A new storm is brewing: Spring Data Flow Server for Kubernetes

CodeMash

2017

https://github.com/lseinc/intro-spring-data-flow-CM2017.git

Java / Enterprise

David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com
@DavidDLucas

Introduction: Ground Rules

You will not hurt my feelings if you …

ask questions

respect others wanting to listen

leave because you are bored

want me to change speed (slower / faster)

This does requires audience participation !!!

Introduction: Assumptions

This is an introduction,
deep dives discussions after presentation

We will scratch the surface,
you will need to take the next step

Examples are for informational purposes,
further assembly required

Most pictures are references from spring.io 😃

Introduction: What will be used?

Local Environment

Docker 1.12.5

Java JDK 1.8

Spring Data Flow Server Local 1.1.1

Kubernetes Environment

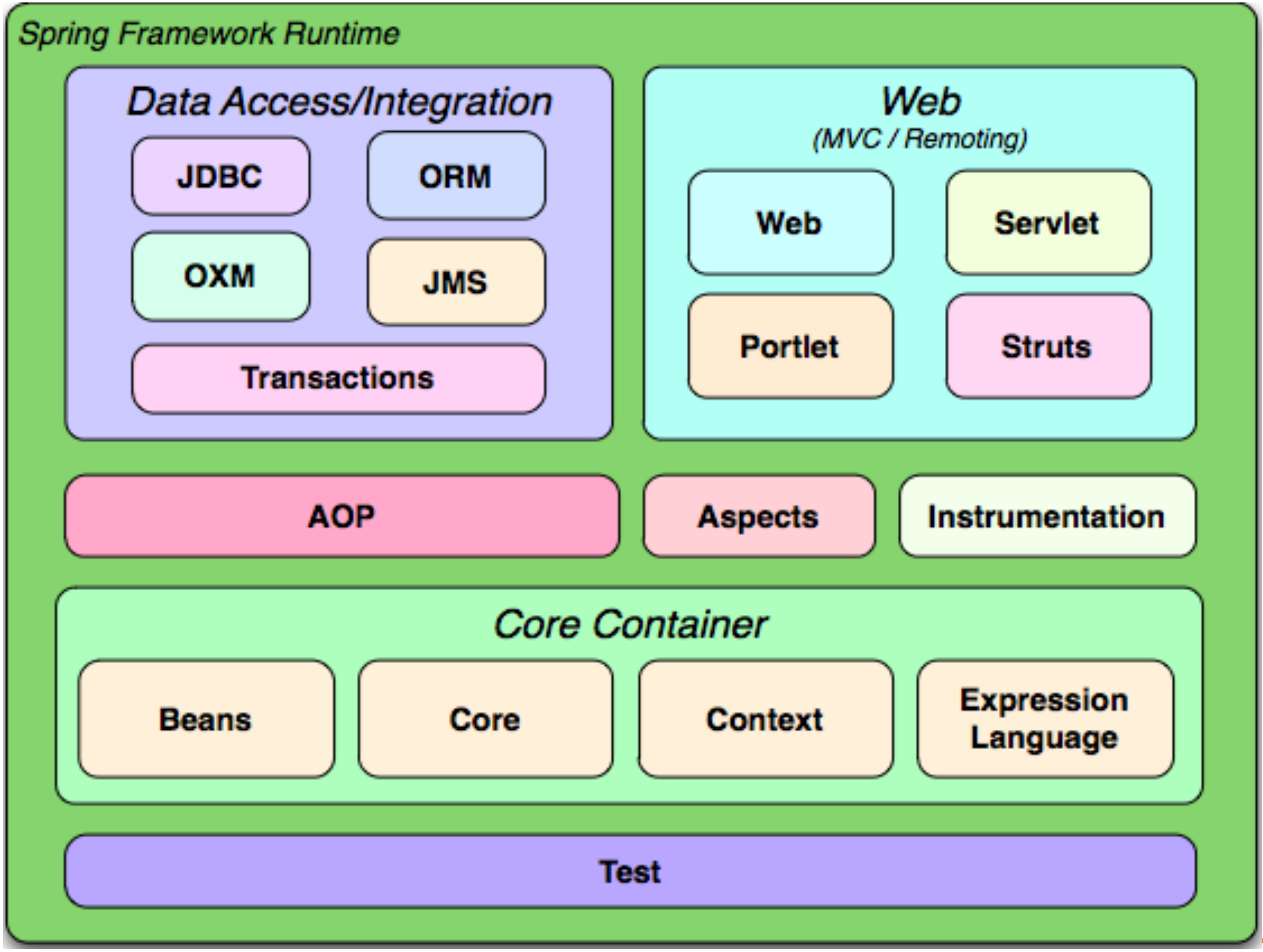
(via minikube 0.14/0.15)

Kubernetes 1.5.1

Docker 1.11

Java JDK 1.8

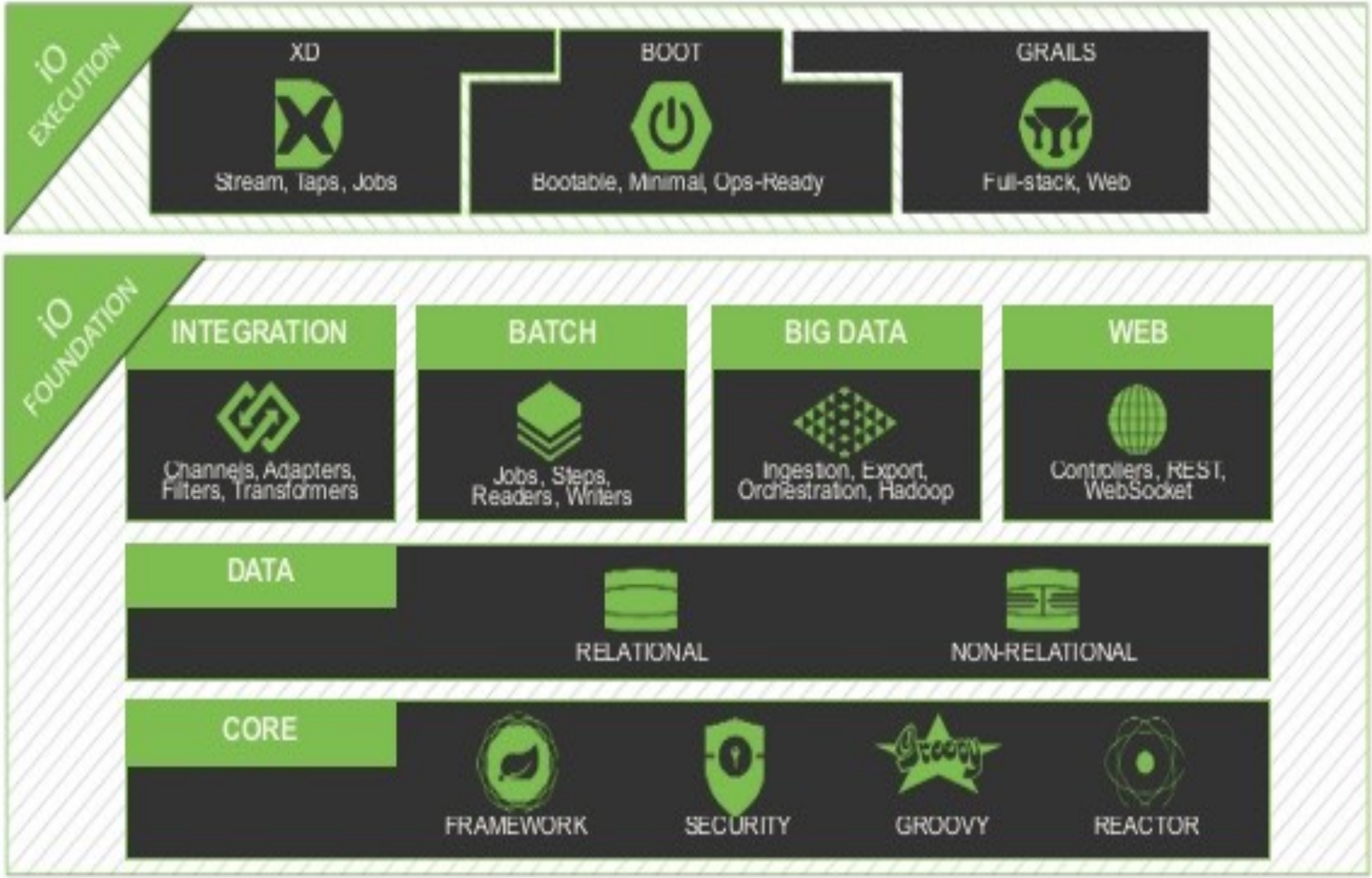
Spring Data Flow Server 1.1.1



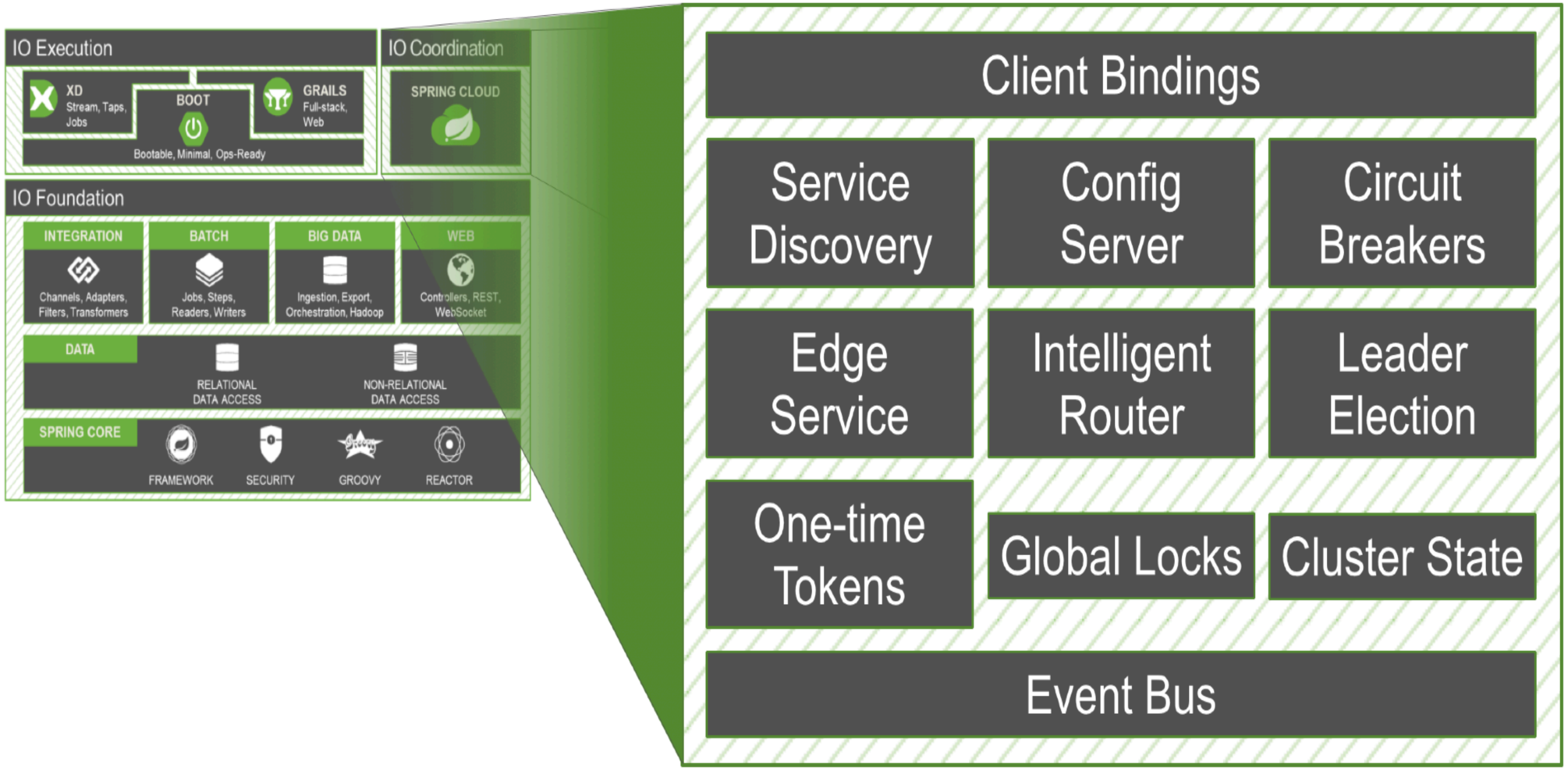
Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
(MVC / Remoting)
Web
Servlet
Portlet
Struts
AOP
Aspects
Instrumentation
Core Container
Beans
Core
Context
Expression Language
Test

NETFLIX OSS

THE SPRING IO PLATFORM (SOME ARCHITECTURE)


iO EXECUTION
XD
Stream, Taps, Jobs
BOOT
Bootable, Minimal, Ops-Ready
GRAILS
Full-stack, Web
iO FOUNDATION
INTEGRATION
Channels, Adapters, Filters, Transformers
BATCH
Jobs, Steps, Readers, Writers
BIG DATA
Ingestion, Export, Orchestration, Hadoop
WEB
Controllers, REST, WebSocket
DATA
RELATIONAL
NON-RELATIONAL
CORE
FRAMEWORK
SECURITY
GROOVY
REACTOR

Introduction: Spring Cloud



Microservices

12 Factors (https://12factor.net)

1. Once codebase tracked in revision control, many deploys
2. Explicitly declare and isolate dependencies
3. Store config in the environment
4. Treat backing services as attached resources
5. Strictly separate build and run stages
6. Execute the app as one or more stateless processes
7. Export services via port binding
8. Scale out via the process model
9. Maximize robustness with fast startup and graceful shutdown
10. Keep development, staging, and production as similar as possible
11. Treat logs as event streams
12. Run admin/management tasks as one-off processes

Microservices

12 Factors (https://12factor.net)

1. Once codebase tracked in revision control, many deploys
2. Explicitly declare and isolate dependencies
3. Store config in the environment
4. Treat backing services as attached resources
5. Strictly separate build and run stages
6. **Execute the app as one or more stateless processes**
7. Export services via port binding
8. **Scale out via the process model**
9. **Maximize robustness with fast startup and graceful shutdown**
10. Keep development, staging, and production as similar as possible
11. Treat logs as event streams
12. Run admin/management tasks as one-off processes

Microservice Types

- Online (frontend)
- Services (backend)

- Event
  (stream, transform, process, analyze, store)
- Task / Job
  (one time or scheduled)

Spring Cloud : Microservices

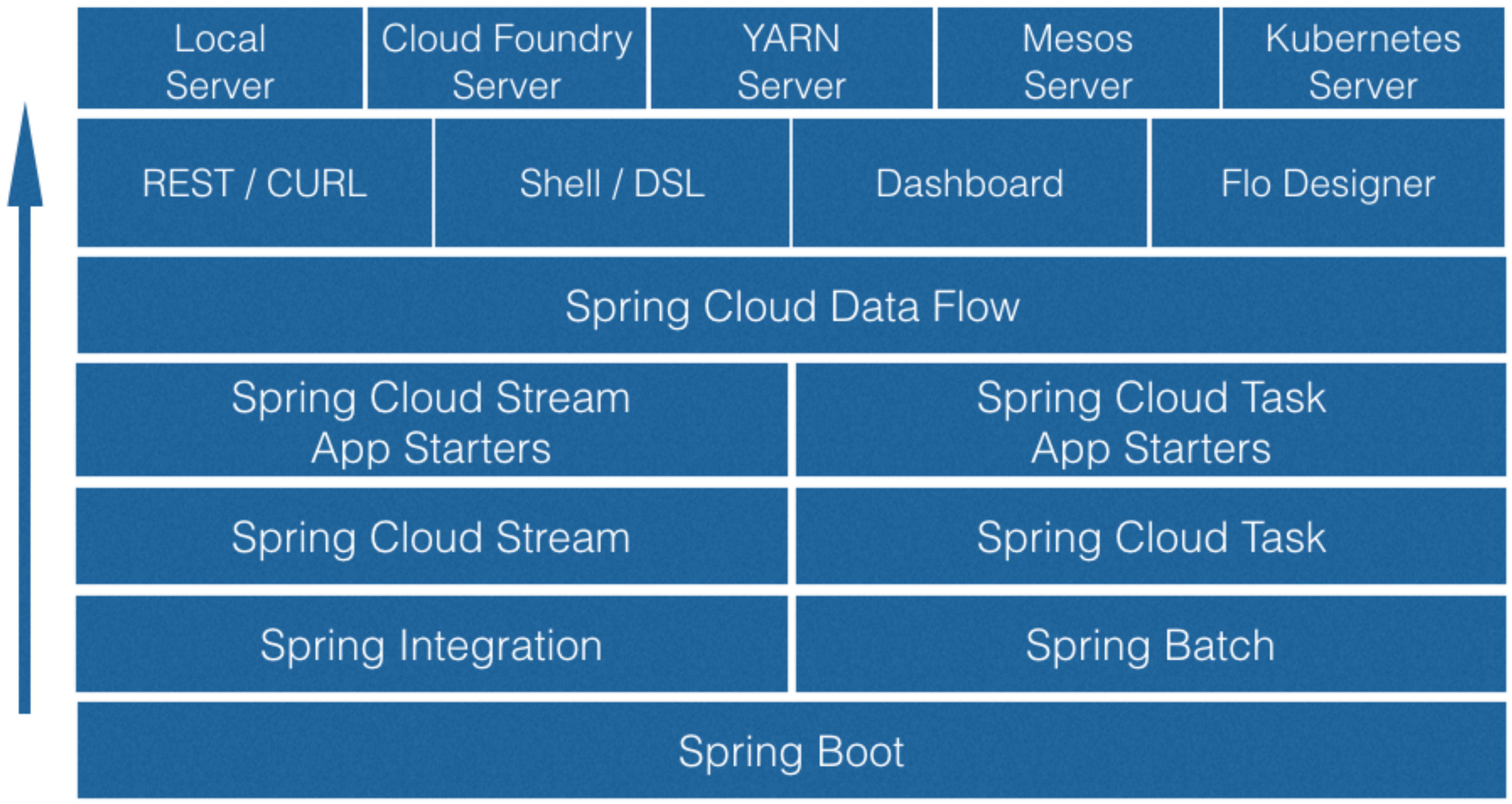
- Zuul: routing / proxy gatekeeper to service contexts
- Eureka: service locator for service contexts
- Hystrix: traffic monitor for circuit breaker clients
- Turbine: monitor multiple Hystrix instances
- Plus lots more (Big Data, Docker, Microservices)
- https://netflix.github.io

Spring Cloud : Data Microservices

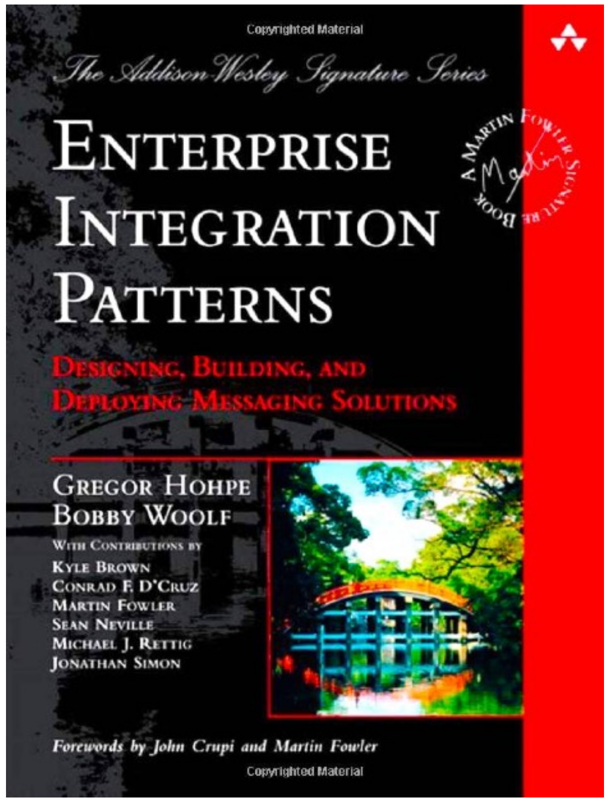
=>

- Spring Integration + Spring Boot
  => Spring XD
- Spring XD + Spring Cloud +
  Spring Stream + Spring Task + Spring Batch
  => Spring Cloud Data Flow

Spring Cloud : Data Microservices

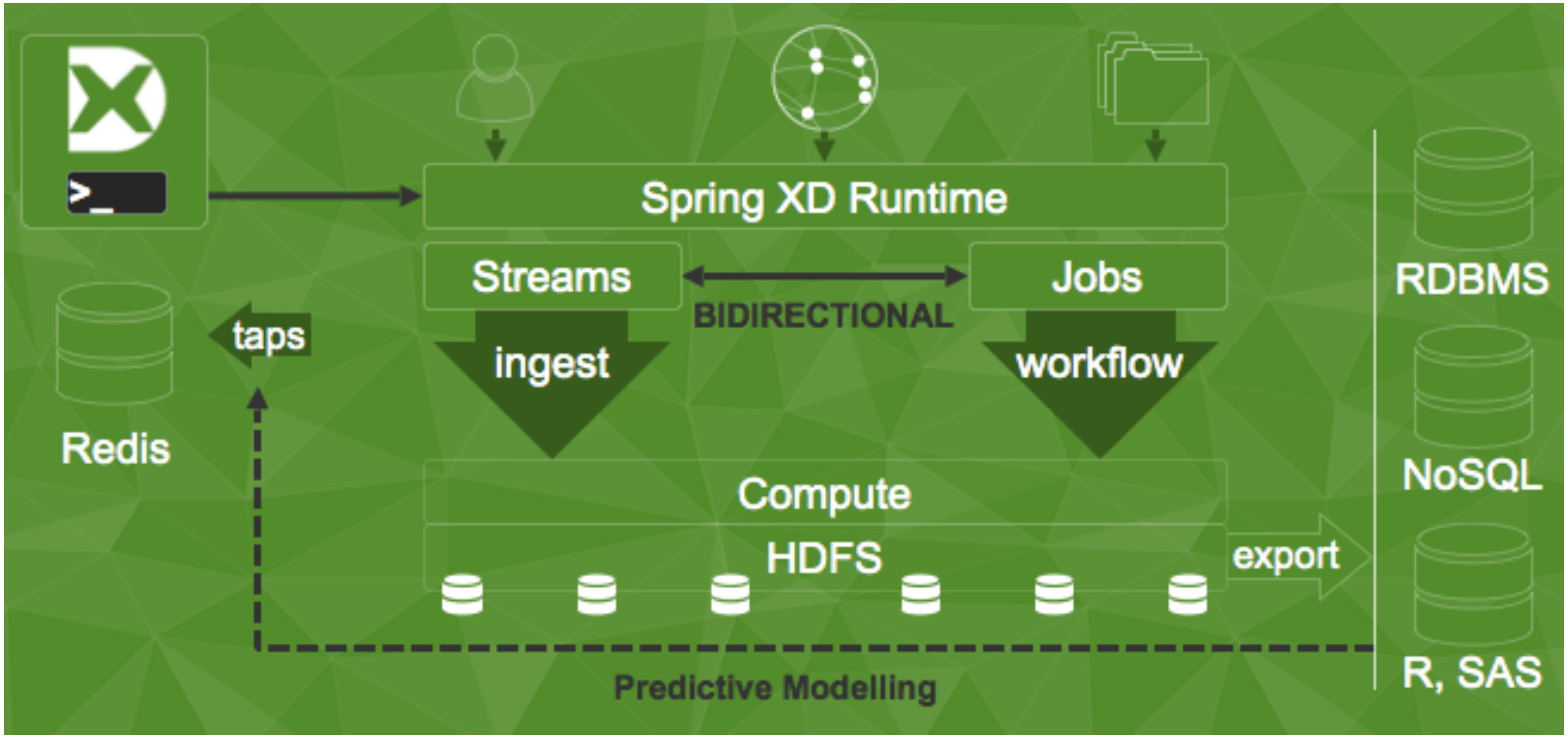


Spring Integration


Copyrighted Material
The Addison-Wesley Signature Series
ENTERPRISE INTEGRATION PATTERNS
DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS
GREGOR HOHPE
BOBBY WOOLF
WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F. D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON
Forewords by John Crupi and Martin Fowler
A MARTIN FOWLER SIGNATURE BOOK

- Based on EAI http://www.enterpriseintegrationpatterns.com/patterns/messaging/
- Channel: mechanism for passing data
- Source: starting point for data to flow
- Processor: manipulates data between a source and sink (filters, transformers)
- Sink: resting place for data
- You can use these without cloud / container (vanilla)

