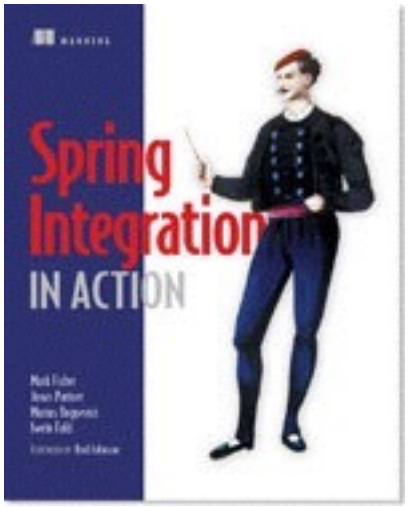
# What's new in Spring Integration 4.x?

**Marius Bogoevici**, Staff Engineer, Pivotal
@mariusbogoevici

# About me

- **Current**
  - Staff Engineer @ Pivotal
  - Spring XD team
- **Past**
  - Open source: SpringSource, JBoss/RedHat, Pivotal
  - Spring Integration contributor
    - co-author “Spring Integration in Action, Manning, 2012”
  - Finance, Asset Management, Mobile Messaging, etc.







spring

# Spring Integration - continuously evolving




spring by Pivotal

DZone
RESEARCH

DZONE RESEARCH PRESENTS

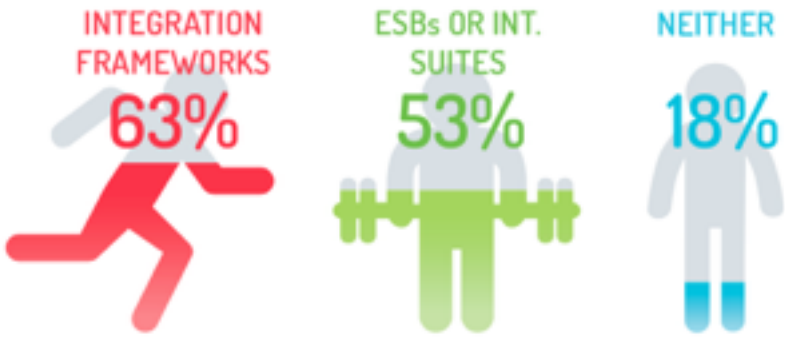
2014 GUIDE TO

# ENTERPRISE INTEGRATION

BROUGHT TO YOU IN PARTNERSHIP WITH

3SCALE ca MuleSoft redhat WSO2

## ESBs VS LIGHTWEIGHT INTEGRATION FRAMEWORKS

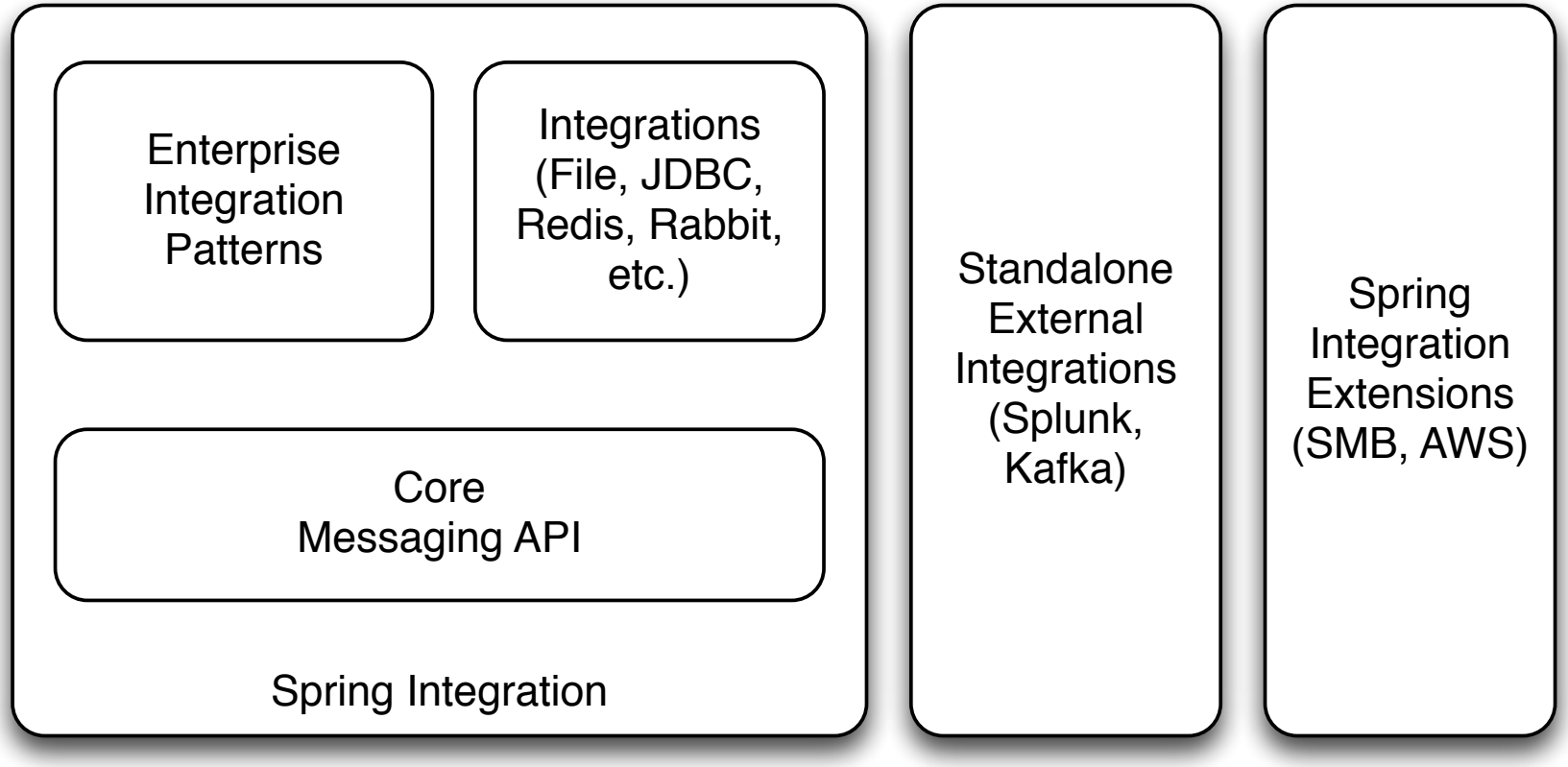



**LIGHTWEIGHT EIP FRAMEWORKS ARE THE NORM FOR MOST SYSTEMS** Out of all integration frameworks (EIP), ESBs, and integration suites, Spring Integration is the most popular (42%) and Apache Camel is a close second (38%). The three most popular ESBs are Mule ESB (16%), WebSphere ESB (15%), and Oracle ESB (13%). Overall, 63% of respondents use an integration framework (e.g. Spring Integration, Camel) and 53% use an ESB or Integration Suite (e.g. Mule ESB, Biztalk), while 18% say they use neither. *Note that 69% of respondents are from large organizations, where bigger integration scenarios would be more common.*




spring by Pivotal

# Spring Integration Ecosystem







spring by Pivotal

# Early releases: 1.0 and 2.0

- **Spring Integration 1.0**
