Getting Superman X-Ray Vision: Bringing Observability to your Stream Processing

Ricardo Ferreira, Developer Advocate
About Me:

- Hi, my name is **Ricardo Ferreira**
- Developer Advocate @ Confluent
- Ex-Oracle, Red Hat, IONA Tech
- Currently ~70% Dev, ~30% Ops
- https://riferrei.net
Agenda

- Part 1: Stream Processing
- Part 2: Observability Challenges
- Part 3: Best Practices for Kafka
Agenda

- Part 1: Stream Processing
- Part 2: Observability Challenges
- Part 3: Best Practices for Kafka
Let’s Tweet the Song!

1. Access your Twitter account.
2. Use \#dotcnz in your tweet for correct tracking of your guess.
3. The song name must be within brackets as shown below.
Apache Kafka and Stream Processing
O'Reilly Books
Application details:

- AWS and Terraform
- Confluent Cloud Cluster
- Spring Boot Application
- Apache Kafka Connect
- Kafka Streams / KSQL
- Redis Cache
- AWS Lambda
- Amazon Alexa
Application details:

- AWS and Terraform
- Confluent Cloud Cluster
- Spring Boot Application
- Apache Kafka Connect
- Kafka Streams / KSQL
- Redis Cache
- AWS Lambda
- Amazon Alexa

You can find the source-code of this application here:
Agenda

- Part 1: Stream Processing
- Part 2: Observability Challenges
- Part 3: Best Practices for Kafka
Why Observability is Important for Modern Software Architectures?

Because it allows us to focus on solving problems instead of finding where the problem is...
OpenTracing has the elements to implement tracing transparently

- Multiple programming languages
- Support for multiple tracers
- Large community supporting it
Does it Works with Software Built using Microservices?

Of course!
So... What are the Problems?

- Collaboration between teams
- Microservices and Serverless
- Binary Protocols with no Headers
- Pub/Sub based Architectures
- Source-Code Unavailability
So... What are the Problems?

- Collaboration between teams
- Microservices and Serverless
- Binary Protocols with no Headers
- Pub/Sub based Architectures
- Source-Code Unavailability
Agenda

- Part 1: Stream Processing
- Part 2: Observability Challenges
- Part 3: Best Practices for Kafka
Headers Support on Kafka 0.11.X

- Span context can be set there
- All headers are persistent
- Headers are agnostic of schema
OpenTracing Contributions

- Apache Kafka Instrumentation
- Provides support for:
  - Decorators
  - Interceptors
  - Spring Framework
  - Kafka Streams

You can find the source-code of this application here:
Tracing Support for Closed JVMs

- Jaeger Tracing support for REST Proxy, Connect and KSQL
- Based on the interceptors model
- Instantiate its own tracers based on JSON configuration file set as an environment variable

You can find the source-code of this application here: