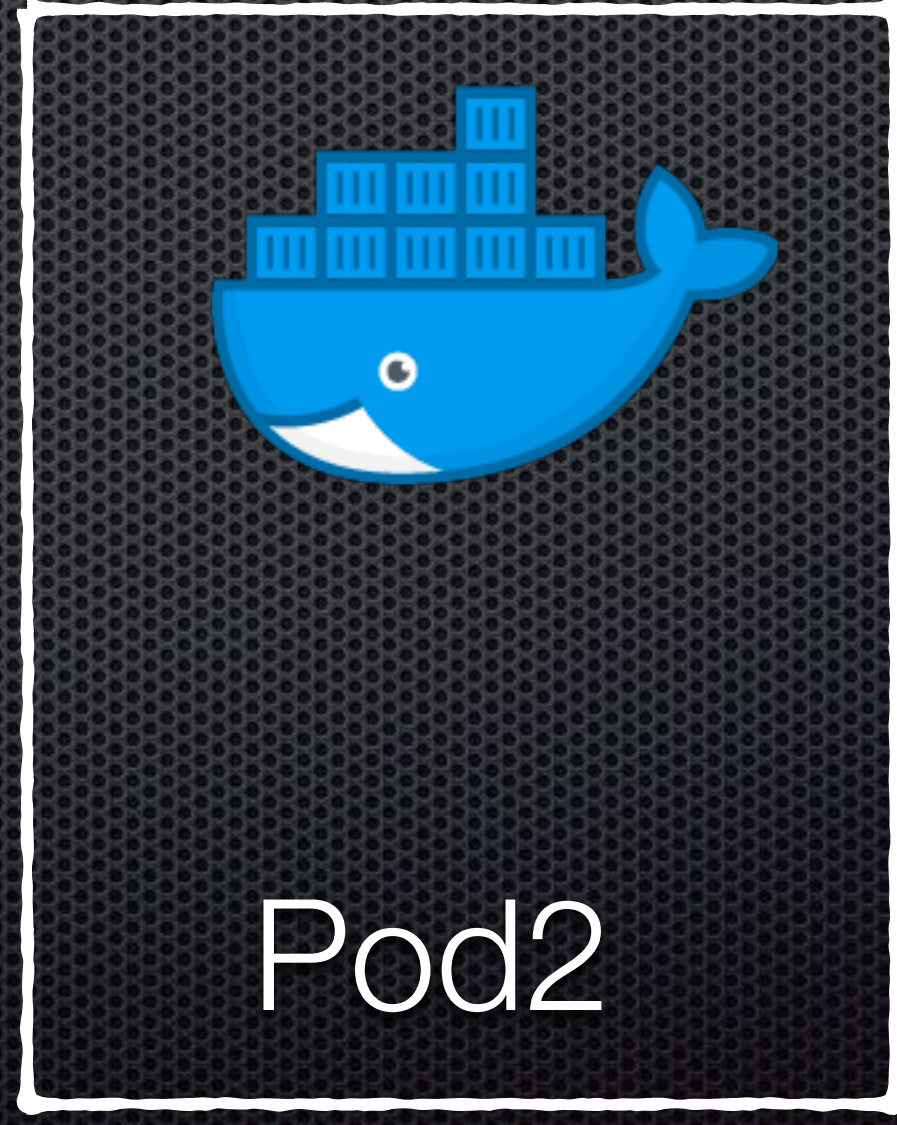
Customer Define Resource



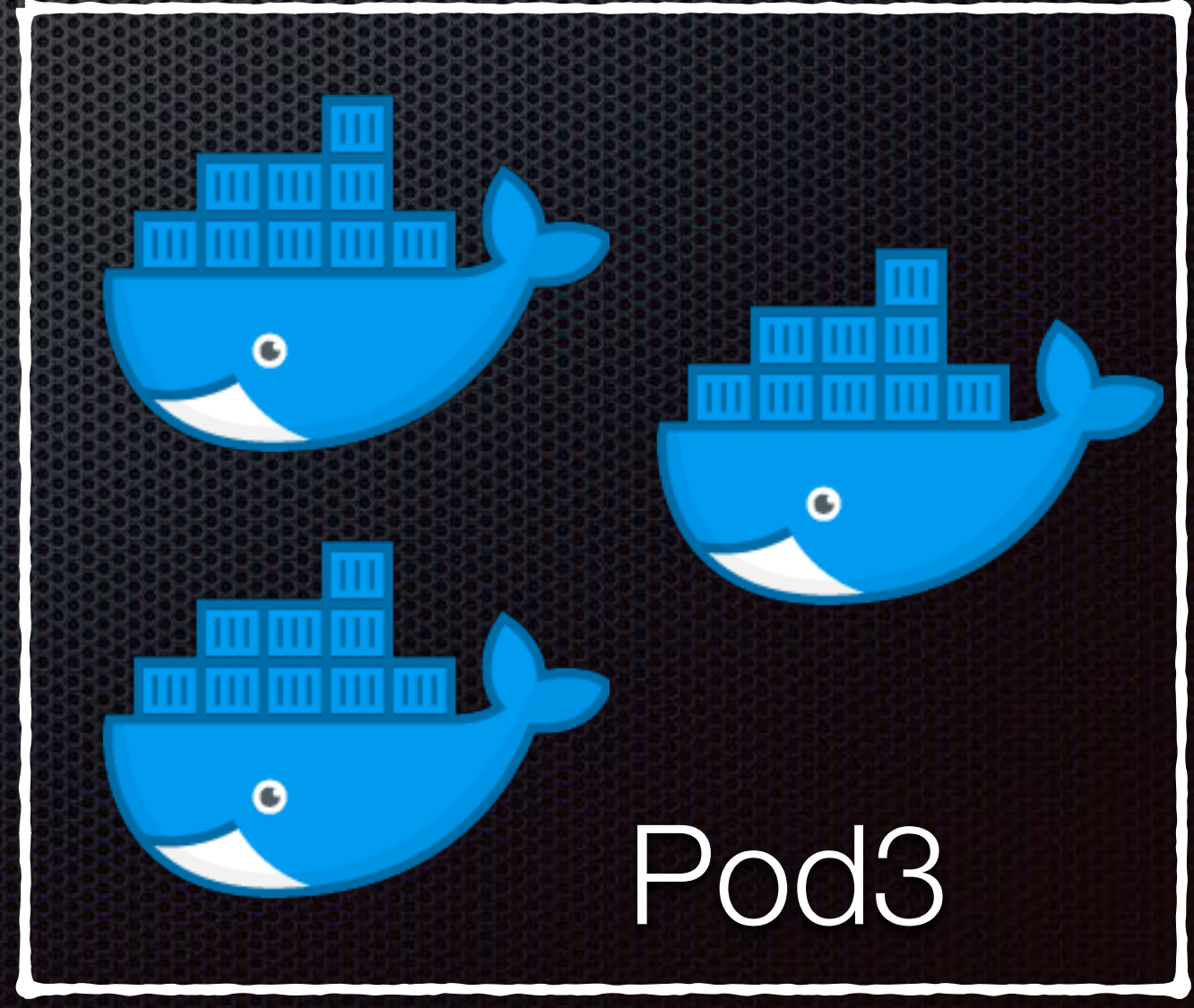
Service

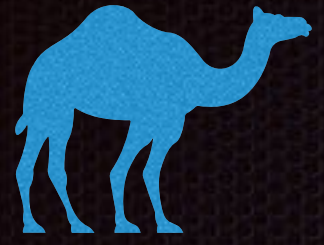


Pod2

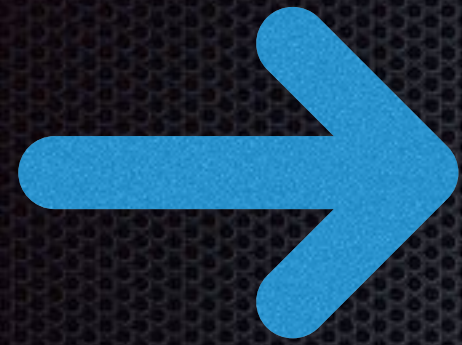


Service





Camel Routes

