Event-Driven Spring

Craig Walls
craig@habuma.com
Twitter: @habuma

What to expect…

Event-Driven vs. Imperative Processing

Handling Events in Spring

What to expect…

Some live coding

Misteaks will be maid

What we'll (try to) do...

11:00 - 12:30 :: Foundations

Introduction, Setting Up, Templates, Message-Driven Beans

1:00 - 2:30 :: Going Higher

Spring Integration, Spring Cloud Stream, Spring Cloud DataFlow

All of this is tentative and flexible

What you'll need

Rabbit (or Kafka) running locally

The easy way:

```
$ docker run -d --name my-rabbit -p 5672:5672 -p 15672:15672 rabbitmq:3-management
```

Other ways:

https://www.rabbitmq.com/download.html

Your IDE of choice

(ideally, Spring ToolSuite or IntelliJ IDEA)

The code and slides?

All code and these slides will be at...

https://github.com/habuma/nfjs-samples/tree/master/EventDrivenSpring-Workshop-Apr-2021

...after the workshop concludes

What is event-driven?

Models the real world

What is event-driven?

Deals in state

What is event-driven?

Lots of small changes to that state every day

What is event-driven?

These small changes are events

What is event-driven?

Event driven architecture (EDA)

What is event-driven?

Patterns for handling events…

What is event-driven?

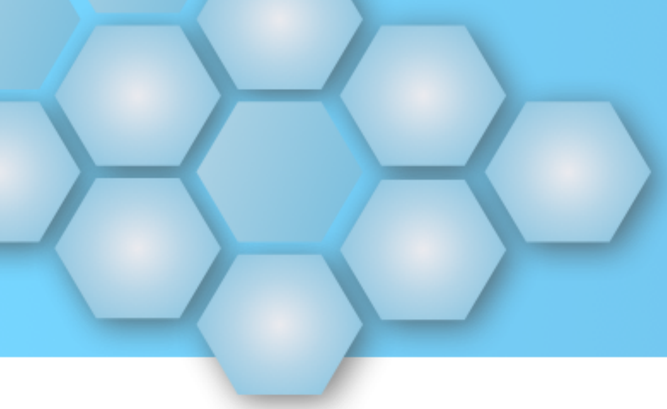
…and possibly creating new events.

What is event-driven?

Achieves loose coupling

What is event-driven?

Especially in the context of microservices



Microservices?

