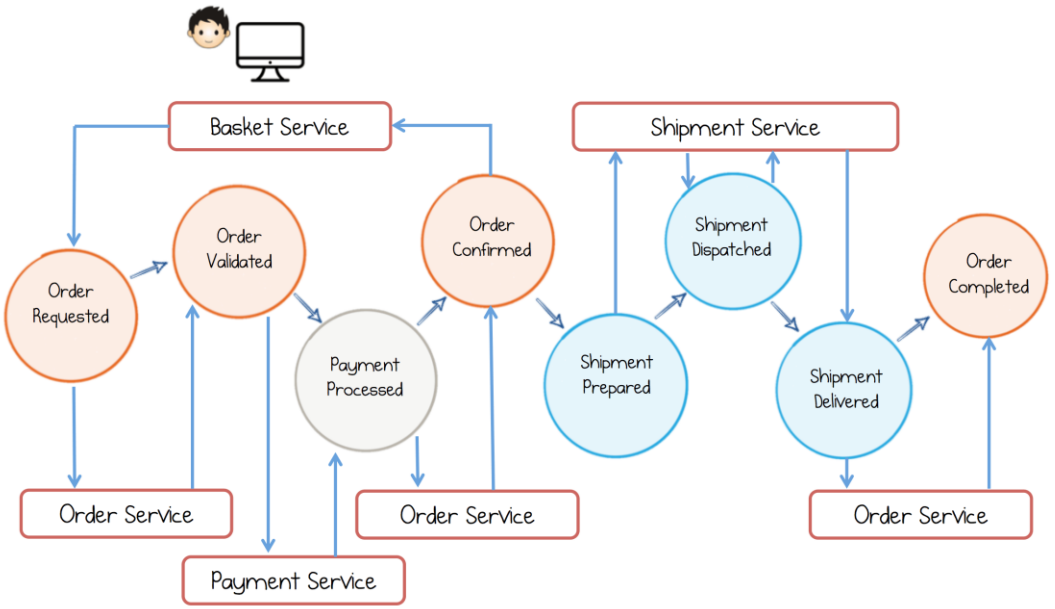
# Messaging and brokers



# Synchronous vs assynchronous

# Synchronous vs assynchronous


# “Do MVC like it’s 1979”

REMEMBER?

### Models

> *“Models represent knowledge. A model could be a single object (rather uninteresting), or it could be some structure of objects.” The original MVC report, 1979.*

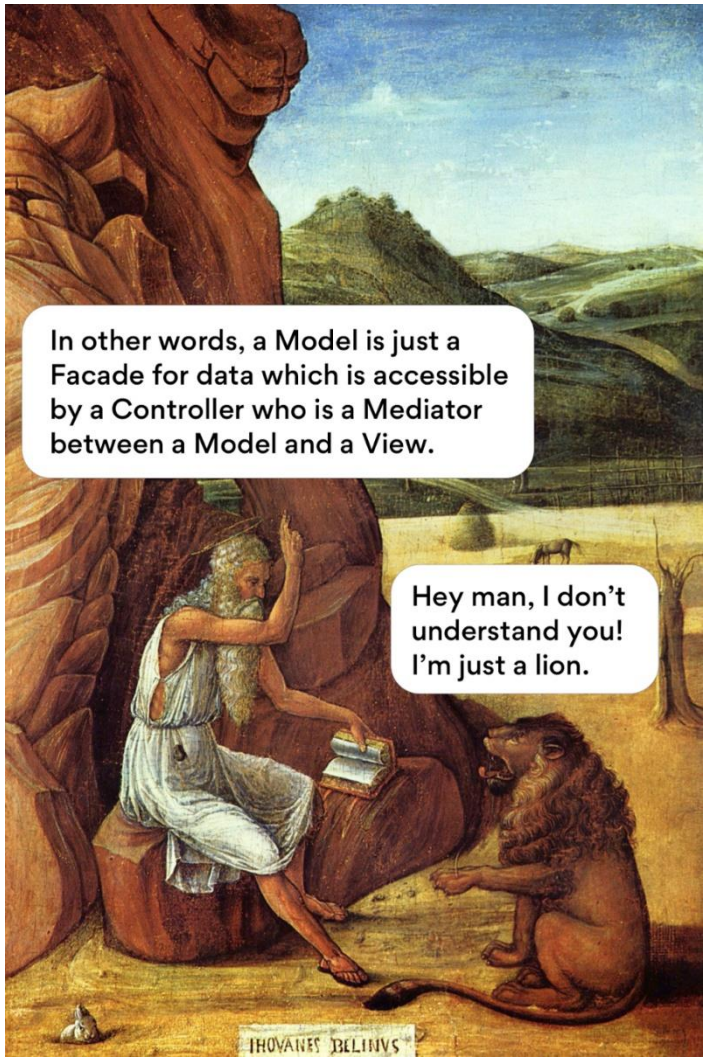
### Views

> *“A view is a (visual) representation of its model. It would ordinarily highlight certain attributes of the model and suppress others.” The original MVC report, 1979.*

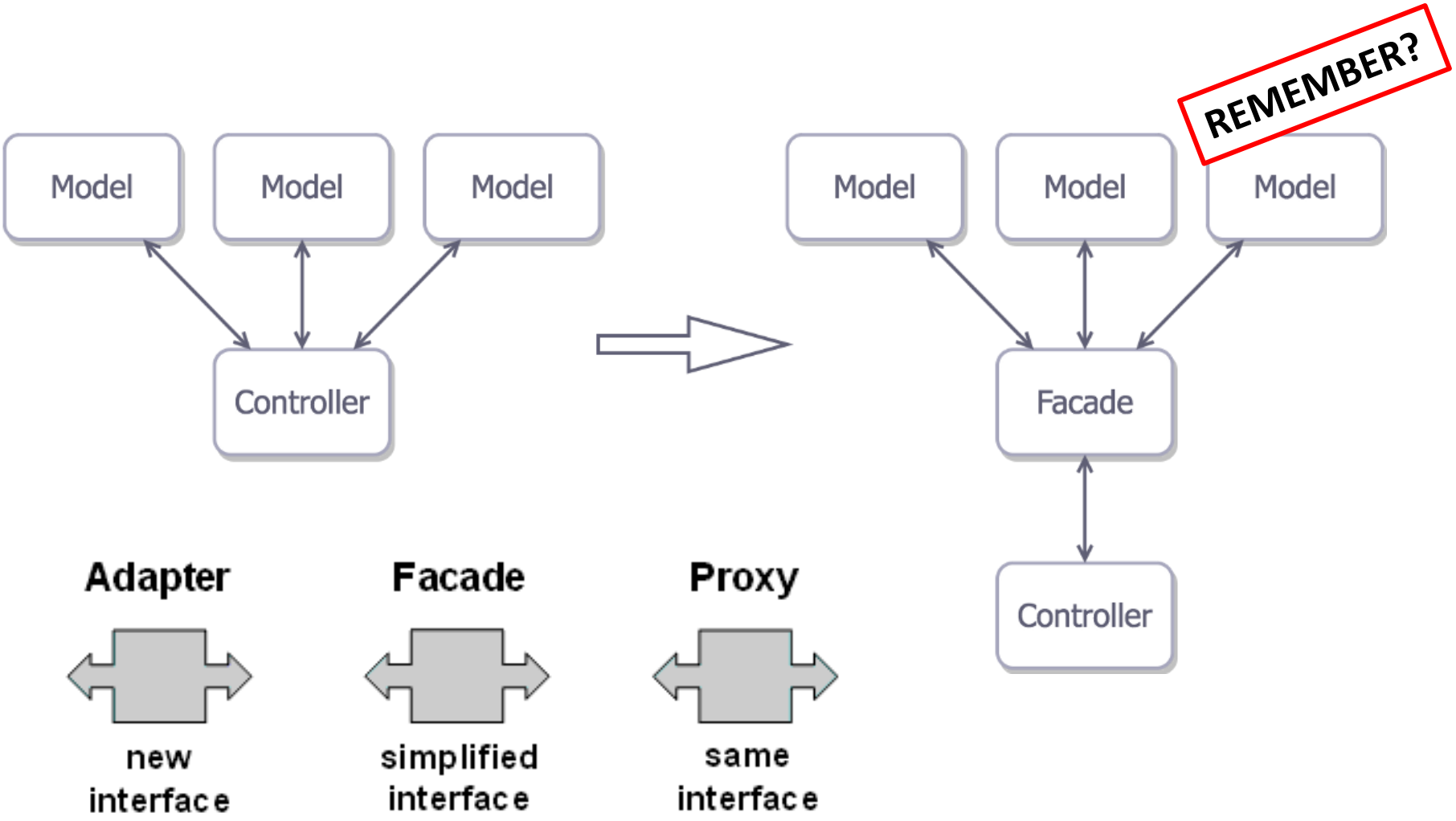
### Controllers

> *“A controller is a link between a user and the system. It provides the user with input by arranging for relevant views to present themselves in appropriate places on the screen.” The original MVC report, 1979.*

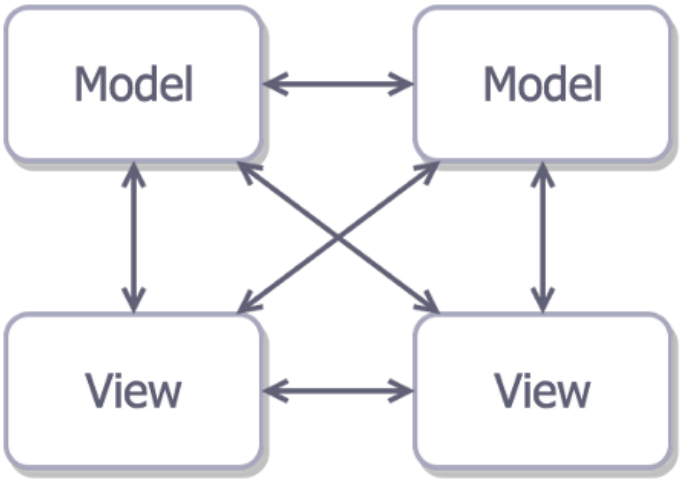
Note: Controllers were initially named Editors, and then the name was changed.

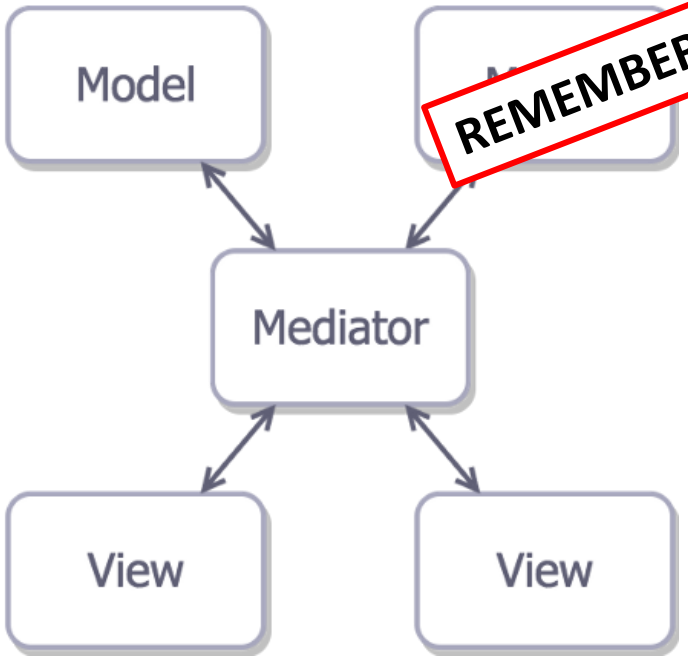


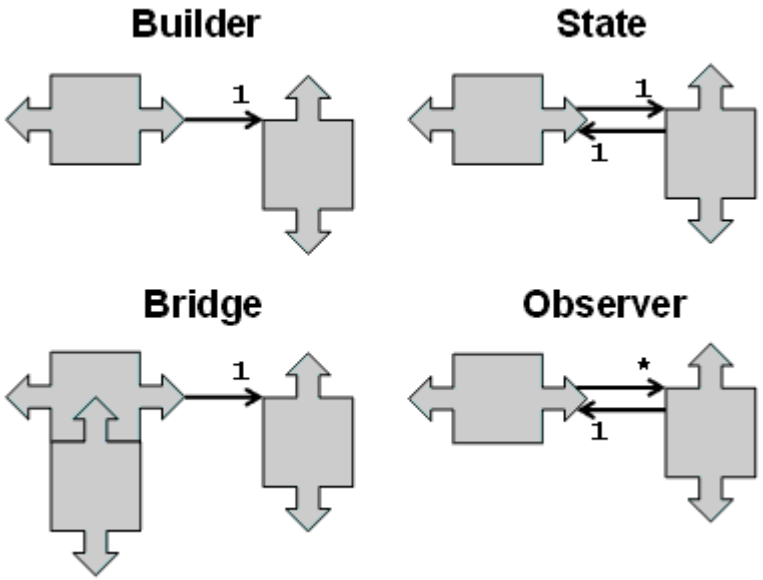

Do MVC like it’s 1979
https://badootech.badoo.com/do-mvc-like-its-1979-da62304f6568




Design Patterns by Vince Huston
http://www.vincehuston.org/dp/



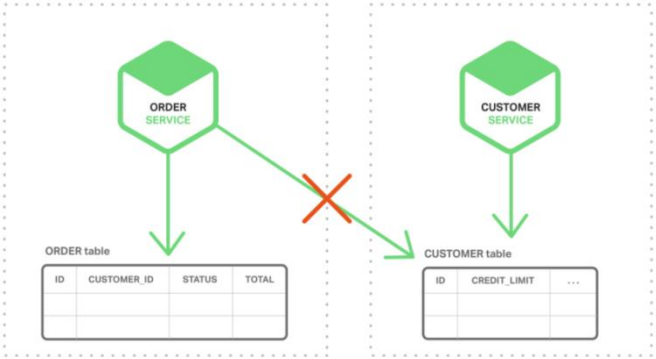



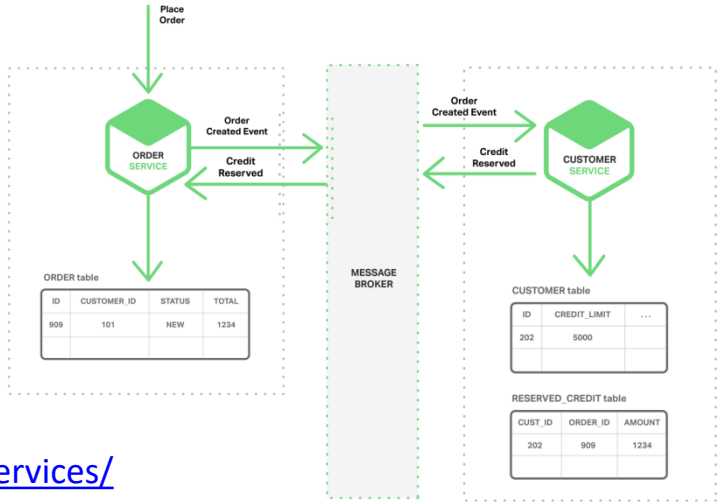




Design Patterns by Vince Huston
http://www.vincehuston.org/dp/

# Messaging

- Asynchronous
- Focus on message
- Reduce coupling
- Scalable & Fault tolerant



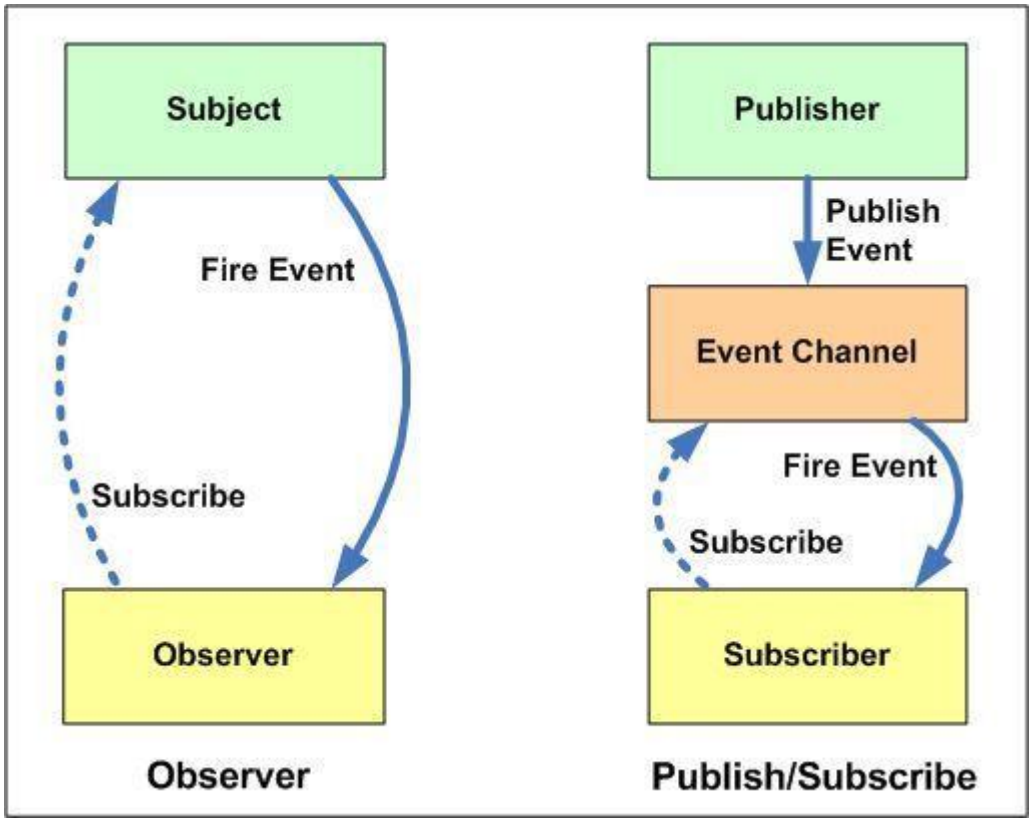




https://www.nginx.com/blog/event-driven-data-management-microservices/

# Messaging

- Conceptual
  - Reduce coupling
  - Focus on domain
  - Scalable
- Patterns and styles associated
  - Publisher / subscriber
  - Cqrs,  Event sourcing
  - Event Driven Architecture
  - …
- **Concept** **is technology agnostic**
- **BUT** commonly associated with **concrete implementations**

# Observer vs **Pub-Sub pattern**




Observer vs Pub-Sub pattern
https://hackernoon.com/observer-vs-pub-sub-pattern-50d3b27f838c

# Observer vs **Pub-Sub pattern**

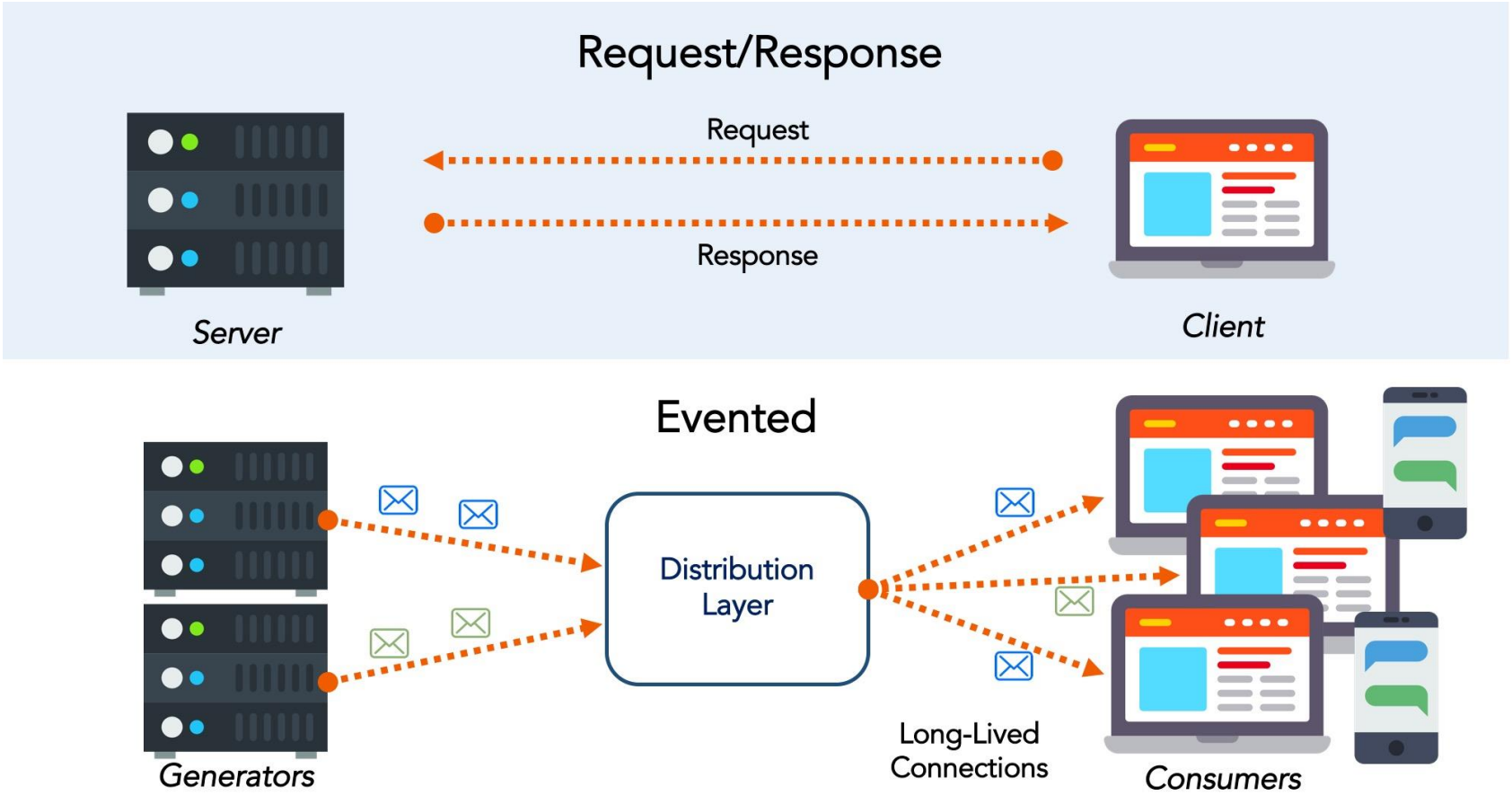
## Observer

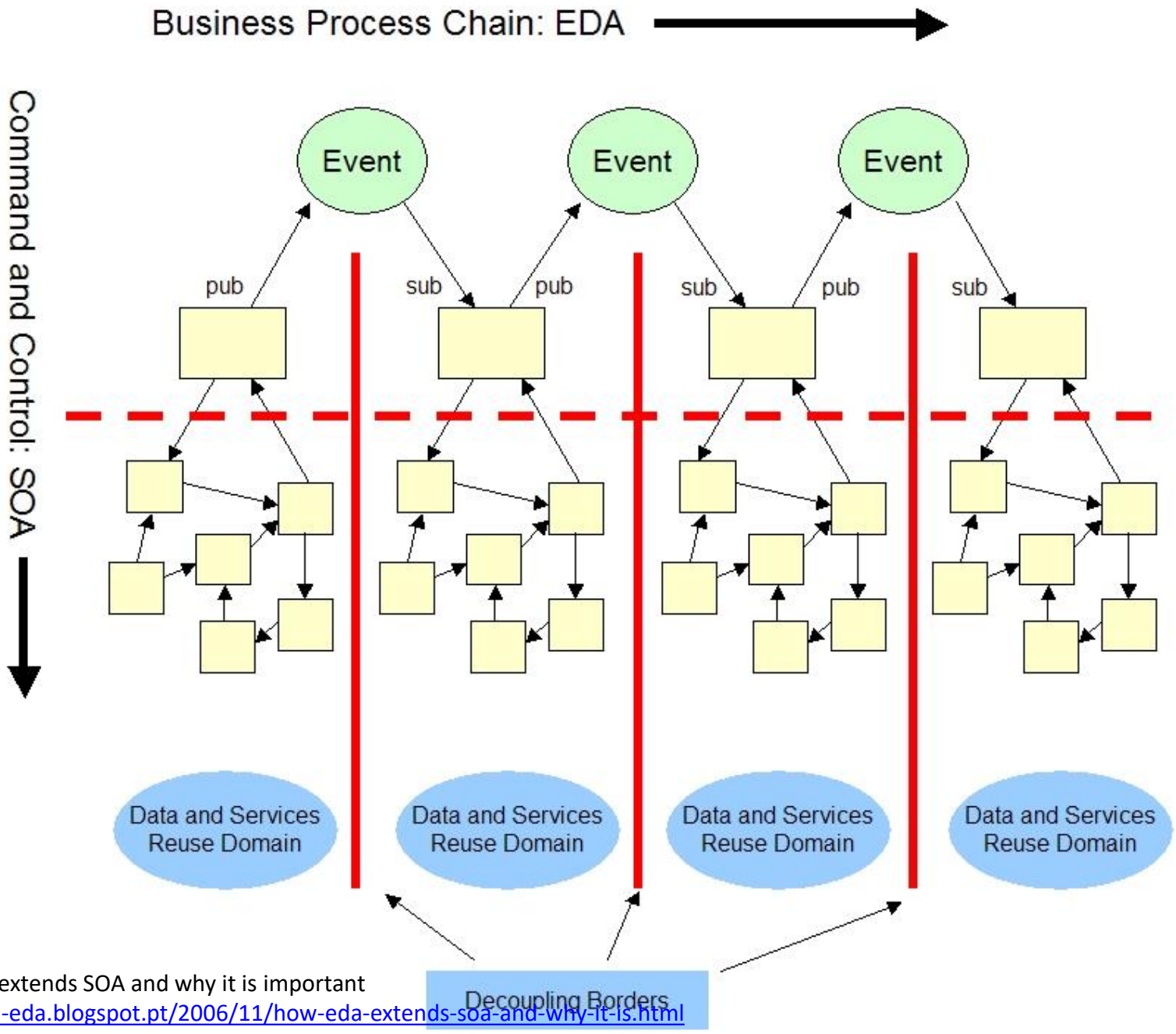
- O***bservers are aware of the Subject, also the Subject maintains a record of the Observers***.
- pattern components **are tightly coupled**
- mostly implemented in a ***synchronous*** way, i.e. the Subject calls the appropriate method of all its observers when some event occurs.
- pattern need to be implemented in a **single application address space.**

## Publisher/Subscriber,

- publishers and subscribers ***don't need to know each other***. They simply communicate with the help of message queues or broker.
- pattern, components are **loosely coupled**
- pattern is mostly implemented in an ***asynchronous*** way (using message queue).
- pattern is more of a **cross application pattern.**

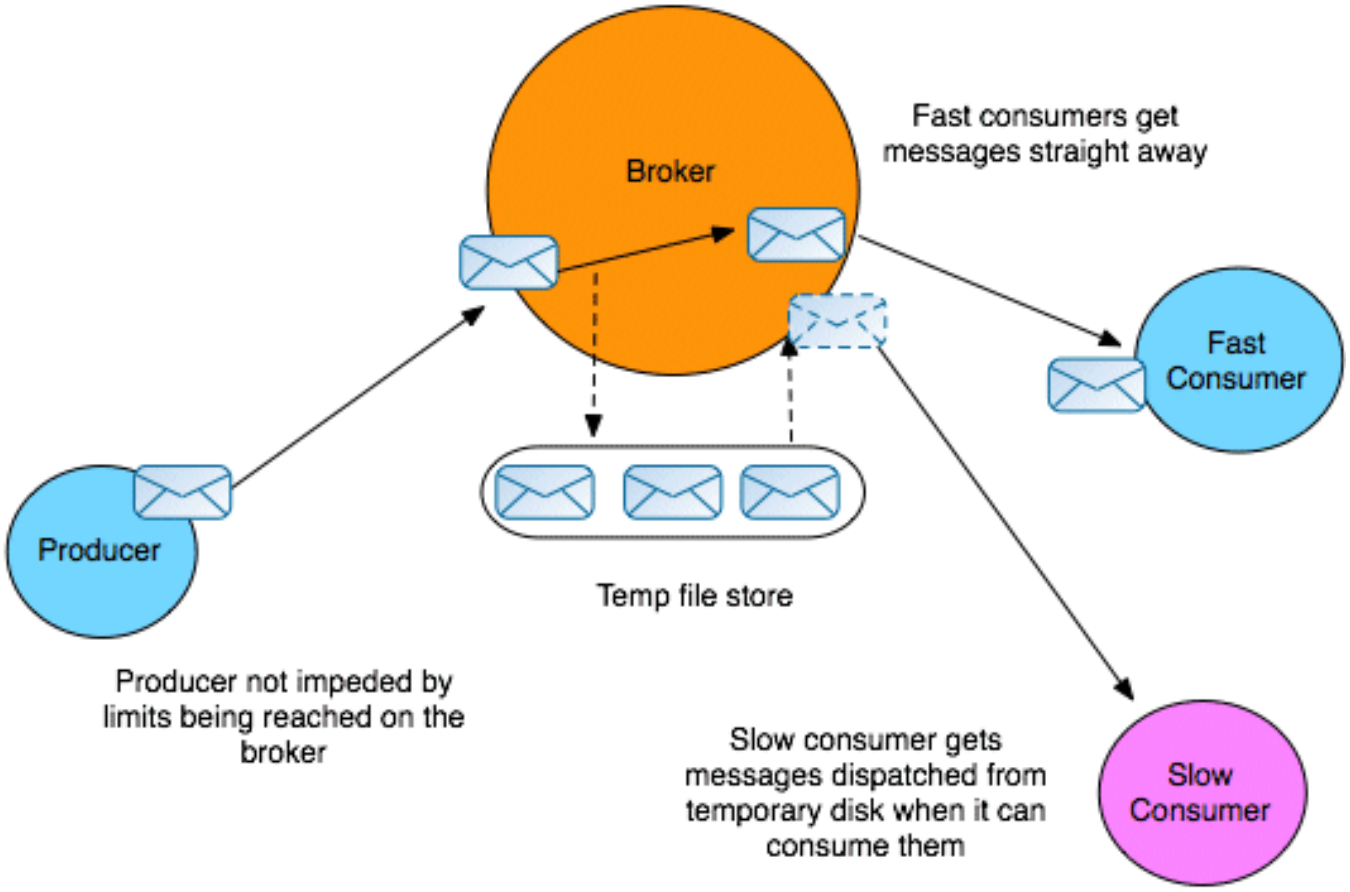
# Messaging vs Request / response






How EDA extends SOA and why it is important
http://soa-eda.blogspot.pt/2006/11/how-eda-extends-soa-and-why-it-is.html

# A solution is a broker

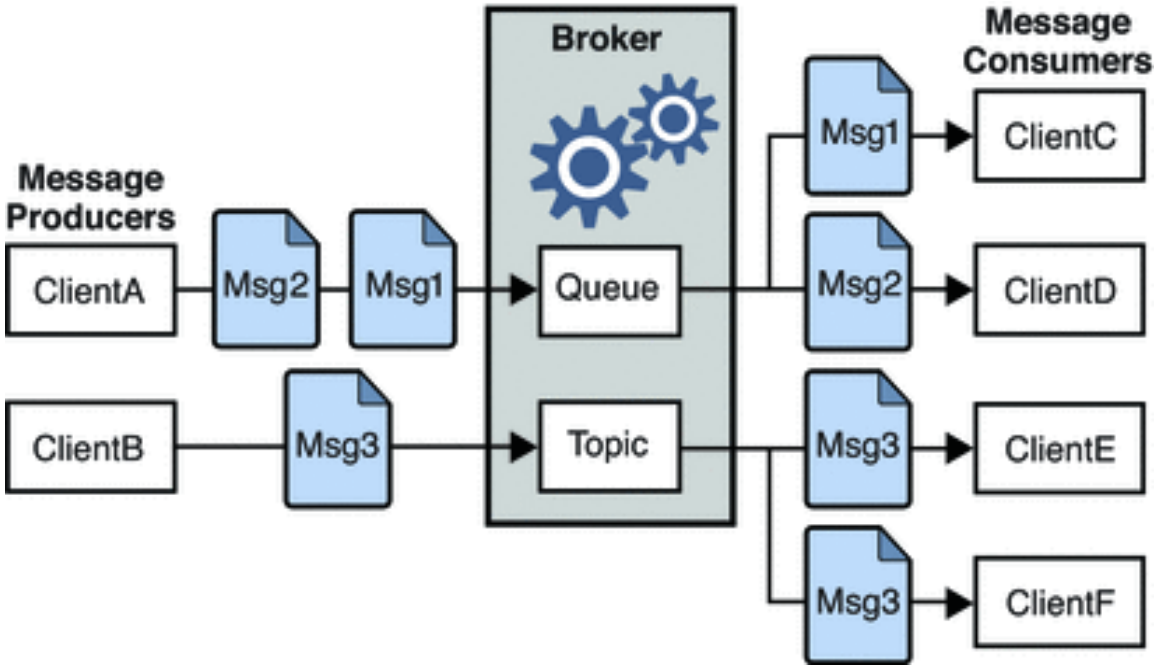



https://access.redhat.com/documentation/en-US/Fuse_ESB_Enterprise/7.0/html-single/ActiveMQ_Tuning_Guide/index.html