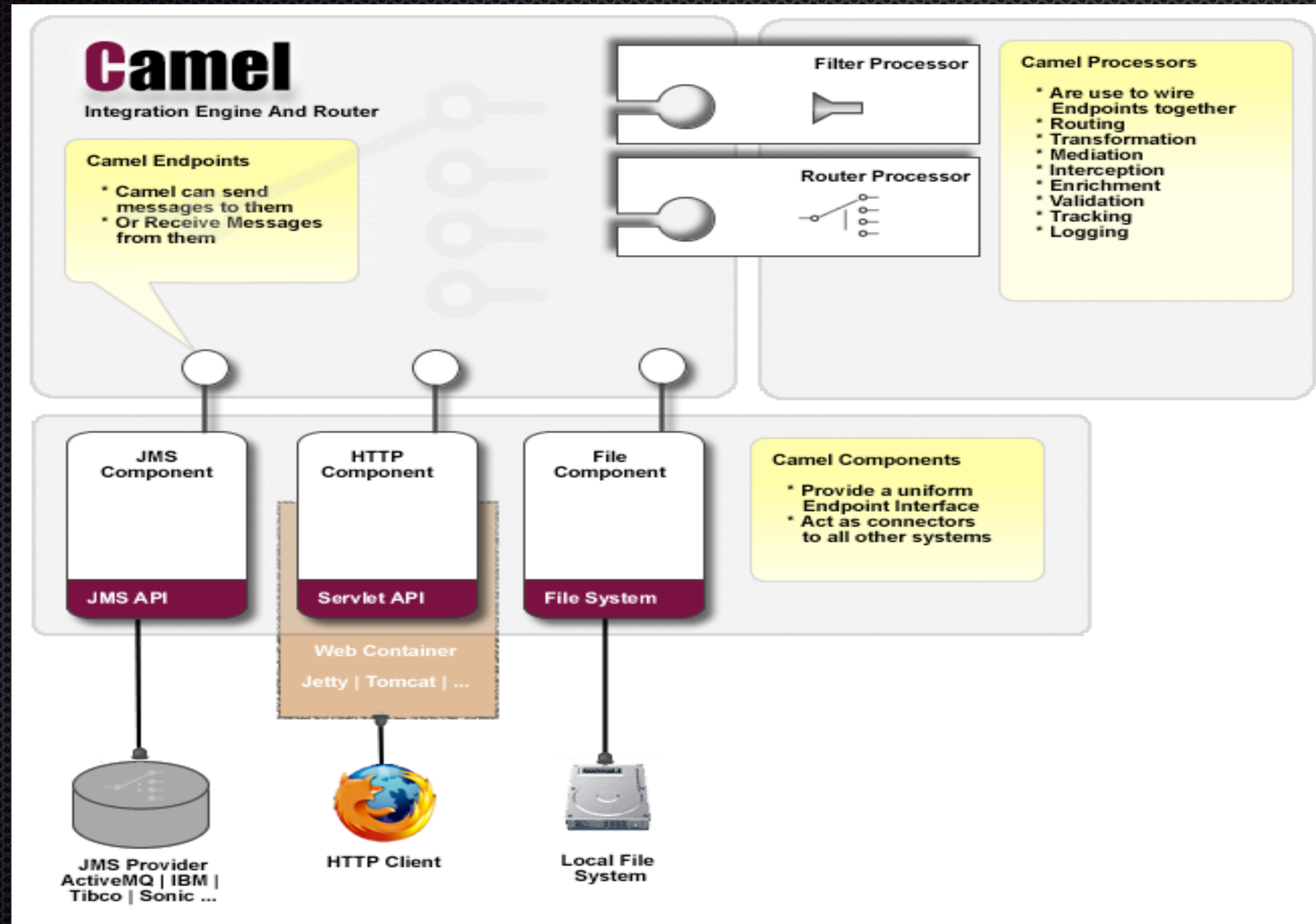


# Reference

- ✦ <https://camel.apache.org>
- ✦ [http://planet.jboss.org/post/camel\\_and\\_enterprise\\_integration\\_patterns](http://planet.jboss.org/post/camel_and_enterprise_integration_patterns)
- ✦ [https://access.redhat.com/documentation/en-us/red\\_hat\\_fuse/7.2/html/apache\\_camel\\_development\\_guide/](https://access.redhat.com/documentation/en-us/red_hat_fuse/7.2/html/apache_camel_development_guide/)



# Camel Architecture



# Message and Exchange