FACT: Microservices != REST

Loose Coupling

Services do not know where events come from

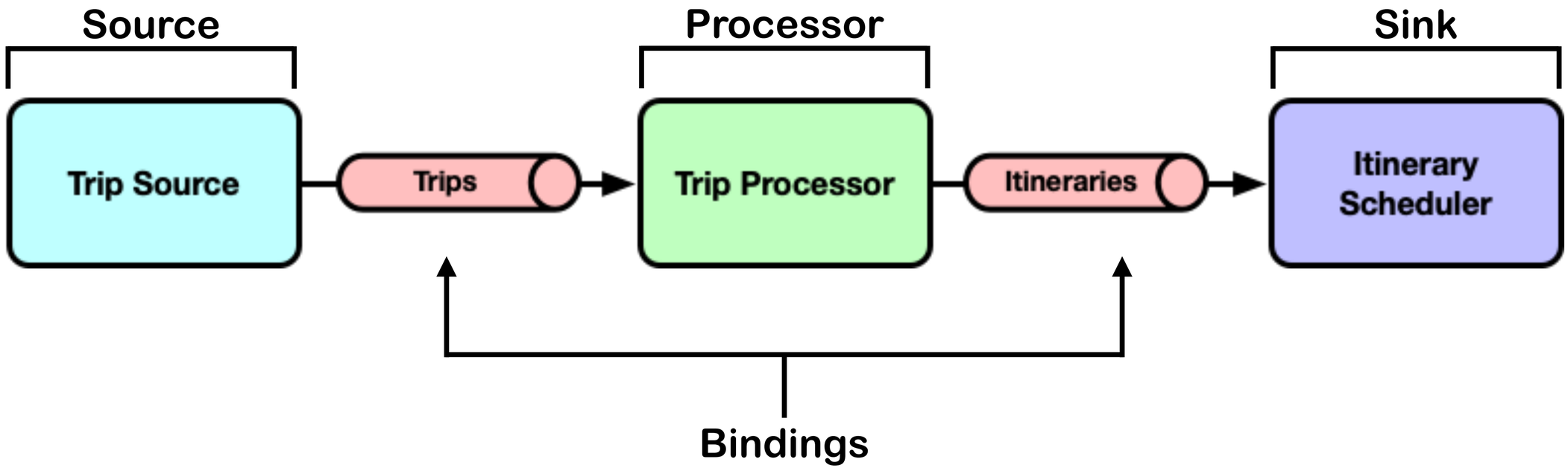
Loose Coupling

Services do not know where events they create will go next

Loose Coupling

Services have one job : Process the event

EDA Illustrated



Event-Driven Spring

Spring offers several choices for EDA

Message Templates

Event-Driven Spring

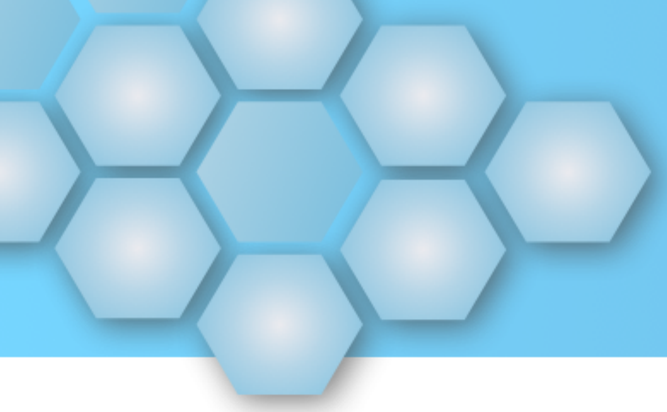
Message-driven beans

Event-Driven Spring

Spring Integration

Event-Driven Spring

Spring Cloud Stream



Event-Driven Spring

Spring Cloud Data Flow

And then there's…

Spring Cloud Function

Let’s see some code…

Message Driven Beans

```
@RabbitListener(queues = "trips")
@SendTo("itineraries")
public Itinerary handleMessage(TripBooking trip) {
  // … process trip …
  return itinerary;
}
```

```
@RabbitListener(queues = "itineraries")
public void handleMessage(Itinerary itinerary) {
  // … process itinerary …
}
```

Message Driven Beans

Reactive; queue-watching is handled by framework

No clear way to break processing into distinct operations

Coupled to message broker; Both consumer and producer must agree on broker

JMS, RabbitMQ, or Kafka

Spring Integration

```
@Bean
public IntegrationFlow flow(
    ConnectionFactory connectionFactory,
    AmqpTemplate amqpTemplate,
    TripProcessor processor) {
  return IntegrationFlows
      .from(new AmqpMessageSource(connectionFactory, "trips"),
          c->c.poller(Pollers.fixedRate(1000)))
      .<TripBooking, Itinerary>transform(trip -> processor.process(trip))
      .handle(Amqp.outboundGateway(amqpTemplate).routingKey(“itineraries"))
      .get();
}
```