# A solution is a broker

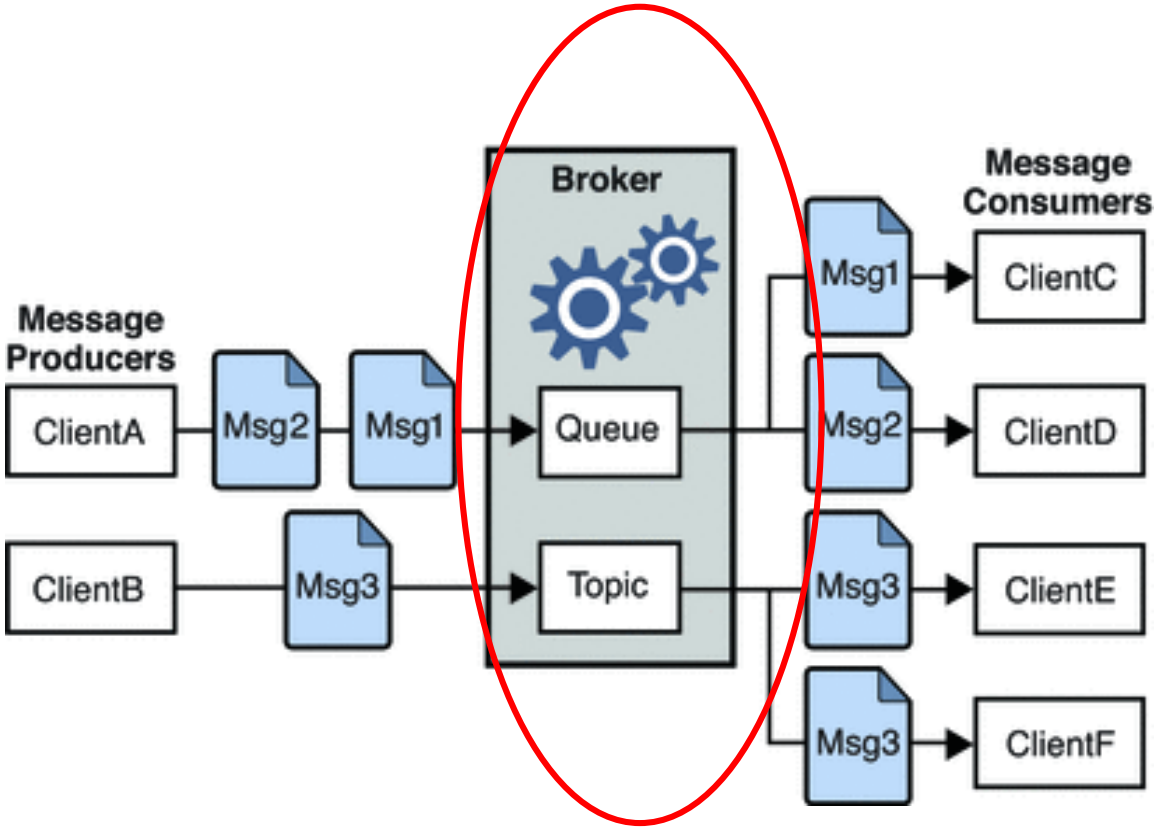
- Receives
  - Input
- Re routes
  - Transport
- Deliver
  - Finds

# Brokers, Messages, queues, topics …




http://docs.oracle.com/cd/E19717-01/819-7759/aeras/index.html

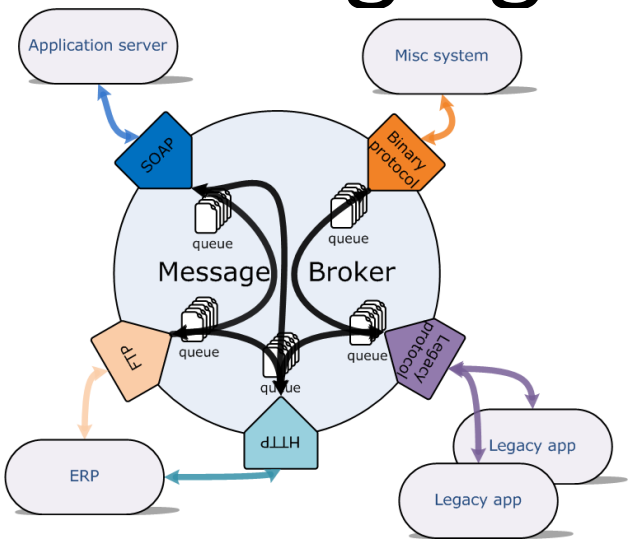
# Brokers, Messages, queues, topics …

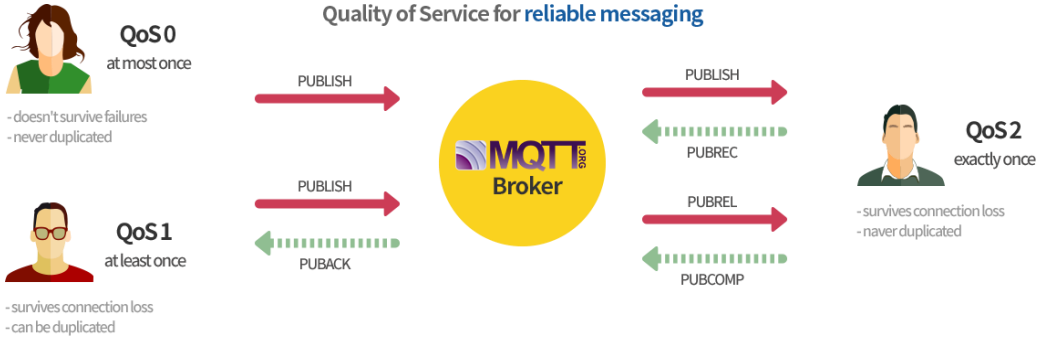


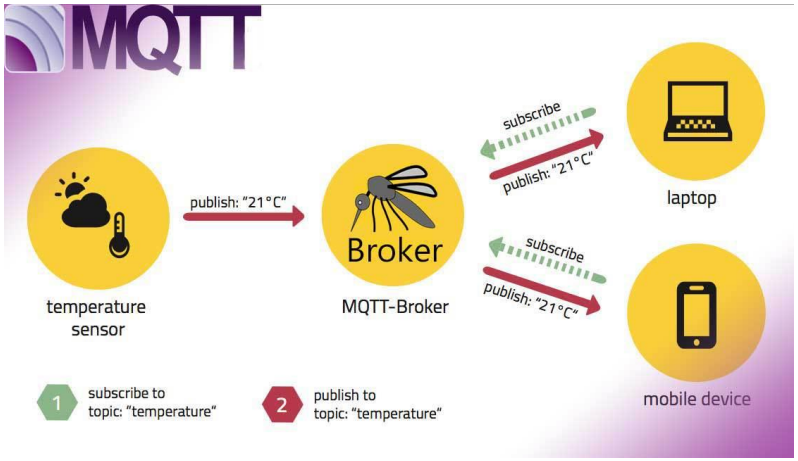

Receives and reroutes

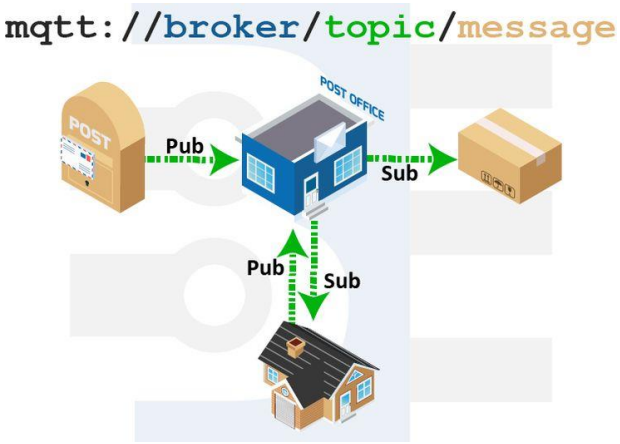
http://docs.oracle.com/cd/E19717-01/819-7759/aeras/index.html

# Messaging as a integration solution



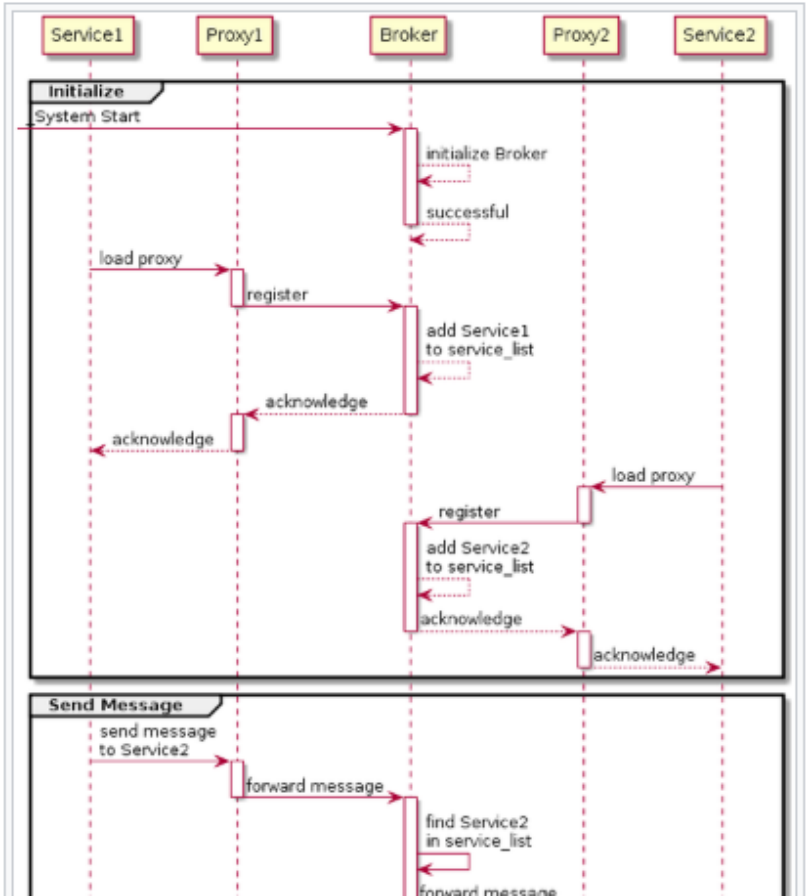



# Message broker

From Wikipedia, the free encyclopedia

A **message broker** (also known as an **integration broker** or **interface engine**[1]) is an intermediary computer program module that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver. Message brokers are elements in telecommunication or computer networks where software applications communicate by exchanging formally-defined messages.[1] Message brokers are a building block of message-oriented middleware (MOM) but are typically not a replacement for traditional middleware like MOM and remote procedure call (RPC).[2][3]




**Contents** [hide]



## Purpose, functionality, and architecture [ edit ]

A message broker is an architectural pattern for message validation

https://en.wikipedia.org/wiki/Message_broker

minimizing the mutual awareness that applications should have of each

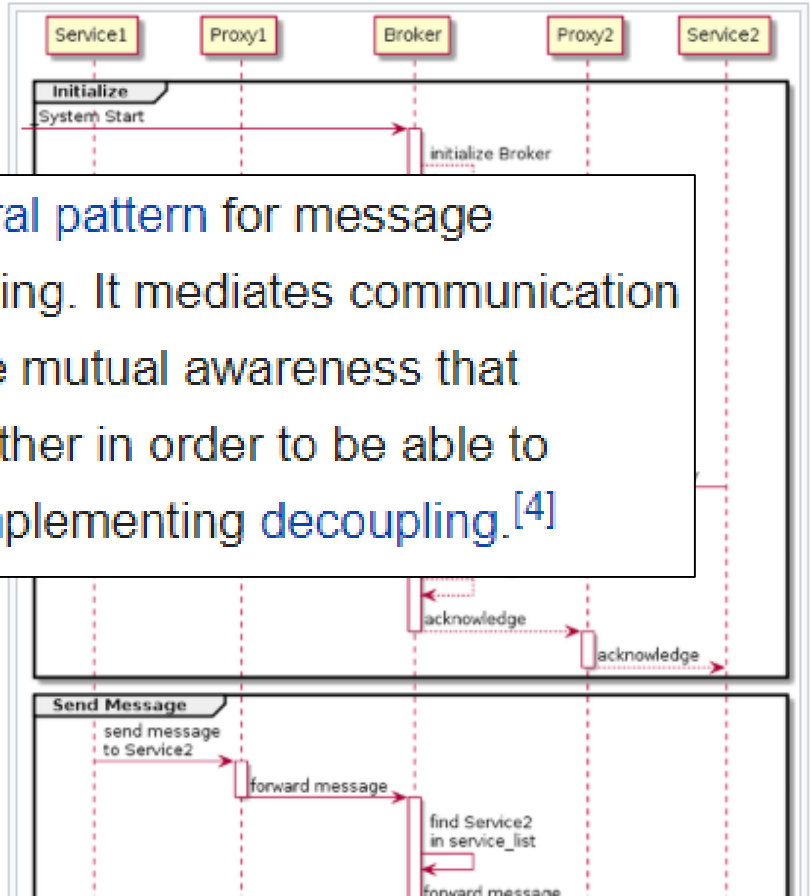
# Message broker

From Wikipedia, the free encyclopedia

A **message broker** (also known as an **integration broker** or **interface engine**[1]) is an intermediary computer program module that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver. Message brokers are elements in telecommu communic brokers ar typically n procedure



A message broker is an architectural pattern for message validation, transformation, and routing. It mediates communication among applications, minimizing the mutual awareness that applications should have of each other in order to be able to exchange messages, effectively implementing decoupling.[4]



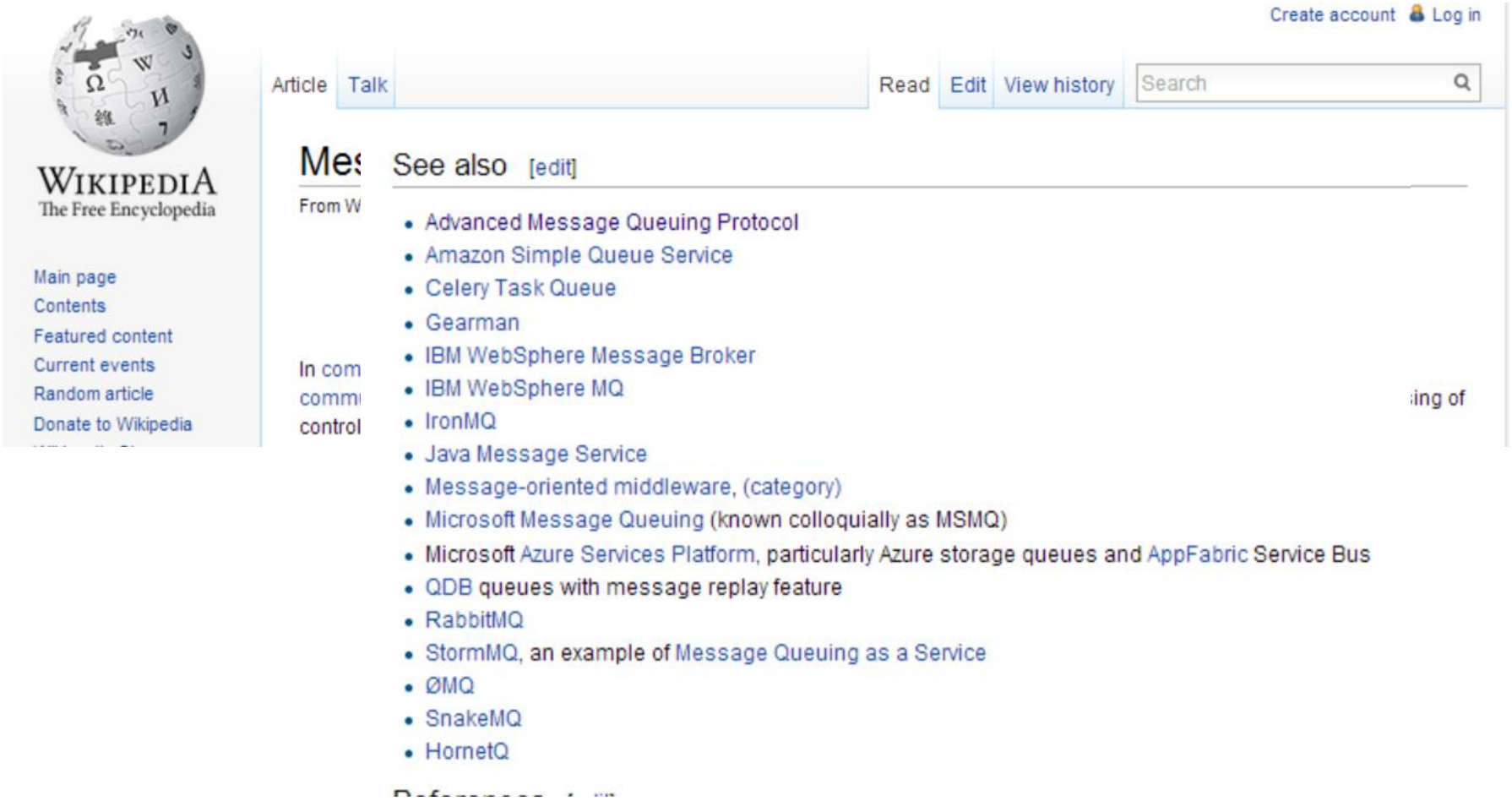

## Purpose, functionality, and architecture [ edit ]

A message broker is an architectural pattern for message validation

https://en.wikipedia.org/wiki/Message_broker

minimizing the mutual awareness that applications should have of each

# Messaging and protocols…




/
http://en.wikipedia.org/wiki/Message_queue
http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol

# Messaging and protocols

- AMQP, which stands for Advanced Message Queuing Protocol
- MQTT (Message Queue Telemetry Transport)
- STOMP (Simple/Streaming Text Oriented Messaging Protocol)
- CoAP (Constrained Application Protocol)

# IoT Protocol Wars: MQTT vs CoAP vs …

**MQTT**

- Field devices with cellular or satellite backhaul – every Kb matters, traffic is expensive.
- Two-way communications over unreliable networks.
- Battery powered devices with low power consumption.
- Devices may sleep, but not 95% of the time. Otherwise please see MQTT-S or CoAP.
- NAT traversal to be addressed as an afterthought – important, but not critical.

**MQTT-S (over UDP)**

- Pretty much the same as MQTT, but with really sleepy devices (sensor networks).
- Potential to scale 10x times more devices comparing to MQTT – question about UDP vs TCP scalability.
- NAT traversal might become a larger issue here comparing to MQTT. So must be addressed during the planning stage.

**CoAP**

- Pretty similar to MQTT-S choice, but with several additions.
- Web services oriented architecture, rather than messaging oriented.
- When you're building a platform for the open development communities in the internet space.

**REST API**

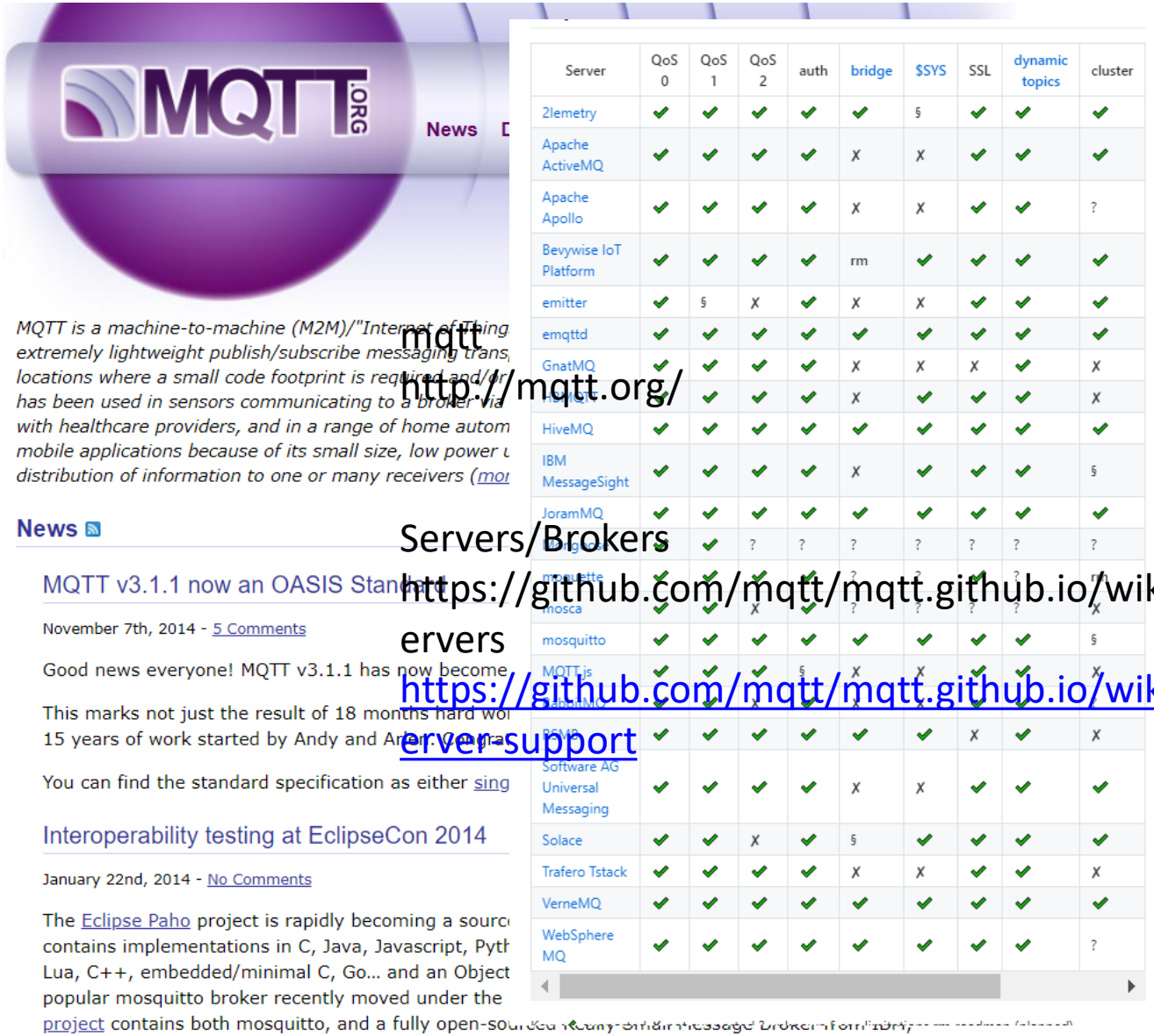
- One-way communications from devices to the cloud.
- NAT traversal around the globe is among the top priorities.

http://www.iotprimer.com/2013/11/iot-protocol-wars-mqtt-vs-coap-vs-xmpp.html

# MQTT: just a mention

- *"MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely **lightweight publish/subscribe messaging** transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium"*
- *http://mqtt.org/*

- *Several brokers, low fingerprint*
  - *Mosquitto most popular*
  - *Popular in IoT*
- *Several drivers & integrations*
  - ActiveMQ , RabbitMQ, Spring, …




mqtt

http://mqtt.org/

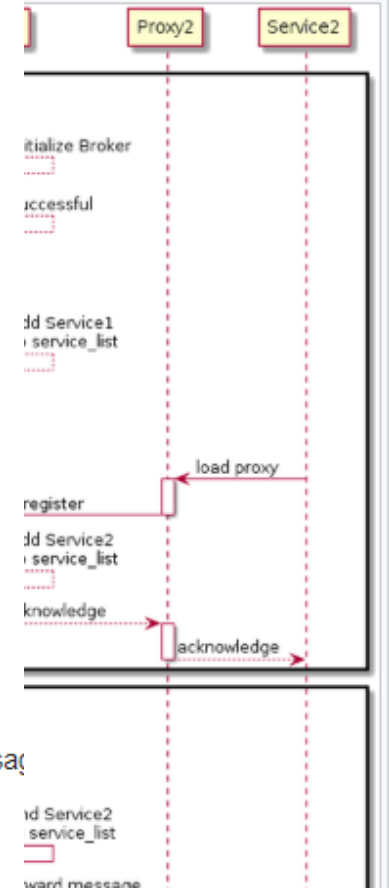
Servers/Brokers

https://github.com/mqtt/mqtt.github.io/wiki/servers

https://github.com/mqtt/mqtt.github.io/wiki/server-support

## List of message broker software [ edit ]

- Amazon Web Services (AWS) Simple Queue Service (SQS)
- Apache ActiveMQ
- Apache Kafka
- Apache Qpid
- Celery
- Cloverleaf (E-Novation Lifeline)
- Comverse Message Broker (Comverse Technology)
- Enduro/X Transactional Message Queue (TMQ)
- Financial Fusion Message Broker (Sybase)
- Fuse Message Broker (enterprise ActiveMQ)
- Gearman
- HornetQ (Red Hat)
- IBM Integration Bus
- IBM Message Queues / IBM WebSphere MQ
- JBoss Messaging (JBoss)
- JORAM
- Microsoft Azure Service Bus (Microsoft)
- Microsoft BizTalk Server (Microsoft)
- NATS (MIT Open Source License, written in Go)
- Open Message Queue
- Oracle Message Broker (Oracle Corporation)
- RabbitMQ (Mozilla Public License, written in Erlang)
- Redis An open source, in-memory data structure store, used as a database, cache and messag
- SAP PI (SAP AG)
- Solace PubSub+




https://en

# Messaging brokers






# Messaging Protocols & clients

| | ActiveMQ | Apollo | HornetQ | Qpid | RabbitMQ | ZeroMQ |
|---|---|---|---|---|---|---|
| AMQP | 1.0 | 1.0 | announced | 1.0 | 0-8, 0-9, 0-9-1 | - |
| MQTT | ✓ | ✓ | - | - | ✓ | - |
| OpenWire | ✓ | ✓ | - | - | - | - |
| REST | ✓ | ✓ | ✓ | - | ✓ | - |
| STOMP | ✓ | ✓ | ✓ | - | ✓ | - |
| STOMP over Websockets | ✓ | ✓ | ✓ | - | ✓ | - |
| XMPP | ✓ | - | - | - | Over Gateway | - |

Table 1: Support for Messaging Protocols

ActiveMQ, Qpid, HornetQ and RabbitMQ in Comparison
https://www.predic8.com/activemq-hornetq-rabbitmq-apollo-qpid-comparison.htm

# Messaging Protocols & clients

| | ActiveMQ | Apollo | HornetQ | Qpid | RabbitMQ | ZeroMQ |
|---|---|---|---|---|---|---|
| AMQP | 1.0 | 1.0 | announced | 1.0 | 0-8, 0-9, 0-9-1 | - |
| MQTT | ✓ | ✓ | - | - | ✓ | - |
| OpenWire | | | | | | |
| REST | | | | | | |
| STOMP | | | | | | |
| STOMP ov | | | | | | |
| XMPP | | | | | | |

Table 1: Su

| | ActiveMQ | Apollo | HornetQ | Qpid | RabbitMQ | ZeroQ |
|---|---|---|---|---|---|---|
| C | ✓ | - | - | ✓ | ✓ | ✓ |
| C++ | - | - | - | ✓ | ✓ | ✓ |
| Erlang | - | - | - | - | ✓ | ✓ |
| Haskell | - | - | - | - | ✓ | ✓ |
| Java JMS | ✓ | - | ✓ | ✓ | - | - |
| Java proprietary | ✓ | - | ✓ | - | ✓ | ✓ |
| .NET | - | - | - | ✓ | ✓ | ✓ |
| Objective-C | - | - | - | - | - | ✓ |
| Perl | - | - | - | - | ✓ | ✓ |
| PHP | - | - | - | - | ✓ | ✓ |
| Python | - | - | - | ✓ | ✓ | ✓ |
| Ruby | - | - | - | ✓ | ✓ | ✓ |

ActiveMQ, Qpid, HornetQ and RabbitMQ in Comparison
https://www.predic8.com/activemq-hornetq-rabbitmq-apollo-qpid-comparison.htm

# Some examples: **just** wildfly


| DISTRIBUTION | BROKER | VERSION | PROTOCOL SUPPORT |
|---|---|---|---|
| Artemis standalone | Artemis | 1.5 | AMQP, STOMP, MQTT, OpenWire, HornetQ Core |
| ActiveMQ 5 standalone | ActiveMQ 5 | 5.14 | AMQP, STOMP, MQTT, OpenWire |
| WildFly 8, 9 | HornetQ | 2.4 | AMQP, STOMP, HornetQ Core |
| JBoss EAP 6.4 | HornetQ | 2.3 | AMQP, STOMP, HornetQ Core |
| WildFly 10 | Artemis | 1.1 | AMQP, STOMP, HornetQ Core |
| JBoss EAP 7 | Artemis | 1.1 | HornetQ Core |

https://blog.akquinet.de/2017/02/22/activemq-confusion-and-what-comes-with-your-jboss-eap-wildfly/

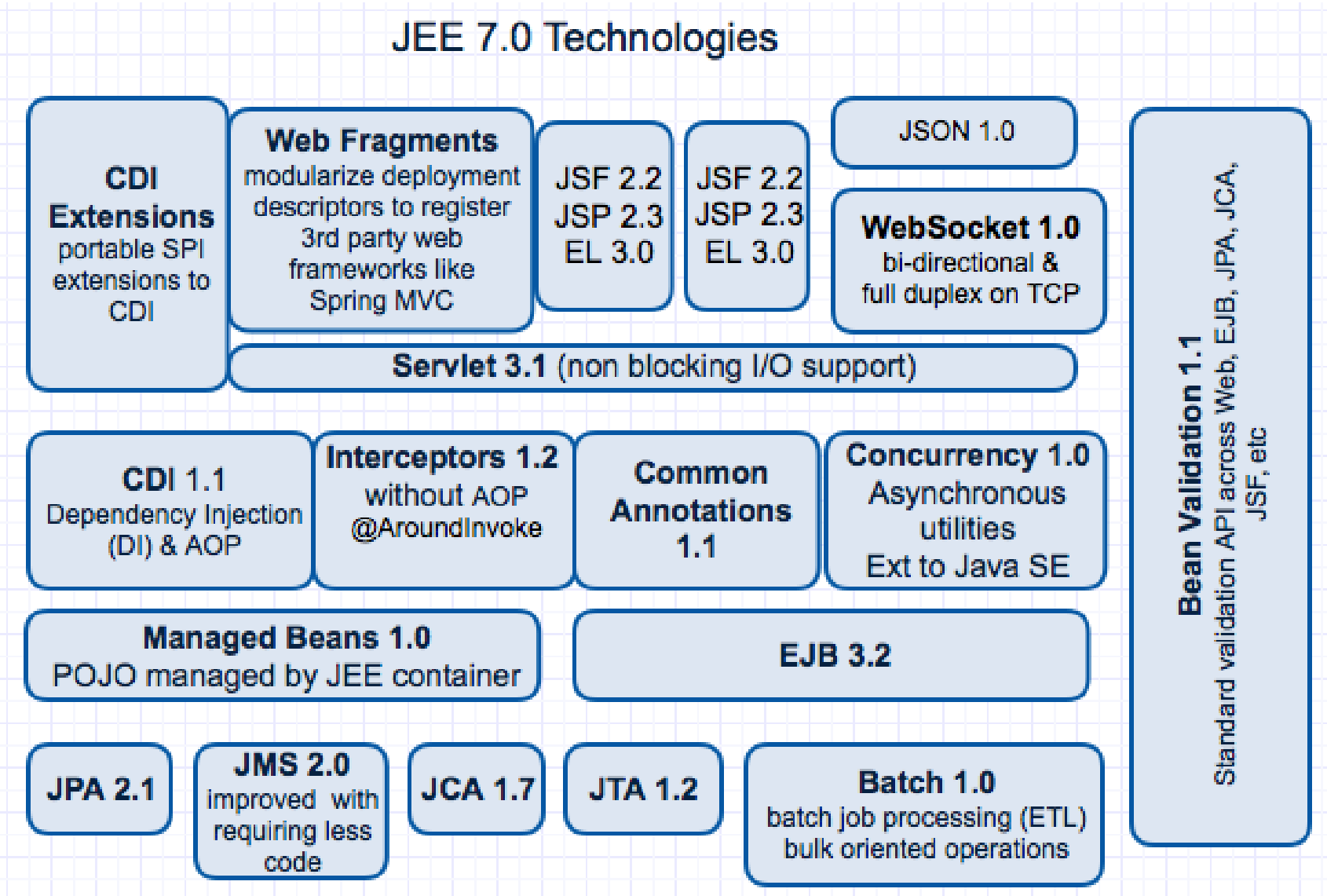
# Brokered solutions

- Several options
  - **EAI and SOA platforms**, such as IBM WebSphere MQ, TIBCO, Vitria, Oracle Service Bus, WebMethods (now Software AG), Microsoft BizTalk, or Fiorano.
  - **Open source ESB's** like Mule ESB, JBoss Fuse, Open ESB, WSo2, Spring Integration, or Talend ESB
  - **Message Brokers** like ActiveMQ, Apache Kafka, or RabbitMQ
  - **Web service- or REST-based integration**, including Amazon Simple Queue Service (SQS) or Google Cloud Pub/Sub
  - **JMS-based messaging systems**
  - **Microsoft technologies** like MSMQ or Windows Communication Foundation (WCF)