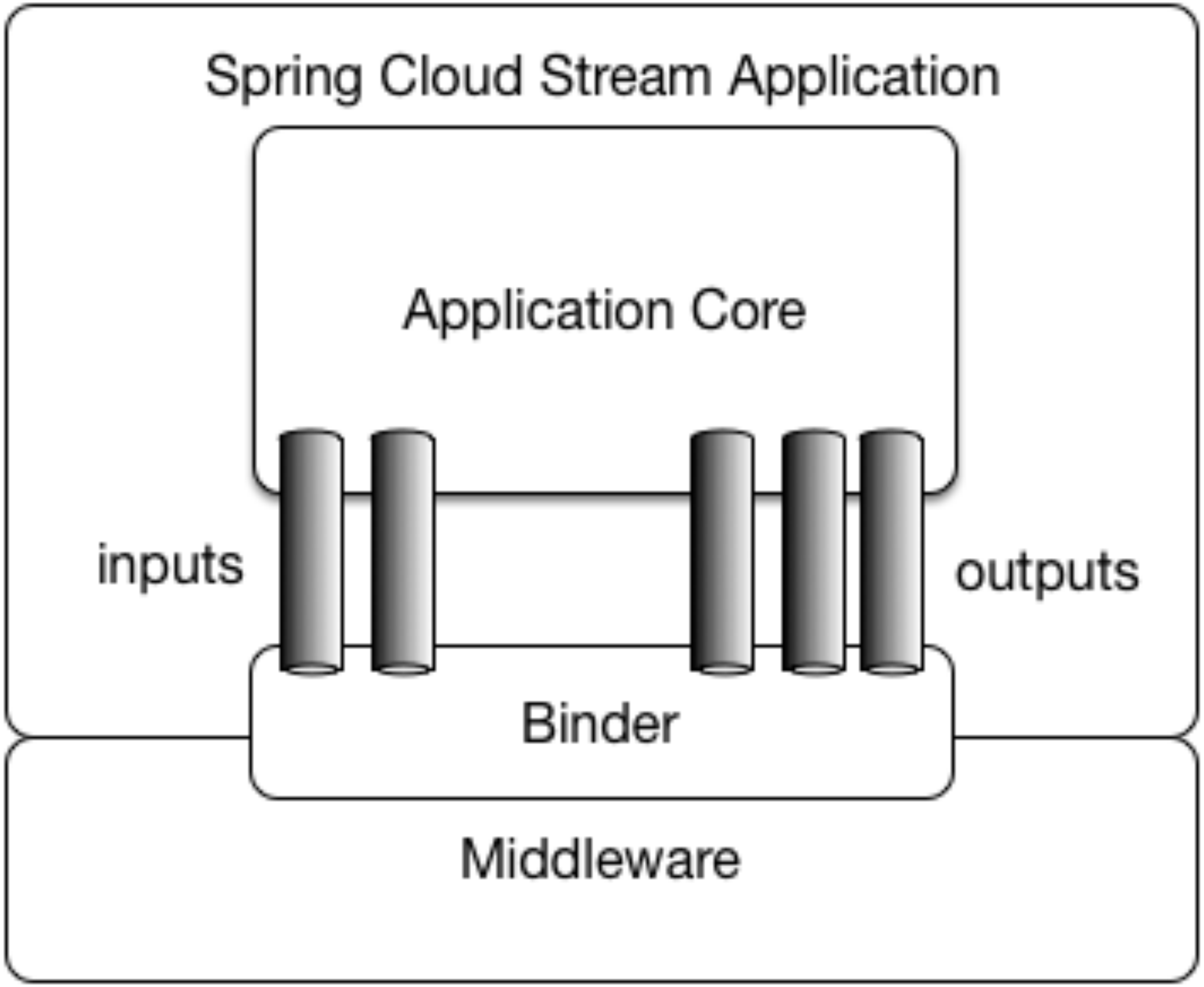
Spring XD



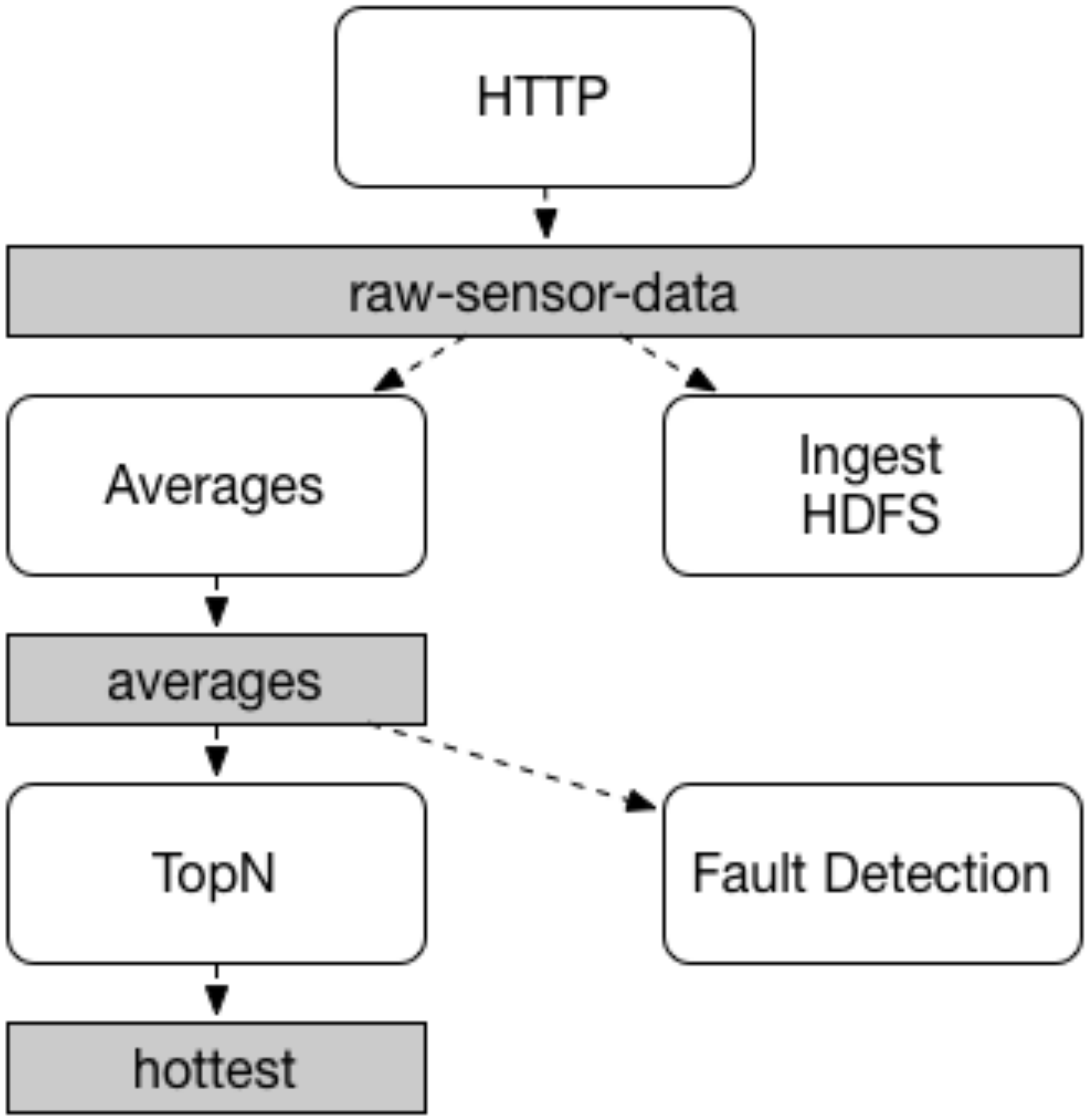
- created environment to deploy streams and jobs like unix processes
- leveraged Spring Integration

Spring Cloud Stream

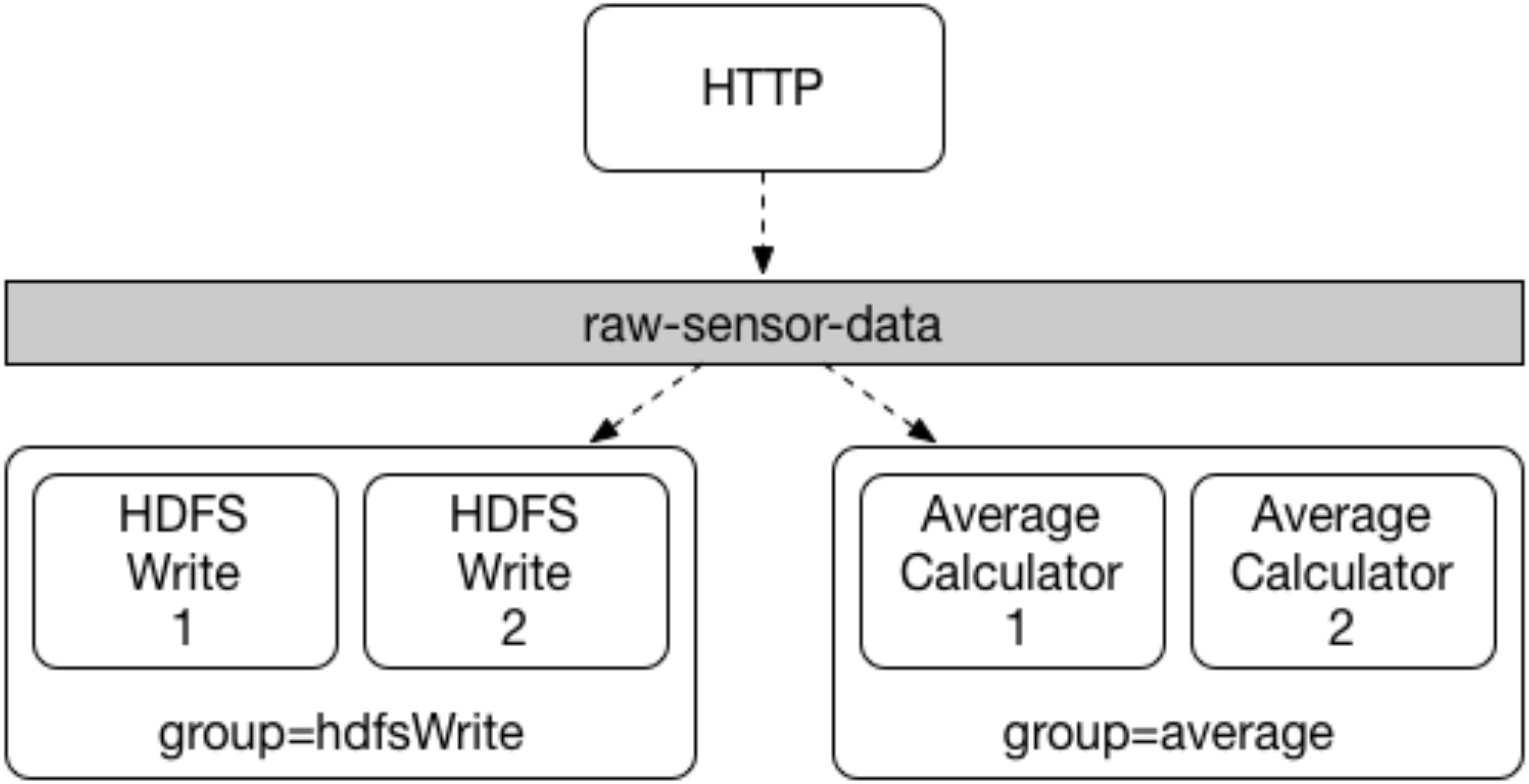
- framework for building message-driven microservices
- leverages Spring Boot and Spring Integration
- integrates with brokers (Kafka, RabbitMQ)
- opinionated configuration
  - message brokers
  - persistent pub/sub semantics
  - consumer groups with partitions
- @EnableBinding sets up main application to connect to message broker
- @StreamListener enables a method to receive events


Spring Cloud Stream Application
Application Core
inputs
outputs
Binder
Middleware

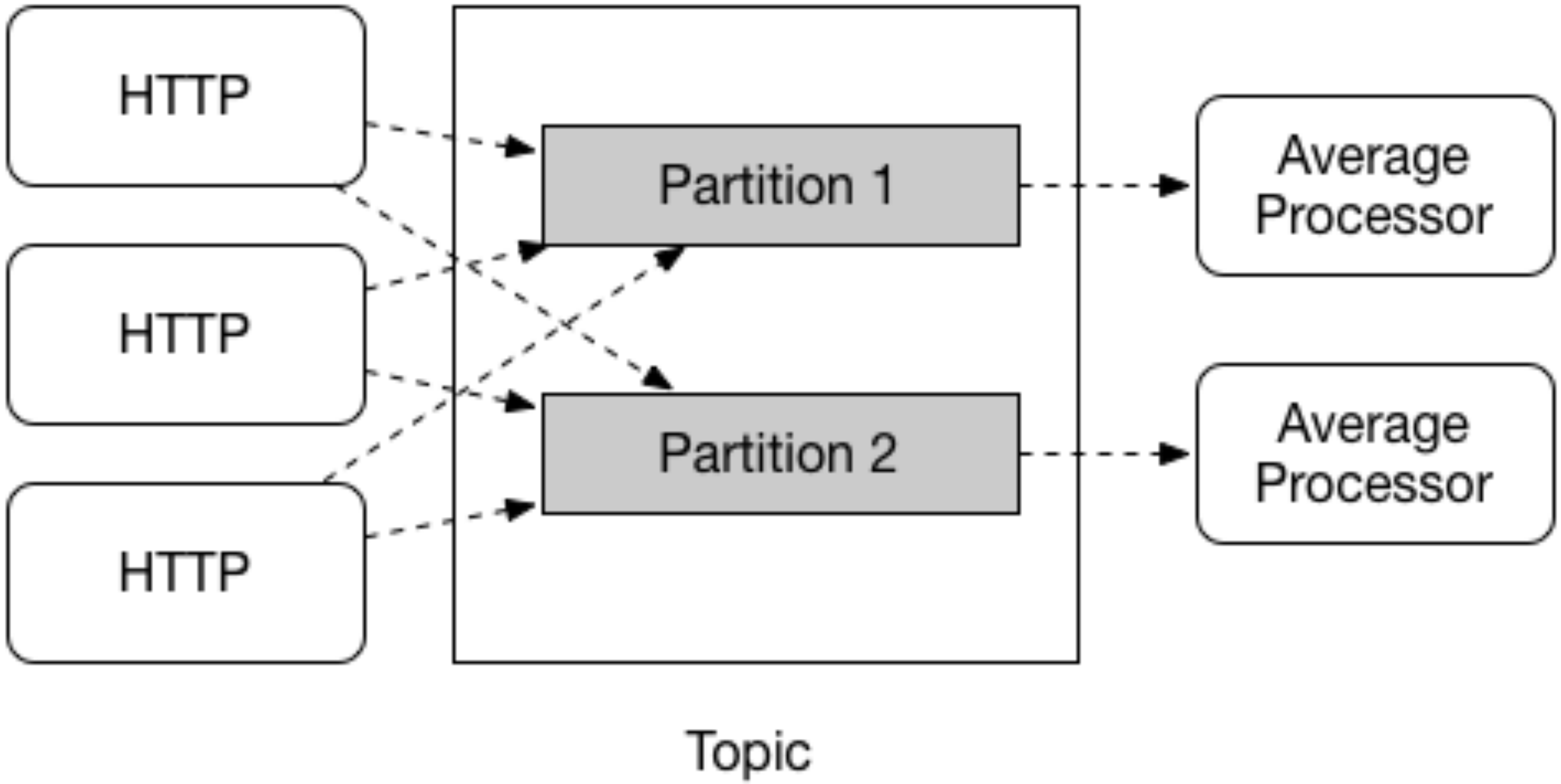
Spring Cloud Stream Publish-Subscribe



Spring Cloud Stream Consumer Groups



Spring Cloud Stream Partitioning



BTW: Unix got it right !

cat data.csv | **grep** “LOGIN“ | **tr** -d’,’ -f1-3 > login.csv

<stream> | <process> | <sink>

- simple
- each module does one thing well
- leverages other capabilities

Spring Cloud Stream: Source

```
@EnableBinding(Source.class)
public class TimerSource {

    @Value("${format}")
    private String format;

    @Bean
    @InboundChannelAdapter(value = Source.OUTPUT, poller =
          @Poller(fixedDelay = "${fixedDelay}", maxMessagesPerPoll = "1"))
    public MessageSource<String> timerMessageSource() {
        return () -> new GenericMessage<>(new SimpleDateFormat(format).format(new Date()));
    }
}
```

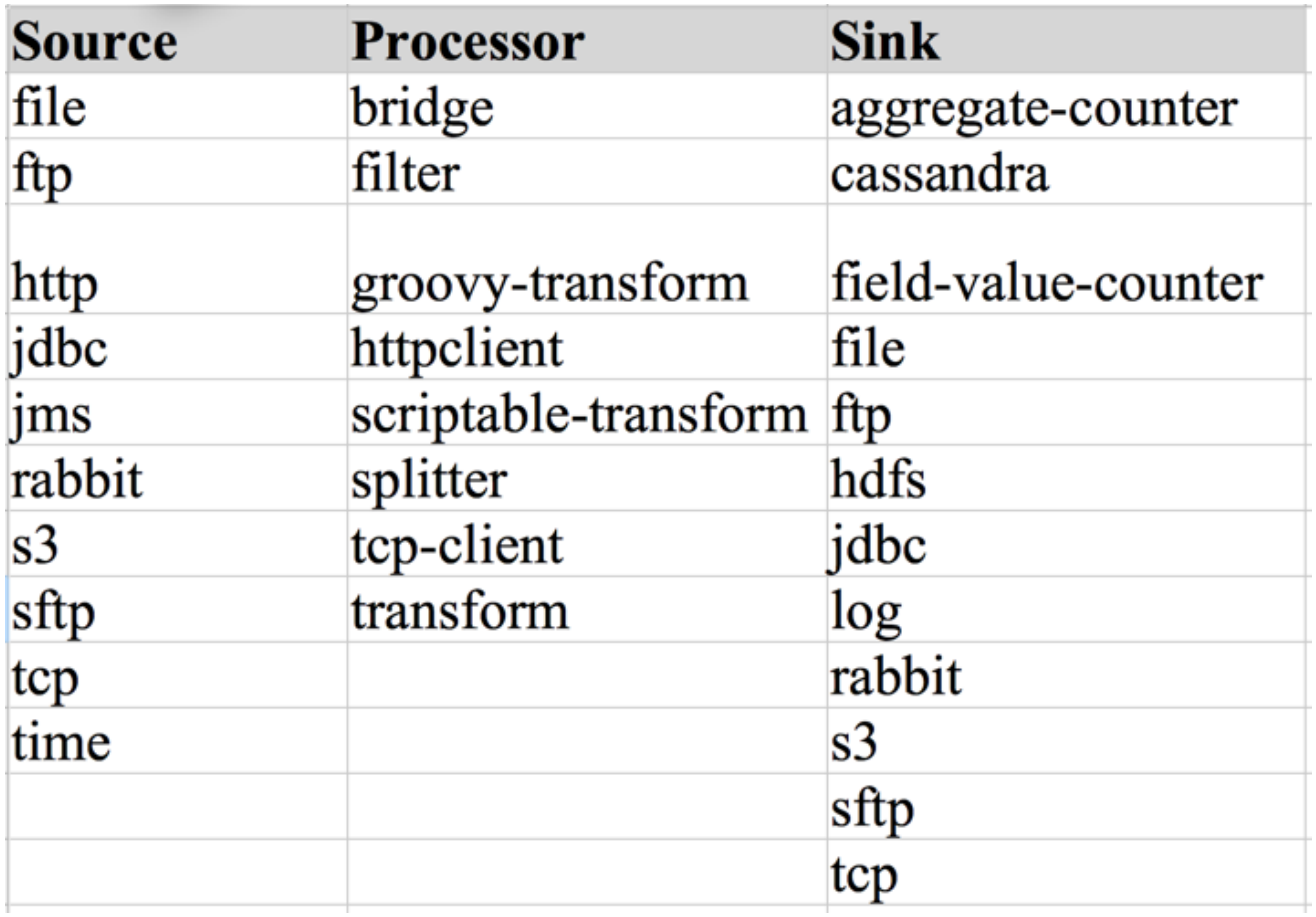
Spring Cloud Stream: Processor

```
@EnableBinding(Processor.class)
public class TransformProcessor {
    @Transformer(inputChannel = Processor.INPUT,
                 outputChannel = Processor.OUTPUT)
    public Object transform(String message) {
        return message.toUpper();
    }
}
```

