Pivotal

DEVNEXUS™

Microservices Messaging

Dan Buchko
Sr. Platform Architect
March 2019

Felipe Gutierrez
Sr. Platform Architect

Agenda

- Introductions
- Evolution of Messaging
- Messaging Patterns
- Microservices
- Microservices Messaging
- Q+A

Background

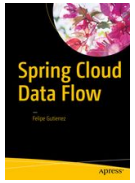
Felipe Gutierrez

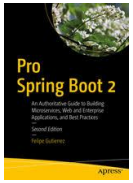
Development and Consulting
- Spring Framework
- RabbitMQ

Technical Instructor
- RabbitMQ
- Core Spring, Spring Boot
- Pivotal Cloud Foundry - PAS

Platform Architect
- PCF (PAS/PKS)

Book Author
- Pro Spring Boot 2nd
- Spring Boot Messaging
- Spring Cloud Data Flow





Dan Buchko

Consulting
- Tibco
- RabbitMQ

Technical Instructor
- RabbitMQ
- Redis
- Pivotal Cloud Foundry - PAS

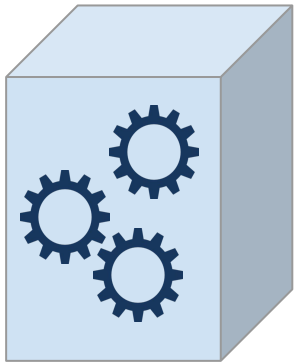
Platform Architect
- Pivotal Cloud Foundry - PAS/PKS
- Spring

Microservices Messaging

Evolution of Messaging

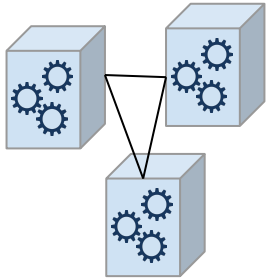
Initially...

- Applications were self-contained
- All interactions were done within the app itself
- No need for applications to communicate

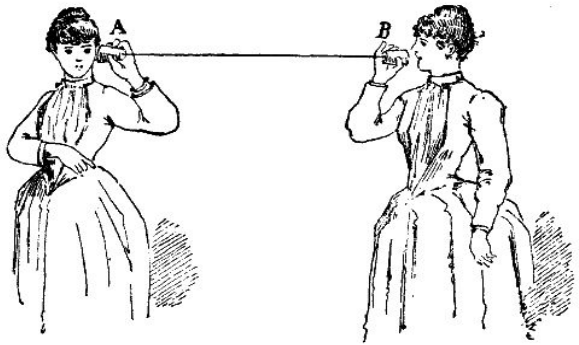


https://www.geeksforgeeks.org

Distributed Systems



- Applications increased in complexity
- Distributing functionality across applications running on separate servers became necessary
- Communication between applications was required
- As a result, various messaging protocols have evolved over time



Evolution of Messaging

Sockets

- Point-to-point
- Low level
- Network layer
- Difficult to write
- Need to code higher-level functionality

TCP/IP

```
// Driver function
int main()
{
    int sockfd, connfd, len;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);

    // Binding newly created socket to given IP and verification
    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
```

https://www.geeksforgeeks.org

Evolution of Messaging

JMS

- Broker-based messaging
- Standardized Java interface
- Vendors implement the interface accordingly
- Widely adopted
- Caveat: Java clients only!


```
QueueConnectionFactory qConnectionFactory = ServiceLocatorManager
                .lookupQueueConnectionFactory(STATIC_CONNECTION_FACTORY_REF_NAME);
Queue queue = ServiceLocatorManager
                .lookupQueue(STATIC_QUEUE_REF_NAME);
QueueConnection qConnection = qConnectionFactory
                .createQueueConnection();
QueueSession qSession = qConnection.createQueueSession(false,
                Session.AUTO_ACKNOWLEDGE);
QueueSender sender = qSession.createSender(queue);
TextMessage message = qSession.createTextMessage();
message.setText("Just a simple message");
sender.send(message);
sender.close();
qSession.close();
qConnection.close();
```

Evolution of Messaging

AMQP

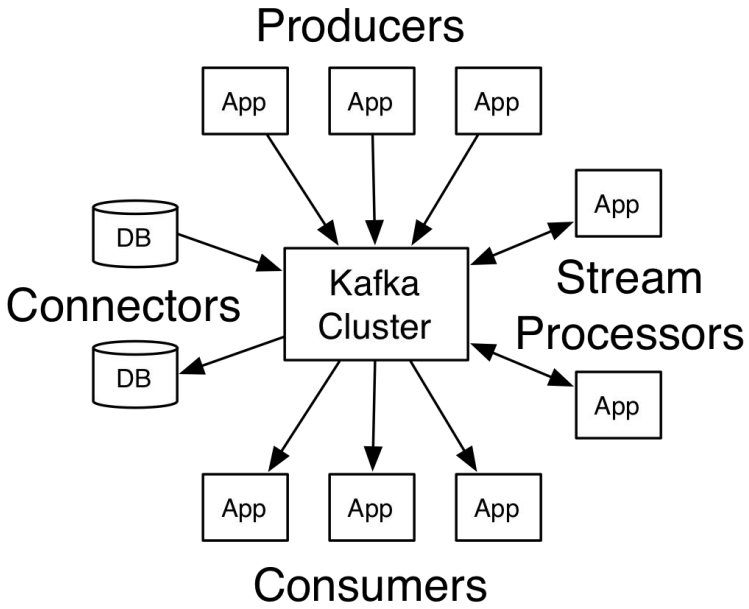
- Broker based messaging
- Standard defines format over the wire
- Allows for complex message routing
- Multiple client libraries available
- Polyglot language support


```
1  ConnectionFactory factory = new ConnectionFactory();
2  factory.setHost("localhost");
3  try (Connection connection = factory.newConnection();
4       Channel channel = connection.createChannel()) {
5       channel.queueDeclare(TASK_QUEUE_NAME, true, false, false, null);
6       String message = String.join(" ", argv);
7       channel.basicPublish("", TASK_QUEUE_NAME,
8                   MessageProperties.PERSISTENT_TEXT_PLAIN,
9                   message.getBytes("UTF-8"));
10      System.out.println(" [x] Sent '" + message + "'");
11  }
```

Evolution of Messaging

Kafka

- Broker-based messaging
- From LinkedIn
- Streaming use case support
- Replay functionality
- Persistent message streams



https://kafka.apache.org

More Players...

ØMQ

solace.

MQTT.ORG

amazon Kinesis

NATS

Stomp

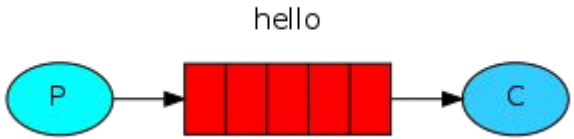
Apache ACTIVEMQ

TIBCO®

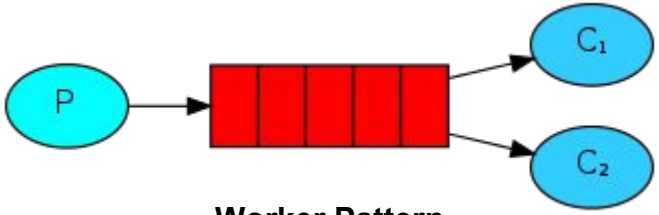
Microservices Messaging

Messaging Patterns

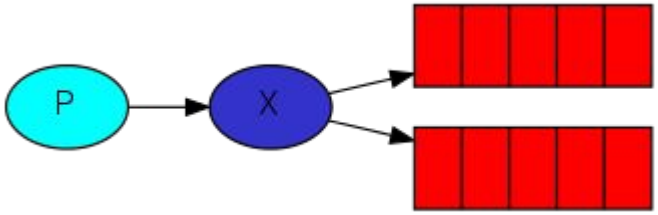
Messaging Patterns



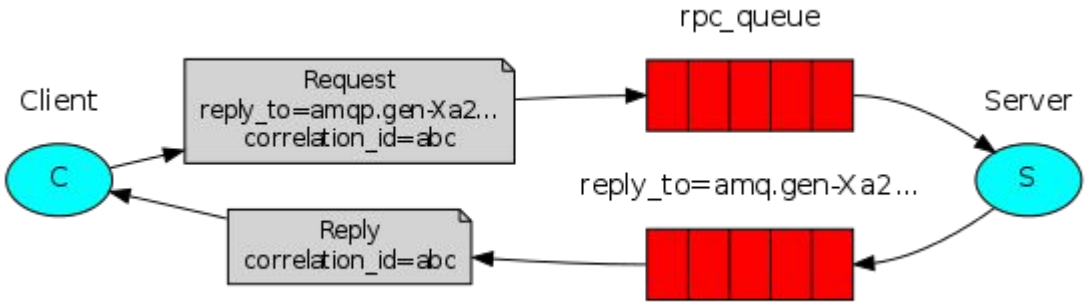
Point-to-Point



Worker Pattern

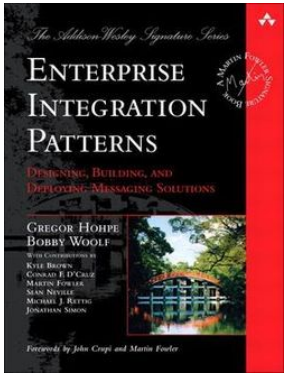


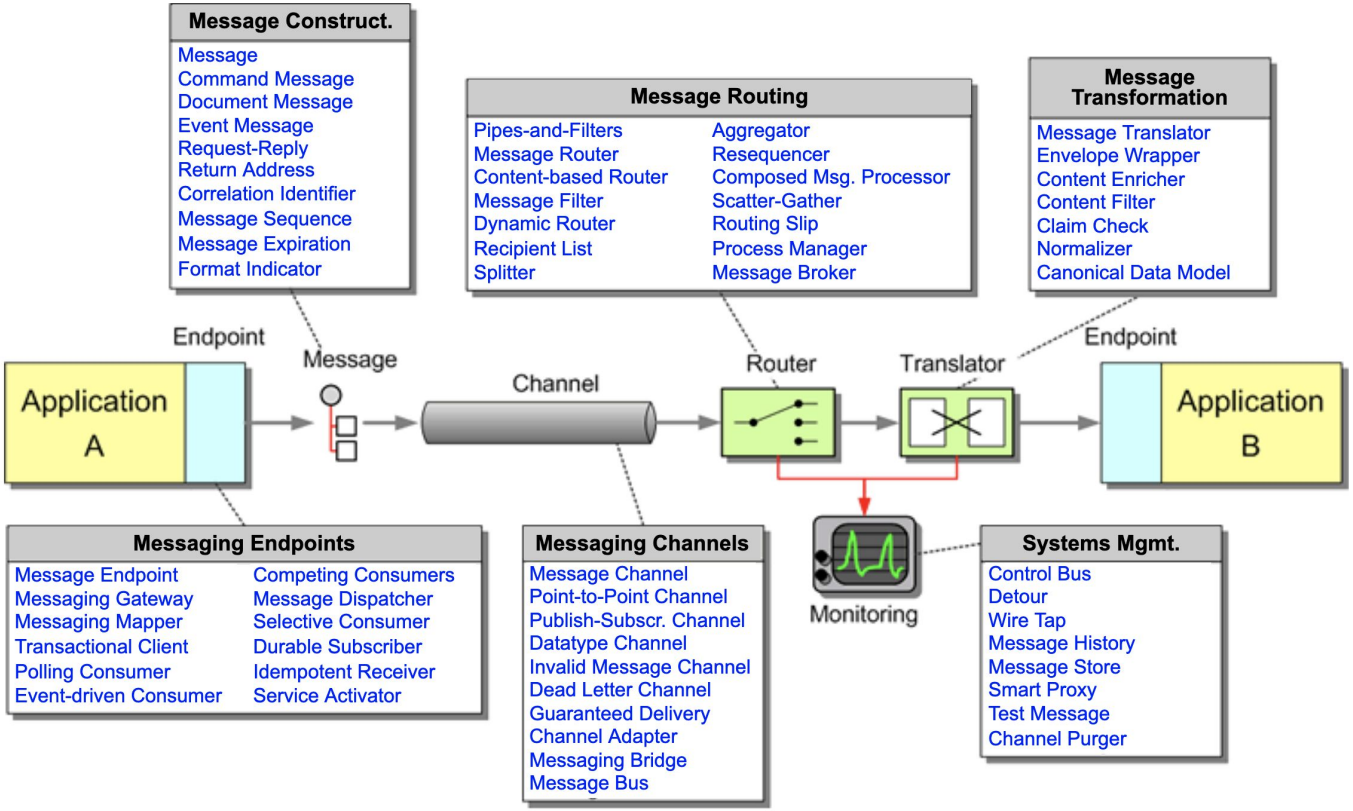
Publish/Subscribe



RPC

Integration Patterns



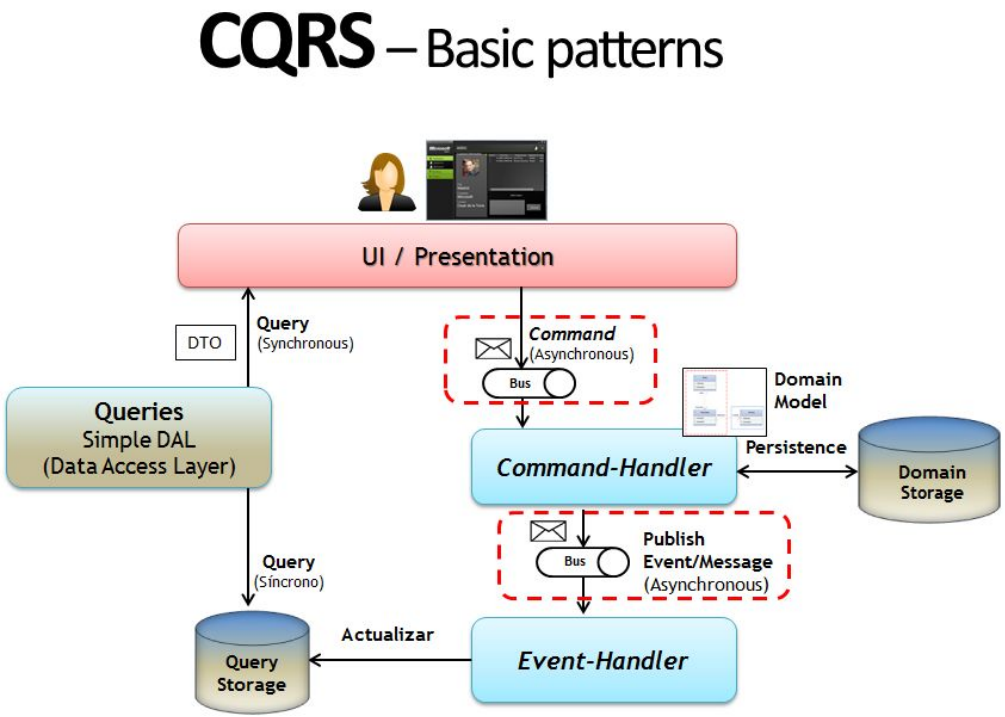


Integration Patterns

Enterprise Service Bus (ESB)

- Integration pattern
- Underlying messaging layer (typically JMS)
- Service orchestration
- Routing
- Clustering

Event Source Patterns

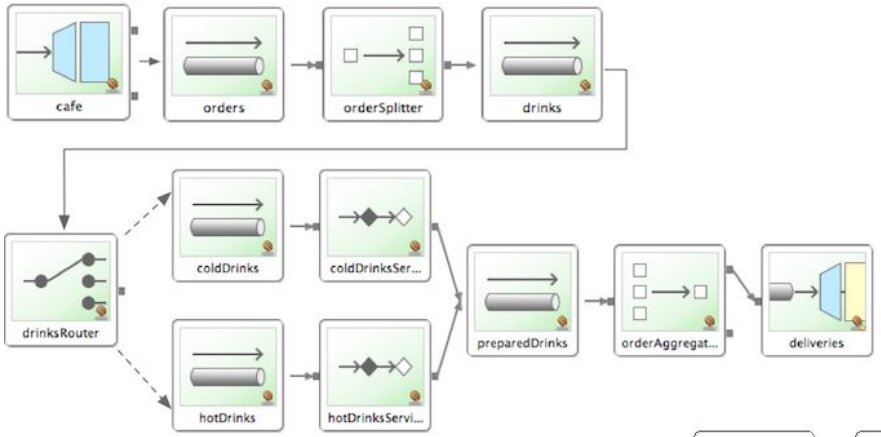


Command Query Responsibility Segregation

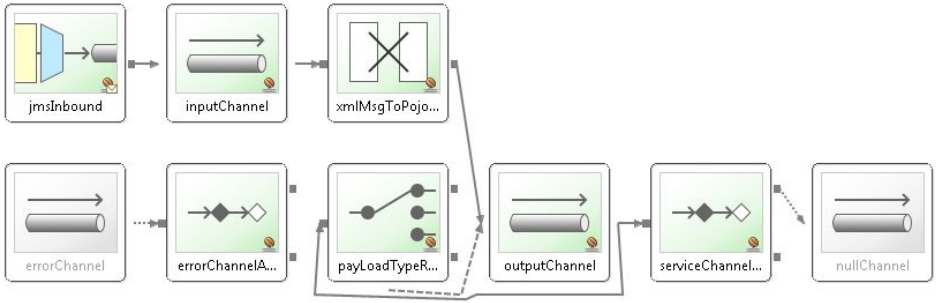
https://blogs.msdn.microsoft.com

Integration Frameworks

Implementation



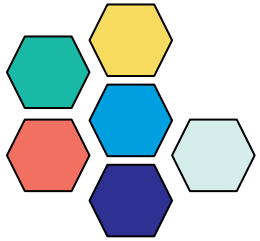
Spring Integration



Microservices Messaging

Microservices

Microservices



Typical Approach

- Synchronous messaging over HTTP
- Must know service endpoints, or use a registry for service discovery
- Scaling and load balancing require platform support, or additional libraries - eg. client-side load balancing (also requires service registry)
- Fault tolerance (apply Circuit Breaker patterns)

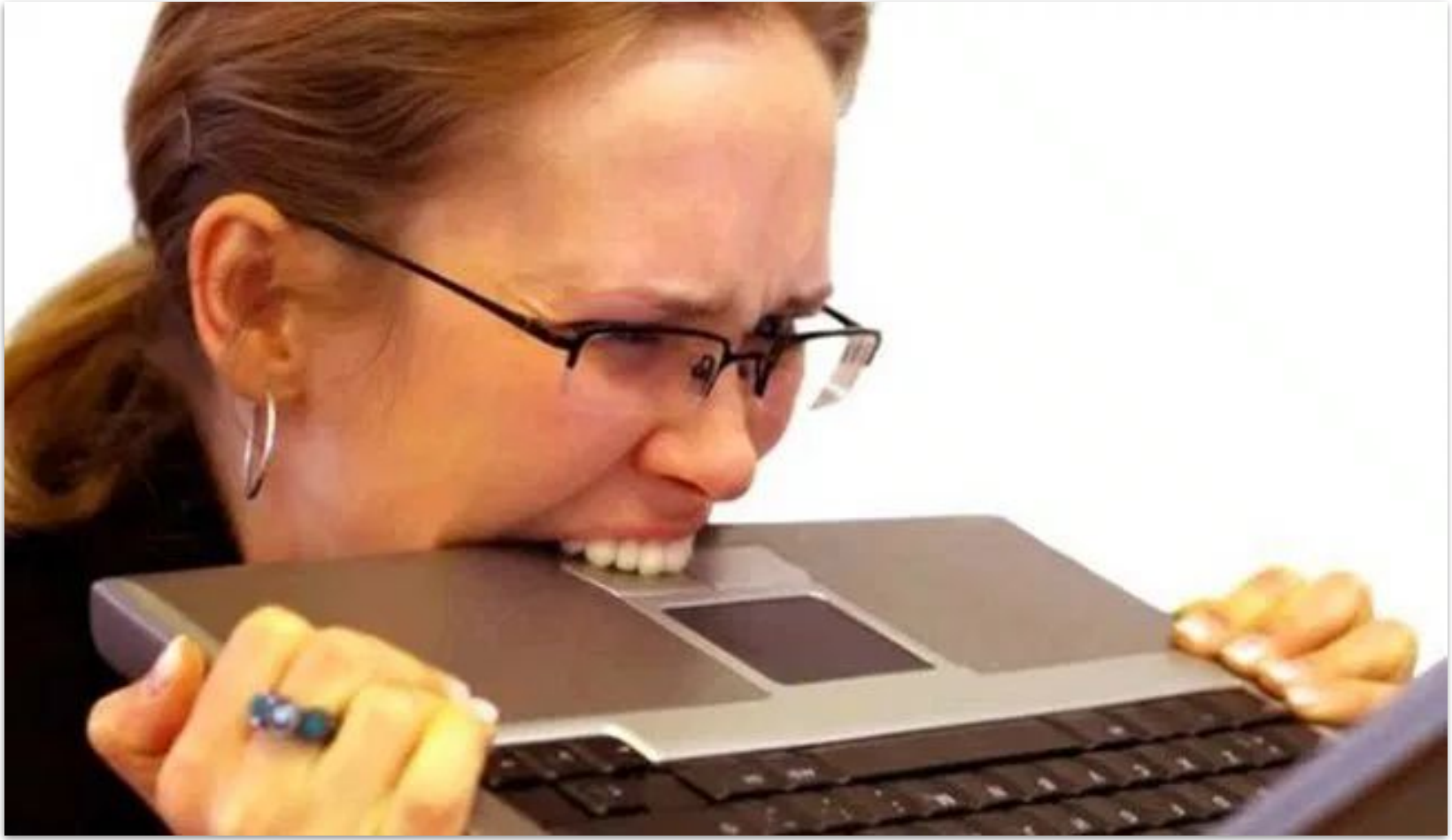
Microservices Messaging

Microservices Messaging

How can we do Microservices Messaging?

- Async ? Sync?
- What protocol or broker should I use? AMQP? STOMP? MQTT?
- What about serialization? Support for Schema versions?
- What about persistence, scalability, HA?
- What about service discovery, fault tolerance?

Should I learn or hire somebody that knows all these: AMQP, MQTT, JMS, STOMP, SQS, KINESIS, PUBSUB, ... technologies?



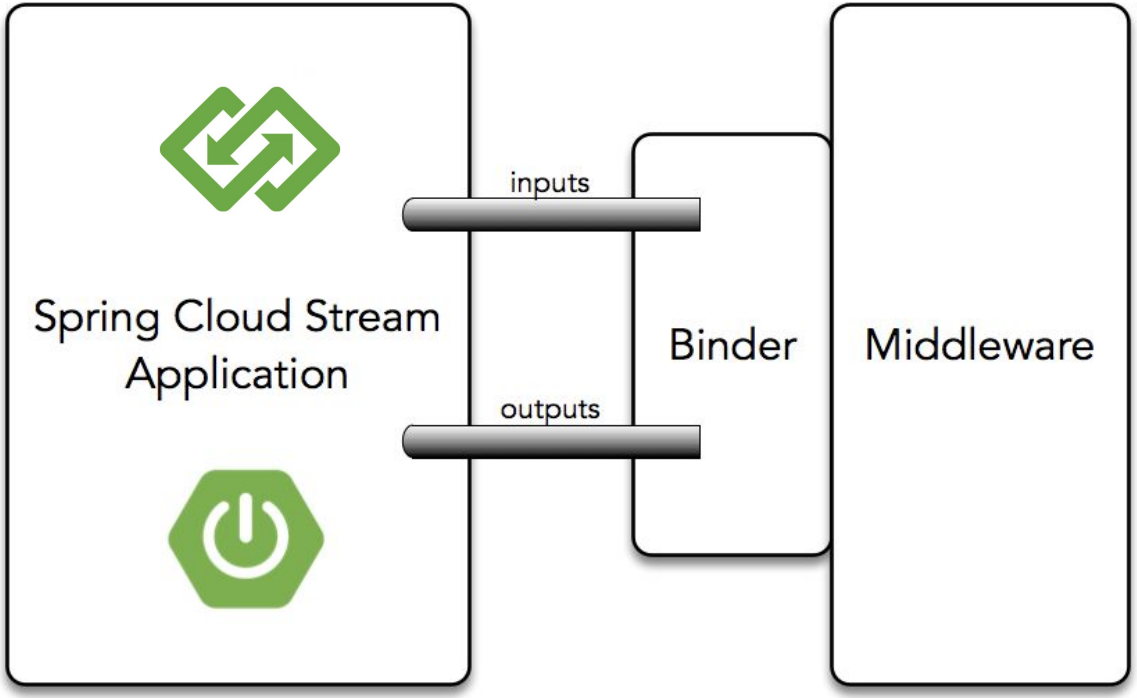
Microservices Messaging

Is there any Solution yet?