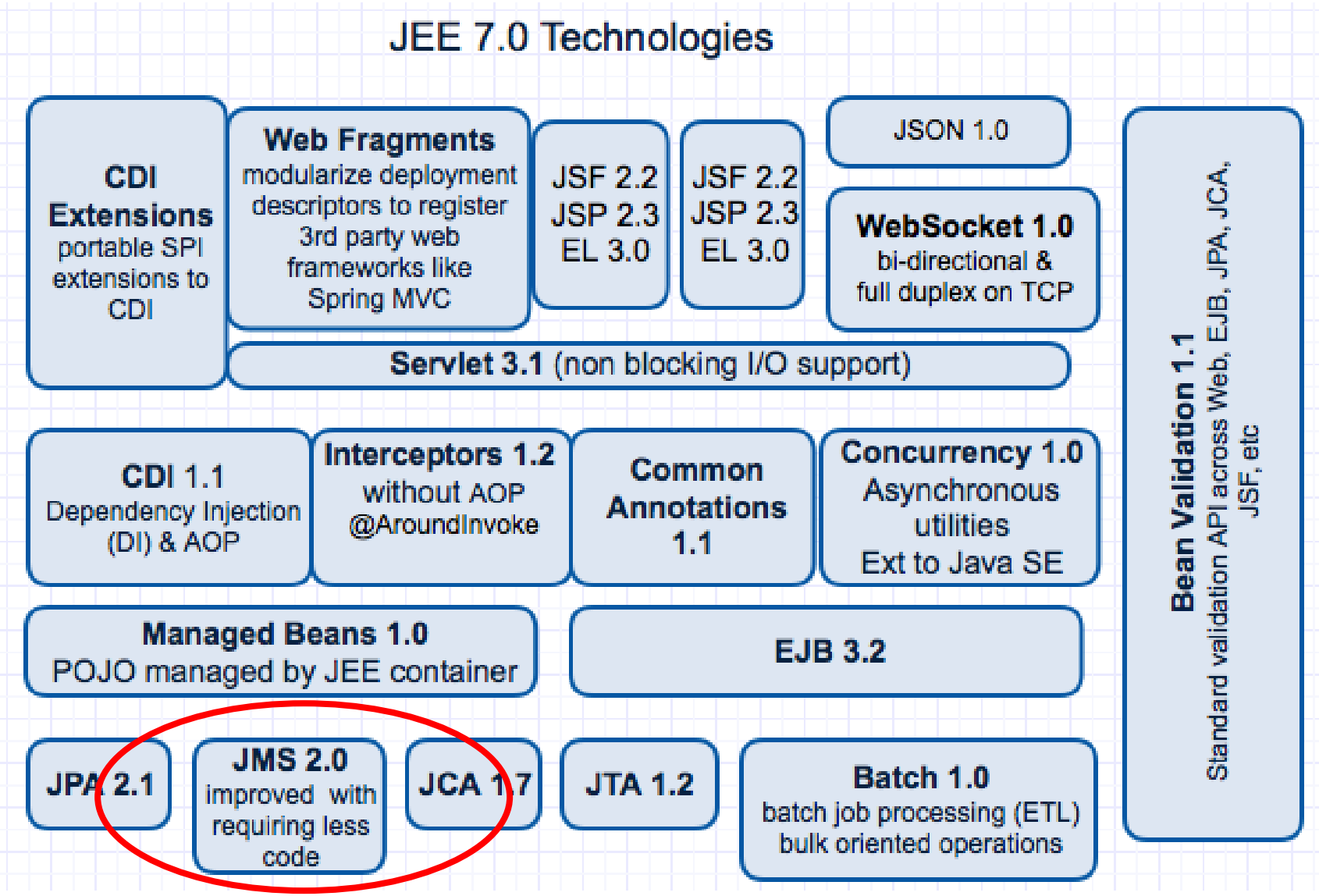
http://www.enterpriseintegrationpatterns.com/patterns/messaging/index.html

# Java Messaging system (JMS): a high level messaging endpoint

Jfernan@ua.pt




❤ ♦ 8 JEE (i.e. Java EE) Overview interview questions & answers
https://www.java-success.com/jee-overview-interview-questions-and-answers/




❤ ♦ 8 JEE (i.e. Java EE) Overview interview questions & answers
https://www.java-success.com/jee-overview-interview-questions-and-answers/

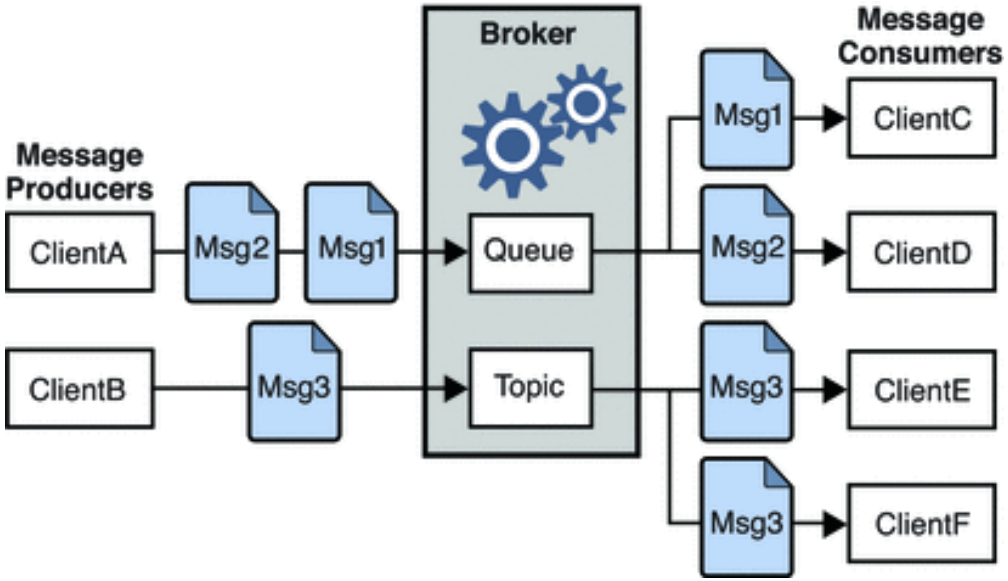
# Java Messaging Service

## Java Message Service

From Wikipedia, the free encyclopedia

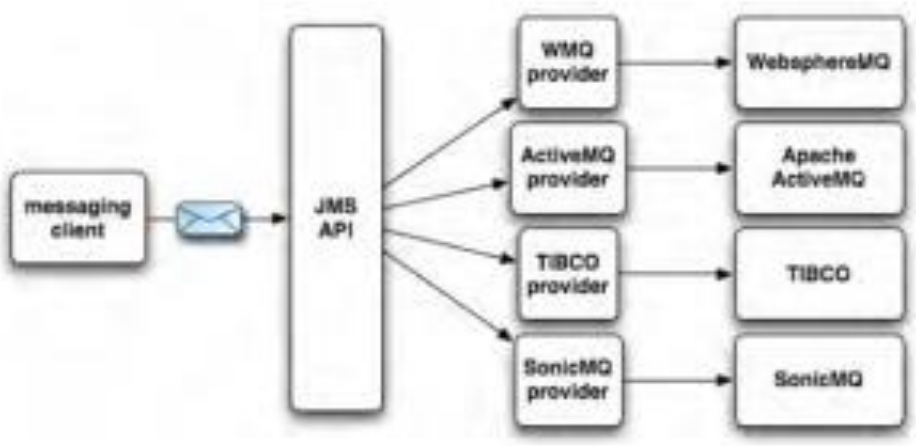
The **Java Message Service** (**JMS**) API is a Java Message Oriented Middleware (MOM) API[1] for sending messages between two or more clients. JMS is a part of the Java Platform, Enterprise Edition, and is defined by a specification developed under the Java Community Process as JSR 914.[2] It is a messaging standard that allows application components based on the Java Enterprise Edition (Java EE) to create, send, receive, and read messages. It allows the communication between different components of a *distributed application* to be loosely coupled, reliable, and asynchronous.[3]

Contents [hide]



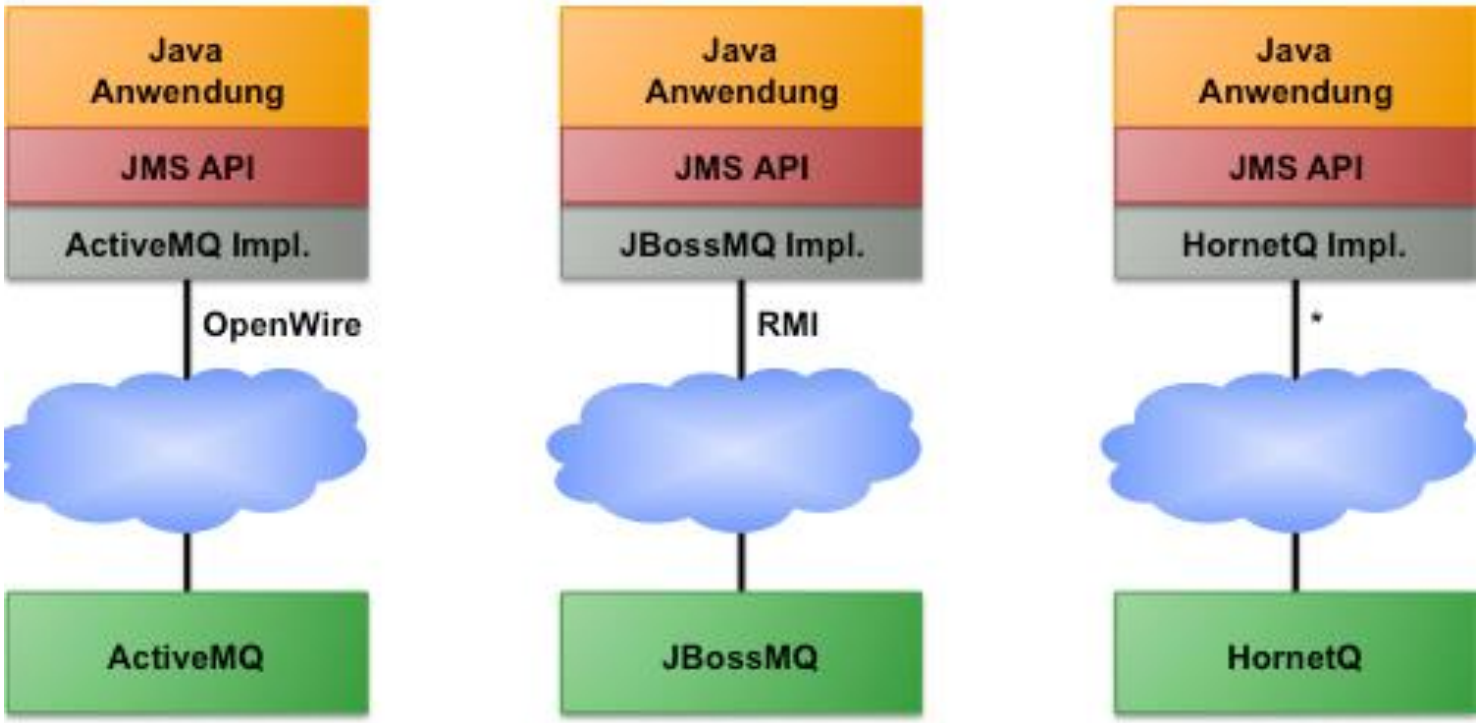
# JMS: middleware with concrete roles

- **JMS client** —
  - An application using Java to send and receive messages.
- **Non-JMS client** —
  - An application a JMS provider's native API to send and receive messages
- **JMS producer** —
  - A client application that creates and sends JMS messages.
- **JMS consumer**—
  - A client application that receives and processes JMS messages.
- **JMS provider**—
  - The implementation of the JMS interfaces,.
- **JMS message**—
  - The most fundamental concept of JMS;
- **JMS domains**—
  - include point-to-point and publish/subscribe.
- Others…




https://anwaarlabs.wordpress.com/2014/04/28/message-queue-part-2-jms/

# JMS




https://www.predic8.com/activemq-hornetq-rabbitmq-apollo-qpid-comparison.htm

# Java Messaging Service

Java M

From Wikipedia

The **Java Me**
part of the Ja
standard that
communicati

Conte

## Provider implementations [edit]

To use JMS, one must have a JMS provider that can manage the sessions and queues. Starting from Java EE version 1.4, JMS provider has to be contai
*all* Java EE application servers. This can be implemented using the message inflow management of the Java EE Connector Architecture, which was firs
made available in that version.

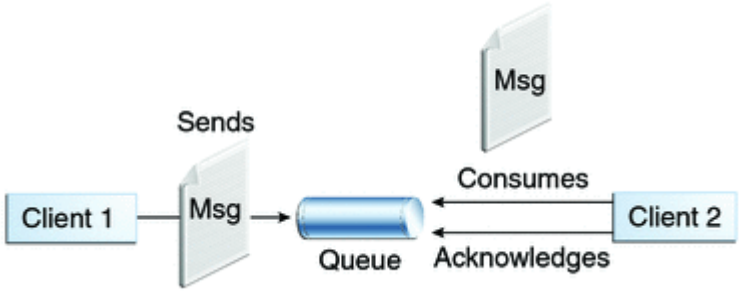
The following is a list of JMS providers:

- Apache ActiveMQ
- Apache Qpid, using AMQP[5]
- Oracle Weblogic (part of the Fusion Middleware suite) and Oracle AQ from Oracle
- EMS from TIBCO
- FFMQ, GNU LGPL licensed
- JBoss Messaging and HornetQ from JBoss
- JORAM, from the OW2 Consortium
- Open Message Queue, from Oracle
- OpenJMS, from The OpenJMS Group
- Solace JMS from Solace Systems
- RabbitMQ by Rabbit Technologies Ltd., acquired by SpringSource
- SAP Process Integration ESB
- SonicMQ from Progress Software
- SwiftMQ
- Tervela
- Ultra Messaging from 29 West (acquired by Informatica)
- webMethods from Software AG
- WebSphere Application Server from IBM, which provides an inbuilt default messaging provider known as the Service Integration Bus (SIBus), or which connect to WebSphere MQ as a JMS provider[6]
- WebSphere MQ (formerly MQSeries) from IBM
- FioranoMQ

A historical comparison matrix of JMS providers from 2005 is available at http://www.theserverside.com/reviews/matrix.tss
http://javakeeexample.blogspot.in/2012/12/integrating-spring-with-jms_19.html

# JMS supports basic patterns

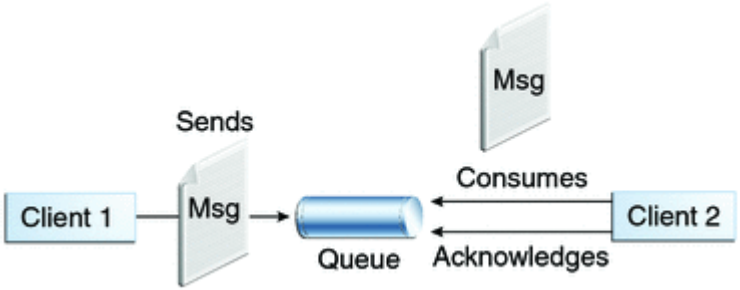
- **Point-to-Point Messaging Domain**




- **Publish/Subscribe Messaging Domain**

# Getting the JMS resources

- **Point-to-Point Messaging Domain**



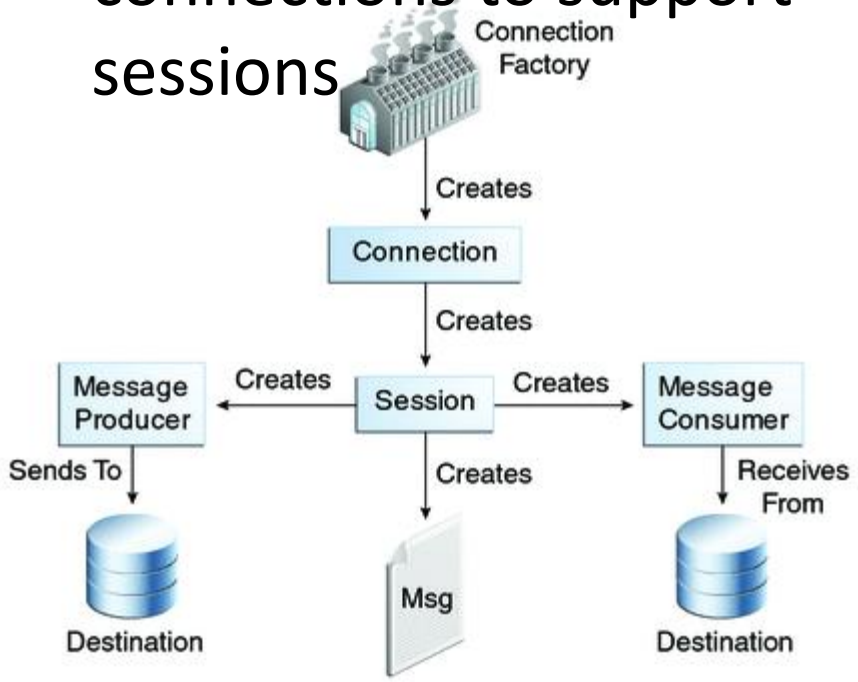

- **Publish/Subscribe Messaging Domain**


```
@Resource(lookup = "jms/ConnectionFactory")
private static ConnectionFactory connectionFactory;

@Resource(lookup = "jms/Queue")
private static Queue queue;
```

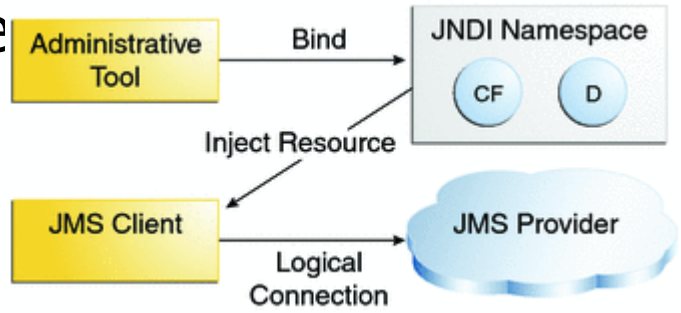
```
@Resource(lookup = "jms/ConnectionFactory")
private ConnectionFactory connectionFactory;

@Resource(lookup = "jms/Topic")
private Topic topic;
```

# JMS hides connection managment

- The factory creates connections to support sessions

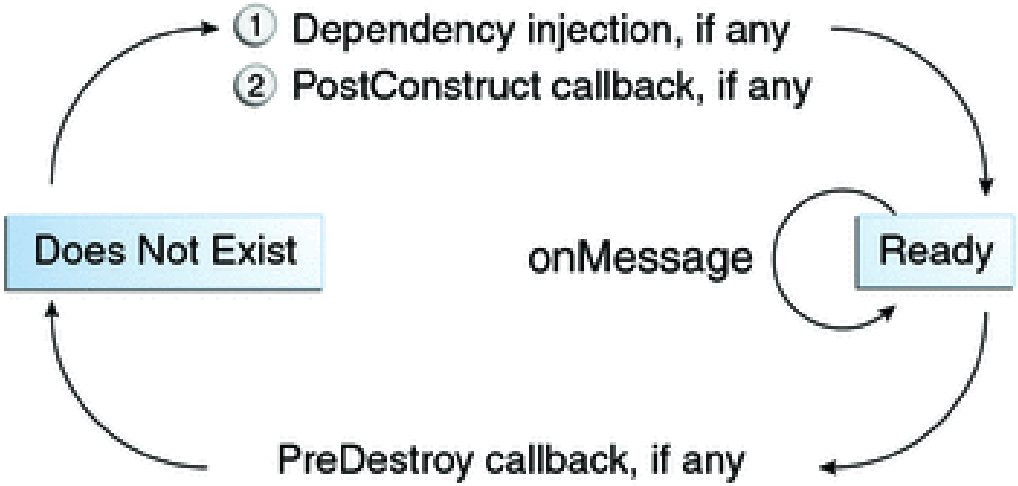



- JMS binding are kept in a JNDI namespace to help JMS clients find the JMS provider me



# Message Driven Bean (MDB)

- Hides the complexity
- Handles to listen to the Java Message Service API



# MDB: A simple example

- The application must perform:
  - Creating a connection and a session
  - Creating message producers and consumers
  - Sending and receiving messages

# A Message Driven Bean (MDB)...

```java
package test;

import java.util.logging.Logger;
import javax.ejb.ActivationConfigProperty;
import javax.ejb.MessageDriven;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;

@MessageDriven(mappedName="testQueue2",
     activationConfig = { @ActivationConfigProperty(
     propertyName="destinationType", propertyValue="javax.jms.Queue")})
public class MessageBean implements MessageListener {
 Logger logger = Logger.getLogger("test");

 public void onMessage(Message msg) {
     try {
         String name = msg.getStringProperty("name");
         logger.info("Received msg " + msg + ", from " + name);
     } catch (JMSException e) {
         throw new RuntimeException(e);
     }
 }
}
```

From http://javahowto.blogspot.pt/2010/04/message-driven-bean-example-with.html

# Is like a Server managed callback

```java
package test;

import java.util.logging.Logger;
import javax.ejb.ActivationConfigProperty;
import javax.ejb.MessageDriven;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;

@MessageDriven(mappedName="testQueue2",
     activationConfig = { @ActivationConfigProperty(
     propertyName="destinationType", propertyValue="javax.jms.Queue")})
public class MessageBean implements MessageListener {
 Logger logger = Logger.getLogger("test");

 public void onMessage(Message msg) {
     try {
         String name = msg.getStringProperty("name");
         logger.info("Received msg " + msg + ", from " + name);
     } catch (JMSException e) {
         throw new RuntimeException(e);
     }
 }
}
```

```
./asadmin create-jms-resource --restype javax.jms.Queue testQueue2
./asadmin create-jms-resource --restype javax.jms.QueueConnectionFactory jms/QueueConnectionFactory
```

From http://javahowto.blogspot.pt/2010/04/message-driven-bean-example-with.html

# MDB can be requested

```
9
10  @WebServlet(urlPatterns = "/MessageServlet")
11  public class MessageServlet extends HttpServlet {
12   @Resource(mappedName = "testQueue2")
13   private Queue queue;
14
15   @Resource(mappedName = "jms/QueueConnectionFactory")
16   private QueueConnectionFactory queueConnectionFactory;
17
18   @Override
19   protected void doGet(HttpServletRequest request, HttpServletResponse response)
20           throws ServletException, IOException {
21       QueueConnection queueConnection = null;
22       try {
23           queueConnection = queueConnectionFactory.createQueueConnection();
24           queueConnection.start();
25           QueueSession queueSession = queueConnection.createQueueSession(false,
26                   Session.AUTO_ACKNOWLEDGE);
27           QueueSender sender = queueSession.createSender(queue);
28
29           TextMessage msg = queueSession.createTextMessage();
30           msg.setText("A message from MessageServlet");
31           msg.setStringProperty("name", "MessageServlet");
32
33           sender.send(msg);
34       } catch (JMSException e) {
35           throw new RuntimeException(e);
36       } finally {
37           try {
38               if (queueConnection != null) {
39                   queueConnection.close();
40               }
41           } catch (JMSException e) { //ignore
42           }
43       }
44   }
45  }
```

From http://javahowto.blogspot.pt/2010/04/message-driven-bean-example-with.html

# … To the server

```
9
10  @WebServlet(urlPatterns = "/MessageServlet")
11  public class MessageServlet extends HttpServlet {
12   @Resource(mappedName = "testQueue2")
13   private Queue queue;
14
15   @Resource(mappedName = "jms/QueueConnectionFactory")
16   private QueueConnectionFactory queueConnectionFactory;
17
18   @Override
19   protected void doGet(HttpServletRequest request, HttpServletResponse response)
20           throws ServletException, IOException {
21       QueueConnection queueConnection = null;
22       try {
23           queueConnection = queueConnectionFactory.createQueueConnection();
24           queueConnection.start();
25           QueueSession queueSession = queueConnection.createQueueSession(false,
26                   Session.AUTO_ACKNOWLEDGE);
27           QueueSender sender = queueSession.createSender(queue);
28
29           TextMessage msg = queueSession.createTextMessage();
30           msg.setText("A message from MessageServlet");
31           msg.setStringProperty("name", "MessageServlet");
32
33           sender.send(msg);
34       } catch (JMSException e) {
35           throw new RuntimeException(e);
36       } finally {
37           try {
38               if (queueConnection != null) {
39                   queueConnection.close();
40               }
41           } catch (JMSException e) { //ignore
42           }
43       }
44   }
45  }
```

From http://javahowto.blogspot.pt/2010/04/message-driven-bean-example-with.html

# … To the server

```
9
10  @WebServlet(urlPatterns = "/MessageServlet")
11  public class MessageServlet extends HttpServlet {
12   @Resource(mappedName = "testQueue2")
13   private Queue queue;
14
15   @Resource(mappedName = "jms/QueueConnectionFactory")
16   private QueueConnectionFactory queueConnectionFactory;
17
18   @Override
19   protected void doGet(HttpServletRequest request, HttpServletResponse response)
20           throws ServletException, IOException {
21       QueueConnection queueConnection = null;
22       try {
23           queueConnection = queueConnectionFactory.createQueueConnection();
24           queueConnection.start();
25           QueueSession queueSession = queueConnection.createQueueSession(false,
26                   Session.AUTO_ACKNOWLEDGE);
27           QueueSender sender = queueSession.createSender(queue);
28
29           TextMessage msg = queueSession.createTextMessage();
30           msg.setText("A message from MessageServlet");
31           msg.setStringProperty("name", "MessageServlet");
32
33           sender.send(msg);
34       } catch (JMSException e) {
35           throw new RuntimeException(e);
36       } finally {
37           try {
38               if (queueConnection != null) {
39                   queueConnection.close();
40               }
41           } catch (JMSException e) { //ignore
42           }
43       }
44   }
45  }
```

From http://javahowto.blogspot.pt/2010/04/message-driven-bean-example-with.html

# … to pass messages

```
9
10  @WebServlet(urlPatterns = "/MessageServlet")
11  public class MessageServlet extends HttpServlet {
12   @Resource(mappedName = "testQueue2")
13   private Queue queue;
14
15   @Resource(mappedName = "jms/QueueConnectionFactory")
16   private QueueConnectionFactory queueConnectionFactory;
17
18   @Override
19   protected void doGet(HttpServletRequest request, HttpServletResponse response)
20           throws ServletException, IOException {
21       QueueConnection queueConnection = null;
22       try {
23           queueConnection = queueConnectionFactory.createQueueConnection();
24           queueConnection.start();
25           QueueSession queueSession = queueConnection.createQueueSession(false,
26                   Session.AUTO_ACKNOWLEDGE);
27           QueueSender sender = queueSession.createSender(queue);
28
29           TextMessage msg = queueSession.createTextMessage();
30           msg.setText("A message from MessageServlet");
31           msg.setStringProperty("name", "MessageServlet");
32
33           sender.send(msg);
34       } catch (JMSException e) {
35           throw new RuntimeException(e);
36       } finally {
37           try {
38               if (queueConnection != null) {
39                   queueConnection.close();
40               }
41           } catch (JMSException e) { //ignore
42           }
43       }
44   }
45  }
```

From http://javahowto.blogspot.pt/2010/04/message-driven-bean-example-with.html

# … That will “trigger” the MDB

```
9
10  @WebServlet(urlPatterns = "/MessageServlet")
11  public class MessageServlet extends HttpServlet {
12   @Resource(mappedName = "testQueue2")
13   private Queue queue;
14
15   @Resource(mappedName = "jms/QueueConnectionFactory")
16   private QueueConnectionFactory queueConnectionFactory;
17
18   @Override
19   protected void doGet(HttpServletRequest request, HttpServletResponse response)
20           throws ServletException, IOException {
21       QueueConnection queueConnection = null;
22       try {
23           queueConnection = queueConnectionFactory.createQueueConnection();
24           queueConnection.start();
25           QueueSession queueSession = queueConnection.createQueueSession(false,
26                   Session.AUTO_ACKNOWLEDGE);
27           QueueSender sender = queueSession.createSender(queue);
28
29           TextMessage msg = queueSession.createTextMessage();
30           msg.setText("A message from MessageServlet");
31           msg.setStringProperty("name", "MessageServlet");
32
33           sender.send(msg);
34       } catch (JMSException e) {
35           throw new RuntimeException(e);
36       } finally {
37           try {
38               if (queueConnection != null) {
39                   queueConnection.close();
40               }
41           } catch (JMSException e) { //ignore
42           }
43       }
44   }
45  }
```

From http://javahowto.blogspot.pt/2010/04/message-driven-bean-example-with.html

# … That will “trigger” the MDB

```
@WebServlet(urlPatterns = "/MessageServlet")
public class MessageServlet extends HttpServlet {
 @Resource(mappedName = "testQueue2")
 private Queue queue;

 @Resource(mappedName = "jms/QueueConnectionFactory")
 private QueueConnectionF

 @Override
 protected void doGet(Htt
         throws ServletEx
     QueueConnection queu
     try {
         queueConnection
         queueConnection.
         QueueSession que
                 Session.
         QueueSender send

         TextMessage msg
         msg.setText("A m
         msg.setStringPro

         sender.send(msg)
     } catch (JMSExceptio
         throw new Runtim
     } finally {
         try {
             if (queueCon
                 queueCon
             }
         } catch (JMSException e) { //ignore
         }
     }
 }
}
```

```
package test;

import java.util.logging.Logger;
import javax.ejb.ActivationConfigProperty;
import javax.ejb.MessageDriven;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;

@MessageDriven(mappedName="testQueue2",
     activationConfig = { @ActivationConfigProperty(
     propertyName="destinationType", propertyValue="javax.jms.Queue")})
public class MessageBean implements MessageListener {
 Logger logger = Logger.getLogger("test");

 public void onMessage(Message msg) {
     try {
         String name = msg.getStringProperty("name");
         logger.info("Received msg " + msg + ", from " + name);
     } catch (JMSException e) {
         throw new RuntimeException(e);
     }
 }
}
```

From http://javahowto.blogspot.pt/2010/04/message-driven-bean-example-with.html

# Note...

- This is a snippet
- Configurations, deployments , ...
- Several subjects not described
  - Transactions
  - Quality of service
  - Connection type
  - Synchronous / asynchronous
  - ...

# JMS is a interface for listener

- Need to use a “real broker”
- Application servers have embedded brokers
  - several options
  - Some information not available/ clear
  - No full compatibility


https://www.rabbitmq.com/#features

RabbitMQ by Pivotal

Features | Get Started | Support | Community | Docs | Blog

## 1 "Hello World!"

The simplest thing that does *something*

Python | Java | Ruby | PHP | C#

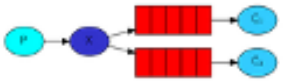
## 2 Work queues

Distributing tasks among workers

Python | Java | Ruby | PHP | C#

## 3 Publish/Subscribe

Sending messages to many consumers at once

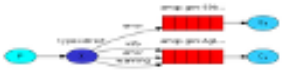


Python | Java | Ruby | PHP | C#

Rabbit
deplo
messa

## 4 Routing

Receiving messages selectively



Python | Java | Ruby | PHP | C#

## 5 Topics

Receiving messages based on a pattern

Python | Java | Ruby | PHP | C#

## 6 RPC

Remote procedure call implementation

Python | Java | Ruby | PHP | C#

With more tha
startups and la
message broker.

RabbitMQ is lightweight and easy to deploy on premises and in the cloud. It supports multiple messaging protocols. RabbitMQ can be deployed in distributed and federated configurations to meet high-scale, high-availability requirements.