Spring Integration

Reactive

Processing can be split into distinct operations

Loosely coupled to message broker

Events are not necessarily sourced from a Message broker

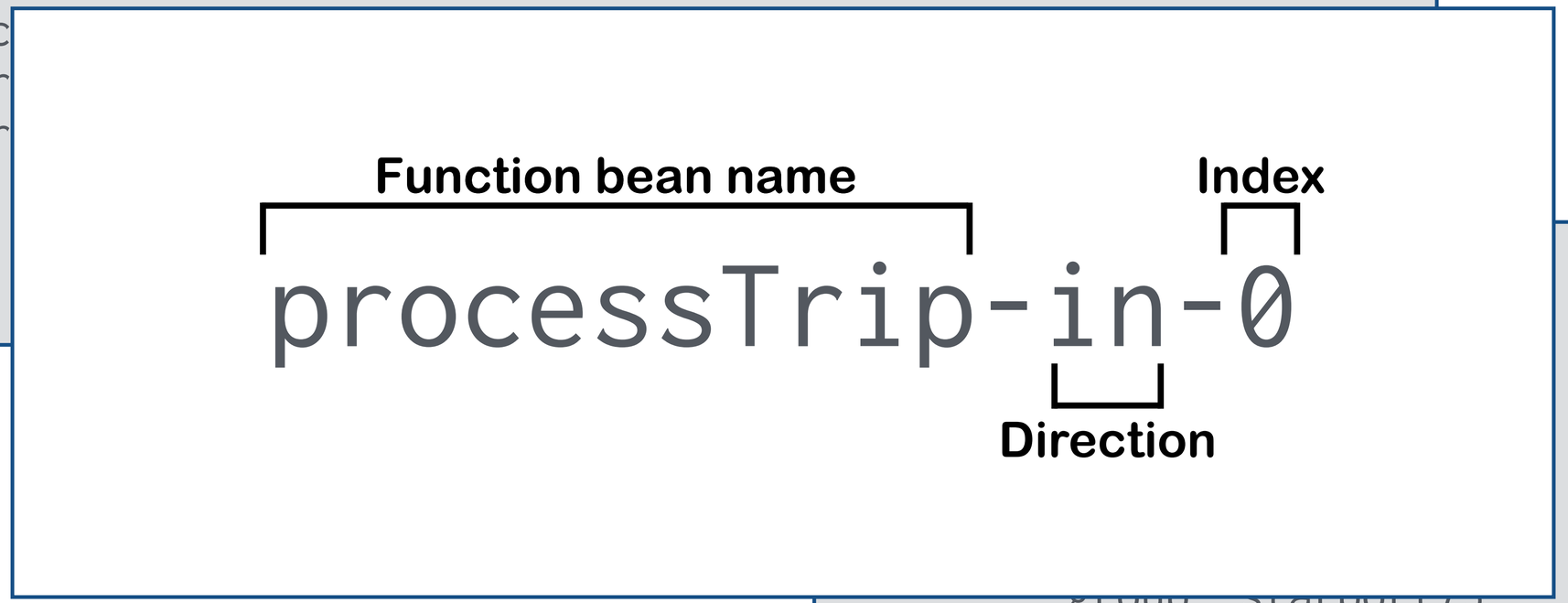
Spring Cloud Stream

```
@Bean
public Function<TripBooking, Itinerary> processTrip() {
  return trip -> {
    // … process trip …
    return trip.getItinerary();
  };
}
```

```
spring:
  cloud:
    stream:
      bindings:
        processTrip-in-0:
          destination: trips
          group: starport75
        processTrip-out-0:
          destination: itineraries
          group: starport75
```

Spring Cloud Stream

```
@Bean
public Func
  return tr
    // … pr
    return
  };
}
```




```
    group: starport75
  processTrip-out-0:
    destination: itineraries
    group: starport75
```

Spring Cloud Stream

```
@Bean
public Consumer<Itinerary> scheduleItinerary() {
  return itinerary -> {
    // … process itinerary …
  };
}
```

```
spring:
  cloud:
    stream:
      bindings:
        scheduleItinerary-in-0:
          destination: itineraries
          group: starport75
```

Spring Cloud Stream

Processing can be easily split into distinct operations

Message-broker is the binding

Events are not necessarily sourced from a Message broker

A bit on Spring Cloud DataFlow

Spring Cloud DataFlow

Out-of-the-box applications

Sources

file
ftp
gemfire
gemfire-cq
http
jdbc
jms
load-generator
loggregator
mail
mongodb
mqtt
rabbit
s3
sftp
sftp-dataflow
syslog
tcp
tcp-client
time
trigger
twitterstream

Processors

aggregator
bridge
counter
filter
groovy-filter
groovy-transform
grpc
header-enricher
httpclient
image-recognition
object-detection
pmml
pose-estimation
python-http
python-jython
scriptable-transform
splitter
tcp-client
tensorflow
transform
twitter-sentiment

Sinks

cassandra
counter
file
ftp
gemfire
gpfdist
hdfs
jdbc
log
mongodb
mqtt
pgcopy
rabbit
redis-pubsub
router
s3
sftp
task-launcher-dataflow
tcp
throughput
websocket

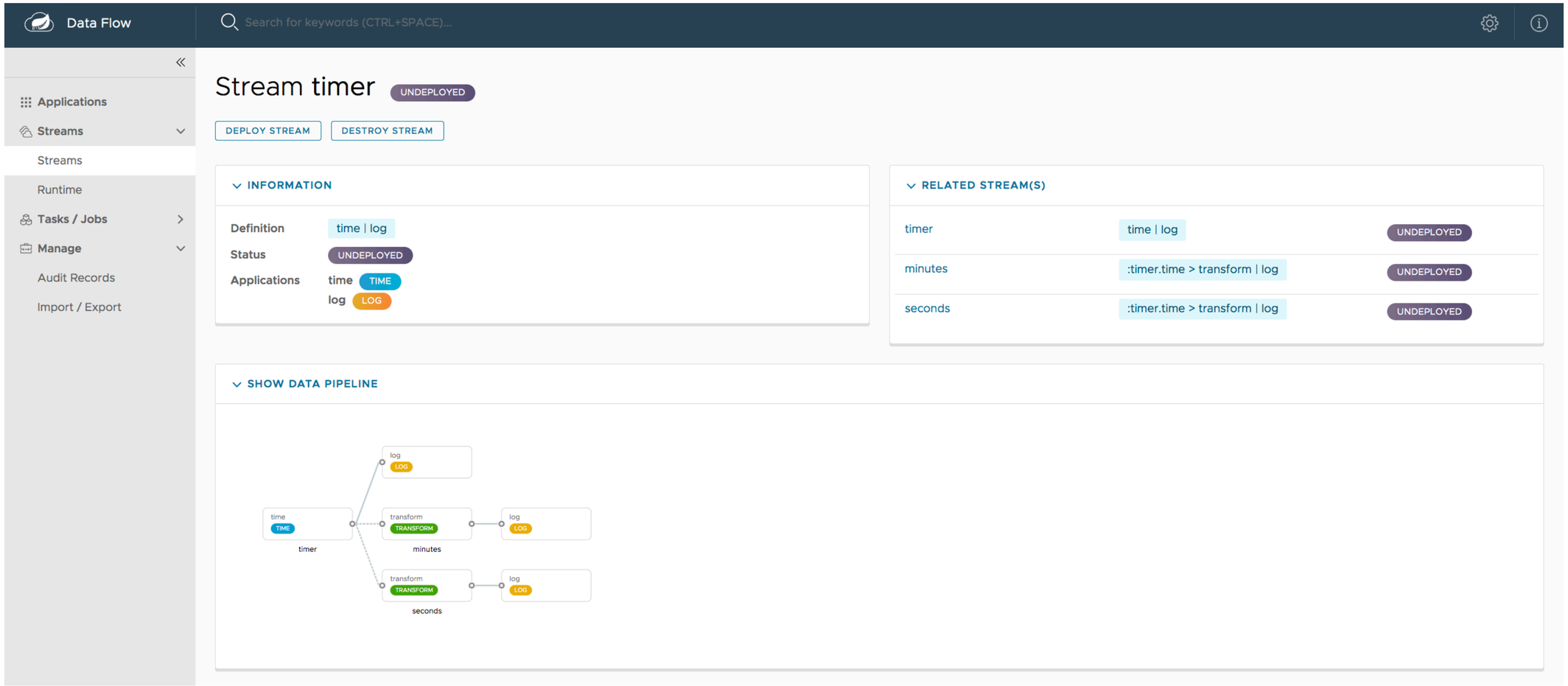
Tasks

compound-task-runner
timestamp
timestamp-batch

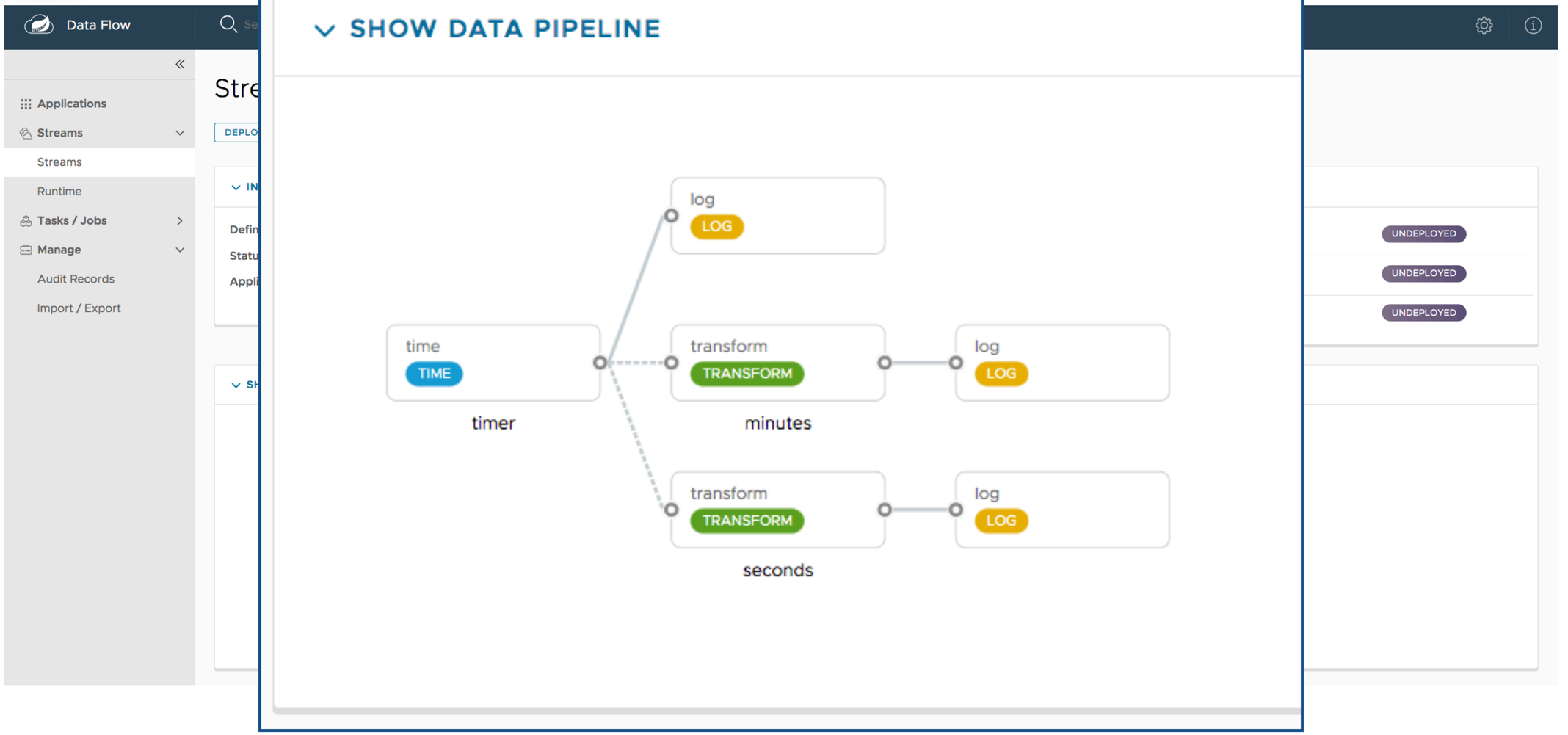
Define streams with DSL

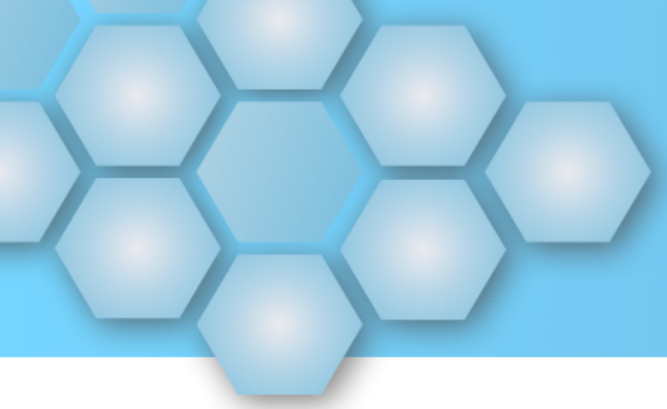
```
http --port=9000 | transform --expression 'hello-world' | log
```

Using the Dashboard



Using the Dashboard





Spring Cloud DataFlow

Processing can be easily split into distinct operations

Message-broker is the binding

Events are not necessarily sourced from a Message broker

A platform on which to build and deploy event-driven microservices

Thank you!

Check out my books:
http://www.habuma.com/sia5_nfjs.html
Discount code (50% off): tsspring