  - First implementation of the framework!
  - Core components: Message, MessageChannel, MessageHandler
  - Core Enterprise Integration Patterns:
    - Filter, Splitter, Aggregator, Resequencer, Service Activator, Gateway
  - Integrations: XML, File, JMS, Mail and more
- **Spring Integration 2.0**
  - Spring 3 and SpEL support
  - More Enterprise Integration Patterns
  - More Integrations
  - Enhanced namespace and annotation support




spring by Pivotal

# Spring Integration 3.0 and 4.x

- **Rapid sequence of changes**
- **Spring Integration 3.0 - October 2013**
- **Spring Integration 4.0, 4.1 throughout 2014**
  - Our topic today!
- **4.2 Forthcoming**




spring by Pivotal

# Spring 4.x Evolution Themes




spring by Pivotal

# Architectural consolidation and internal refactoring




spring by Pivotal

# Messaging API moved to Spring Framework 4

- **New packages in Spring Framework 4**
  - `org.springframework.messaging`
    - `Message, MessageHeaders`
    - `MessageHandler`
    - `MessageChannel,PollableChannel,SubscribableChannel`
    - `MessagingException,MessageHandlingException, MessageDeliveryException`
    - `MessagePostProcessor`
  - `org.springframework.messaging.support`
    - `ChannelInterceptorAdapter`
    - `GenericMessage,ErrorMessage`
    - `ChannelResolver`
- **Spring Integration 4 depends on Spring 4**




spring

# Advantages

- **Shared messaging abstractions with other Spring components, e.g. Spring Core**
  - WebSocket
  - STOMP
  - SockJS
- **Spring Integration 4 internals make use of the new abstraction**