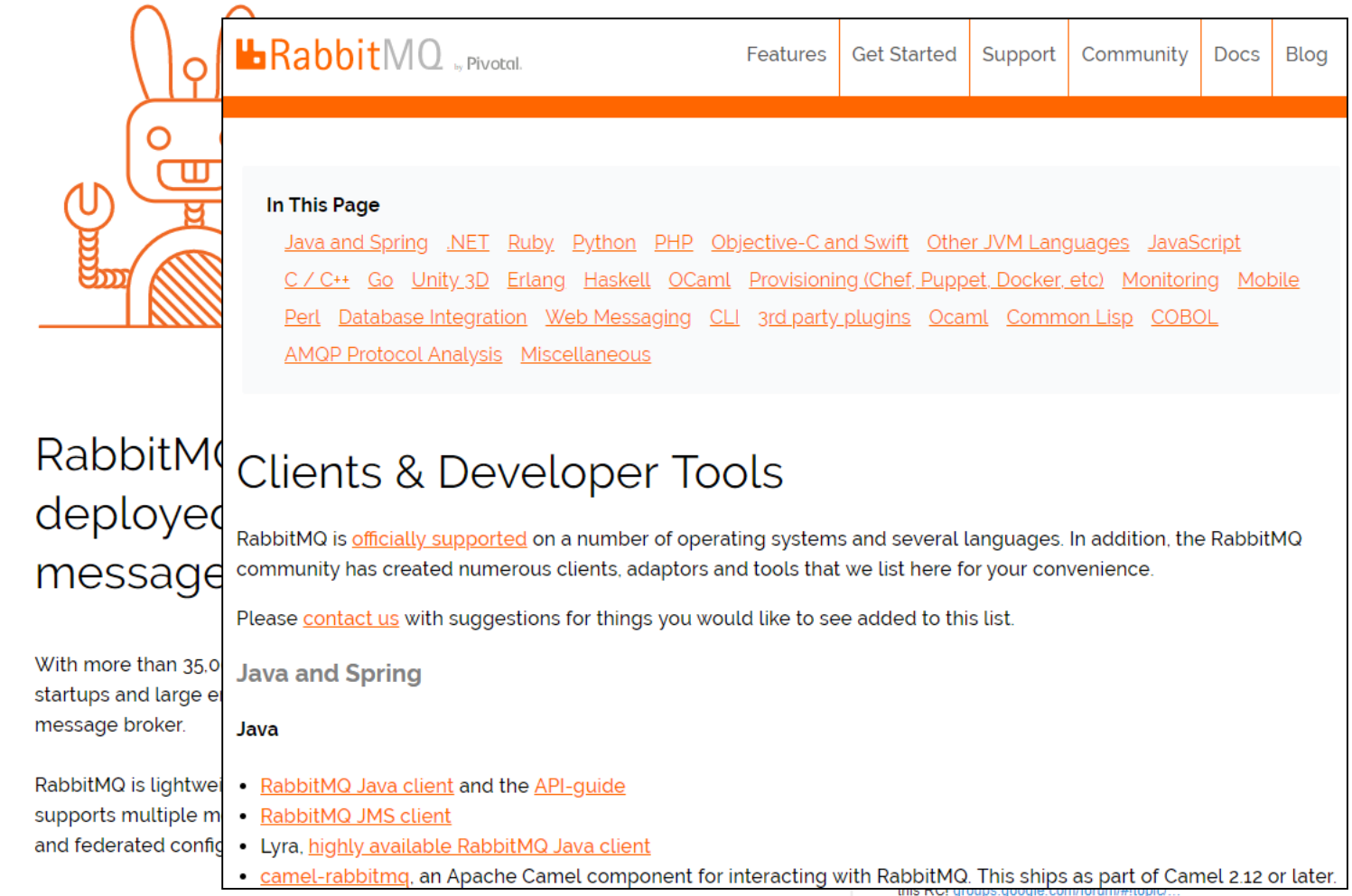
loud environments, and provides

Tweets

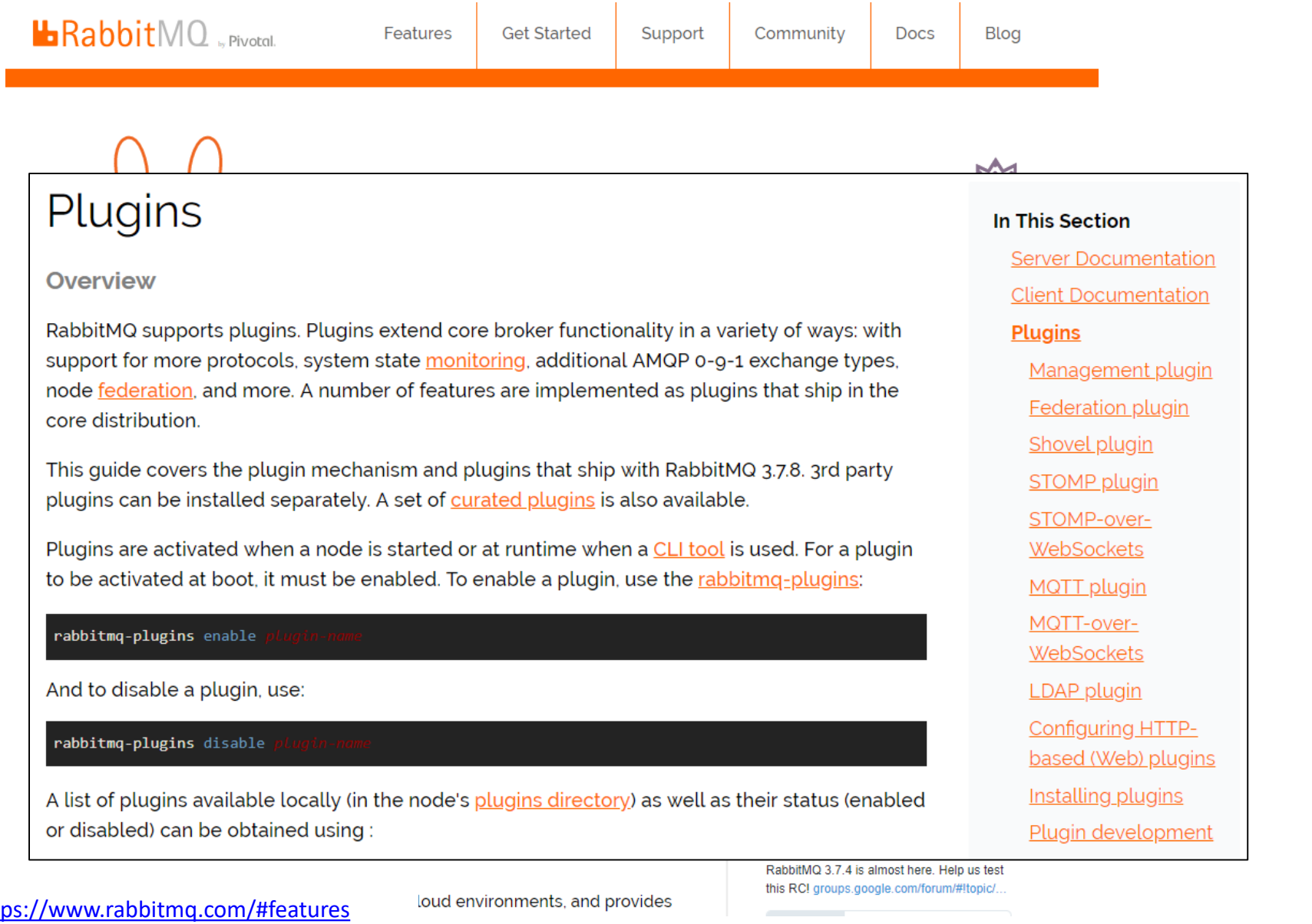
RabbitMQ
@RabbitMQ

RabbitMQ 3.7.4 is almost here. Help us test this RC! groups.google.com/forum/#!topic/...

https://www.rabbitmq.com/#features
https://www.rabbitmq.com/devtools.html

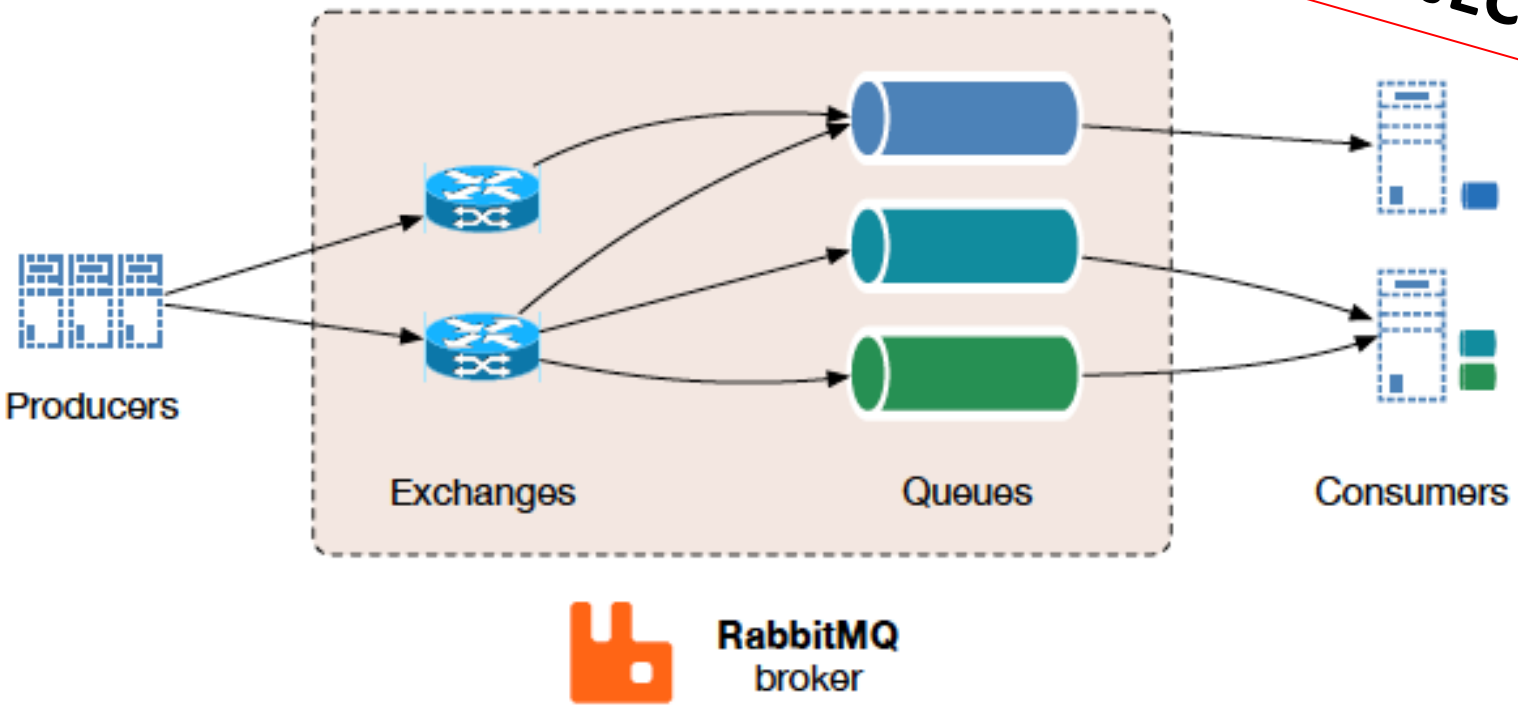



https://www.rabbitmq.com/#features
https://www.rabbitmq.com/devtools.html




https://www.rabbitmq.com/#features
https://www.rabbitmq.com/devtools.html

# CAN BE USEFULL IN SOME PROJECTS

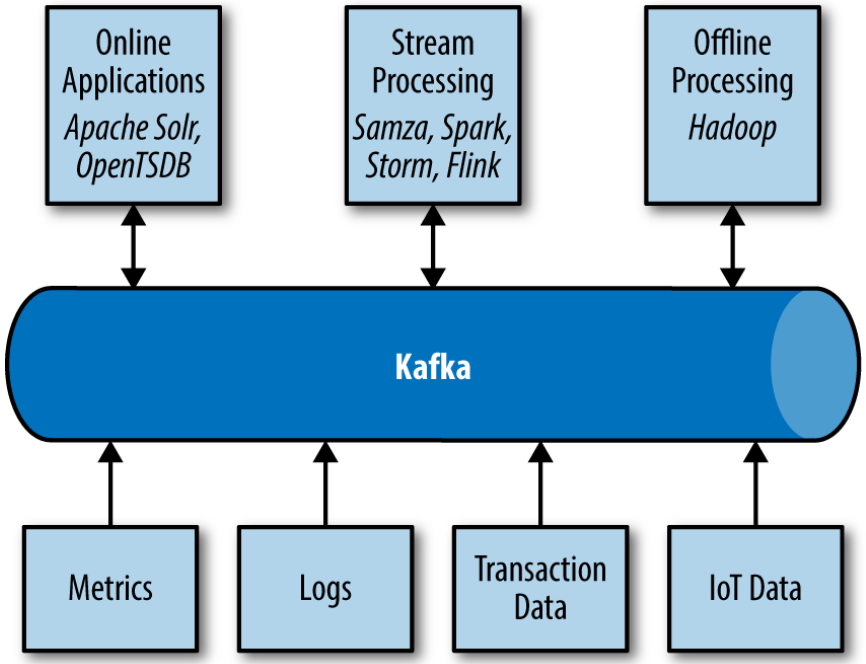



Understanding When to use RabbitMQ or Apache Kafka
https://content.pivotal.io/blog/understanding-when-to-use-rabbitmq-or-apache-kafka

# Kafka: a worthy option

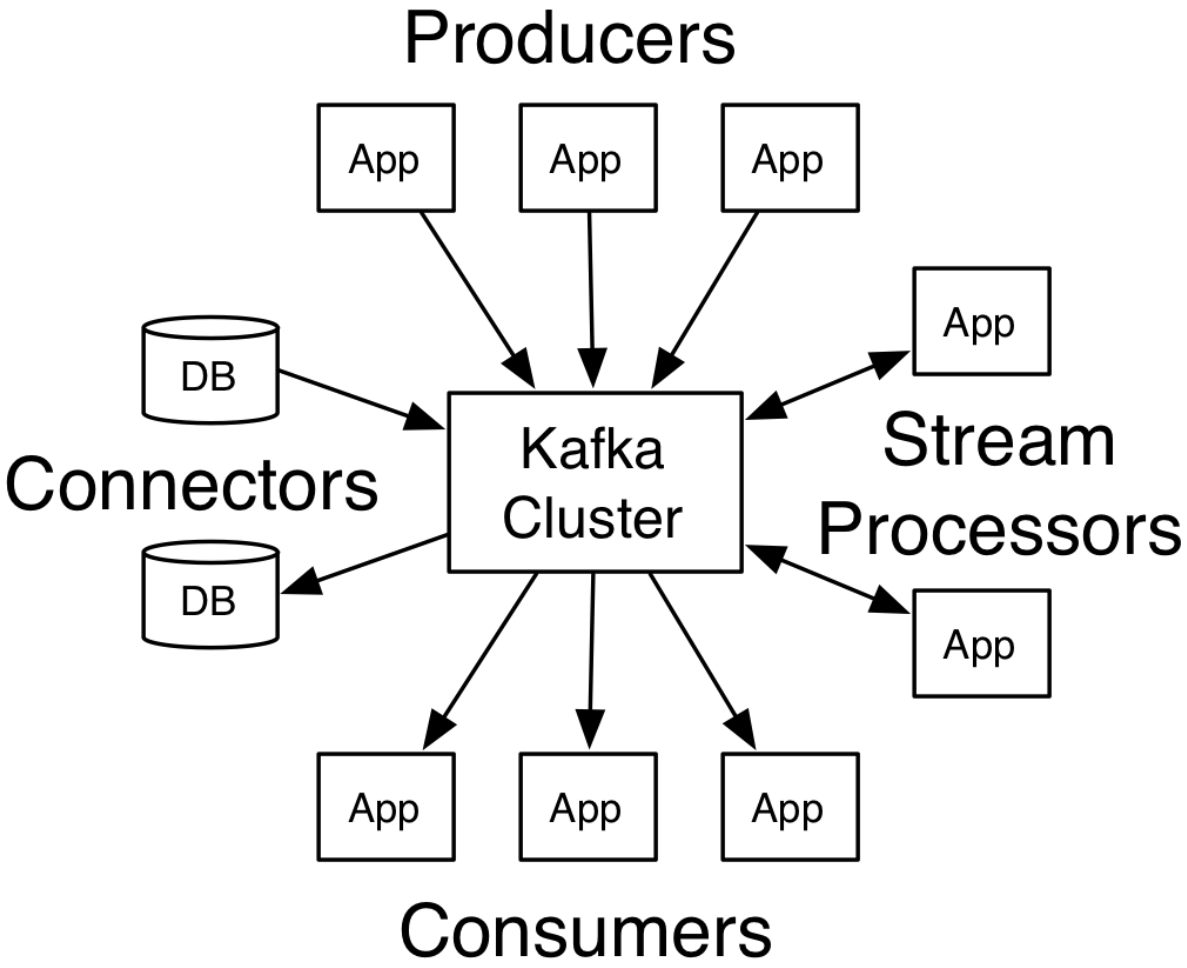
# Kafka

- As a Messaging System
- As a Storage System
- for Stream Processing



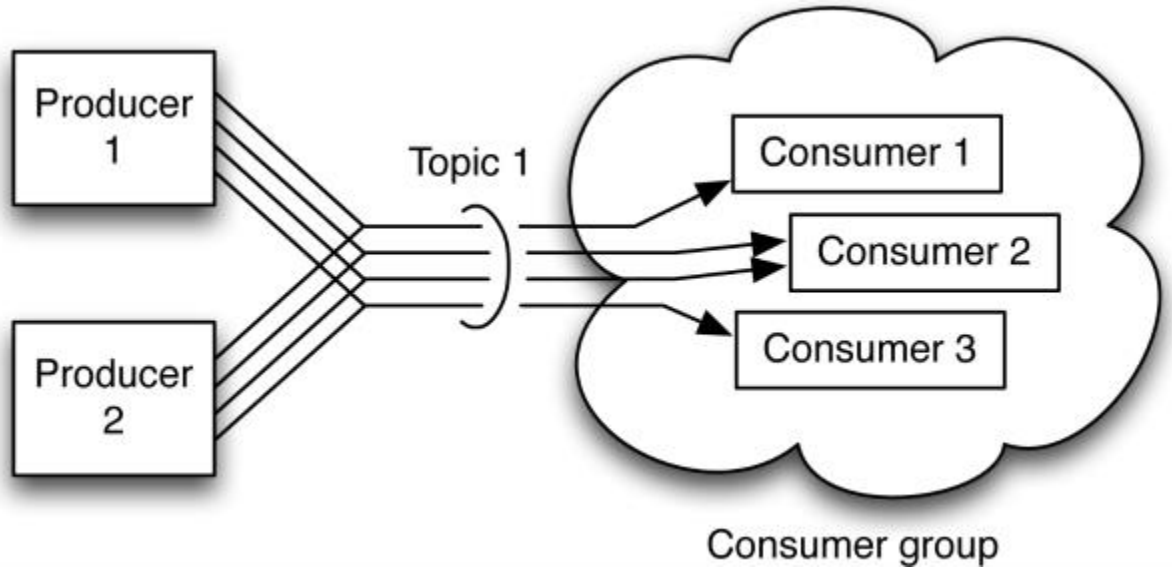

https://balamaci.ro/kafka-streams-for-stream-processing/
https://www.safaribooksonline.com/library/view/kafka-the-definitive/9781491936153/ch01.html

# Kafka is a broker

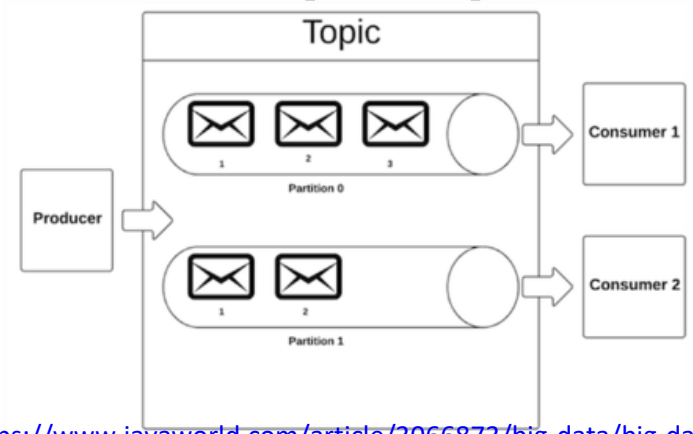



http://kafka.apache.org/intro

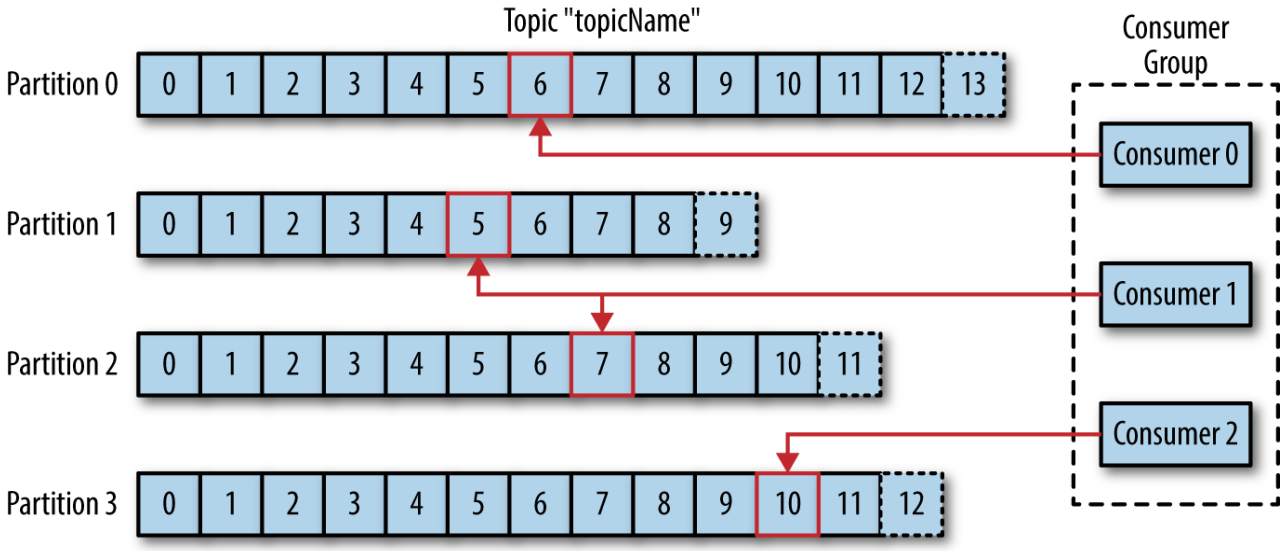
# It has topic to publish “events”



# Uses topic partition to scale

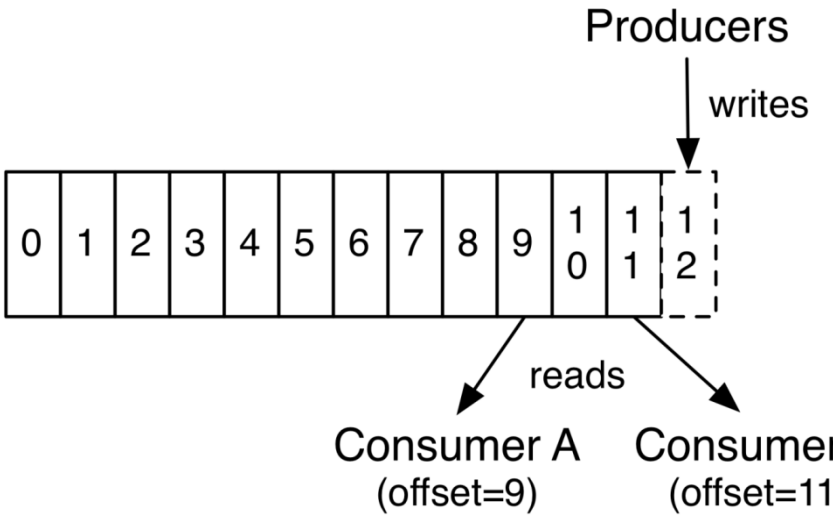



https://www.javaworld.com/article/3066873/big-data/big-data-messaging-with-kafka-part-2.html



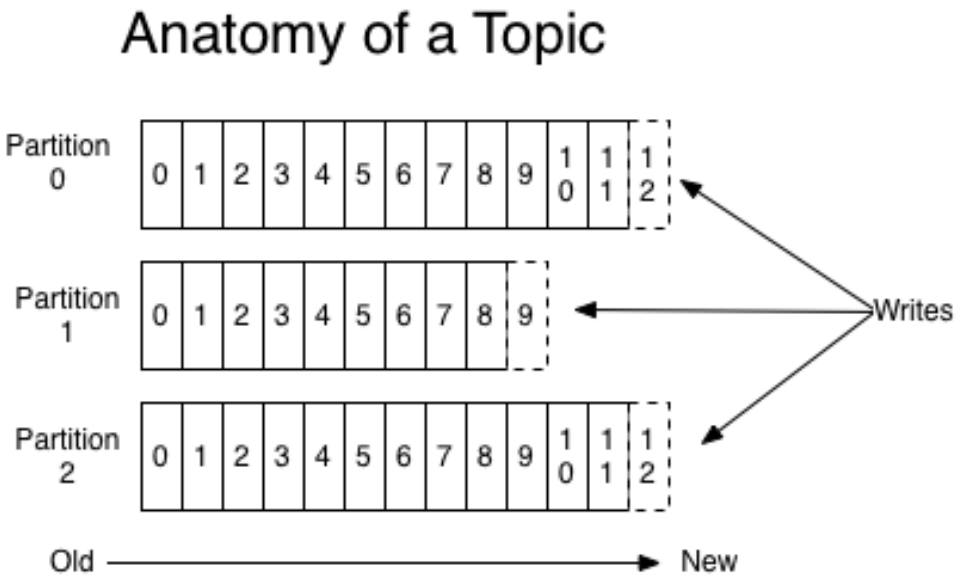

https://www.safaribooksonline.com/library/view/kafka-the-definitive/9781491936153/ch01.html

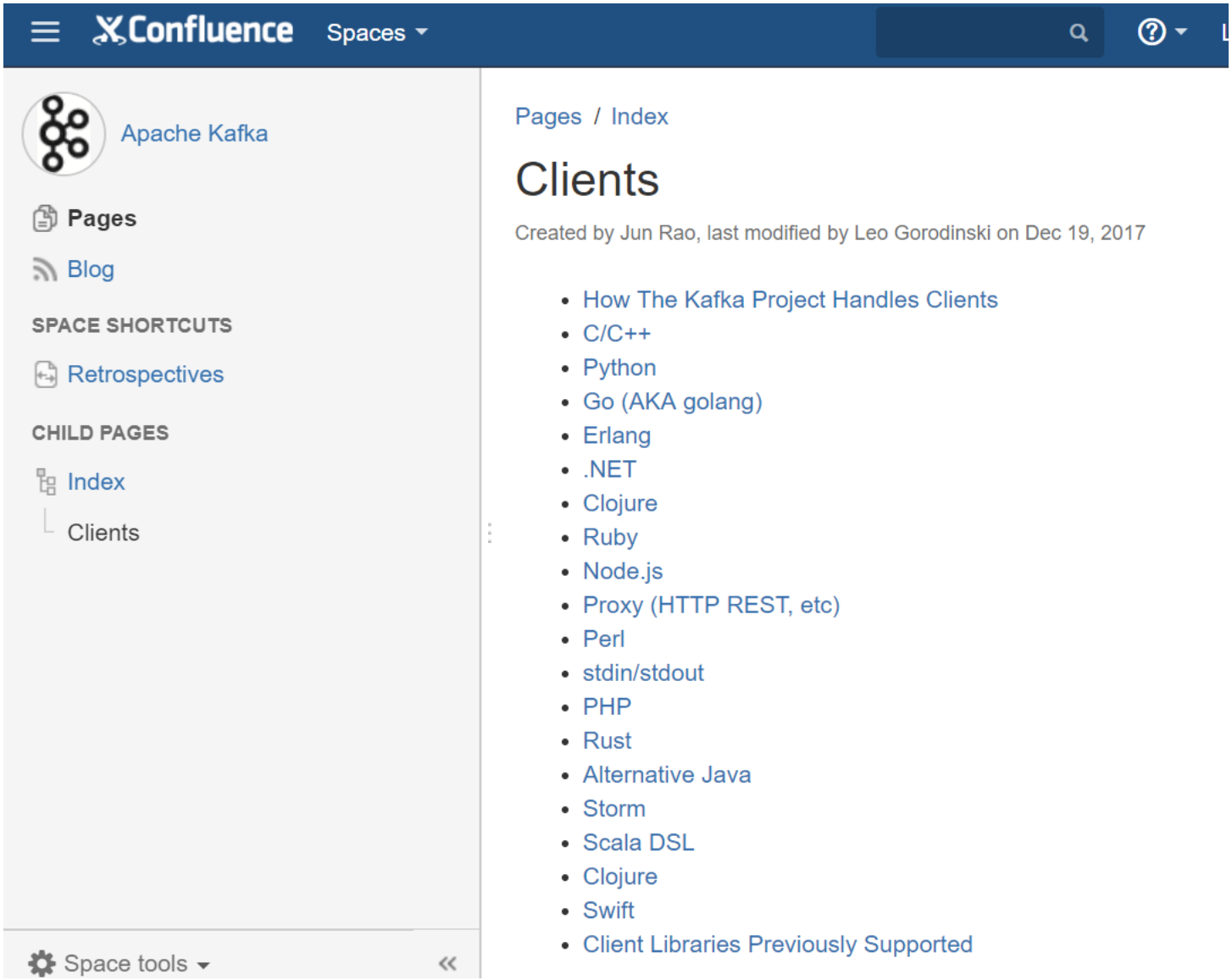
# Uses memory files to store




Message agnostic i.e. appends message, pointers are file offsets. No message processing

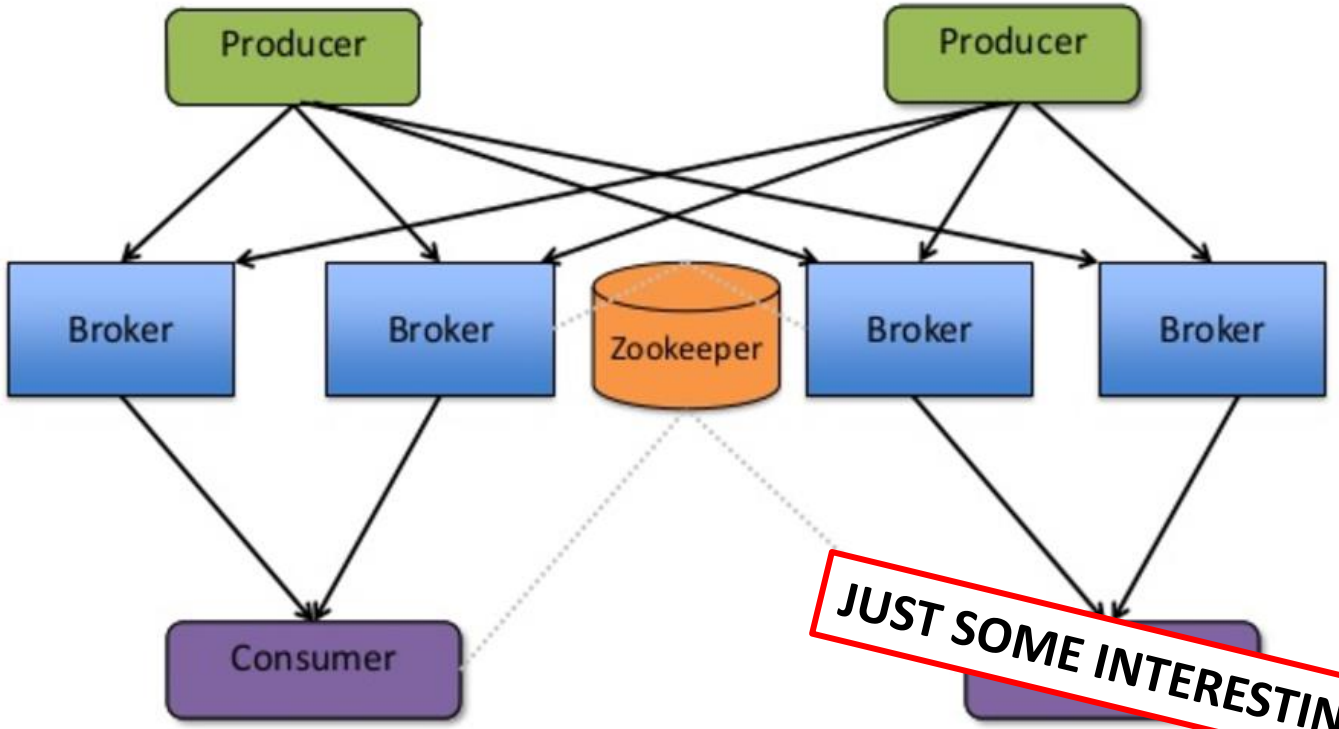
Concurrent access depends on file concurrent OS access support