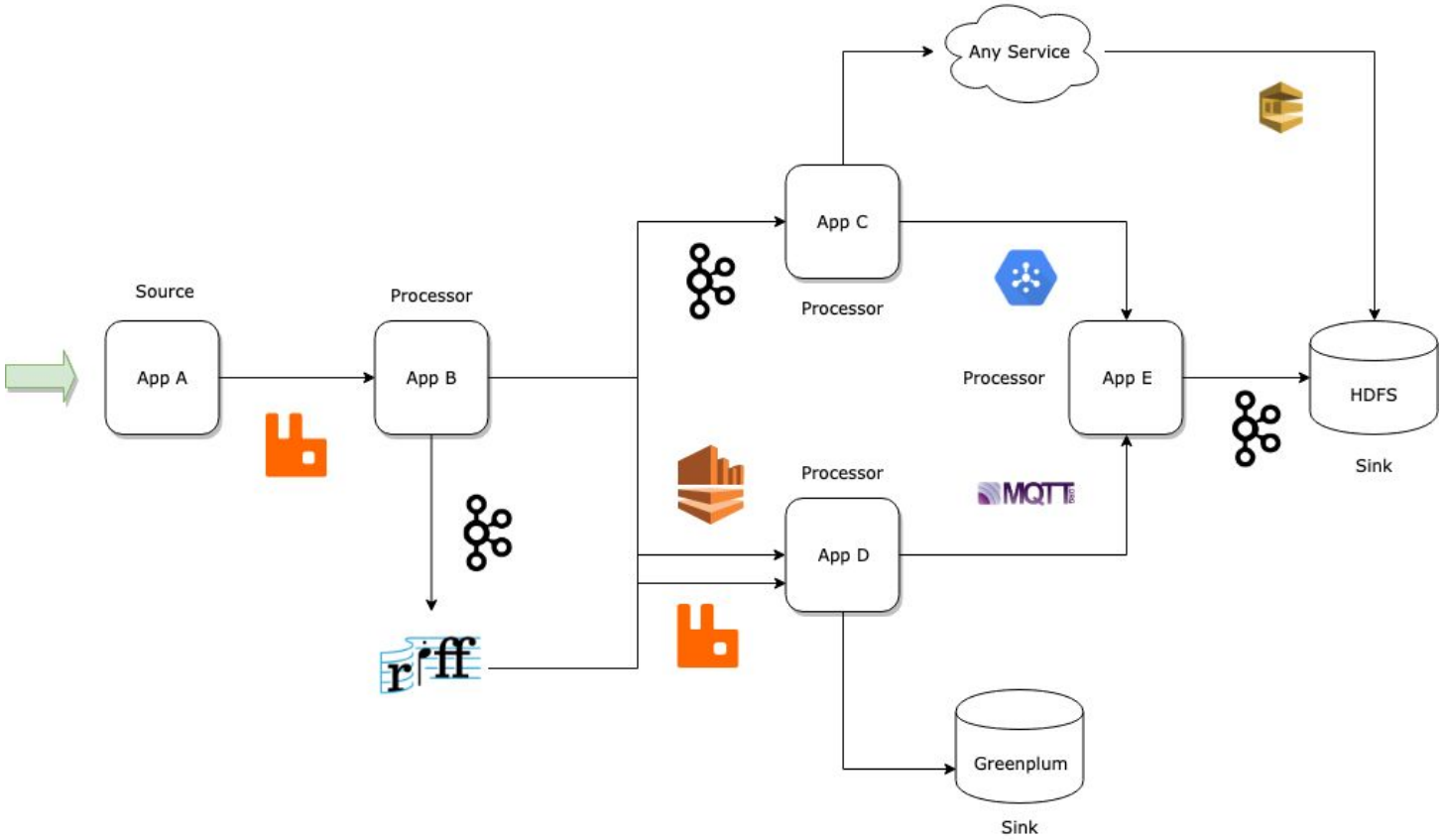
Spring Cloud Stream

Spring Cloud Stream

Spring Cloud Stream



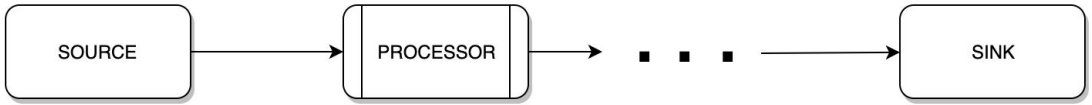
Spring Cloud Stream Example Topology



Spring Cloud Stream

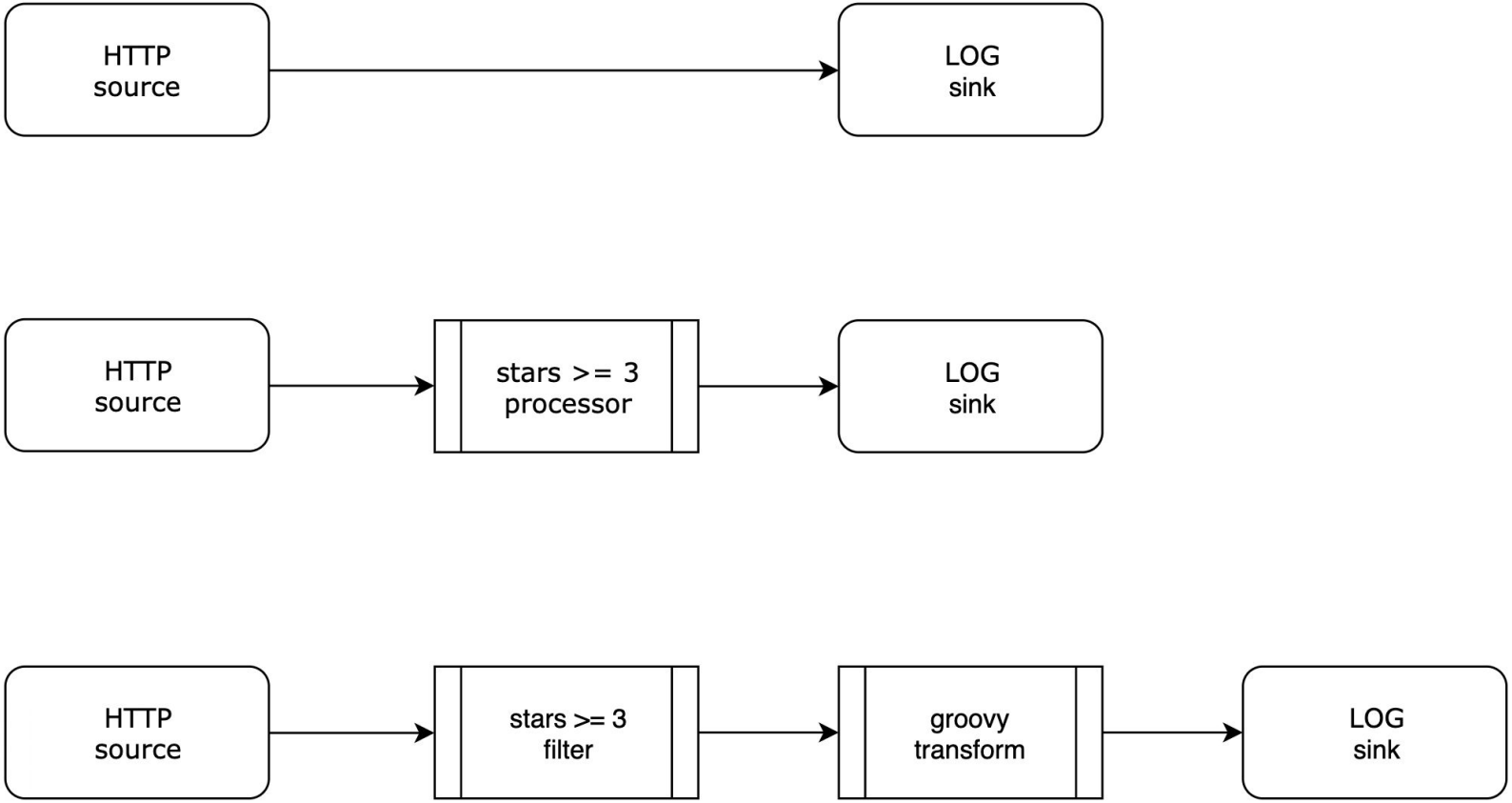
Spring Cloud Stream App Starters

Spring Cloud Stream Application Starters are Spring Boot based Spring Integration applications that provide integration with external systems. Spring Cloud Stream Applications can be used with Spring Cloud Data Flow to create, deploy, and orchestrate message-driven microservice applications.



Source	Processor	Sink
file	aggregator	aggregate-counter
ftp	bridge	cassandra
gemfire	filter	counter
gemfire-cq	groovy-filter	field-value-counter
http	groovy-transform	file
jdbc	header-enricher	ftp
jms	httpclient	gemfire
load-generator	pmml	gpfdist
loggregator	python-http	hdfs
mail	python-jython	hdfs-dataset
mongodb	scriptable-transform	jdbc
mqtt	splitter	log
rabbit	tasklaunchrequest-transform	mongodb
s3	tcp-client	mqtt
sftp	tensorflow	pgcopy
syslog	transform	rabbit
tcp	twitter-sentiment	redis-pubsub
tcp-client	grpc	router
time		s3
trigger		sftp
triggertask		task-launcher-cloudfoundry
twitterstream		task-launcher-local
		task-launcher-dataflow
		task-launcher-yarn