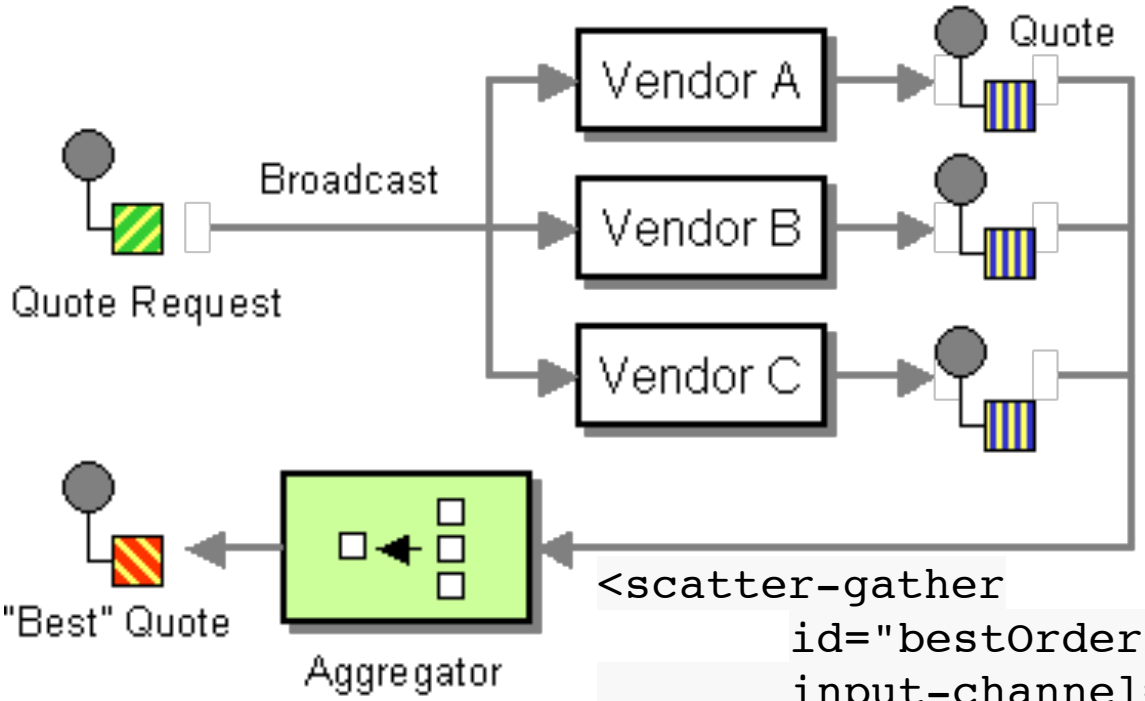

spring

# New integration patterns




spring by Pivotal

# Scatter-Gather

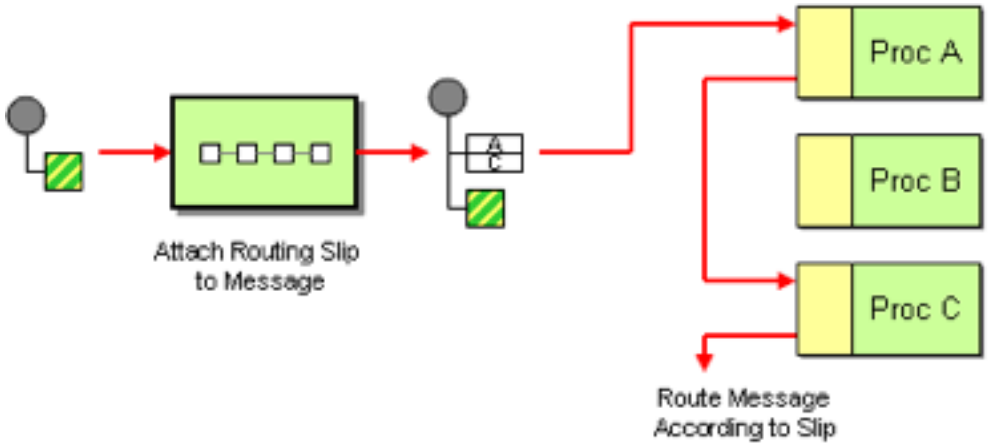



```
<scatter-gather
        id="bestOrder"
        input-channel="orders"
        output-channel="bestOrder" >
            <scatterer>
              <recipient channel="vendorA"/>
              <recipient channel="vendorB"/>
              <recipient channel="vendorC"/>
            </scatterer>
            <gatherer ref="bestOrderCompute">
</scatter-gather>
```




spring by Pivotal

# Routing Slip




```
<header-enricher input-channel="input" output-channel="process">
        <routing-slip value="procA; procB; procC; finishChannel”/>
</header-enricher>
```

SpEL expressions are supported, too




spring by Pivotal

# Idempotent Receiver

- Messages with the same ID, played only once
- Useful in retry, execute at most once scenarios

```
<idempotent-receiver
      id="receiver"
      endpoint="orderProcessor"
      discard-channel="failures"
      metadata-store="metadataStore"
      key-expression="#{payload.orderId}"/>
```




spring by Pivotal

# Java DSL, Java 8 and Boot

# New annotations for configuring endpoints

- @MessageGateway
- @GlobalChannelInterceptor
- @IntegrationConverter
- @Poller
- @InboundChannelAdapter




spring by Pivotal

# Java Configuration Support

- **@Configuration**
  - @EnableIntegration
  - @IntegrationComponentScan
  - @EnableMessageHistory
  - @EnableIntegrationMBeanExport
  - @BridgeFrom
  - @BridgeTo
- **Spring Boot**
  - via **spring-boot-starter-integration**
    - some adapters included by default (file, stream, http, ip)
    - others must be added explicitly
  - @EnableAutoConfiguration




spring by Pivotal

# Spring Integration Java DSL

- Fluent API for creating Integration flows
- Perfect complement for @Configuration and Spring Boot
- Core concept of `IntegrationFlow`

```
@Bean
IntegrationFlow flow() {
        return IntegrationFlows.from(foo())
                .transform("payload + payload")
                .handle(String.class, (p, h) -> p.toUpperCase())
                .get();
}
```




spring by Pivotal

# Demo time




spring by Pivotal

# ListenableFuture and Asynchronous Programming

- Since Spring Integration 4
- ListenableFuture from Spring Core Framework 4

```
@MessagingGateway(defaultReplyTimeout = 0)
public interface MathGateway {

  @Gateway(requestChannel = "gatewayChannel")
    ListenableFuture<Integer> multiplyByTwo(int number);

}
```

```
ListenableFuture<String> result = this.asyncGateway.async("foo");
result.addCallback(new ListenableFutureCallback<String>() {

    @Override
    public void onSuccess(String result) {
        ...
    }

    @Override
    public void onFailure(Throwable t) {
        ...
    }
});
```




spring by Pivotal

# Reactor Promise<?>

- Based on Project Reactor Promise
- Streaming component (not Java 8) for reactive programming

```
@MessagingGateway(defaultReplyTimeout = 0, reactorEnvironment = "reactorEnv")
  public interface MathGateway {
    @Gateway(requestChannel = "gatewayChannel")
    Promise<Integer> multiplyByTwo(int number);
}
```

```
Streams.defer(Arrays.asList("1", "2", "3", "4", "5"))
            .env(this.environment)
            .get()
            .map(Integer::parseInt)
            .mapMany(integer -> testGateway.multiply(integer))
            .collect()
            .consume(integers -> ...)
            .flush();
```




spring by Pivotal

# Distributed processing




spring by Pivotal

# Distributed Metadata Stores

- **What is a Metadata Store ?**
  - Maintains state across restarts
  - Used by Twitter, File, FTP channel adapters, Idempotent receiver
- **MetadataStore abstraction available since 2.0**
- **Distributed implementations for operating in a cluster**
  - Redis (since 3.0)
  - Gemfire (since 4.0)




spring by Pivotal

# Distributed Lock Registries

- **Global lock support in a clustered environment**
- **Used by Aggregator, Resequencer (and any subclass of AbstractCorrelatingMessageHandler)**
- **LockRegistry available since 2.1.1**
- **Distributed implementations for operating in cluster**
  - Redis (since 4.0)
  - Gemfire (since 4.0)
  - Forthcoming: Zookeeper (4.2), Hazelcast (incubating)

# Internet of Things




spring by Pivotal

# WebSockets

- **Based on Spring Framework WebSocket support**
- **Channel adapters:**
  - Inbound
  - Outbound
- **Integrate in Spring Web Applications through the Message abstraction**

```
<int-websocket:inbound-channel-adapter … />

<int-websocket:outbound-channel-adapter … />

@MessagingGateway
@Controller
public interface WebSocketGateway {

    @MessageMapping("/greeting")
    @SendToUser("/queue/answer")
    @Gateway(requestChannel = "greetingChannel")
    String greeting(String payload);

}
```




spring by Pivotal

# MQTT support

- **Inbound message gateway**
- **Machine-to-machine connectivity protocol**
  - Large net of small devices that need to be monitored remotely
- **Lightweight pub-sub**
- **Device data collection,**
  - Sensors
  - Smartphones
  - Cars

```
<int-mqtt:message-driven-channel-adapter id="mqttInbound"
    client-id="${mqtt.default.client.id}.src"
    url="${mqtt.url}"
    topics="sometopic"
    client-factory="clientFactory"
    channel="output"/>
```




spring by Pivotal

# Data Streaming




spring by Pivotal

# Apache Kafka Integration

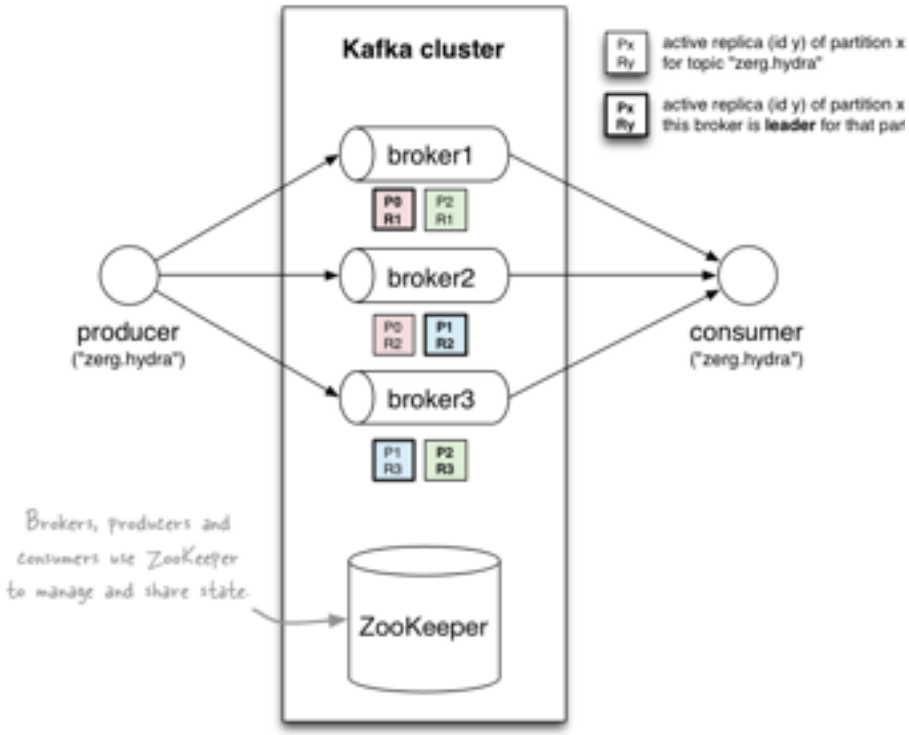
- **Available as extension via spring-integration-kafka**
- **High performance messaging platform**
  - Distributed commit log
- **Heavy reliance of linear disk writes**
- **High data retention**
- **Consumer-controlled read positioning**