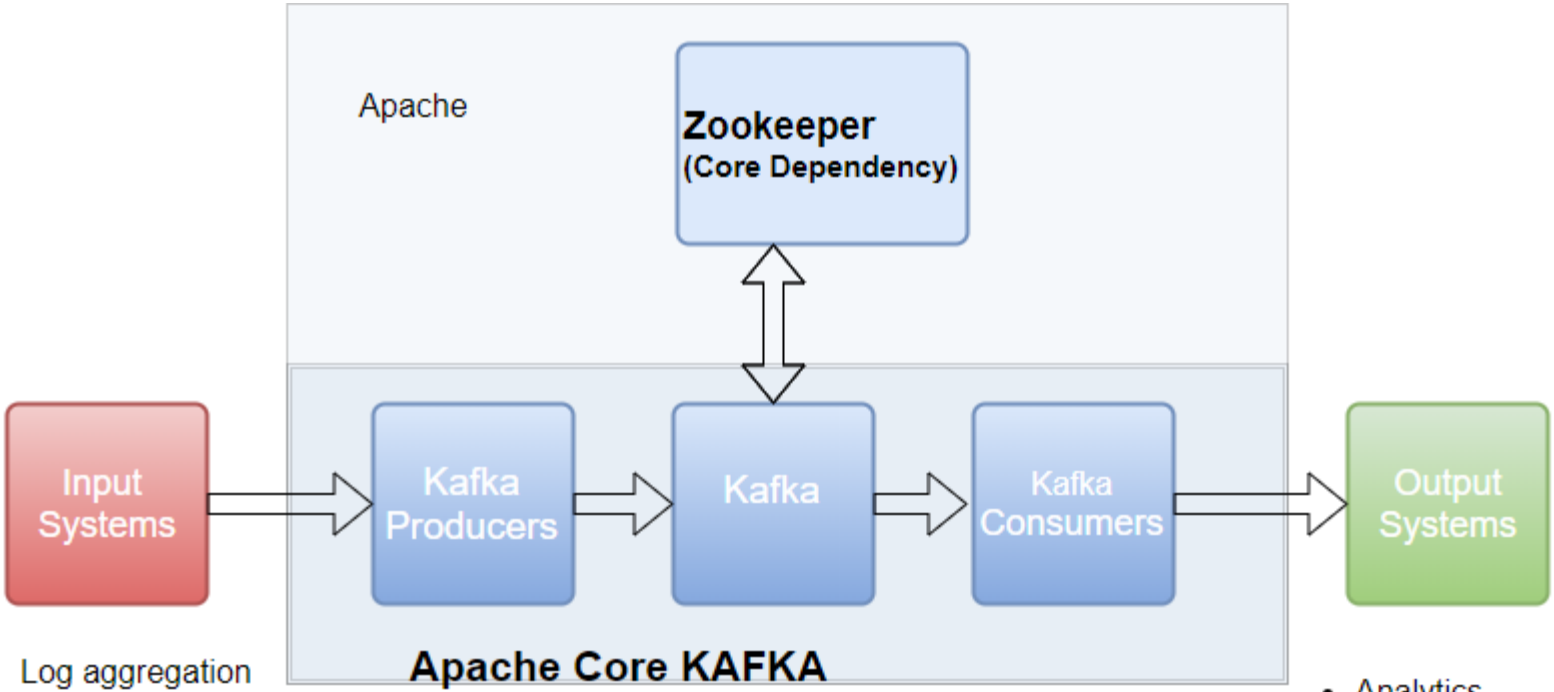

https://cwiki.apache.org/confluence/display/KAFKA/Clients

# Kafka Architecture




JUST SOME INTERESTING DETAILS

# Kafka architecture




- Log aggregation
- Metrics
- KPIs
- Batch imports
- Audit trail
- User activity logs
- Web logs

***Not*** part of core

- Schema Registry
- Avro
- Kafka REST Proxy
- Kafka Connect
- Kafka Streams

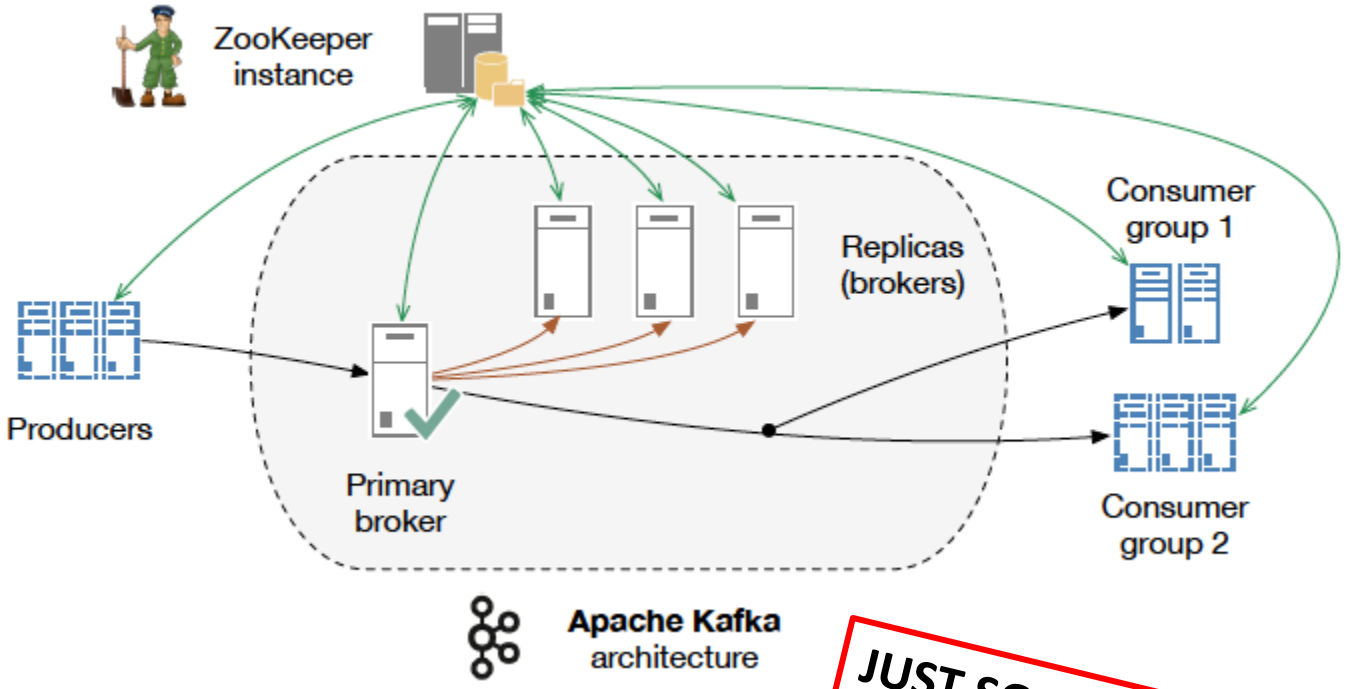
Apache Kafka Core

- Server/Broker
- Scripts to start libs
- Script to start up Zookeeper
- Utils to create topics
- Utils to monitor stats

- Analytics
- Databases
- Machine Learning
- Dashboards
- Indexed for Search
- Business Intelligene

https://dzone.com/articles/kafka-architecture

# Support clustering



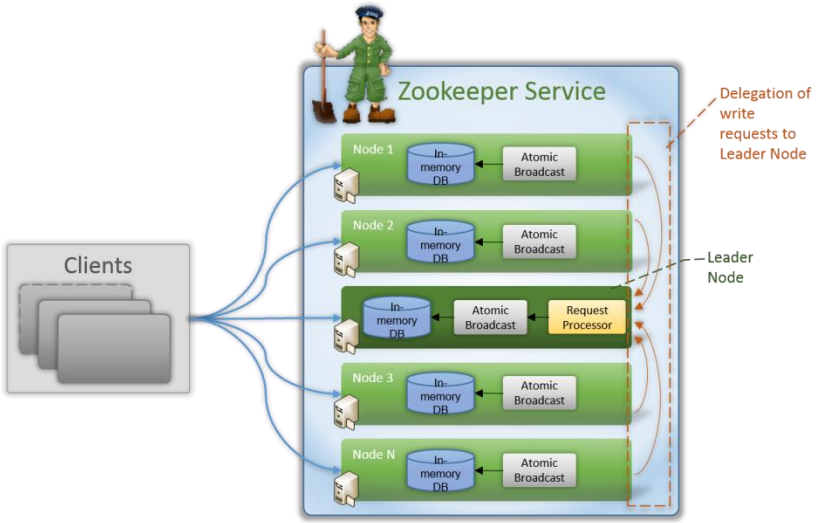

JUST SOME INTERESTING DETAILS

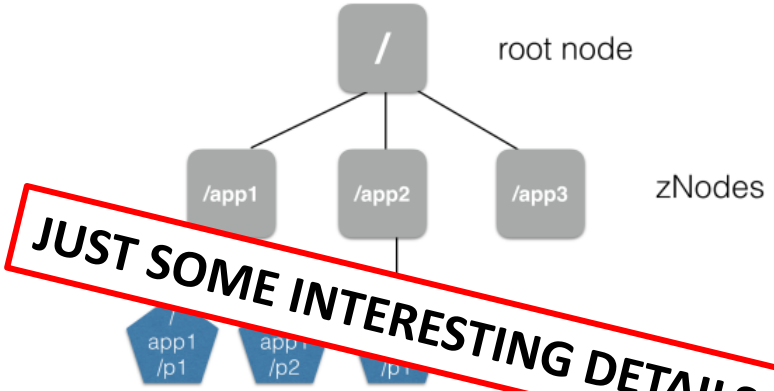
Understanding When to use RabbitMQ or Apache Kafka
https://content.pivotal.io/blog/understanding-when-to-use-rabbitmq-or-apache-kafka

# Apache Zookeeper

GEEK ZONE

- Apache open source
- Distributed file system
  - Focused of being a repository for configuration files and the services needed around that.
  - Supprot co-ordinating services for distributed solutions.
- it is itself, a distributed system.
- used by Hadoop components
  - HDFS file system, HBase, Kafka, and others.



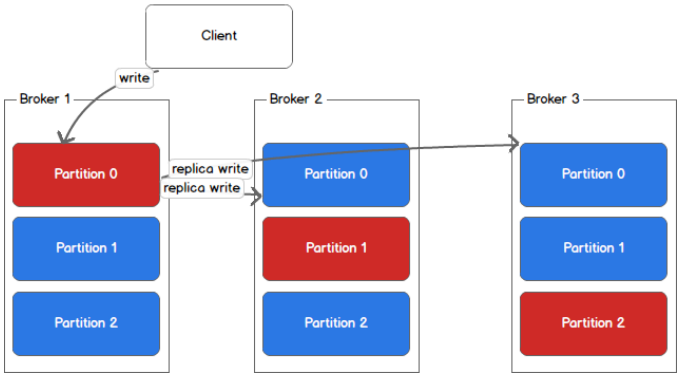




http://www.allprogrammingtutorials.com/tutorials/introduction-to-apache-zookeeper.php

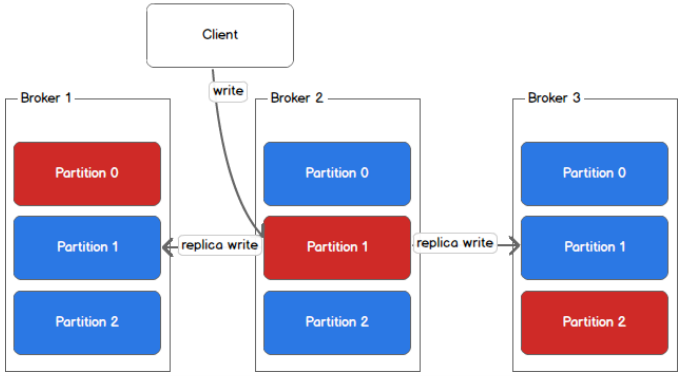
# Kafka consistency and failover

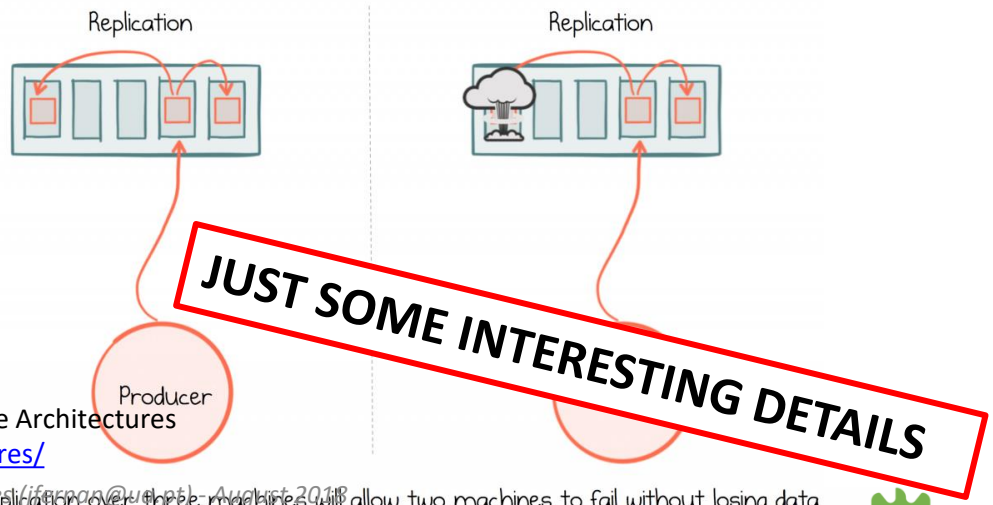
Apache Zookeeper

Leader (red) and replicas (blue)



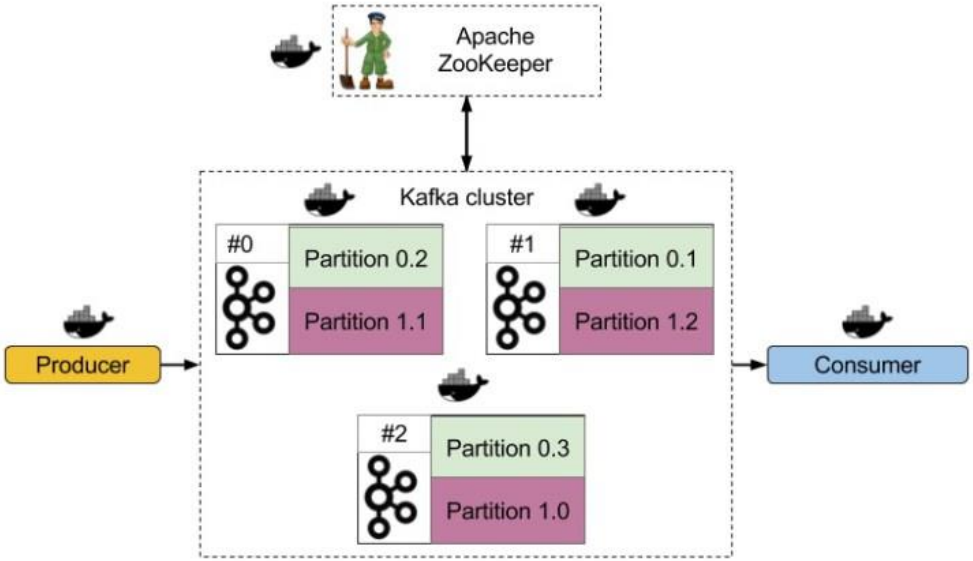

Leader (red) and replicas (blue)



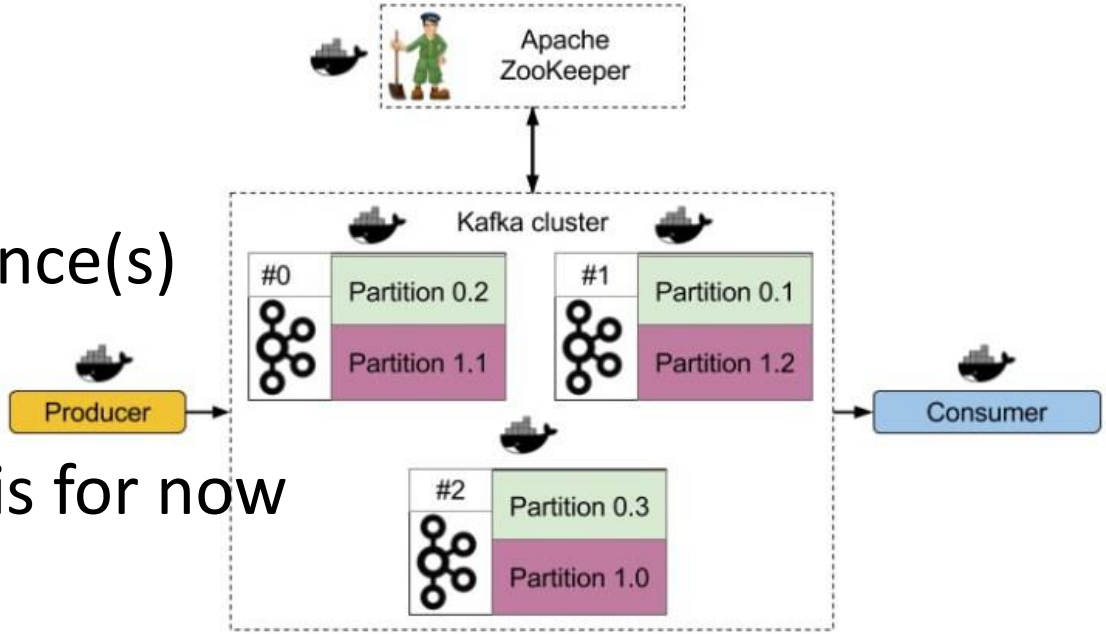

Using Apache Kafka as a Scalable, Event-Driven Backbone for Service Architectures
https://www.confluent.io/blog/apache-kafka-for-service-architectures/

# Deploying kafka



# Deploying kafka

- Basic concern
  - access Zookeeper
  - access kafka instance(s)
- Native OS
  - Probably using this for now
- Docker
  - several options




https://codeblog.dotsandbrackets.com/highly-available-kafka-cluster-docker/

# Start ZK and kafka

- in linux
  - bin/kafka-server-start.sh config/server.properties
  - bin/zookeeper-server-start.sh config/zookeeper.properties

- in windows
  - bin\windows\kafka-server-start.bat config\server.properties
  - bin\windows\zookeeper-server-start.bat config\zookeeper.properties

https://kafka.apache.org/quickstart

Search Explore Help Sign up Sign In

PUBLIC | AUTOMATED BUILD

# wurstmeister/kafka ☆

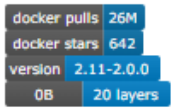
Last pushed: 16 hours ago

Repo Info Tags Dockerfile Build Details

Short Description

Multi-Broker Apache Kafka Image

Full Description




## kafka-docker

Dockerfile for Apache Kafka

The image is available directly from Docker Hub

## Announcements

- **03-Apr-2018** - *BREAKING* - `KAFKA_ADVERTISED_PROTOCOL_NAME` and `KAFKA_PROTOCOL_NAME` removed. Please update to canonical kafka settings.
- **03-Apr-2018** - *BREAKING* - `KAFKA_ZOOKEEPER_CONNECT` is now a mandatory environment var.

## Pre-Requisites

- install docker-compose https://docs.docker.com/compose/install/
- modify the `KAFKA_ADVERTISED_HOST_NAME` in docker-compose.yml to match your docker host IP (Note: Do not use loca
- if you want to c
`compose.yml`, e.g. in order to increase the `message.max.bytes` parameter set the environment to `YTES: 2000000`. To turn off automatic topic creation set

Docker Pull Command

```
docker pull wurstmeister/kafka
```

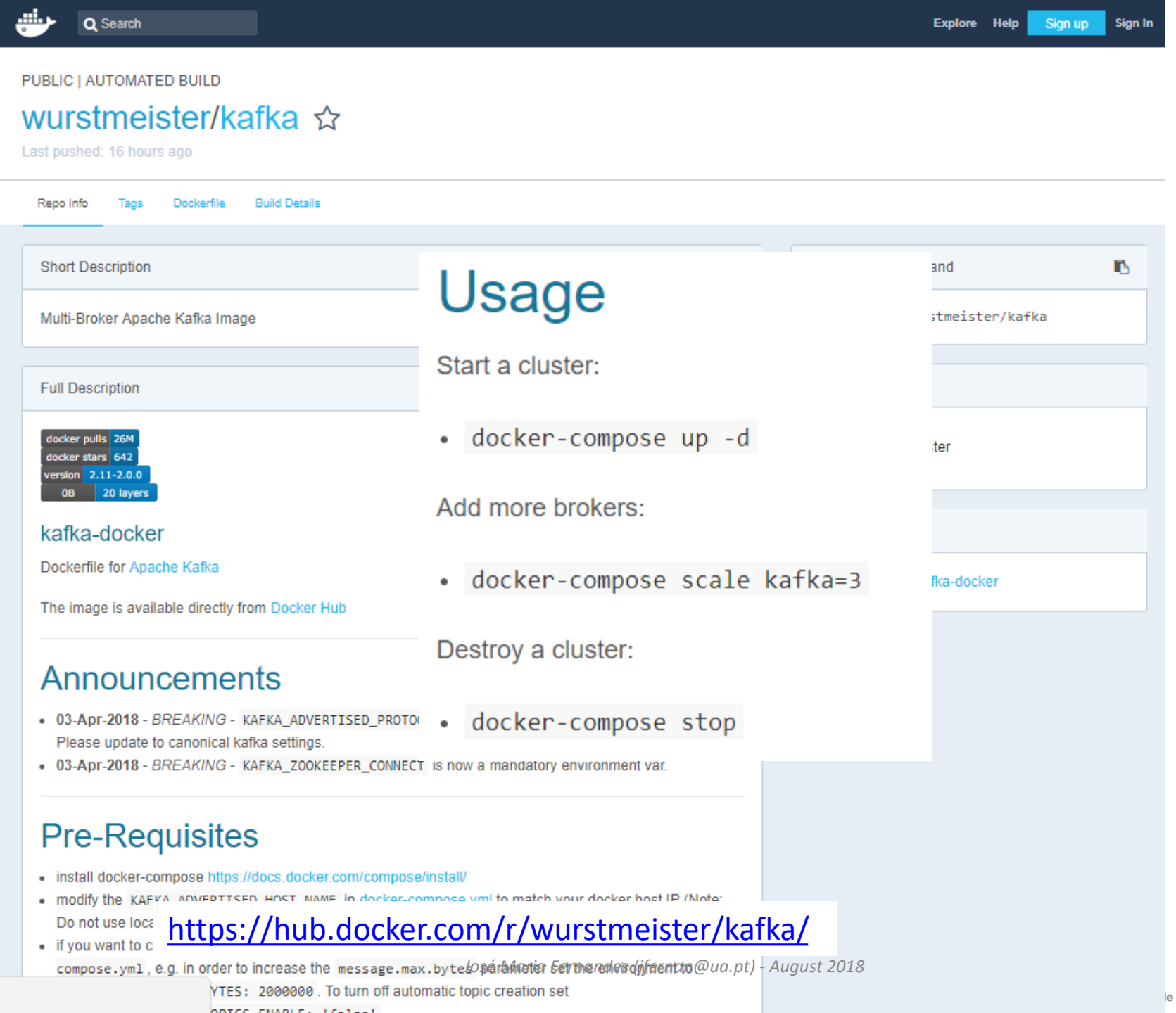
Owner

wurstmeister

Source Repository

wurstmeister/kafka-docker

https://hub.docker.com/r/wurstmeister/kafka/




https://hub.docker.com/r/wurstmeister/kafka/

# Connectivity

sscaling edited this page on Mar 29 · 1 revision

## Kafka connectivity

A large percentage of questions and issues opened against the kafka-docker project concern configuring Kafka networking. This is often a case of not understanding the Kafka requirements, or not understanding docker networking.

This page aims to explain some of the basic requirements to help people resolve any initial issues they may encounter. It is not an exhaustive guide to configuring Kafka and/or Docker.

### Kafka requirements

There are three main requirements for configuring Kafka networking.

1. Each Broker must be able to talk to Zookeeper - for leader election etc.
2. Each Broker must be able to talk to every other Broker - for replication etc.
3. Each Consumer/Producer must be able to talk to every Broker - for reading/writing data etc.

The following diagram represents the different communication paths:

Pages 3

Find a Page...

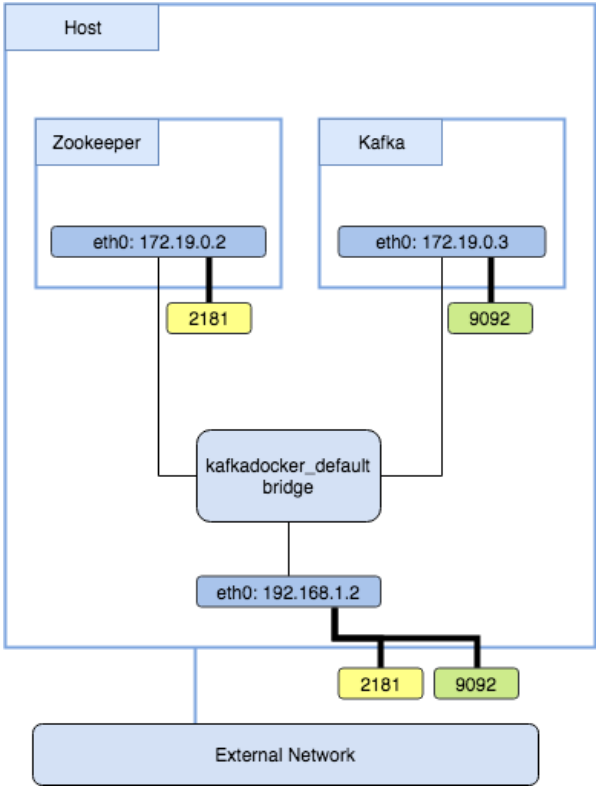
Home

Connectivity

Swarm

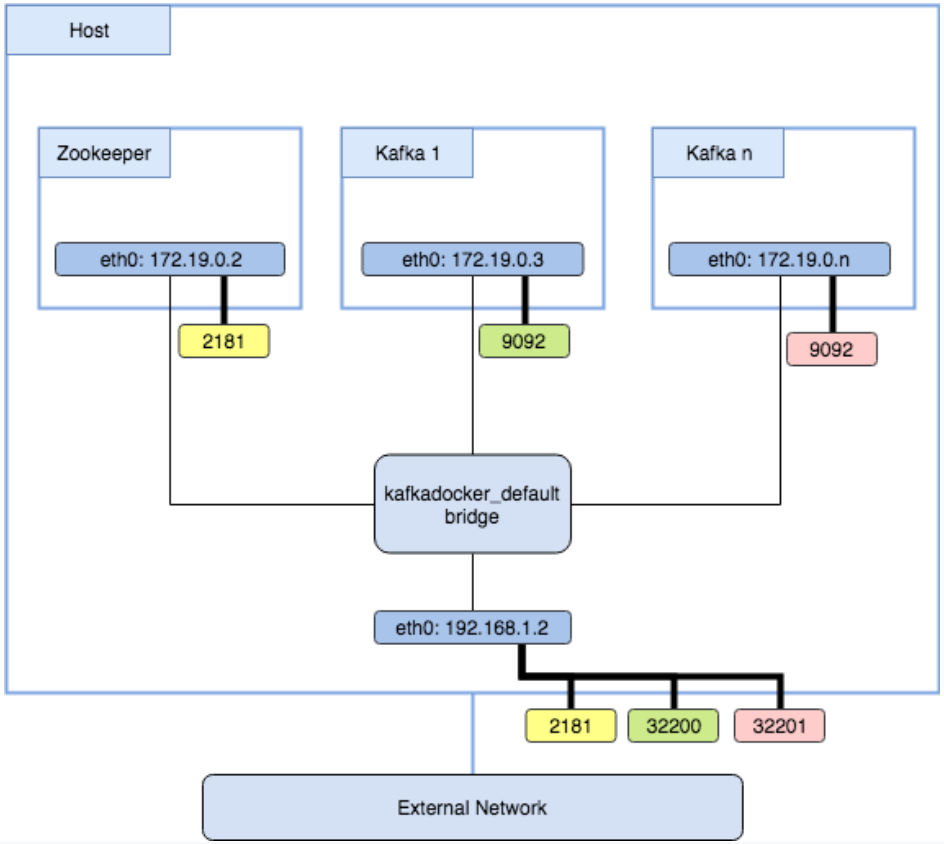


https://github.com/wurstmeister/kafka-docker/wiki/Connectivity




```
# KAFKA_ADVERTISED_HOST_NAME: localhost
# ZOOKEEPER_CONNECT: zookeeper:2181
docker-compose -f docker-compose-single-broker.yml up -d
```

Kafka connectivity
https://github.com/wurstmeister/kafka-docker/wiki/Connectivity




```
# KAFKA_ADVERTISED_HOST_NAME: 192.168.1.2
# ZOOKEEPER_CONNET: zookeeper:2181
docker-compose up -d zookeeper
docker-compose scale kafka=2
```

Kafka connectivity
https://github.com/wurstmeister/kafka-docker/wiki/Connectivity

Search Explore Help Sign up Sign

PUBLIC | AUTOMATED BUILD

# spotify/kafka ☆

Last pushed: 2 years ago

Repo Info Tags Dockerfile Build Details

Short Description

A simple docker image with both Kafka and Zookeeper.

Full Description

## Kafka in Docker

This repository provides everything you need to run Kafka in Docker.

For convenience also contains a packaged proxy that can be used to get data from a legacy Kafka 7 cluster into a dockerized Kafka 8.