		tcp
		throughput
		websocket


Demo
DEVNEXUS™
Pivotal


HTTP source
LOG sink
HTTP source
stars >= 3 processor
LOG sink
HTTP source
stars >= 3 filter
groovy transform
LOG sink

Custom Spring Cloud Stream Apps

- It is possible to create custom Spring Cloud Stream Apps

Application Model

- `@EnableBinding`
- Define a `Source`, `Processor` or `Sink`
- Define Input and Output Channels
- `@StreamListener` / `@SendTo` / Spring Integration DSL
- Use a Transport Layer (RabbitMQ / Kafka / ...)
- **NO MORE MESSAGING API TO LEARN !!**

Custom Spring Cloud Stream App Example

```
@EnableBinding(Source.class)
public static class JmsSource {

    @Bean
    public IntegrationFlow jmsListenerFlow(JmsTemplate jmsTemplate){
        return IntegrationFlows.from(Jms.inboundAdapter(jmsTemplate).destination("ITEM"),
                c -> c.poller(Pollers.fixedRate( period: 100)))
                .channel(Source.OUTPUT)
                .get();
    }

}

@EnableBinding(Sink.class)
public static class LogSink {

    @StreamListener(Sink.INPUT)
    public void log(Item item){
        System.out.println(String.format("CLOUD STREAM: " + item));
    }
}
```

Microservices Challenges

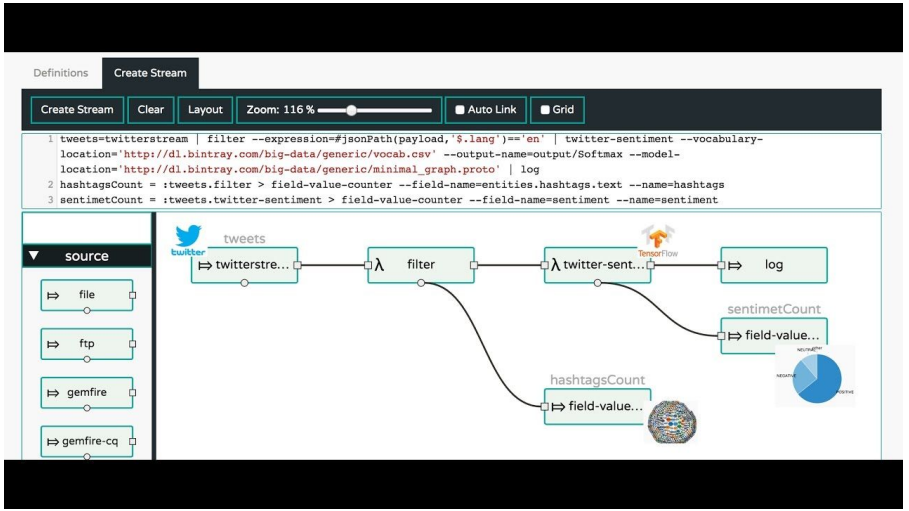
Now, I have 30+ microservices ...

- Is there an easy way to deploy them?
- Is there any tool to orchestrate them?
- Is there any way to monitor them?
- What about scalability, HA, ...?

Spring Cloud Data Flow

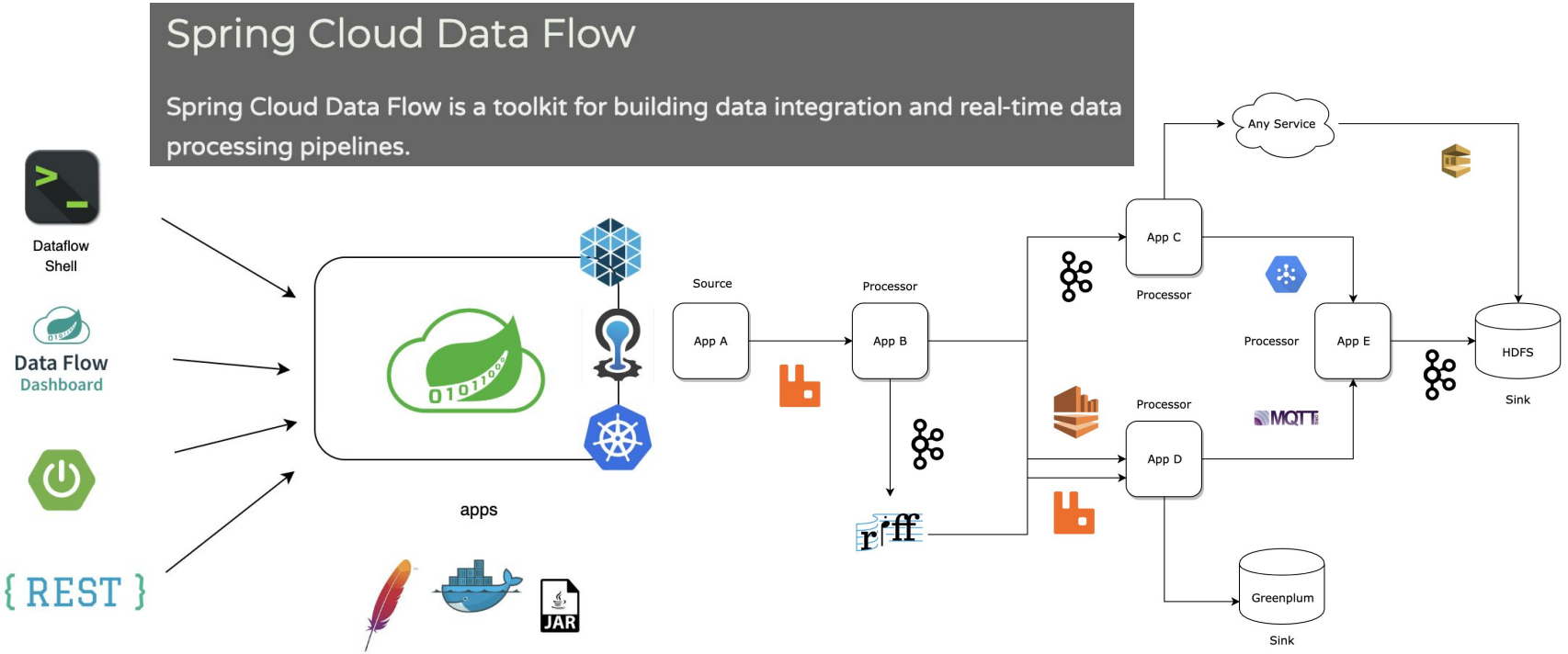
Spring Cloud Data Flow

- Data integration and real-time data processing pipelines.
- Reusable Stream and Task Creation
- Stream Pipeline DSL (Unix like)
- Dashboard / Shell / Rest API



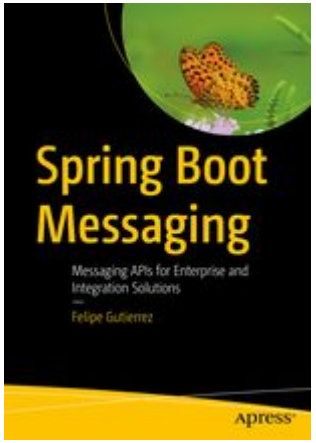

Demo
DEVNEXUS™

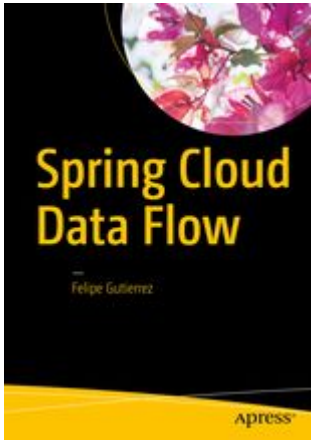
Microservices Messaging

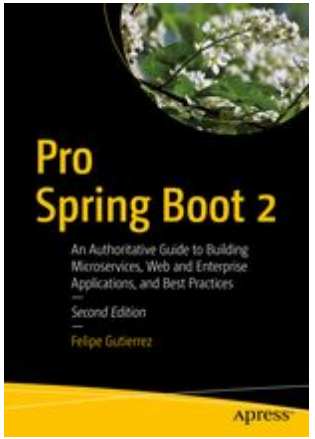


A Little GIFT...

- Source Code:
  - https://github.com/felipeg48/devnexus2019-talk
- Apress Publishing
  - https://apress.com
  - CODE: **SpringBoot2**








Q & A
DEVNEXUS™
Pivotal