Spring Cloud Stream: Sink

```
@EnableBinding(Sink.class)
public class VoteHandler {

  @Autowired
  VotingService votingService;

  @StreamListener(Sink.INPUT)
  public void handle(Vote vote) {
    votingService.record(vote);
  }
}
```

Spring Cloud Stream Apps



Source	Processor	Sink
file	bridge	aggregate-counter
ftp	filter	cassandra
http	groovy-transform	field-value-counter
jdbc	httpclient	file
jms	scriptable-transform	ftp
rabbit	splitter	hdfs
s3	tcp-client	jdbc
sftp	transform	log
tcp		rabbit
time		s3
		sftp
		tcp

Spring Cloud Task

- Simple Task Execution
- @EnableTask in main app
- Launched by Spring Data Flow
- uses CommandLineRunner interface

Spring Cloud Task

```
@SpringBootApplication
@EnableTask
public class MyApp {

   @Bean
   public MyTaskApplication myTask() {
       return new MyTaskApplication();
   }

   public static void main(String[] args) {
       SpringApplication.run(MyApp.class);
   }

   public static class MyTaskApplication implements CommandLineRunner {

       @Override
       public void run(String... strings) throws Exception {
           System.out.println("Hello World");
       }
   }
}
```

Spring Cloud Task

- Simple Task Execution
- @EnableTask in main app
- Launched by Spring Data Flow
- uses CommandLineRunner interface

Spring Batch Integration

- Batch “Step” Execution
- @EnableBatchProcessing
- Launched by Spring Data Flow
- uses JobLaucher

Spring Cloud Data Flow Server

- cloud-native redesign of Spring XD
  - stream (Spring Cloud Stream)
  - batch / task (Spring Cloud Task)
- goal to simplify Big Data applications
- Pre-Built collection of microservices
- Unix Style DSL microservices pipeline
- Consumes versioned artifacts
  - maven
  - docker

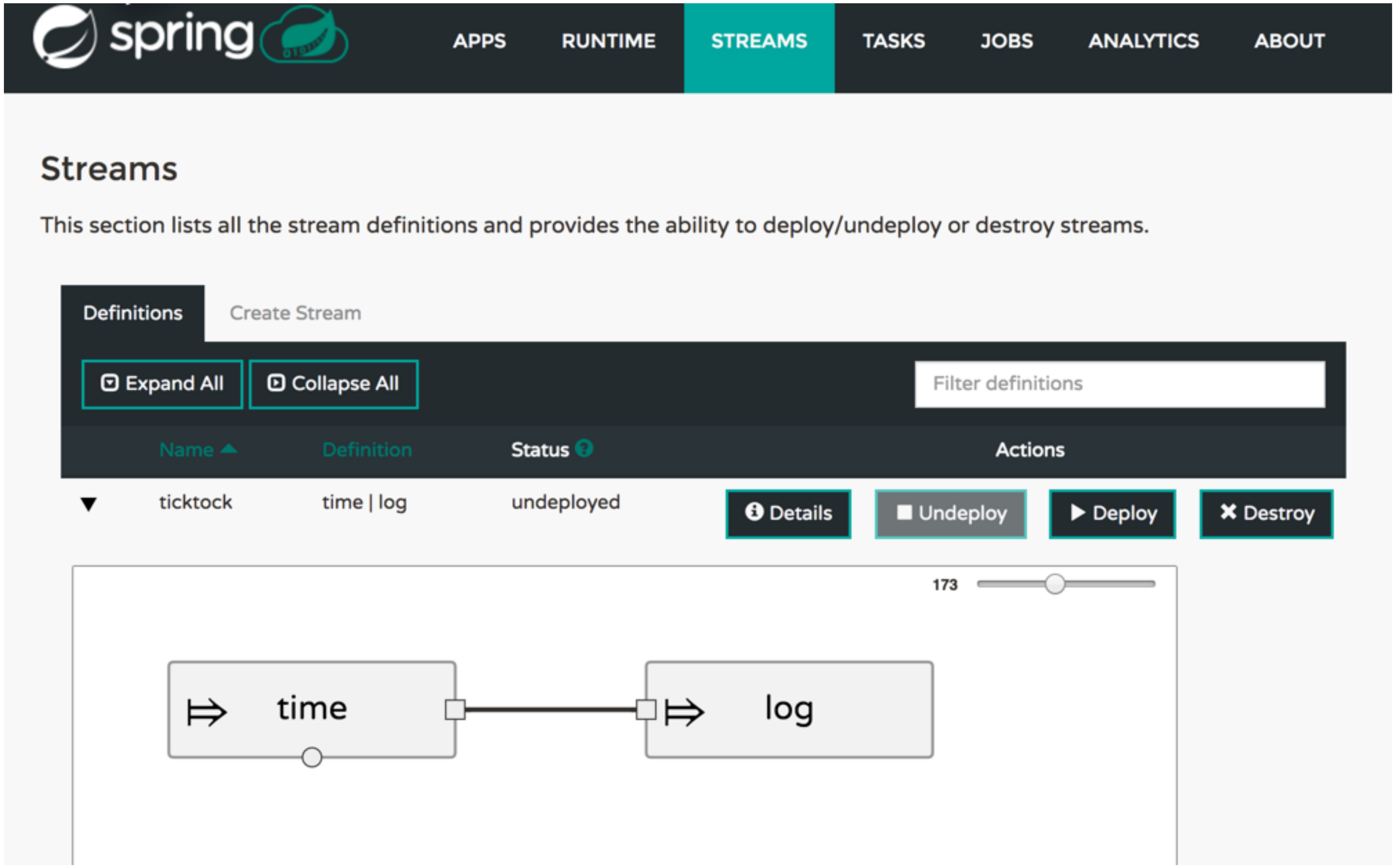
Spring Cloud Data Flow Server

- Support for multiple environments
  - Cloud Foundry
  - Apache YARN
  - Apache Mesos
  - Kubernetes
- Metrics and health checks
- Remote management

Spring Cloud Data Flow Dashboard

- Support for multiple environments
  - Cloud Foundry
  - Apache YARN
  - Apache Mesos
  - Kubernetes
- Metrics and health checks
- Remote management

Spring Cloud Data Flow Server Dashboard

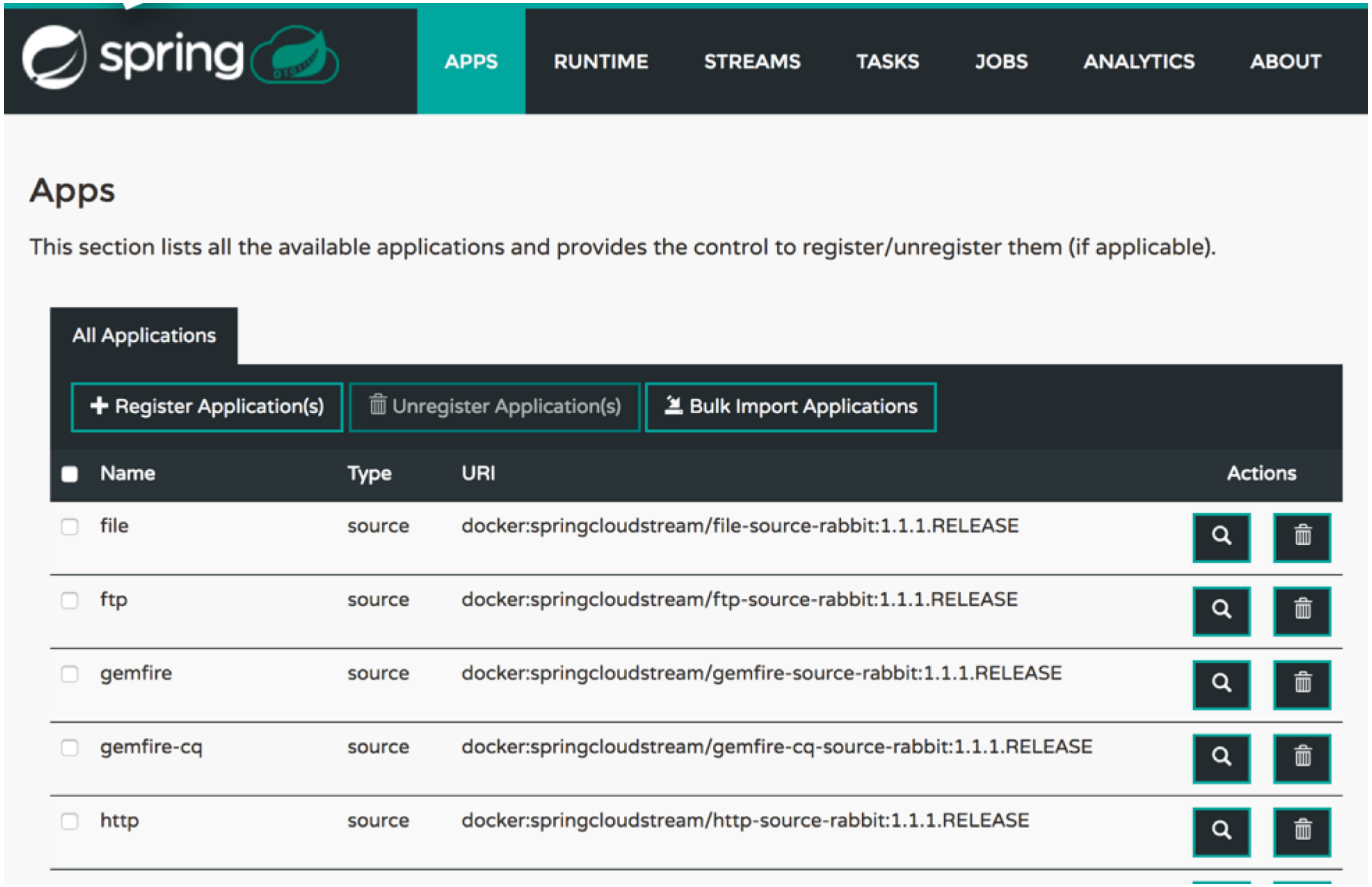


Streams

This section lists all the stream definitions and provides the ability to deploy/undeploy or destroy streams.

Name	Definition	Status	Actions
ticktock	time | log	undeployed	Details / Undeploy / Deploy / Destroy

Spring Cloud Data Flow Server Dashboard

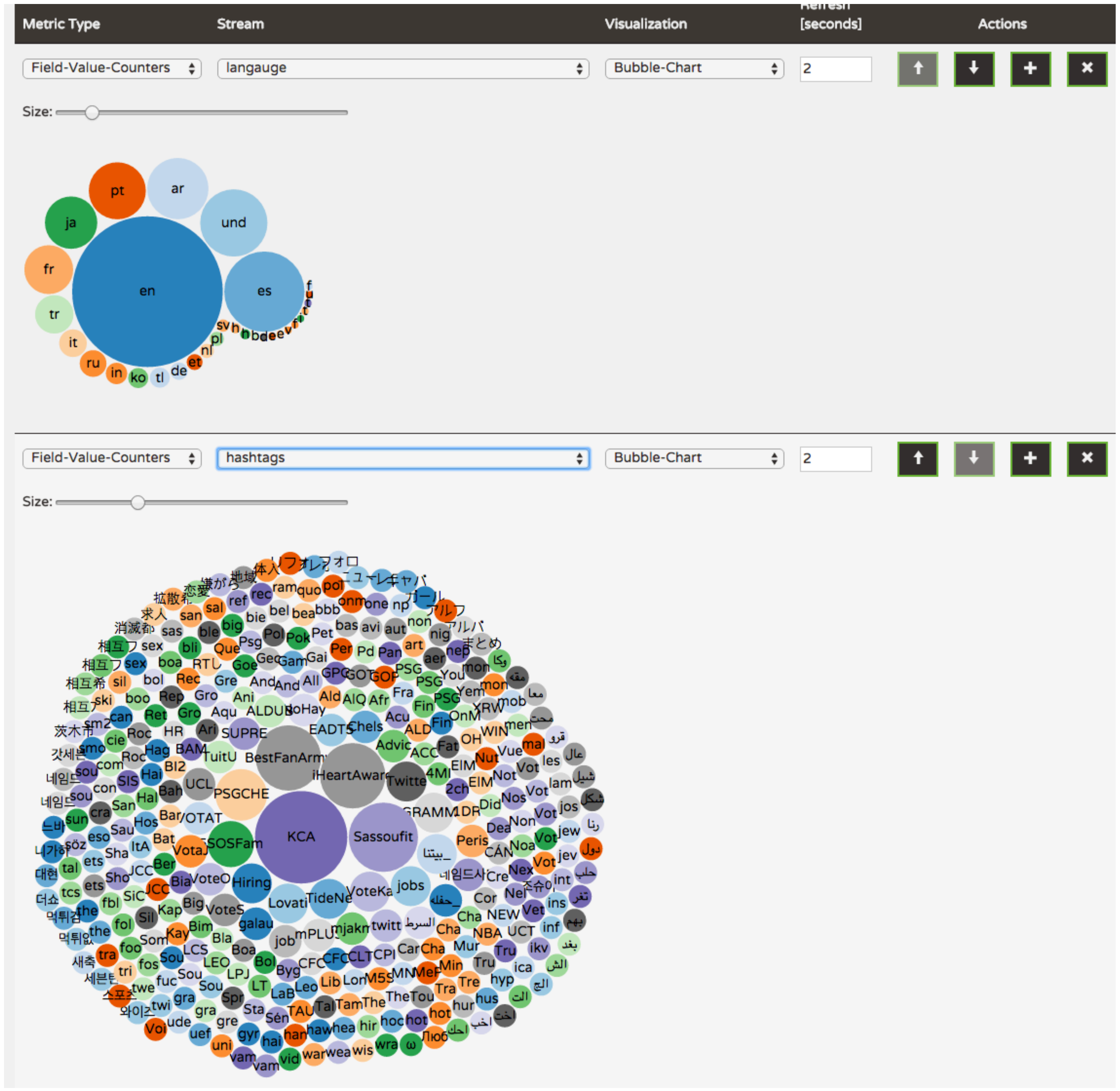


Apps

This section lists all the available applications and provides the control to register/unregister them (if applicable).

Name	Type	URI	Actions
file	source	docker:springcloudstream/file-source-rabbit:1.1.1.RELEASE	
ftp	source	docker:springcloudstream/ftp-source-rabbit:1.1.1.RELEASE	
gemfire	source	docker:springcloudstream/gemfire-source-rabbit:1.1.1.RELEASE	
gemfire-cq	source	docker:springcloudstream/gemfire-cq-source-rabbit:1.1.1.RELEASE	
http	source	docker:springcloudstream/http-source-rabbit:1.1.1.RELEASE	

Spring Cloud Data Flow Server Dashboard



Metric Type	Stream	Visualization	Refresh [seconds]	Actions
Field-Value-Counters	langauge	Bubble-Chart	2	
Field-Value-Counters	hashtags	Bubble-Chart	2	

Size:

Spring Cloud Data Flow Server Shell

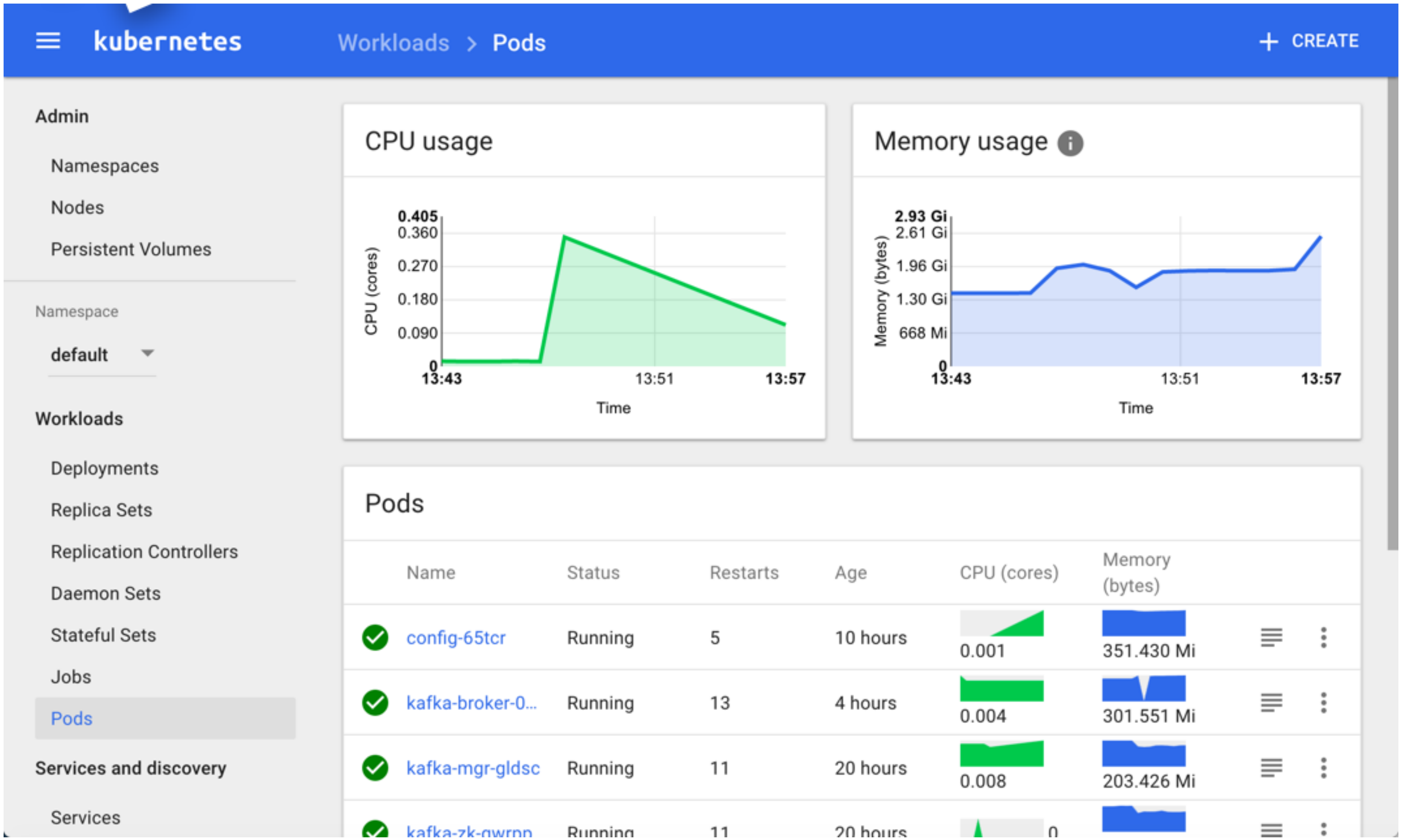
```
  ____                              ____ _                __
 / ___| _ __  _ __(_)_ __   __ _   / ___| | ___  _   _  __| |
 \___ \| '_ \| '__| | '_ \ / _` | | |   | |/ _ \| | | |/ _` |
  ___) | |_) | |  | | | | | (_| | | |___| | (_) | |_| | (_| |
 |____/| .__/|_|  |_|_| |_|\__, |  \____|_|\___/ \__,_|\__,_|
  ____ |_|    _          __|___/                 __________
 |  _ \  __ _| |_ __ _  |  ___| | _____      __  \ \ \ \ \ \
 | | | |/ _` | __/ _` | | |_  | |/ _ \ \ /\ / /   \ \ \ \ \ \
 | |_| | (_| | || (_| | |  _| | | (_) \ V  V /    / / / / / /
 |____/ \__,_|\__\__,_| |_|   |_|\___/ \_/\_/    /_/_/_/_/_/

1.1.1.BUILD-SNAPSHOT

Welcome to the Spring Cloud Data Flow shell. For assistance hit TAB or type "help".
dataflow:>app import --uri file:../Avogadro-GA-stream-applications-rabbit-maven
dataflow:>stream create --name httptest --definition "http --server.port=9000 | log" --deploy
Created new stream 'httptest'
Deployment request has been sent
dataflow:>
```

DEMO
Spring Cloud Data Flow Server Local

Kubernetes

- production grade orchestration
- Pods
- Services
- Cordination
- Cluster? (Federation coming soon)

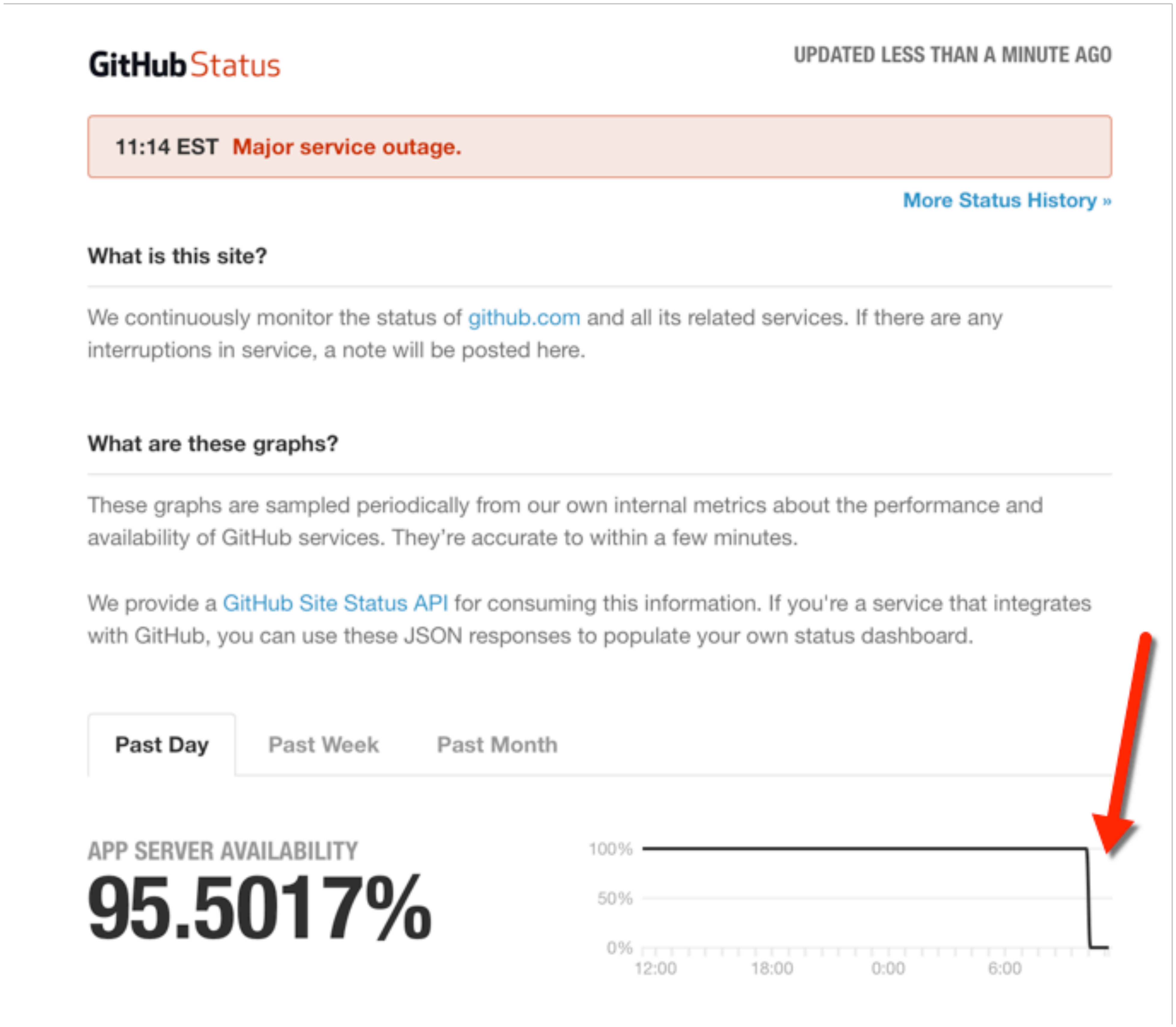
Kubernetes



Pods

	Name	Status	Restarts	Age	CPU (cores)	Memory (bytes)
✓	config-65tcr	Running	5	10 hours	0.001	351.430 Mi
✓	kafka-broker-0...	Running	13	4 hours	0.004	301.551 Mi
✓	kafka-mgr-gldsc	Running	11	20 hours	0.008	203.426 Mi
✓	kafka-zk-qwrpp	Running	11	20 hours	0	

DEMO
Spring Cloud Data Flow Kubernetes



GitHub Status

UPDATED LESS THAN A MINUTE AGO

11:14 EST **Major service outage.**

More Status History »

What is this site?

We continuously monitor the status of github.com and all its related services. If there are any interruptions in service, a note will be posted here.

What are these graphs?

These graphs are sampled periodically from our own internal metrics about the performance and availability of GitHub services. They're accurate to within a few minutes.

We provide a GitHub Site Status API for consuming this information. If you're a service that integrates with GitHub, you can use these JSON responses to populate your own status dashboard.

APP SERVER AVAILABILITY

95.5017%

Please remember CodeMash speaker
Jim Holmes
and his family
in your thoughts and prayers !!!

Thank You !!!

Please fill out EventsXD Survey !!!

Contact Info

David D Lucas

Email: ddlucas@lse.com

GitHub: https://github.com/lseinc

Twitter: @DavidDLucas

LinkedIn: https://www.linkedin.com/in/ddlucas