# Why?

The main hurdle of running Kafka in Docker is that it depends on Zookeeper. Compared to other Kafka docker images, this one runs both Zookeeper and Kafka in the same container. This means:

Docker Pull Command

```
docker pull spotify/kafka
```

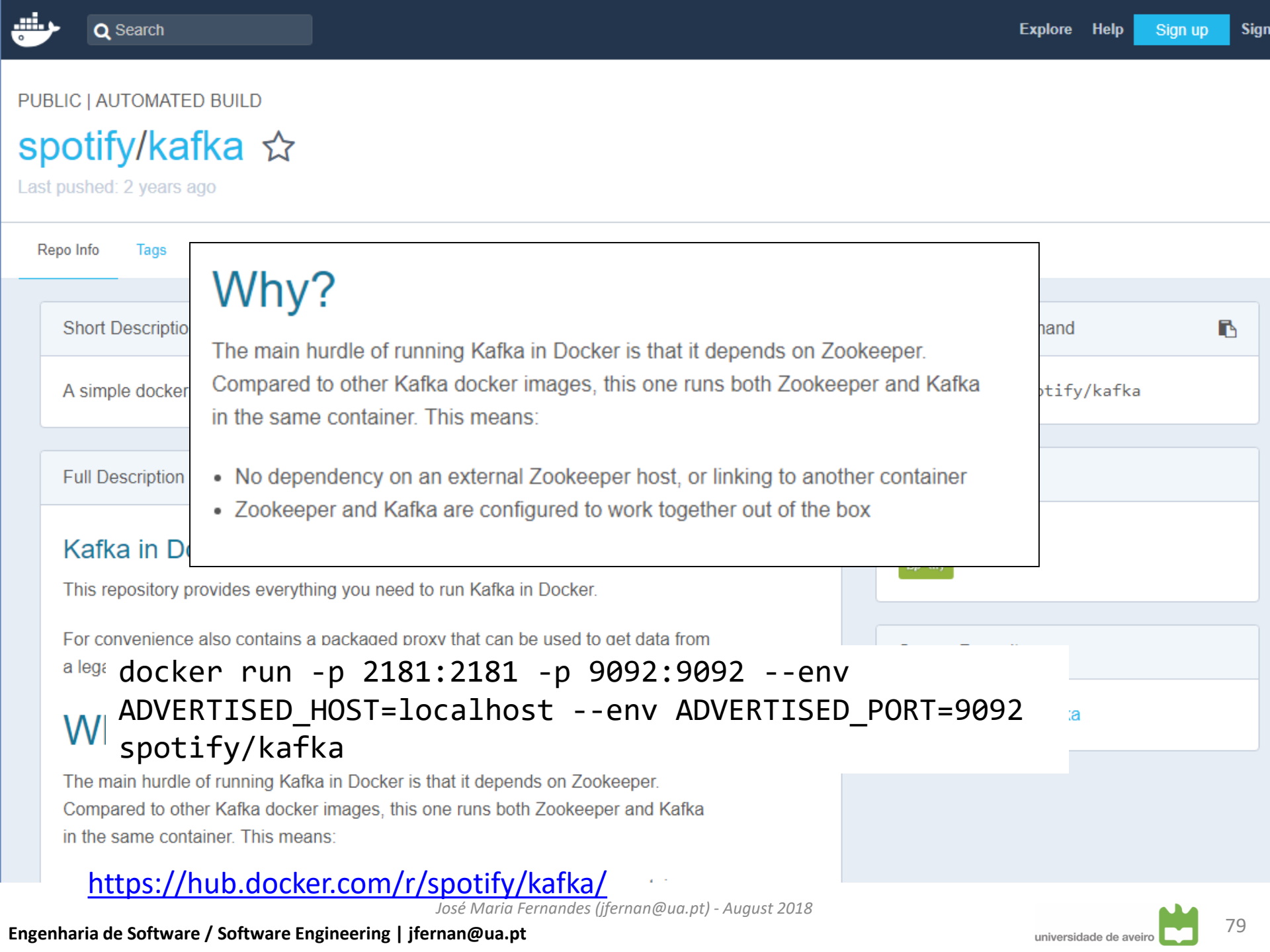
Owner

spotify

Source Repository

spotify/docker-kafka

https://hub.docker.com/r/spotify/kafka/

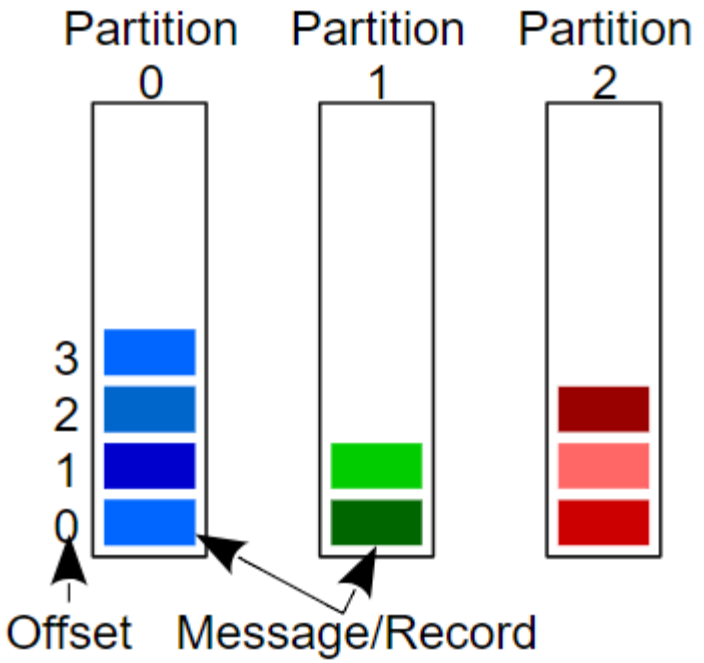

Search
Explore
Help
Sign up
PUBLIC | AUTOMATED BUILD
spotify/kafka
Last pushed: 2 years ago
Repo Info
Tags
Short Description
A simple docker
Full Description
Kafka in D
This repository provides everything you need to run Kafka in Docker.
For convenience also contains a packaged proxy that can be used to get data from
a lega
W
The main hurdle of running Kafka in Docker is that it depends on Zookeeper.
Compared to other Kafka docker images, this one runs both Zookeeper and Kafka
in the same container. This means:
Why?
The main hurdle of running Kafka in Docker is that it depends on Zookeeper.
Compared to other Kafka docker images, this one runs both Zookeeper and Kafka
in the same container. This means:
No dependency on an external Zookeeper host, or linking to another container
Zookeeper and Kafka are configured to work together out of the box
docker run -p 2181:2181 -p 9092:9092 --env
ADVERTISED_HOST=localhost --env ADVERTISED_PORT=9092
spotify/kafka
https://hub.docker.com/r/spotify/kafka/

# Producer / Consumer using shell i.e. no code

# Create topic

```
bin/kafka-topics.sh --create
--zookeeper localhost:2181
--replication-factor 1 --partitions 1
--topic test
```

```
bin\windows\kafka-topics.bat --create
--zookeeper localhost:2181
--replication-factor 1 --partitions 1
--topic test
```

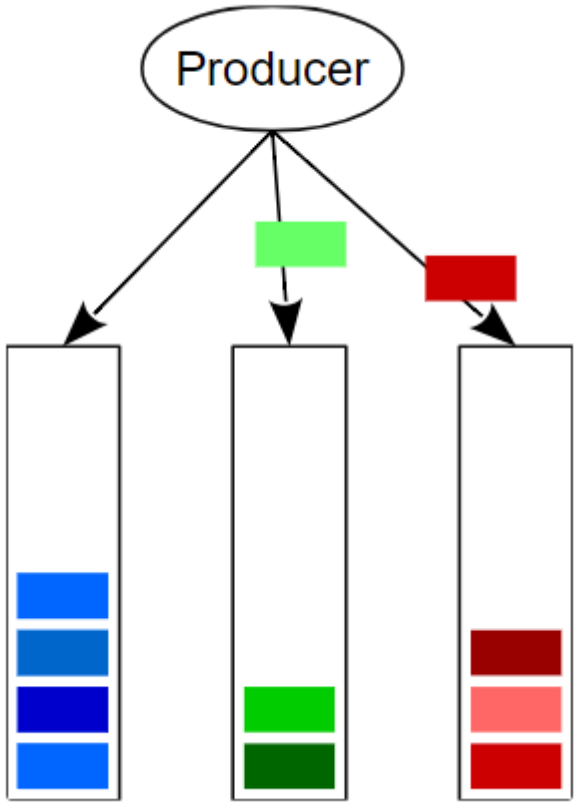



https://kafka.apache.org/quickstart

# Run producer

```
bin/kafka-console-producer.sh
--broker-list localhost:9092
--topic test
```

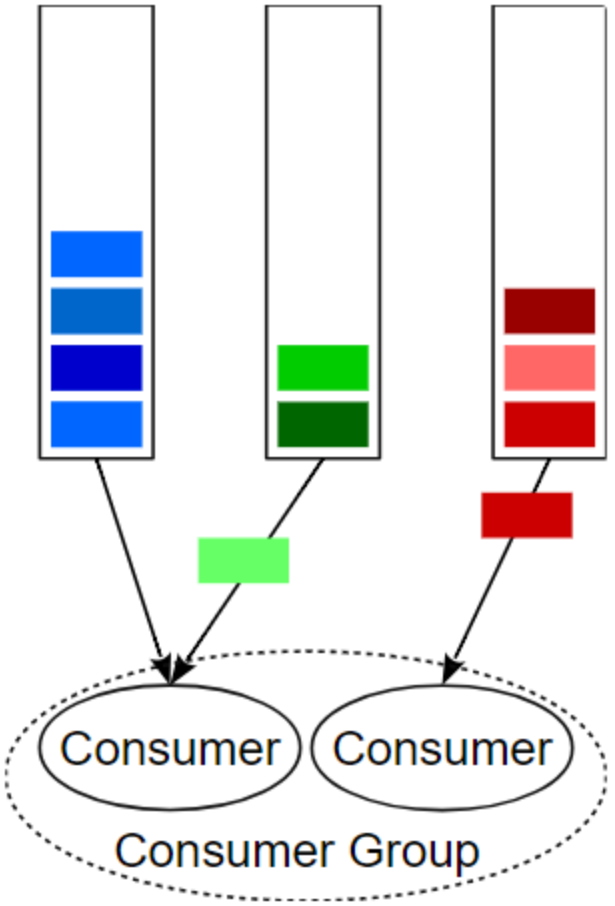
```
bin\windows\kafka-console-producer.bat
--broker-list localhost:9092
--topic test
```




https://kafka.apache.org/quickstart

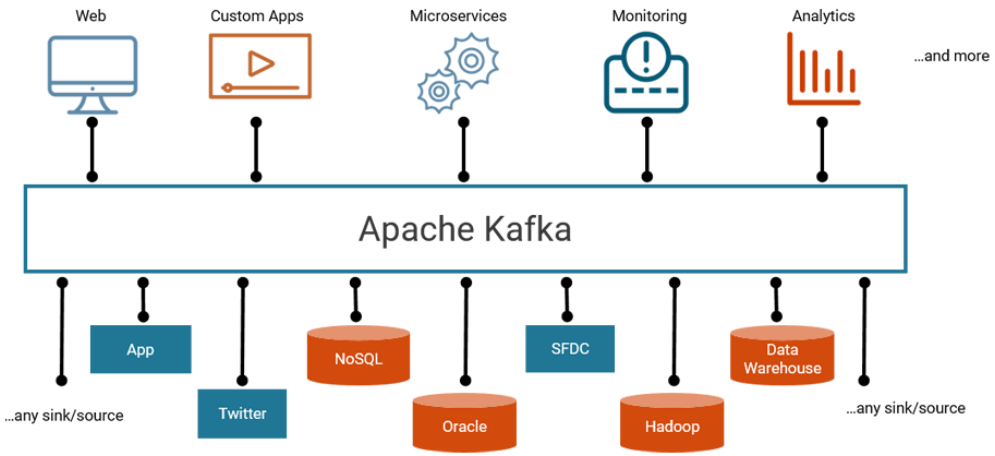
# Run consumer

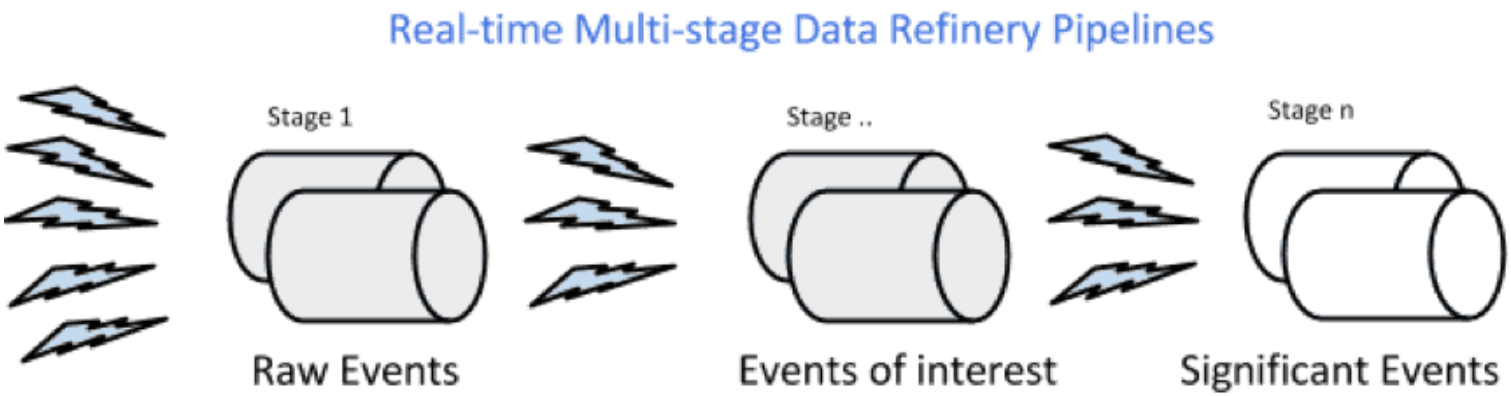
```
bin/kafka-console-consumer.sh
--bootstrap-server localhost:9092
--topic test --from-beginning

bin\windows\kafka-console-consumer.bat
--bootstrap-server localhost:9092
--topic test --from-beginning
```




https://kafka.apache.org/quickstart

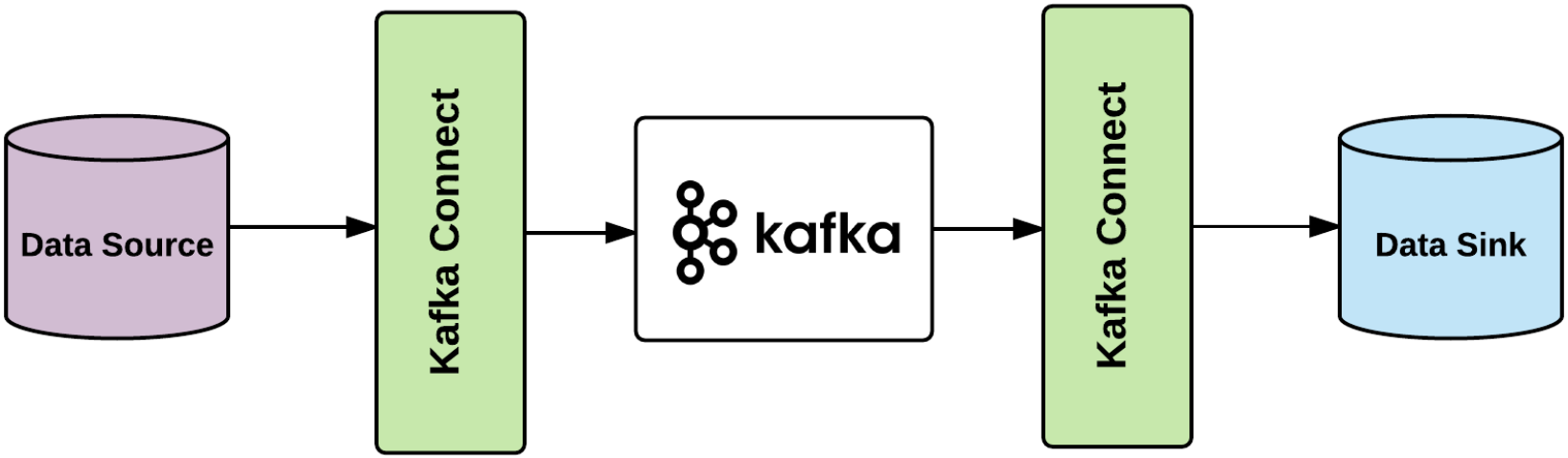
# Pipeline & integration




https://hortonworks.com/blog/storm-kafka-together-real-time-data-refinery/

# Kafka connect




https://kafka.apache.org/documentation/#connect
https://www.confluent.io/product/connectors/
https://docs.confluent.io/current/connect/quickstart.html

# From file to kafka

## sink

```
name=local-file-source
connector.class=FileStreamSource
tasks.max=1
file=c:\\tmp\\my-test.txt
topic=my-connect-test1
```

## “connector”

```
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter


# always want to use the built-in default. Offset and config data is never visible outside of Kafka Connect in this format.
internal.key.converter=org.apache.kafka.connect.json.JsonConverter
internal.value.converter=org.apache.kafka.connect.json.JsonConverter
internal.key.converter.schemas.enable=false
internal.value.converter.schemas.enable=false

offset.storage.file.filename=c:\\tmp\\connect.offsets
offset.flush.interval.ms=1000
```

```
bin\windows\connect-standalone.bat my-standalone1.properties my-file-source.properties
bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic my-connect-test1 --from-beginning
```

# From kafka to file

## sink

```
name=local-file-sink
connector.class=FileStreamSink
tasks.max=1
file=c:\tmp\test.sink.txt
topics=my-connect-test
```

## “connector”

```
bootstrap.servers=localhost:9092

key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter


internal.key.converter=org.apache.kafka.connect.json.JsonConverter
internal.value.converter=org.apache.kafka.connect.json.JsonConverter
internal.key.converter.schemas.enable=false
internal.value.converter.schemas.enable=false

offset.storage.file.filename=c:\tmp\connect.offsets
# Flush much faster than normal, which is useful for testing/debugging
offset.flush.interval.ms=10000
```

```
bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic my-connect-test
bin\windows\connect-standalone.bat my-standalone.properties my-file-sink.properties
```

# Simple producer

```
public class Producer {

    public static void main(String[] args){
        Properties properties = new Properties();
        properties.put("bootstrap.servers", "localhost:9092");
        properties.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        properties.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        KafkaProducer kafkaProducer = new KafkaProducer(properties);
        try{
            for(int i = 0; i < 100; i++){
                System.out.println(i);
                kafkaProducer.send(new ProducerRecord("devglan-test", Integer.toString(i), "test message - " +
            }
        }catch (Exception e){
            e.printStackTrace();
        }finally {
            kafkaProducer.close();
        }
    }
}
```

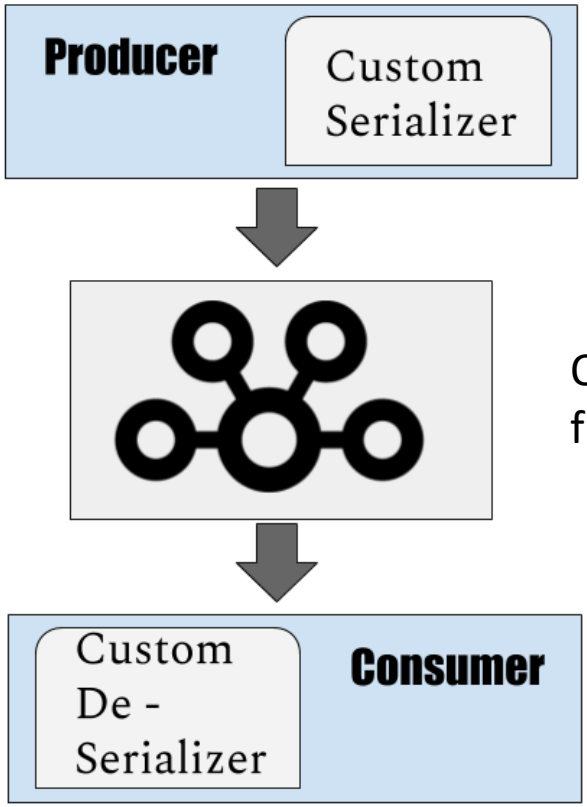
https://www.devglan.com/corejava/apache-kafka-java-example

# Simple consumer

```
public class Consumer {

    public static void main(String[] args) {
        Properties properties = new Properties();
        properties.put("bootstrap.servers", "localhost:9092");
        properties.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        properties.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        properties.put("group.id", "test-group");

        KafkaConsumer kafkaConsumer = new KafkaConsumer(properties);
        List topics = new ArrayList();
        topics.add("devglan-test");
        kafkaConsumer.subscribe(topics);
        try{
            while (true){
                ConsumerRecords records = kafkaConsumer.poll(10);
                for (ConsumerRecord record: records){
                    System.out.println(String.format("Topic - %s, Partition - %d, Value: %s", record.topic(), record.partitio
n(), record.value()));
                }
            }
        }catch (Exception e){
            System.out.println(e.getMessage());
        }finally {
            kafkaConsumer.close();
        }
    }
```

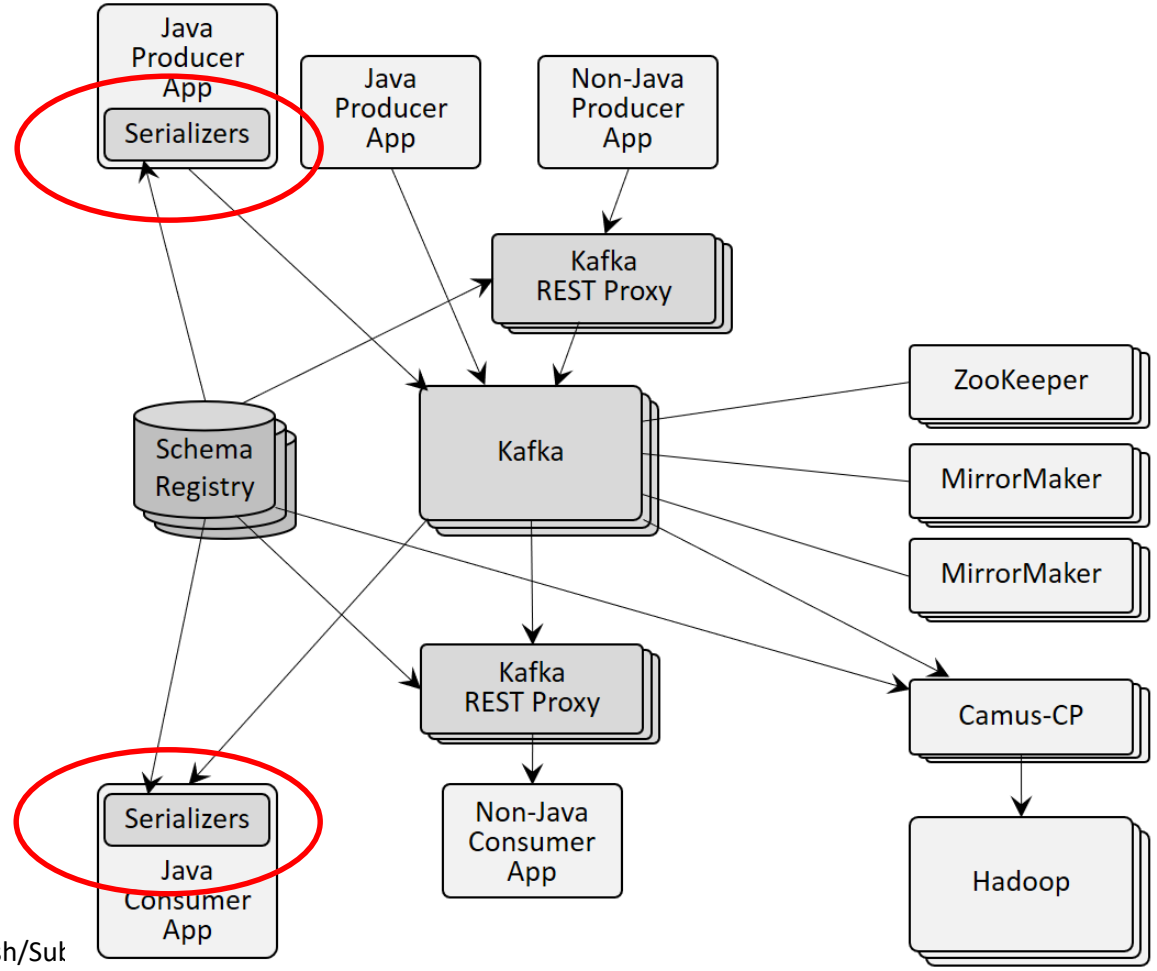
https://www.devglan.com/corejava/apache-kafka-java-example

# Kafka is data agnostic




Only knows bytes ( is filesystem based)

# Components architecture




Comparing Kafka and Solace for Publish/Sub
http://dev.solace.com/kafka/product-architecture/

# You need to provide SerDes

- SerDes (Serializer/Deserializer)
  - Needed in kafka
  - Mandatory (i.e. explicit) when using streams
- for the data types of record keys and record values
- Several existing
  - https://kafka.apache.org/10/documentation/streams/developer-guide/datatypes
- Can define your own

- Available SerDes
  - Primitive and basic types
  - Avro
  - JSON
  - Further serdes

# SERDES: Key & value (de)serialization

It (de/en) codes the KEY and VALUE

```java
public class Producer {

    public static void main(String[] args){
        Properties properties = new Properties();
        properties.put("bootstrap.servers", "localhost:9092");
        properties.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        properties.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        KafkaProducer kafkaProducer = new KafkaProducer(properties);
        try{
            for(int i
                System
                kafkaP
            }
        }catch (Except
            e.printSta
        }finally {
            kafkaProdu
        }
    }
}
```

```java
public class Consumer {

    public static void main(String[] args) {
        Properties properties = new Properties();
        properties.put("bootstrap.servers", "localhost:9092");
        properties.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        properties.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        properties.put("group.id", "test-group");

        KafkaConsumer kafkaConsumer = new KafkaConsume
        List topics = new ArrayList();
        topics.add("devglan-test");
        kafkaConsumer.subscribe(topics);
        try{
            while (true){
                ConsumerRecords records = kafkaConsumer.poll(10);
                for (ConsumerRecord record: records){
                    System.out.println(String.format("Topic - %s, Partition - %d, Value: %s", record.topic(), re
n(), record.value()));
```

SIMPLE… using just strings


# How to Create Serializers With Kafka

Kafka provides the ability to publish and subscribe to streams of records. You can create your own custom serializer and deserializer.

by Prabhat Kashyap MVB · Jan. 31, 17 · Integration Zone · Tutorial

Like (6) Comment (1) Save Tweet 20.71k Views



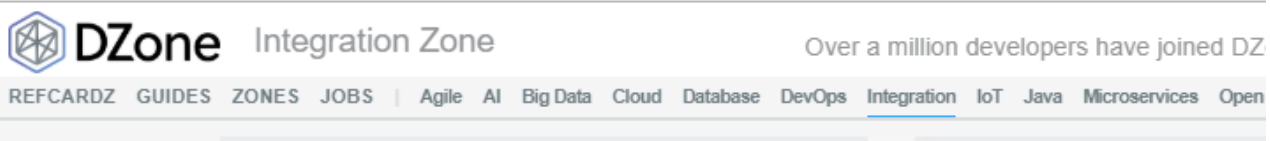
Kafka lets us publish and subscribe to streams of records and the records can be of any type (JSON, String, POJO, etc.) Kafka gives users the ability to creates our own serializer and deserializer so that we can transmit different data type using it.

In this article, I will demonstrate how to create a custom serializer and deserializer — but first, let's understand what serialization is and why we would serialize.

## Serialization and Deserialization

r transmission.

How to Create Serializers With Kafka
https://dzone.com/articles/kafka-sending-object-as-a-message




```java
public class UserSerializer implements Serializer {

  @Override public void configure(Map<String, ?> map, boolean b) {

  }

  @Override public
    byte[] retVal
    ObjectMapper
    try {
      retVal = obj
    } catch (Excep
      e.printStack
    }
    return retVal;
  }

  @Override public

  }

}
```

```java
public class UserDeserializer implements Deserializer {

  @Override public void close() {

  }

  @Override public void configure(Map<String, ?> arg0, boolean arg1) {

  }

  @Override
  public User deserialize(String arg0, byte[] arg1) {
    ObjectMapper mapper = new ObjectMapper();
    User user = null;
    try {
      user = mapper.readValue(arg1, User.class);
    } catch (Exception e) {

      e.printStackTrace();
    }
    return user;
  }

}
```

Kafka lets us publish and subsc
POJO, etc.) Kafka gives users the
transmit different data type usi

In this article, I will demonstrate how to create a custom serializer and deserializer — but first, let's understand what serialization is and why w

```
props.put("value.serializer", "com.knoldus.serializers.UserSerializer");
```

r transmission.

How to Create Serializers With Kafka
https://dzone.com/articles/kafka-sending-object-as-a-message

# If curious… or need…

- How to Create Serializers With Kafka
  - https://dzone.com/articles/kafka-sending-object-as-a-message
- Kafka tutorial #3 - JSON SerDes
  - https://aseigneurin.github.io/2018/08/01/kafka-tutorial-3-json-serdes.html
- Kafka tutorial #7 - Kafka Streams SerDes and Avro
  - https://aseigneurin.github.io/2018/08/06/kafka-tutorial-7-kafka-streams-serdes-and-avro.html
- Robust Message Serialization in Apache Kafka Using Apache Avro, Part 1
  - http://blog.cloudera.com/blog/2018/07/robust-message-serialization-in-apache-kafka-using-apache-avro-part-1/

# Some notes

- Issues due to access to port or correct ip address
  - Zookeeper, kafka
  - Within network, network vs container…

- Typical: Leader Not Available Kafka in Console Producer
  - https://stackoverflow.com/questions/35788697/leader-not-available-kafka-in-console-producer
  - (…)
  - It could be related to advertised.host.name setting in your server.properties.
  - (…)
  - …what worked for me was this advertised.listeners=PLAINTEXT://my.ip:9092

payara

Home About Services Commu

Blog Archive Blog Español 日本のブログ Beginner's Guides Productic

Java EE

# Payara Micro JCA Adapters - Apache Kafka

08 Jun 2017

Tweet Share Like 0 Share G+

by Matthew Gill

In this blog, which follows on from the introduction to Cloud Connectors in Payara Micro, we'll guide you through the process of setting up Payara Micro as a *Kafka Client*, which can produce and consume data from Apache Kafka.

## What is Apache Kafka?

Put simply, Apache Kafka is a publish-subscribe messaging system. It is similar to a

https://blog.payara.fish/payara-micro-jca-adapters-apache-kafka

prioritises the message log over consumers, and because of this is optimised to process

payara

Home About Services Commu

Java EE

Blog Archive Blog Español 日本のブログ Beginner's Guides Productic

Payara

08 Jun 2017

Tweet in Shar

In this blog, v
Micro, we'll g
*Client*, which

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "clientId",
        propertyValue = "testClient"),
    @ActivationConfigProperty(propertyName = "groupIdConfig",
        propertyValue = "test-consumer-group"),
    @ActivationConfigProperty(propertyName = "topics",
        propertyValue = "testing"),
    @ActivationConfigProperty(propertyName = "bootstrapServersConfig",
        propertyValue = "192.168.29.139:9092"),
    @ActivationConfigProperty(propertyName = "autoCommitInterval",
        propertyValue = "100"),
    @ActivationConfigProperty(propertyName = "retryBackoff",
        propertyValue = "1000"),
    @ActivationConfigProperty(propertyName = "keyDeserializer",
        propertyValue = "org.apache.kafka.common.serialization.StringDeserializer"),
    @ActivationConfigProperty(propertyName = "valueDeserializer",
        propertyValue = "org.apache.kafka.common.serialization.StringDeserializer"),
    @ActivationConfigProperty(propertyName = "pollInterval", propertyValue = "1000"),
})
public class KafkaMDB implements KafkaListener {

    @OnRecord( topics={"testing"})
    public void getMessageTest(ConsumerRecord record) {
        System.out.println("Got record on topic testing " + record);
    }
}
```

## What is Apache Kafka?

Put simply, Apache Kafka is a publish-subscribe messaging system. It is similar to a

prioritises the message log over consumers, and because of this is optimised to process

https://blog.payara.fish/payara-micro-jca-adapters-apache-kafka

payara

Home About Services Commu

Java EE

## Deploying to Payara Micro

You need the following three files to make this work:

1. Payara Micro Jar (obviously).
2. Kafka Example Jar (the project we've created).
3. Kafka RAR Adapter (the JCA connector project).

The latter two should be in the directory tree shown below (if you can't find these files then you may not have built the projects with "**mvn package**").

```
└── target
    ├── dependency
    │   └── kafka-rar-0.1.0.rar
    └── kafka-tutorial-example-1.0-SNAPSHOT.war
```

To deploy these to Payara Micro, run the following command.

```
java -jar payara-micro.jar --deploy target/dependency/kafka-rar-0.1.0.rar --dep
```

## What is Apache Kafka?

Put simply, Apache Kafka is a publish-subscribe messaging system. It is similar to a

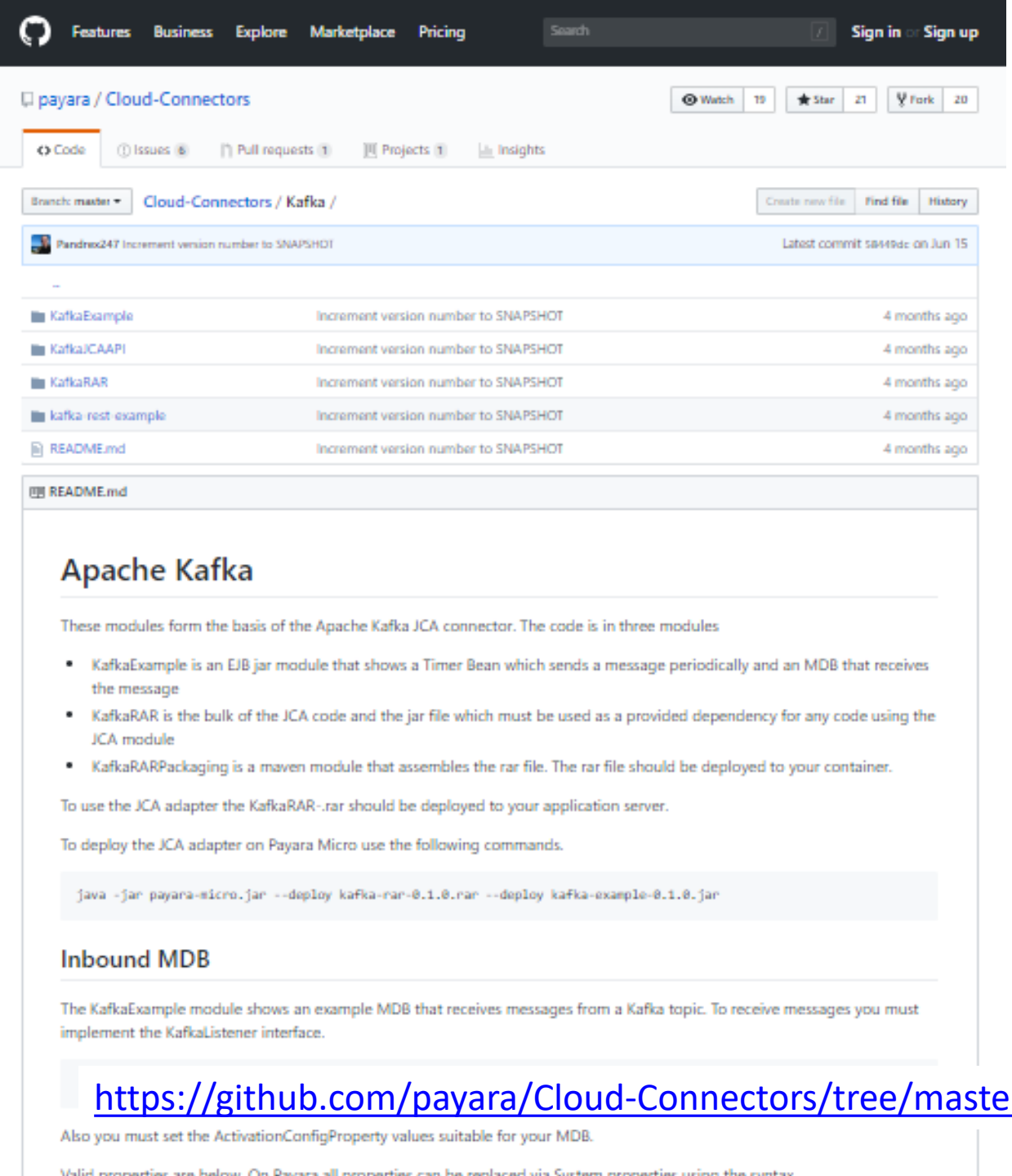
https://blog.payara.fish/payara-micro-jca-adapters-apache-kafka



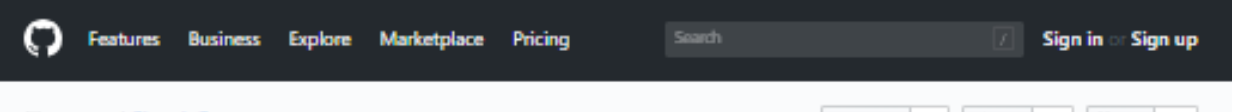
```
@ConnectionFactoryDefinition(name = "java:module/env/KafkaConnectionFactory",
        description = "Kafka Connection Factory",
        interfaceName = "fish.payara.cloud.connectors.kafka.KafkaConnectionFactory",
        resourceAdapter = "kafka-rar-0.1.0",
        minPoolSize = 2,
        maxPoolSize = 2,
        transactionSupport = TransactionSupport.TransactionSupportLevel.NoTransaction,
        properties = {
            "bootstrapServersConfig=192.168.29.139:9092",
            "clientId=PayaraMicroMessenger"
        })
@Stateless
public class KafkaMessenger {

    @Resource(lookup = "java:module/env/KafkaConnectionFactory")
    KafkaConnectionFactory factory;

    @Schedule(hour = "*", minute = "*", second = "*/5", persistent = false)
    public void sendMessage() {
        try (KafkaConnection conn = factory.createConnection()) {
            conn.send(new ProducerRecord("testing", "Sent from Payara Micro."));
        } catch (Exception ex) {
            Logger.getLogger(getClass().getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

https://github.com/payara/Cloud-Connectors/tree/master/Kafka

https://blog.payara.fish/payara-micro-jca-adapters-apache-kafka




https://github.com/payara/Cloud-Connectors/tree/master/Kafka




payara / Cloud-Connectors

Watch 19 Star 21 Fork 20

Code

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "clientId", propertyValue = "testClient"),
    @ActivationConfigProperty(propertyName = "groupIdConfig", propertyValue = "testGroup"),
    @ActivationConfigProperty(propertyName = "topics", propertyValue = "test,test2"),
    @ActivationConfigProperty(propertyName = "bootstrapServersConfig", propertyValue = "localhost:9092"),
    @ActivationConfigProperty(propertyName = "autoCommitInterval", propertyValue = "100"),
    @ActivationConfigProperty(propertyName = "retryBackoff", propertyValue = "1000"),
    @ActivationConfigProperty(propertyName = "keyDeserializer", propertyValue = "org.apache.kafka.common.
    @ActivationConfigProperty(propertyName = "valueDeserializer", propertyValue = "org.apache.kafka.commo
    @ActivationConfigProperty(propertyName = "pollInterval", propertyValue = "3000"),
})
public class KafkaMDB implements KafkaListener {

    public KafkaMDB() {
    }

    @OnRecords( matchOtherMethods = true)
    public void getMessageTest2(ConsumerRecords records) {
        System.out.println("Bulk processing called with " + records.count() + " records");
    }

    @OnRecord( topics={"test"})
    public void getMessageTest(ConsumerRecord record) {
        System.out.println("Got record on topic test " + record);
    }

    @OnRecord( topics={"test2"})
    public void getMessageTest2(ConsumerRecord record) {
        System.out.println("Got record on topic test2 " + record);
    }

}
```

spring boot

Projects
Spring Boot
Spring Framework
Spring Data
Spring Cloud
Spring Cloud Data Flow
Spring Security
Spring Session
Spring Integration
Spring HATEOAS
Spring REST Docs
Spring Batch
Spring IO Platform
Spring AMQP
Spring for Android
Spring CredHub
Spring for Apache Kafka
Spring LDAP
Spring Mobile
Spring Roo
Spring Shell
Spring Vault

# Spring for Apache Kafka 2.1.10

Overview Learn

The Spring for Apache Kafka (spring-kafka) project applies core Spring concepts to the development of Kafka-based messaging solutions. It provides a "template" as a high-level abstraction for sending messages. It also provides support for Message-driven POJOs with `@KafkaListener` annotations and a "listener container". These libraries promote the use of dependency injection and declarative. In all of these cases, you will see similarities to the JMS support in the Spring Framework and RabbitMQ support in Spring AMQP.

## Features

- `KafkaTemplate`
- `KafkaMessageListenerContainer`
- `@KafkaListener`
- `KafkaTransactionManager`
- `spring-kafka-test` jar with embedded kafka server

## Kafka Client Compatibility

Spring for Apache Kafka is based on the pure java `kafka-clients` jar. The following is the compatibility matrix:

| Spring for Apache Kafka Version | Spring Integration for Apache Kafka Version | `kafka-clients` |
|---|---|---|
| 2.2.x | 3.1.x | 2.0.0 |
| 2.2.0.M1 | 3.1.0.M1 | 1.1.x |
| | | 1.0.x, 1.1.x |
| | | 0.11.0.x, 1.0.x |
| | | 0.11.0.x, 1.0.x |
| 1.2.x | 2.2.x | 0.10.2.x |

https://spring.io/projects/spring-kafka

spring boot

# Getting Started with Kafka in Spring Boot


Bartosz Jedrzejewski

May 21, 2018

Choreography, Microservices, Spring Boot

Kafka seems to only be gaining in popularity. A few years ago you could mostly see it in Big Data engineering context. These days, Kafka is starting to power more common message-oriented architectures. In this article, I want to give you a basic introduction to working with Spring Boot and Kafka.

## Installing Kafka on your machine

One common barrier to entry for people to *hack around* on their machines with Kafka is how tricky the installation can be. For that very reason, I have written *"How to easily run Kafka with Docker for development"* – I think you will find it especially useful if you are on Windows!

If you are running macOS or Linux, you can still follow the aforementioned tutorial, but you could also run Kafka and Zookeeper without Docker. To do that I recommend Confluent.io platform and their Quick Start tutorial.

## Enabling Kafka in Spring Boot

Assuming that you have Kafka accessible on *kafka:9092* what follows is basic instruction on integrating your Spring Boot application with Kafka.

With Spring Boot, to use Kafka, you need a single dependency added to your POM file

```
1 <dependency>
```



### ABOUT

E4developer is a place where I share my open and honest views on software development, technology and working with people. The name – e4 comes from a chess move, this is how I start most of my games. Follow me on twitter – @e4developer

I also blog at Scott Logic – a great consulting company where I work as a Lead Developer.

When it comes to my chess passion, I also run chesscollecting.com where you can see some beautiful chess sets.

### CATEGORIES

- Algorithms (3)
- Architecture (20)
- Books (10)
- Building teams (11)
- Career (22)
- Chess (2)
- Choreography (8)
- DevOps (4)
- Java (13)
- Microservices (47)
- Newsletter (5)
- Orchestration (4)
- Personal (13)
- Public speaking (3)
- Reactive (7)
- Spring Boot (17)
- Spring Cloud (11)
- Spring Cloud Data Flow (4)

https://www.e4developer.com/2018/05/21/getting-started-with-kafka-in-spring-boot/

JEDRZEJEWSKI

E4developer
Java Microservices Blog

Start Here Blog Newsletter – Sign Up Cc

spring boot

## Getting Started with

```
Application.properties

spring.kafka.consumer.group-id=kafka-intro
spring.kafka.bootstrap-servers=kafka:9092
```

APACHE kafka®
A distributed streaming platform

```
@Autowired
private KafkaTemplate<String, String> kafkaTemplate;

public void send(String topic, String payload) {
    kafkaTemplate.send(topic, payload);
    System.out.println("Message: "+payload+" sent to topic: "+topic);
}
```

Bartosz Jedrzejewski
May 21, 2018
Choreography, Microser
Spring Boot

ABOUT

E4developer is a place where I share my open

how tricky the installation can be. For that very reason, I have written *"How to easily run Kafka with Docker for development"*– I think you will find it especially useful if you are on Windows!

If you are running macOS or Linux, you can still follow the aforementioned tutorial, but you could also run Kafka and Zookeeper without Docker. To do that I recommend

- Algorithms (3)
- Architecture (20)
- Books (10)
- Building teams (11)
- Career (22)
- Chess (2)
- Choreography (8)

```
@KafkaListener(topics = "topic1")
public void receiveTopic1(ConsumerRecord<?, ?> consumerRecord) {
    System.out.println("Receiver on topic1: "+consumerRecord.toString());
}
```

- Spring Cloud Data Flow (4)

With Spring Boot, to use Kafka, you need a single dependency added to your POM file

```
1 <dependency>
```

JEDRZEJEWSKI

https://www.e4developer.com/2018/05/21/getting-started-with-kafka-in-spring-boot/

E4developer
Java Microservices Blog

Start Here Blog Newsletter – Sign Up Cc

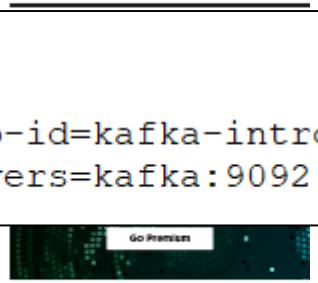
spring boot

## Getting Started with

```
Application.properties

spring.kafka.consumer.group-id=kafka-intro
spring.kafka.bootstrap-servers=kafka:9092
```


```
@Autowired
private KafkaTemplate<String, String> kafkaTemplate;

public void send(String topic, String payload) {
    kafkaTemplate.send(topic, payload);
    System.out.println("Message: "+payload+" sent to topic: "+topic);
}
```

Bartosz Jedrzejewski
May 21, 2018
Choreography, Microser
Spring Boot



ABOUT

E4developer is a place where I share my open

how tricky the installation can be. For that very reason, I have written *"How to easily run Kafka with Docker for development"*– I think you will find it especially useful if you are on Windows!

If you are running macOS or Linux, you can still follow the aforementioned tutorial, but you could also run Kafka and Zookeeper without Docker. To do that I recommend

- Algorithms (3)
- Architecture (20)
- Books (10)
- Building teams (11)
- Career (22)
- Chess (2)
- Choreography (8)

```
@KafkaListener(topics = "topic1")
public void receiveTopic1(ConsumerRecord<?, ?> consumerRecord) {
    System.out.println("Receiver on topic1: "+consumerRecord.toString());
}
```

- Spring Cloud Data Flow (4)

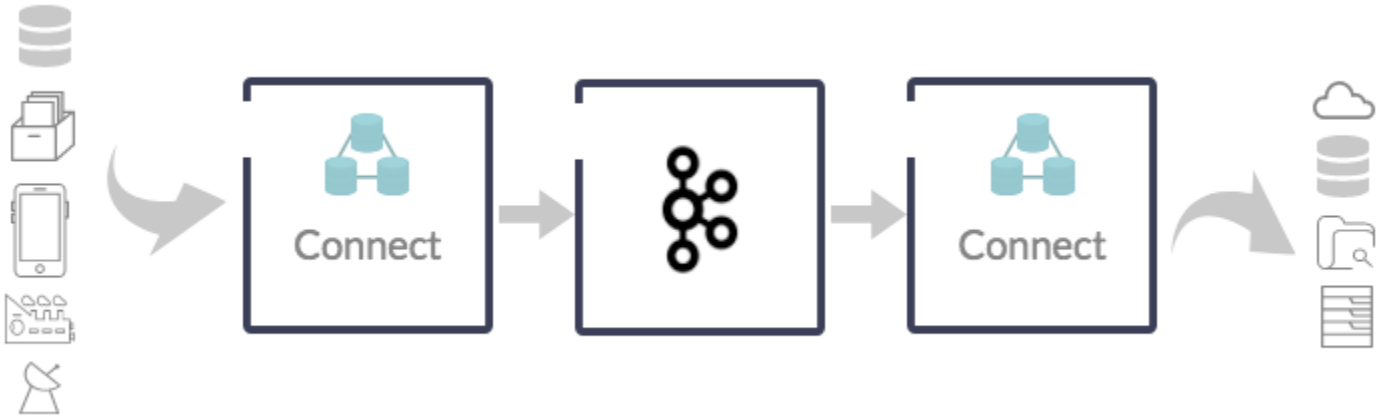
With Spring Boot, to use Kafka, you need a single dependency added to your POM file

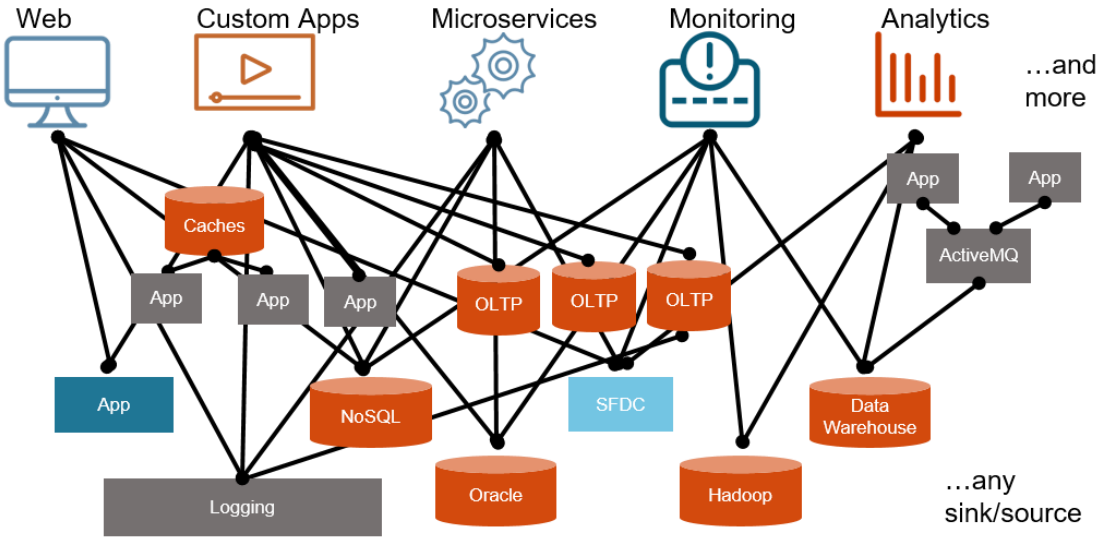
```
1 <dependency>
```

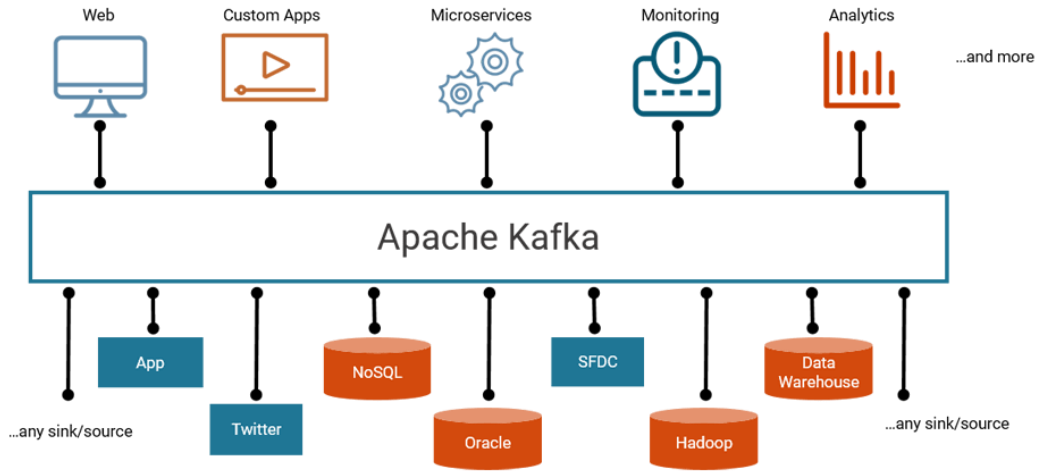
JEDRZEJEWSKI

https://www.e4developer.com/2018/05/21/getting-started-with-kafka-in-spring-boot/

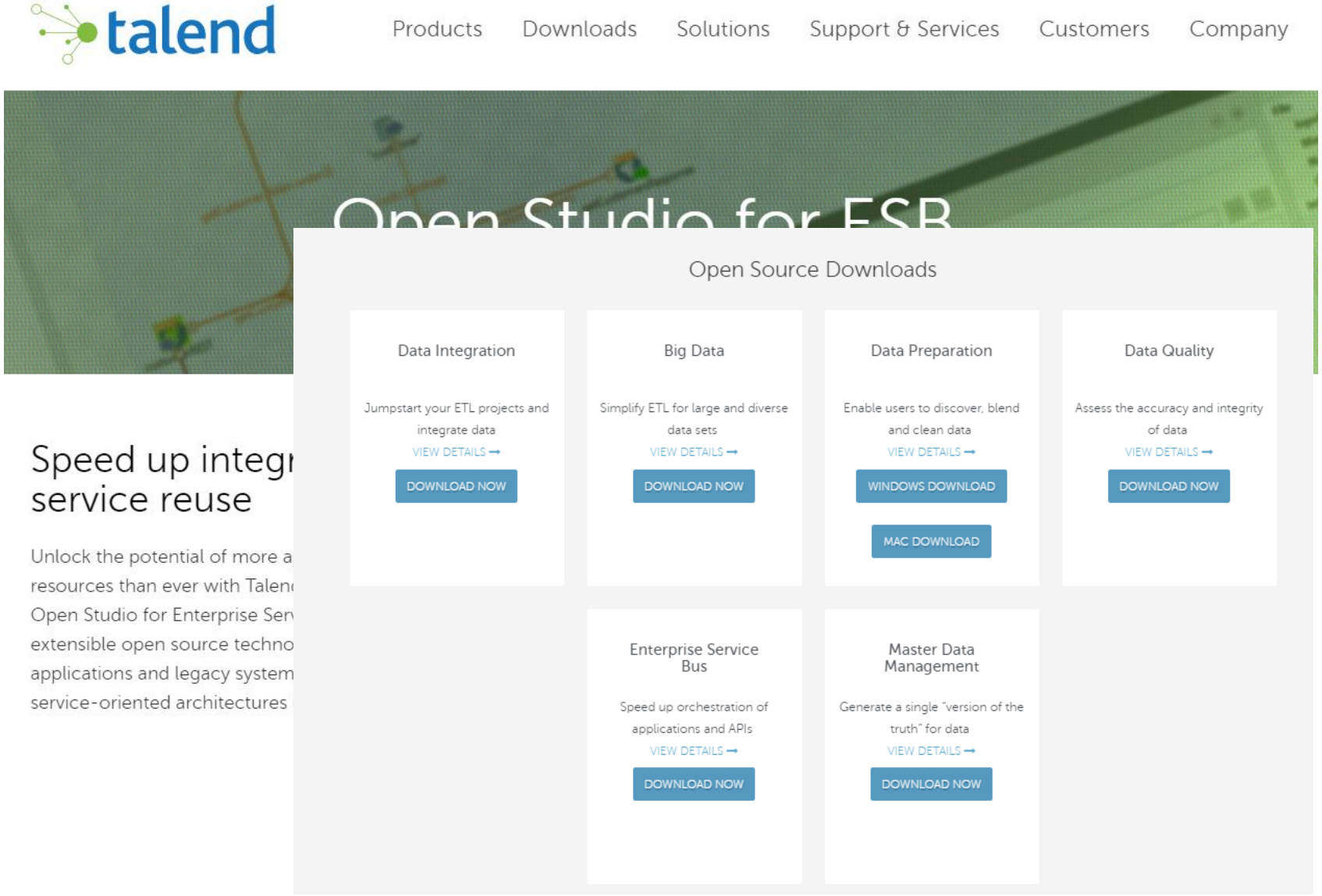
# “connectors”






https://www.confluent.io/product/confluent-platform/




https://www.talend.com/products/application-integration/esb-open-studio/

confluent

Product Cloud Developers Blog Docs Download

# Confluent Open Source

A developer-optimized distribution of Apache Kafka®. 100% open source.

Confluent, founded by the original creators of Apache Kafka® at LinkedIn, created a more complete distribution of Kafka, adding schema management, connectors, clients, and the highest level of testing in the industry.

Download


## Why Confluent Open Source?


https://www.confluent.io/product/confluent-open-source/

More Languages

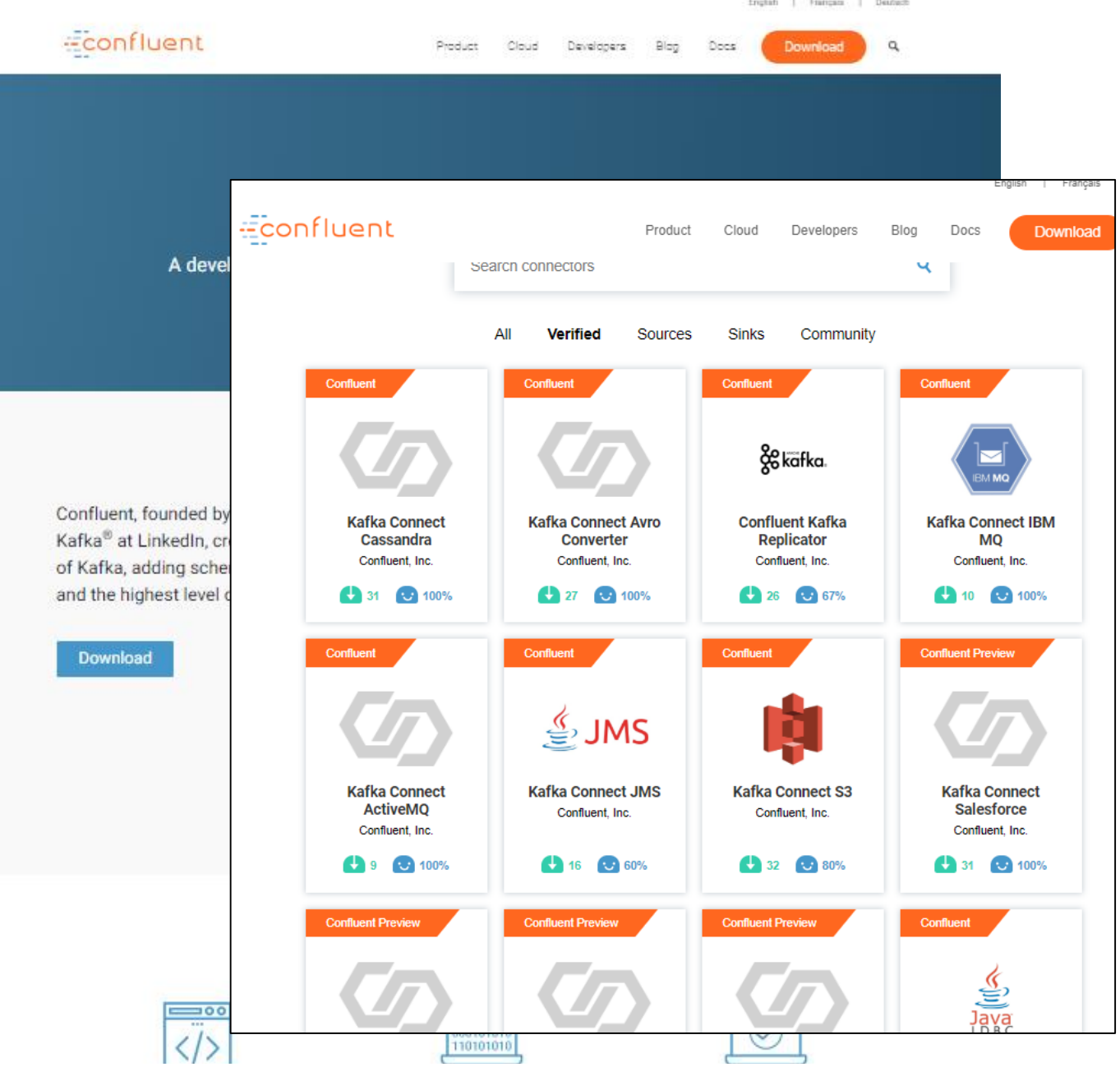
The majority of teams use two or

Better Data

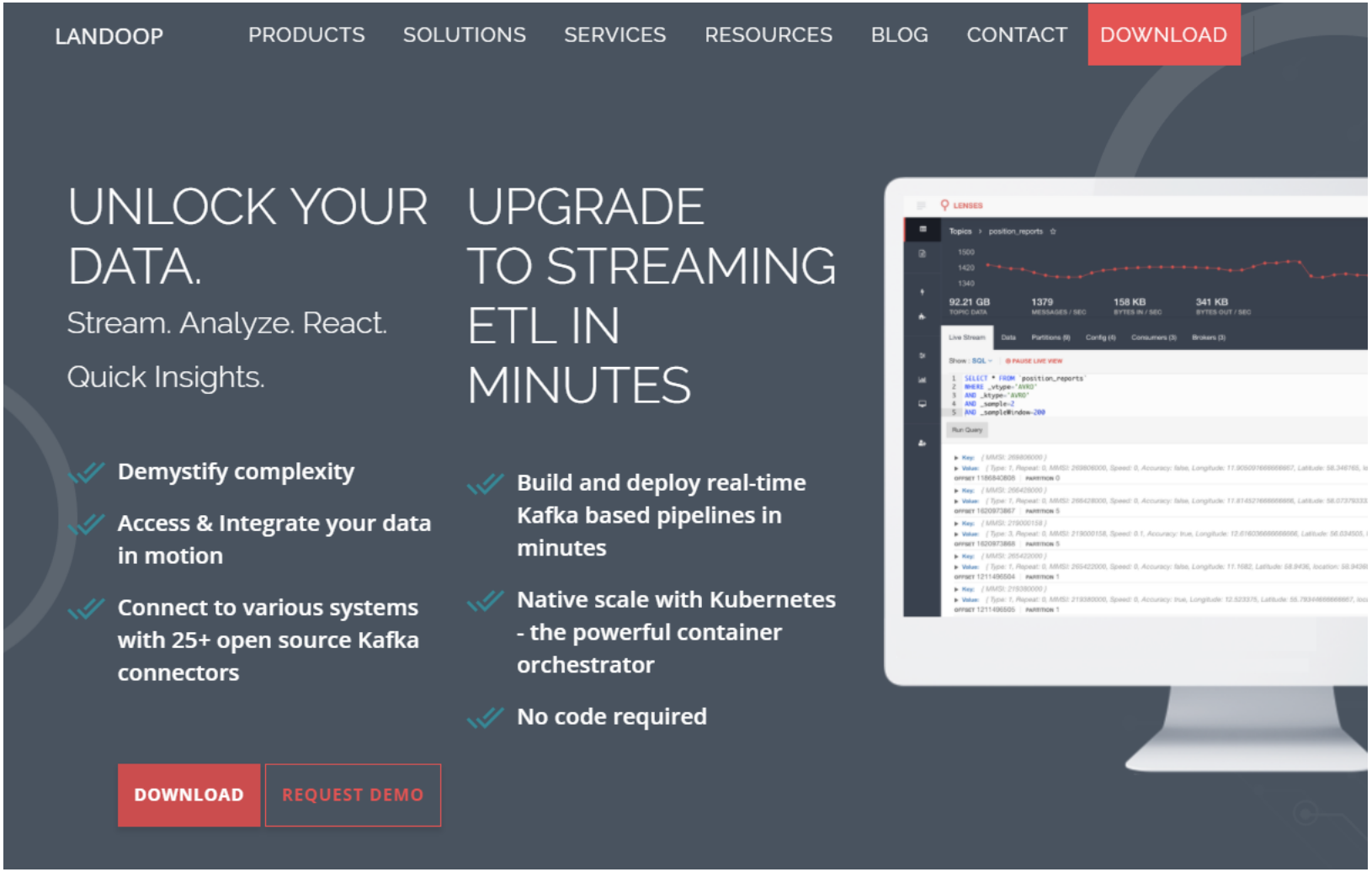
Schema registry sets schemas for the

Fewer Problems

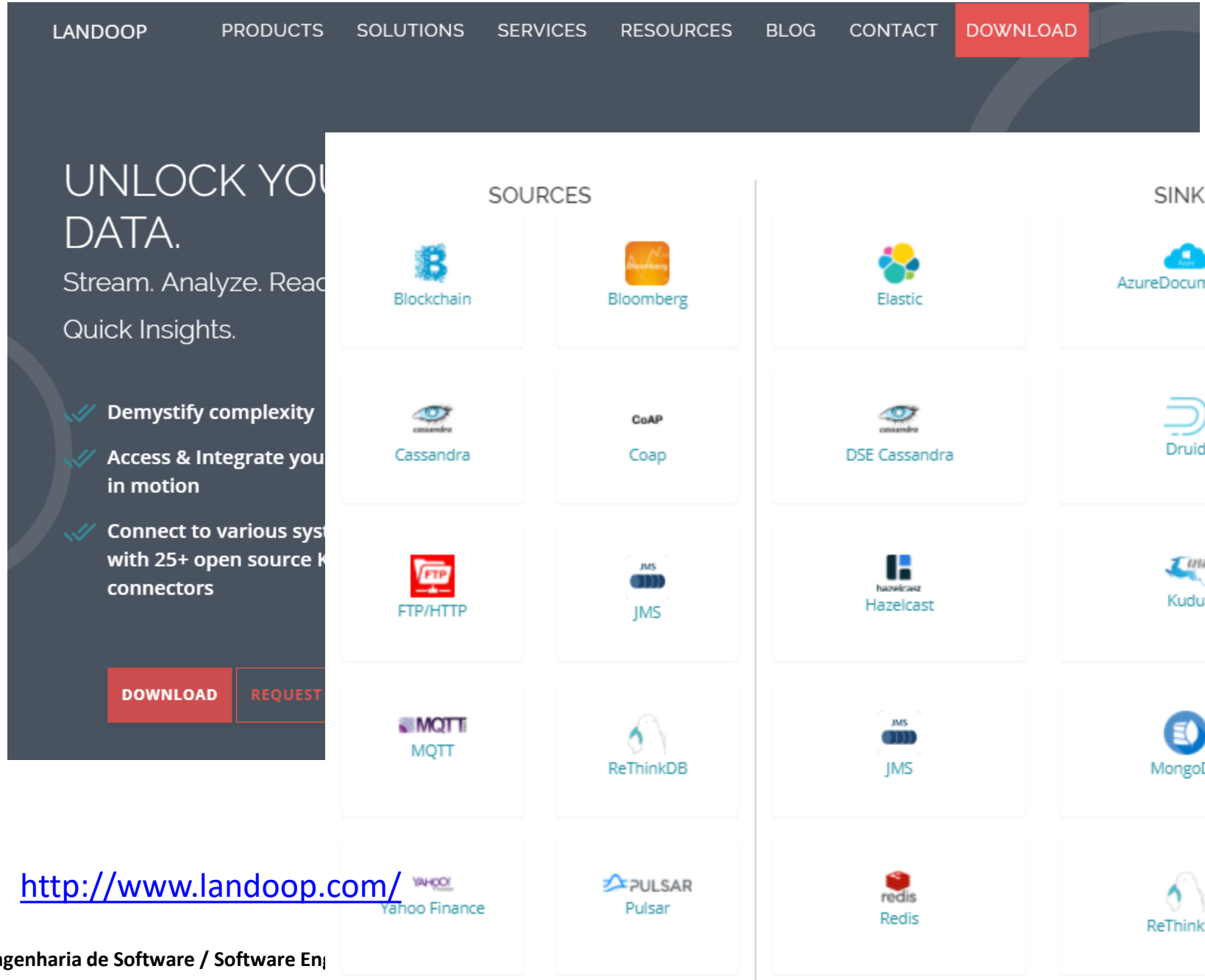
Packaged by the Kafka experts at




https://www.confluent.io/product/confluent-open-source/




http://www.landoop.com/
http://www.landoop.com/kafka-lenses/

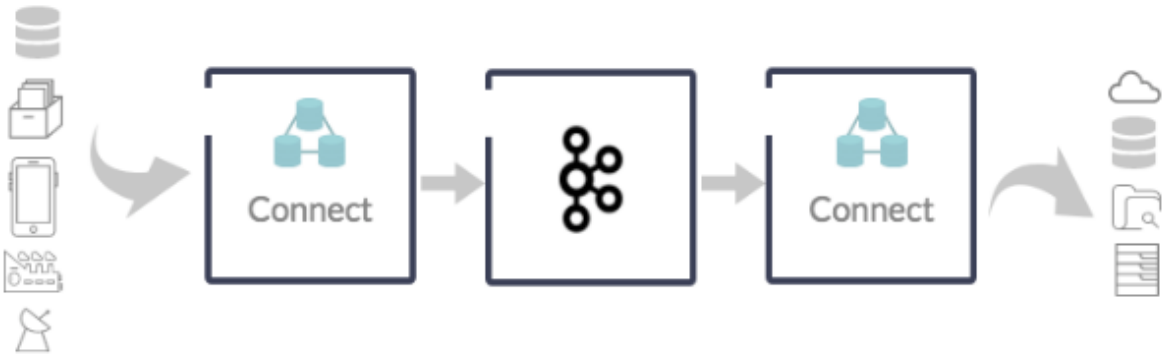



http://www.landoop.com/

# Connectors

Kafka Connect is a tool to rapidly stream events in and out of Kafka. It has a narrow focus on data ingress in and egress out of the central nervous system of modern streaming frameworks. Connect allows developers to quickly build robust, durable and scalable pipelines in and out of Kafka.

Landoop provides a large collection of Connectors bundled in the Stream Reactor. This open source set of components can be used as the entry point for building your streaming data flows.




Kafka Connect can run in standalone or distributed mode. The former is targetting testing or an one-off jobs whereas the later provides a scalable, fault-tolerant service supporting an entire organization. Connect can scale down for development, to support production workloads.

https://lenses.stream/1.1/connectors/index.html

LANDOOP DOCS / LENSES

PRODUCT SER CONTACT

v1.1 v2.1 v2.0 v1.0

LENSES 1.1

Search docs

- Installation & Setup
- User Guide
- Developer Guide
- Lenses SQL Engine
- Connectors
  - Connect CLI
  - Source Connector
  - Sink Connectors
  - Cloudera Integrati
  - Connector Install
  - FAQ
- Release Notes

Source connectors load

- Blockchain Source
- Bloomberg Source
- Cassandra Source
- Cassandra CDC
- CoAP Source
- FTP Source
- JMS Source
- Mqtt Source
- Pulsar Source
- ReThink Source

Connect

Sink connectors stream data f

- Azure DocumentDB Sink
- Cassandra Sink
- CoAP Sink
- Elastic Sink
- Elastic 5 Sink
- HazelCast Sink
- HBase Sink
- Influx Sink
- JMS Sink
- Kudu Sink
- Mongo Sink
- MQTT Sink
- Pulsar Sink
- Redis Sink
- ReThink Sink
- VoltDB Sink

https://lenses.stream/1.1/connectors/index.html

payara

Buy Support Online

Home About Services Community Blog Downloads Co

Blog Ar

Cloud Co

02 Jun 2017

Tweet Share 0

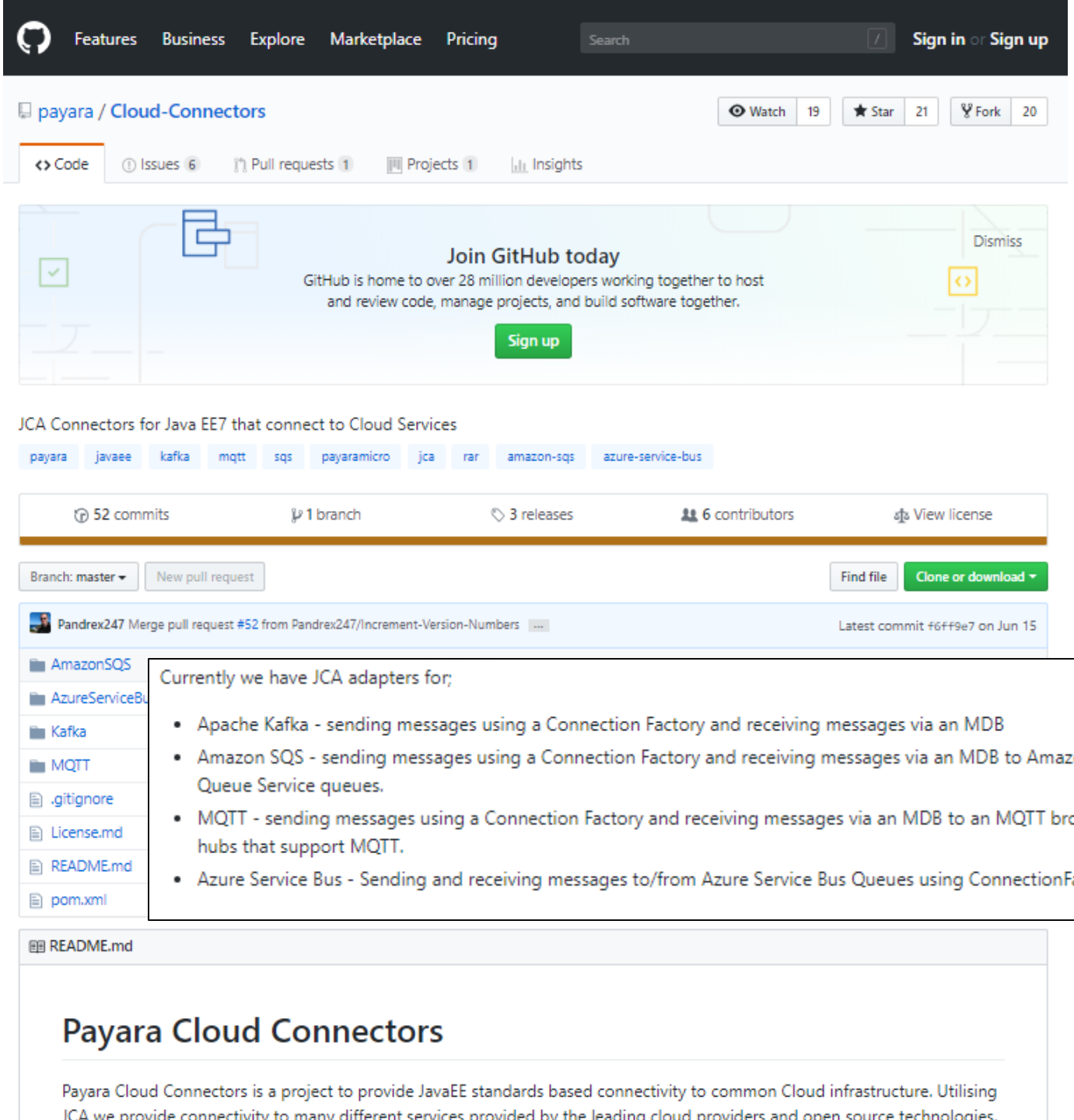
Currently we have JCA adapters for:

- **Apache Kafka** - sending messages using a Connection Factory ar receiving messages via an MDB.
- **Amazon SQS** - sending messages using a Connection Factory and messages via an MDB to Amazon Simple Queue Service queues.
- **MQTT** - sending messages using a Connection Factory and recei messages via an MDB to an MQTT broker or to IOT hubs that su MQTT.
- **Azure Service Bus** - Sending and receiving messages to/from Azu Bus Queues using ConnectionFactory via MDB.

Payara Micro 172 brings with it support for JCA adapters, meaning it can be u

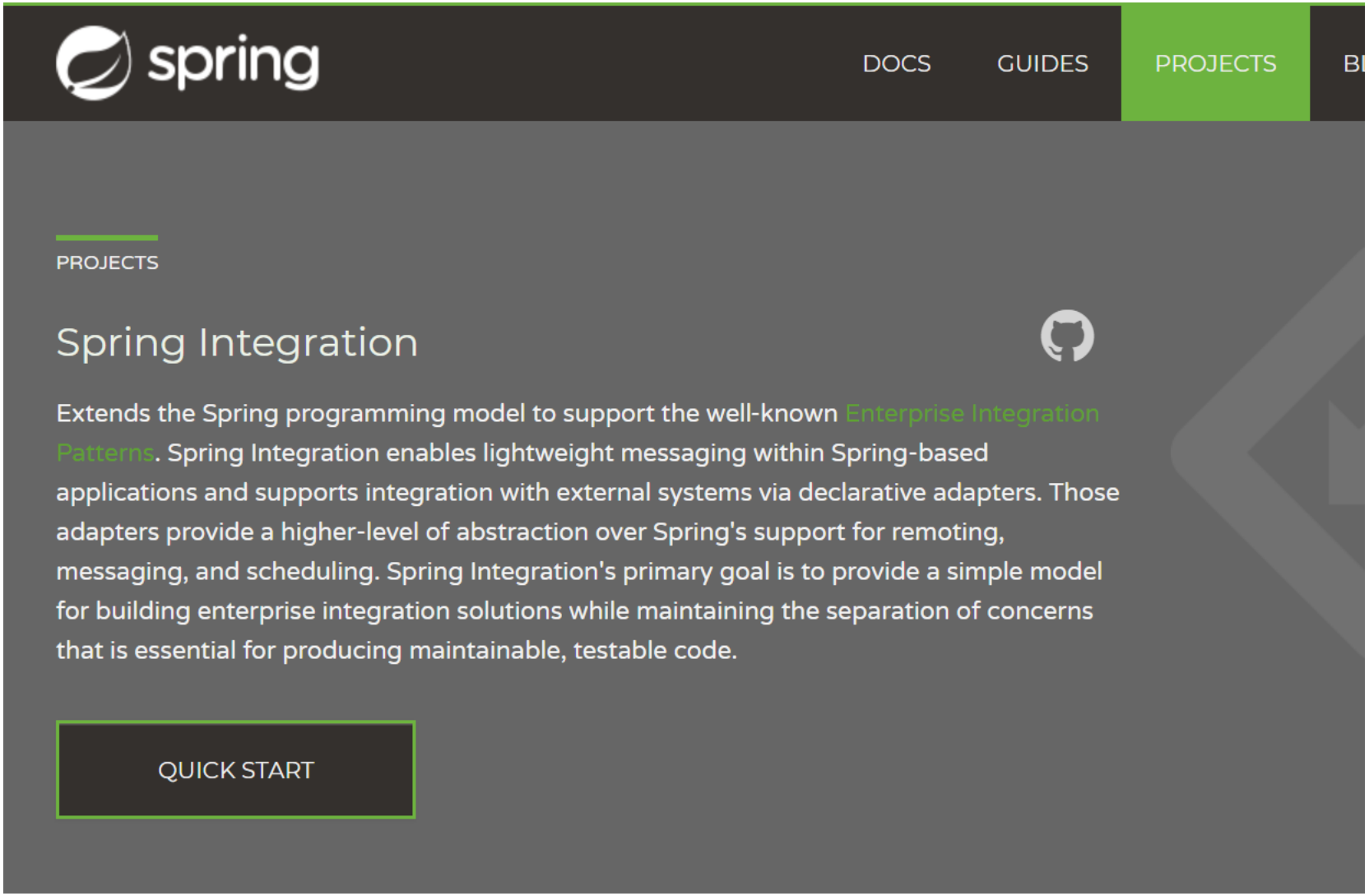
http://blog.payara.fish/cloud-connectors-in-payara-micro

Subscribe




Currently we have JCA adapters for;

- Apache Kafka - sending messages using a Connection Factory and receiving messages via an MDB
- Amazon SQS - sending messages using a Connection Factory and receiving messages via an MDB to Amazon Simple Queue Service queues.
- MQTT - sending messages using a Connection Factory and receiving messages via an MDB to an MQTT broker or to IOT hubs that support MQTT.
- Azure Service Bus - Sending and receiving messages to/from Azure Service Bus Queues using ConnectionFactory via MDB




https://projects.spring.io/spring-integration/

spring by Pivotal

PROJECTS GUIDES BLOG

# Guides

Whatever you're bui
releases and technic

Have a suggestion for

Find a guide...

Getti

Desig
Worl

Building a RESTfu

Learn how to create
with Spring.

## Building a RESTful Web Service

Learn how to create a RESTful web service with Spring.

## Scheduling Tasks

Learn how to schedule tasks with Spring.

## Consuming a RESTful Web Service

Learn how to retrieve web page data with Spring's RestTemplate.

## Building Java Projects with Gradle

Learn how to build a Java project with Gradle.

## Building Java Projects with Maven

Learn how to build a Java project with Maven.

## Accessing Relational Data using JDB with Spring

Learn how to access relational data with Spring.

## Uploading Files

Learn how to build a Spring application that accepts multi-part file uploads.

## Authenticating a User with LDAP

Learn how to secure an application with LDAP.

## Registering an Application with Facebook

Learn how to register an application to integrate with Facebook.

## Messaging with Redis

Learn how to use Redis as a message broker.

## Registering an Application with Twitter

Learn how to register apps with Twitter.

## Messaging with RabbitMQ

Learn how to create a simple publish-and-subscribe application with Spring and RabbitMQ.

## Accessing Twitter Data

Learn how to access user data from Twitter.

## Accessing Facebook Data

Learn how to access Facebook information from an application.

## Accessing Data with Neo4j

Learn how to persist objects and relationships in Neo4j's NoSQL data store.

## Validating Form Input

Learn how to perform form validation with Spring

## Building a RESTful Web Service with Spring Boot Actuator

Learn how to create a RESTful Web service with Spring Boot Actuator.

## Messaging with JMS

Learn how to publish and subscribe to messages using a JMS broker.

## Creating a Batch Service

## Securing a Web Application

## Building a Hypermedia-Driven

https://spring.io/guides

# Take home

- Messaging
  - High level solution, not only brokers
  - Supports reactive, events , decoupling
- Same (new) patterns
  - Mostly different scope
  - Conceptually equivalent
  - Cloud, soa, integration
- Don’t mix architecture & implementation
  - Mix if need asynch, synch , message , REST, …

# Messaging

- Conceptual
  - Reduce coupling
  - Focus on domain
  - Scalable
- Patterns and styles associated
  - Publisher / subscriber
  - Cqrs,  Event sourcing
  - Event Driven Architecture
  - …
- Technology agnostic
  - However commonly associated with concrete broker solutions

# Take home…

- Messaging
  - A (good) trend
  - Brokers, message, events, …
- Kafka
  - One option
  - is a worthy tool:
    - Integration – data & developer
  - Many roles and some to be invented
- Connectors
  - New trend, not only for messaging
  - Look and don't reinvent the wheel

# Just a mention

- Runtime cannot be ignored
  - limited resources Kafka runtime may be too “heavy”
  - Different versions / protocols
- Websocket
  - P2P solutions may enough
  - Similar reasoning, different solution
- MQTT solutions may be interesting
  - Mosquitto and plenty of lightweight clients
- RabbitMQ
  - Multi protocol support
  - Multi platform support
- Addressed on a need to know basis (mostly)

126Kr 126kr.com Google Search Home Programming

# ActiveMQ Confusion and What comes with your JBoss EAP / WildFly

Datetime:2017-02-22 05:26:58 Topic: Jboss Share Original >>

Here to See The Original Article!!!

Test Option Applications Performance Management Actually infolinks

| DISTRIBUTION | BROKER | VERSION | PROTOCOL SUPPORT |
|---|---|---|---|
| Artemis standalone | Artemis | 1.5 | AMQP, STOMP, MQTT, OpenWire, HornetQ Core |
| ActiveMQ 5 standalone | ActiveMQ 5 | 5.14 | AMQP, STOMP, MQTT, OpenWire |
| WildFly 8, 9 | HornetQ | 2.4 | AMQP, STOMP, HornetQ Core |
| JBoss EAP 6.4 | HornetQ | 2.3 | AMQP, STOMP, HornetQ Core |
| WildFly 10 | Artemis | 1.1 | AMQP, STOMP, HornetQ Core |
| JBoss EAP 7 | Artemis | 1.1 | HornetQ Core |

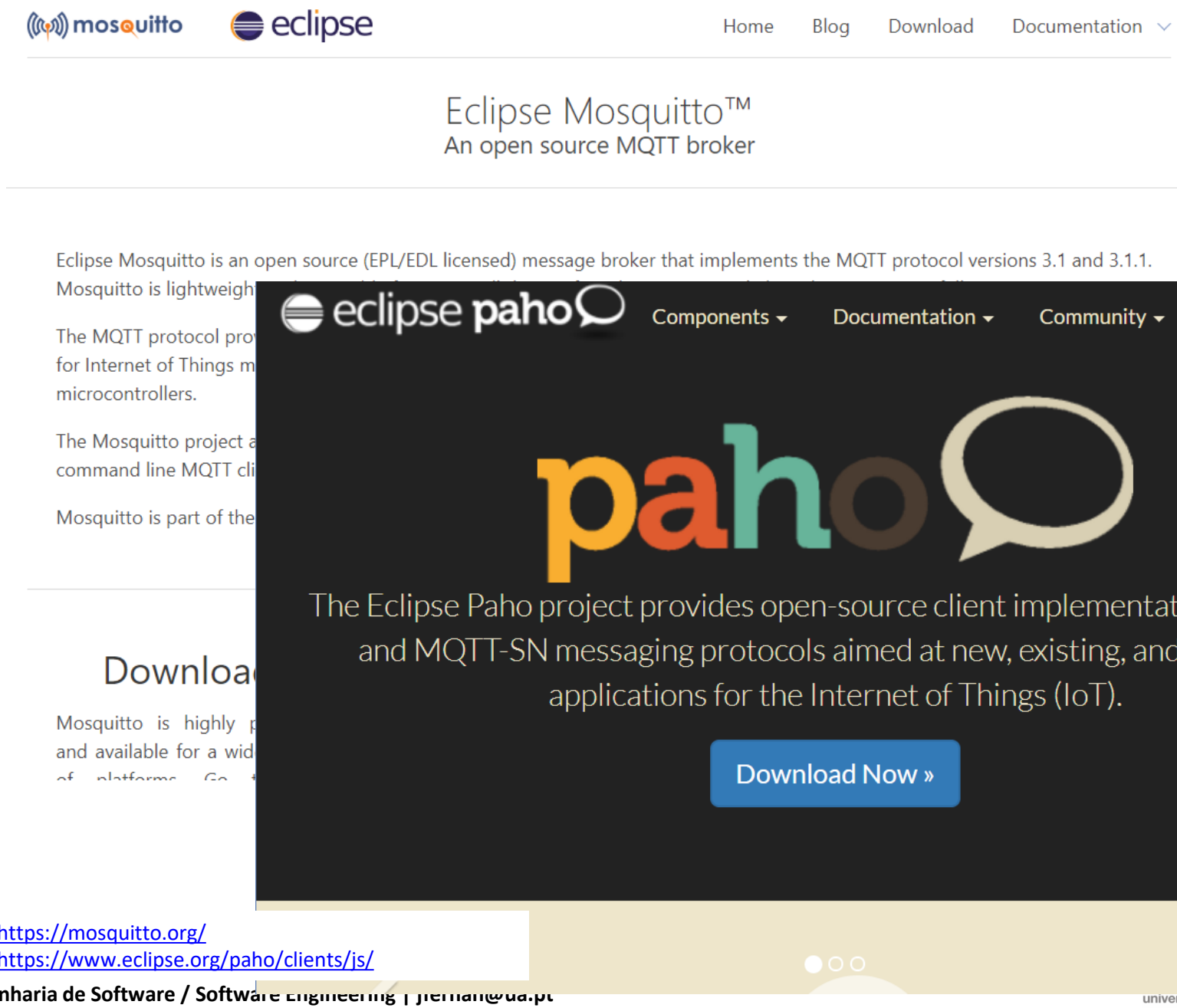
Oftentimes
to different
their name
a broker wit
name 'Activ

So there are

- ' Acti
  proje
  is use
  many
  confi
- ' Acti
  Active
  ActiveMQ 5 with some radical architecture changes to improve performance, reliability and maintainablity. This project seems to have become dormant since the introduction of the 3rd ActiveMQ project:
- ' ActiveMQ Artemis ' is not a fork or a reimplementation but rather started as the HornetQ broker. HornetQ's code was donated to the ActiveMQ project and is now developed by the developers of HornetQ under the new name ActiveMQ Artemis. So basically it's a completely different and field proven broker. Artemis is known to be rather fast for a JMS broker. Artemis received and is still receiving a lot of changes to build it up to the feature set of ActiveMQ 5. The last official statement I heard is that Artemis might at some point become the successor of ActiveMQ 5, but that is not decided yet. The Wikipedia article for

ActiveMQ Confusion and What comes with your JBoss EAP / WildFly
http://126kr.com/article/6vk09hqaq7n

the latest WildFly, as well as RedHat's EAP 7 , distribute ActiveMQ Artemis as their JMS broker.




https://mosquitto.org/
https://www.eclipse.org/paho/clients/js/

# RabbitMQ, ActiveMQ,…

RabbitMQ
Messaging that just works

ActiveMQ™

## How to Apply Messaging to Cloud Apps with RabbitMQ, Node.js, Redis, and More

August 19, 2013 • **PRODUCTS** • By Adam Bloom


Recently, Cloud Foundry developer advocate, Alvaro Videla, gave a talk at the Norwegian Developer's

below.

## 800,000 Messages/Minute: How Nokia's Uses RabbitMQ to Make Real-time Traffi

April 30, 2013 • **CASE STUDIES** • By Stacey Schneider

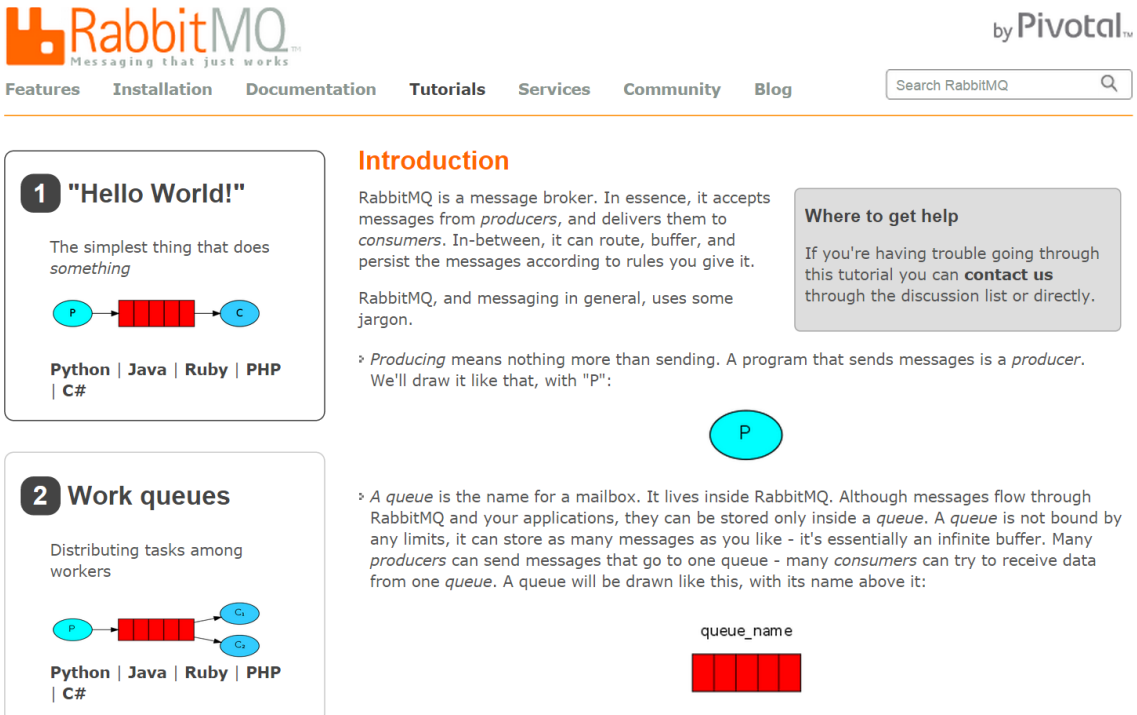

Back in 2008, Nokia acq
Maps which has since be
of Nokia's Location and
business, renamed HERE

Based in Chicago, IL., the
team ingests tens of tho
points every second from
around the globe includ

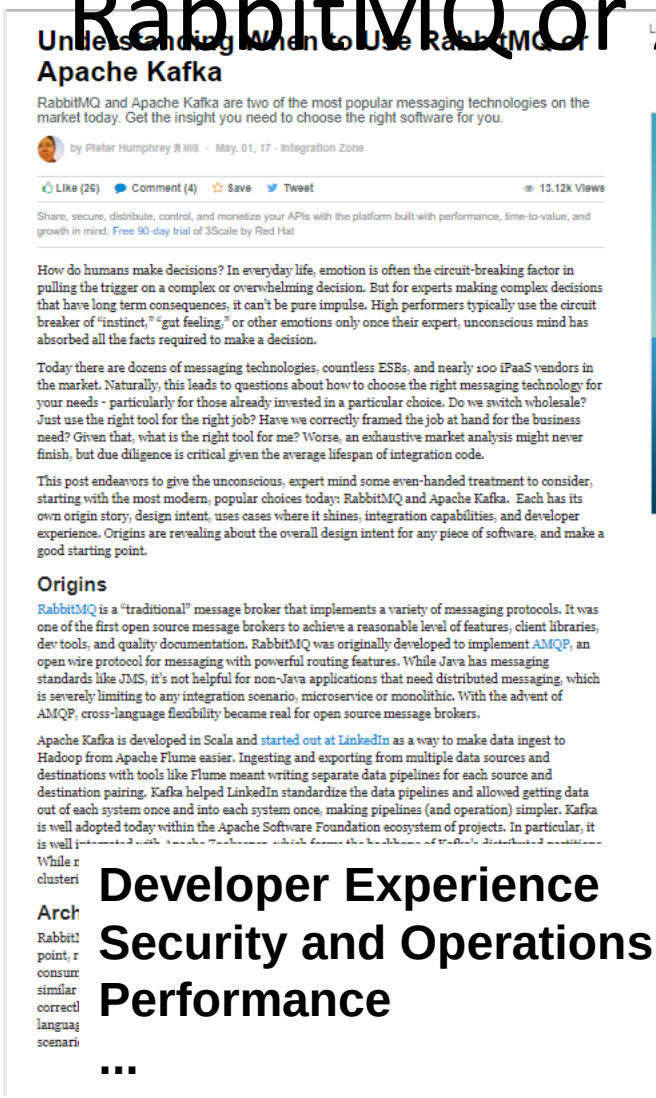
https://www.rabbitmq.com/
http://activemq.apache.org/
http://www.gridshore.nl/2011/07/28/combining-java-and-
http://blog.gopivotal.com/products/how-to-apply-messac
http://blog.cloudfoundry.com/tag/redis/

# Why present RabbitMQ?

- A good example, not the only one
- Multilanguage and framework solution
- Good solution to integrate software



# Understanding When to Use RabbitMQ or Apache Kafka



**Understanding When to Use RabbitMQ or Apache Kafka**

RabbitMQ and Apache Kafka are two of the most popular messaging technologies on the market today. Get the insight you need to choose the right software for you.

by Pieter Humphrey · May. 01, 17 · Integration Zone

Like (26) Comment (4) Save Tweet 13.12k Views

Share, secure, distribute, control, and monetize your APIs with the platform built with performance, time-to-value, and growth in mind. Free 90-day trial of 3Scale by Red Hat

How do humans make decisions? In everyday life, emotion is often the circuit-breaking factor in pulling the trigger on a complex or overwhelming decision. But for experts making complex decisions that have long term consequences, it can't be pure impulse. High performers typically use the circuit breaker of "instinct," "gut feeling," or other emotions only once their expert, unconscious mind has absorbed all the facts required to make a decision.

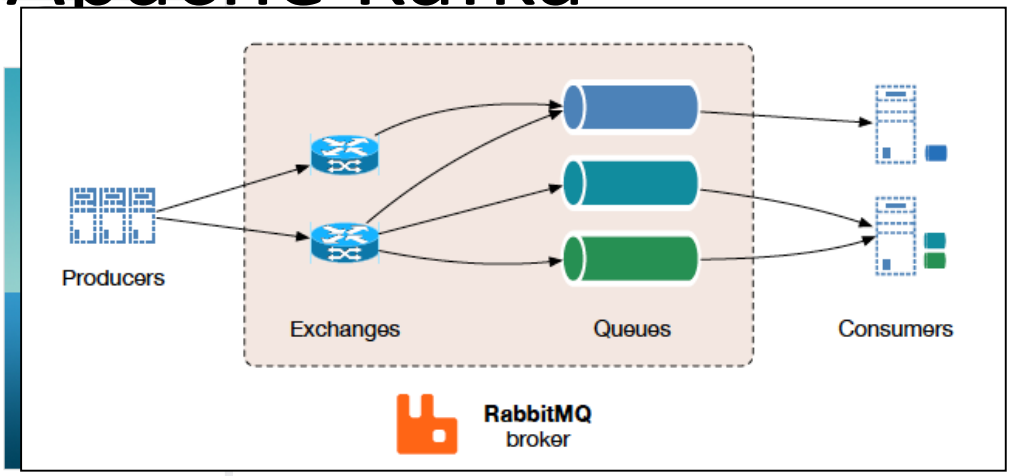
Today there are dozens of messaging technologies, countless ESBs, and nearly 100 iPaaS vendors in the market. Naturally, this leads to questions about how to choose the right messaging technology for your needs - particularly for those already invested in a particular choice. Do we switch wholesale? Just use the right tool for the right job? Have we correctly framed the job at hand for the business need? Given that, what is the right tool for me? Worse, an exhaustive market analysis might never finish, but due diligence is critical given the average lifespan of integration code.

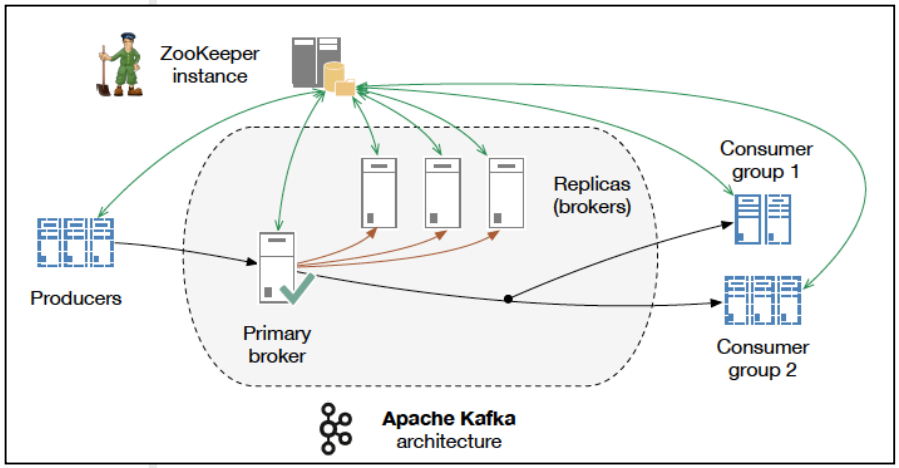
This post endeavors to give the unconscious, expert mind some even-handed treatment to consider, starting with the most modern, popular choices today: RabbitMQ and Apache Kafka. Each has its own origin story, design intent, uses cases where it shines, integration capabilities, and developer experience. Origins are revealing about the overall design intent for any piece of software, and make a good starting point.

**Origins**

RabbitMQ is a "traditional" message broker that implements a variety of messaging protocols. It was one of the first open source message brokers to achieve a reasonable level of features, client libraries, dev tools, and quality documentation. RabbitMQ was originally developed to implement AMQP, an open wire protocol for messaging with powerful routing features. While Java has messaging standards like JMS, it's not helpful for non-Java applications that need distributed messaging, which is severely limiting to any integration scenario, microservice or monolithic. With the advent of AMQP, cross-language flexibility became real for open source message brokers.

Apache Kafka is developed in Scala and started out at LinkedIn as a way to make data ingest to Hadoop from Apache Flume easier. Ingesting and exporting from multiple data sources and destinations with tools like Flume meant writing separate data pipelines for each source and destination pairing. Kafka helped LinkedIn standardize the data pipelines and allowed getting data out of each system once and into each system once, making pipelines (and operation) simpler. Kafka is well adopted today within the Apache Software Foundation ecosystem of projects. In particular, it is well




**Developer Experience**
**Security and Operations**
**Performance**
**...**

https://dzone.com/articles/understanding-when-to-use-rabbitmq-or-apache-kafka

# The END