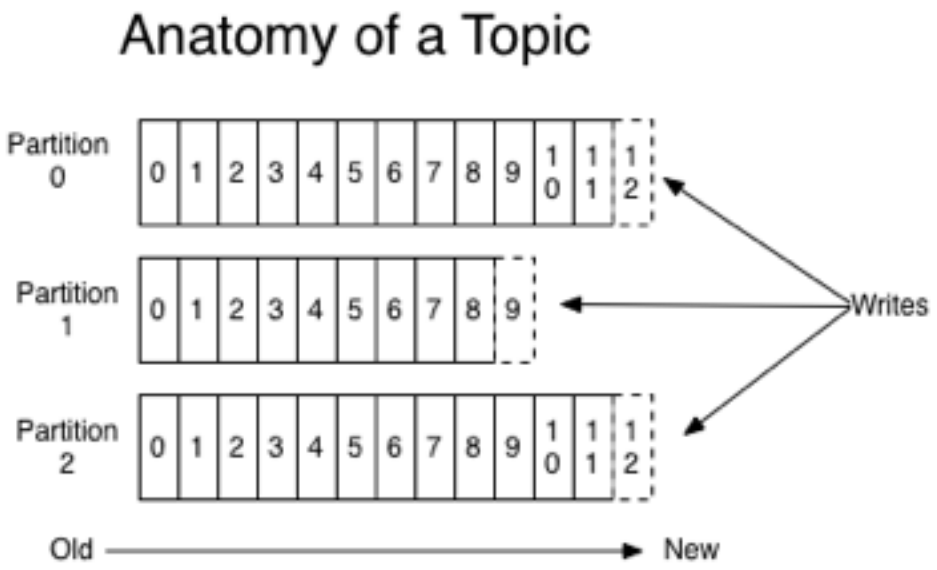
http://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying




spring by Pivotal

# Apache Kafka in a nutshell




http://www.michael-noll.com/blog/2013/03/13/running-a-multi-broker-apache-kafka-cluster-on-a-single-node/




spring by Pivotal

# Spring Integration Kafka Support

- **Message Source (based on high level consumer)**
- **Producer Channel Adapter**
- **Event-Driven Channel Adapter**
  - Offset Control and Replay
  - Pluggable Offset Management
  - Listened Partition Control
  - Guaranteed ordering by partition

```
<int-kafka:message-driven-channel-adapter
                id="kafkaListener"
                channel="“output"
                connection-factory="connectionFactory"
                key-decoder="keyDecoder"
                payload-decoder="payloadDecoder"
                offset-manager="offsetManager"
                concurrency="${concurrency:10}"
                topics="${topics:foo,bar}"/>
```




spring by Pivotal

# Spring XD

# Common problems

Batch and Streaming often handled by multiple platforms

Fragmented Big Data Ecosystem

Simple things are not simple






spring by Pivotal


SPRING XD
EXTREME DATA
"One stop shop for developing and deploying Big Data Applications"

# Spring XD in a nutshell

- **Spring eXtreme Data**
- **Unified, distributed and extensible service for:**
  - Data Ingestion
  - Real time analytics
  - Batch processing
  - Data export
- **Built on existing solutions**
  - Spring Integration
  - Spring Batch
  - Spring Platform (Core, Boot, Data, etc)




spring by Pivotal

# Stream processing


HTTP
Tail
File
Mail
Twitter
Gemfire
Syslog
TCP
UDP
JMS
RabbitMQ
MQTT
Trigger
Reactor TCP/UDP

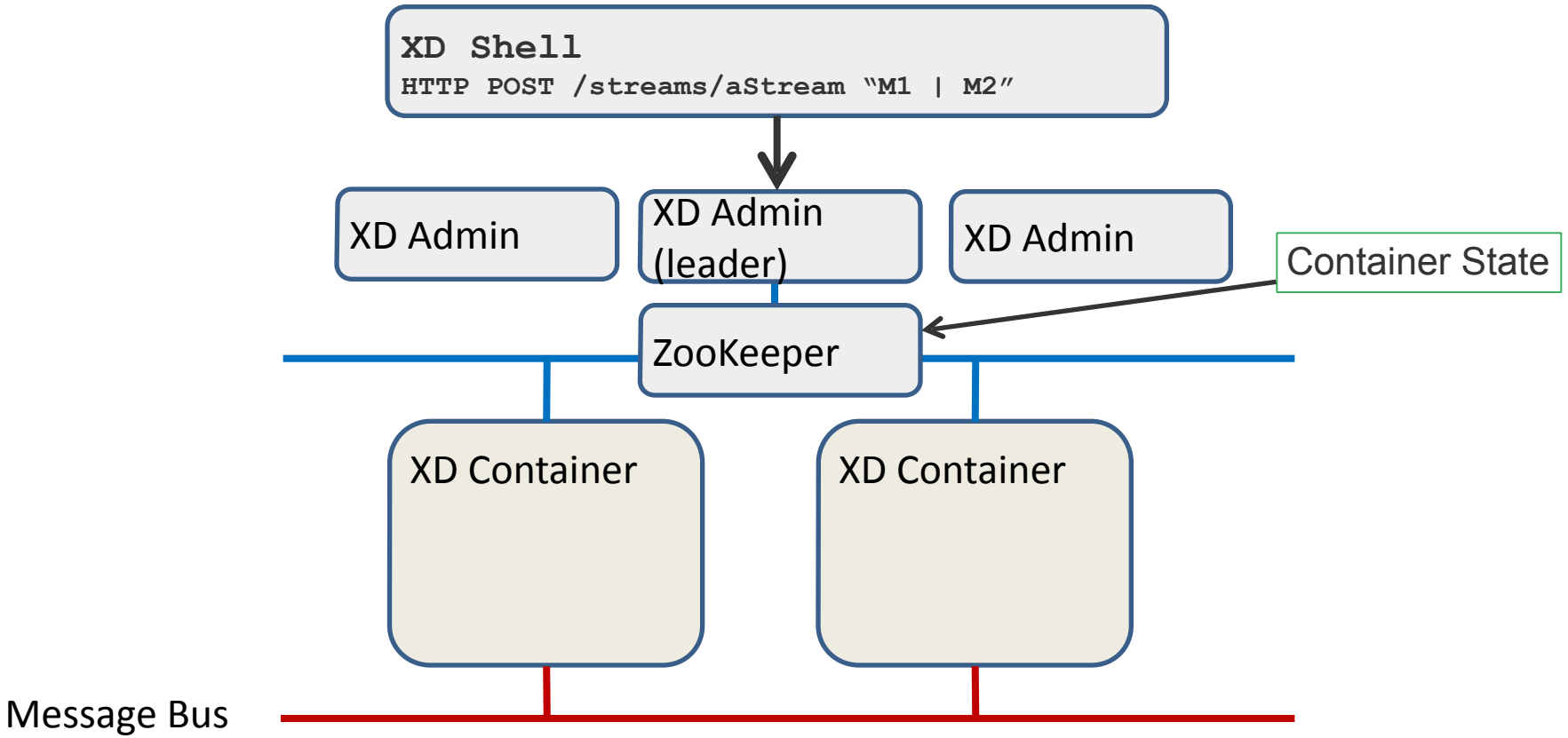
Filter
Transformer
Object-to-JSON
JSON-to-Tuple
Splitter
Aggregator
HTTP Client
JPMML Evaluator
Shell
Groovy
Python
Java

File
HDFS
JDBC
TCP
Log
Mail
RabbitMQ
Gemfire
Splunk
MQTT
Dynamic Router
Counters




spring by Pivotal

# Distributed runtime







spring by Pivotal

# Spring XD and Spring Integration

- **Spring Integration is a foundational component of Spring XD**
  - Message Bus based on Spring Integration Channels
  - Modules are Spring Integration contexts
    - Automatically bridged together in Streams
    - Conventions for Input and Output Channels
    - Runtime designed for concurrency/scaling/failover
- **Spring XD is designed for distributed, modular solutions**
  - Spring Integration components are a natural fit
  - Most Spring XD integrations are reusing Spring Integration channel adapters
- **Spring XD is for less extreme data too :)**




spring by Pivotal

# Into the future




spring by Pivotal

# Spring 4.2 Roadmap (tentative)

- Rework of JMX monitoring and management for performance
- Spring Integration AWS based on Spring Cloud AWS
- More adapters based on existing Spring Data Support
- Kafka enhancements
- Reactive programming support




spring by Pivotal

# Learn More.  Stay Connected.


**GitHub:** github.com/spring-projects/spring-integration

**GitHub:** github.com/spring-projects/spring-integration-samples

**Twitter:** twitter.com/springcentral

**YouTube:** spring.io/video

**LinkedIn:** spring.io/linkedin

**Google Plus:** spring.io/gplus




spring