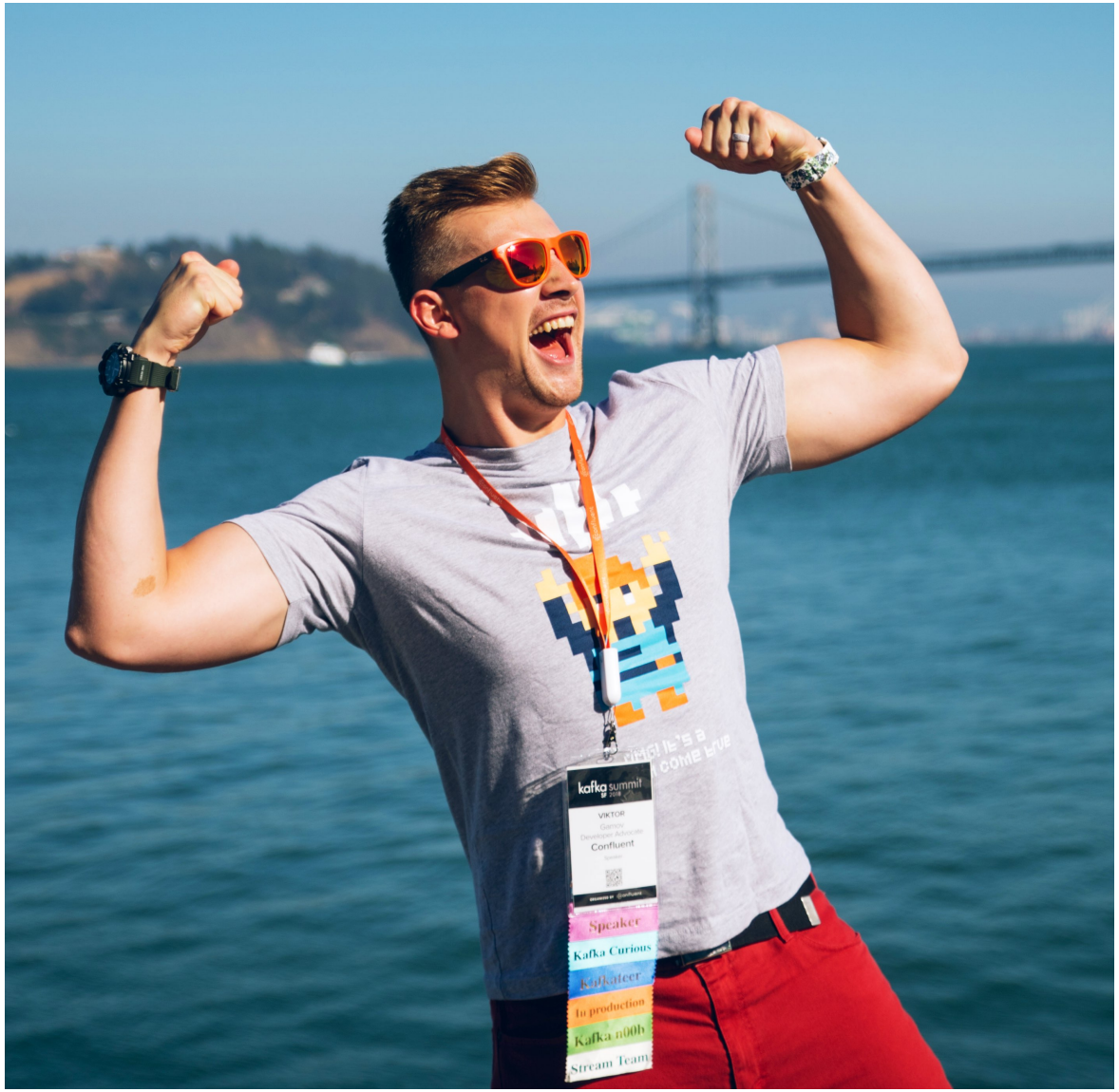
BRINGING OBSERVABILITY TO YOUR STREAM PROCESSING

SEPTEMBER, 17TH, 2019

@GAMUSSA | #CODEONE | @CONFLUENTINC



@GAMUSSA | #CODEONE | @CONFLUENTINC

I BUILD HIGHLY SCALABLE

Hello World

APPS

I BUILD HIGHLY SCALABLE

Hello World

APPS

@kennybastani

confluent

RAFFLE, YEAH 

confluent

RAFFLE, YEAH 🚀

- ✓ Follow @gamussa @confluentinc
- ✓ 📸🖼️🏋️‍♂️
- ✓ Tag @gamussa
- ✓ With #CodeOne

Photo by Khai Sze Ong on Unsplash


Photo by Javier Ridao on Unsplash

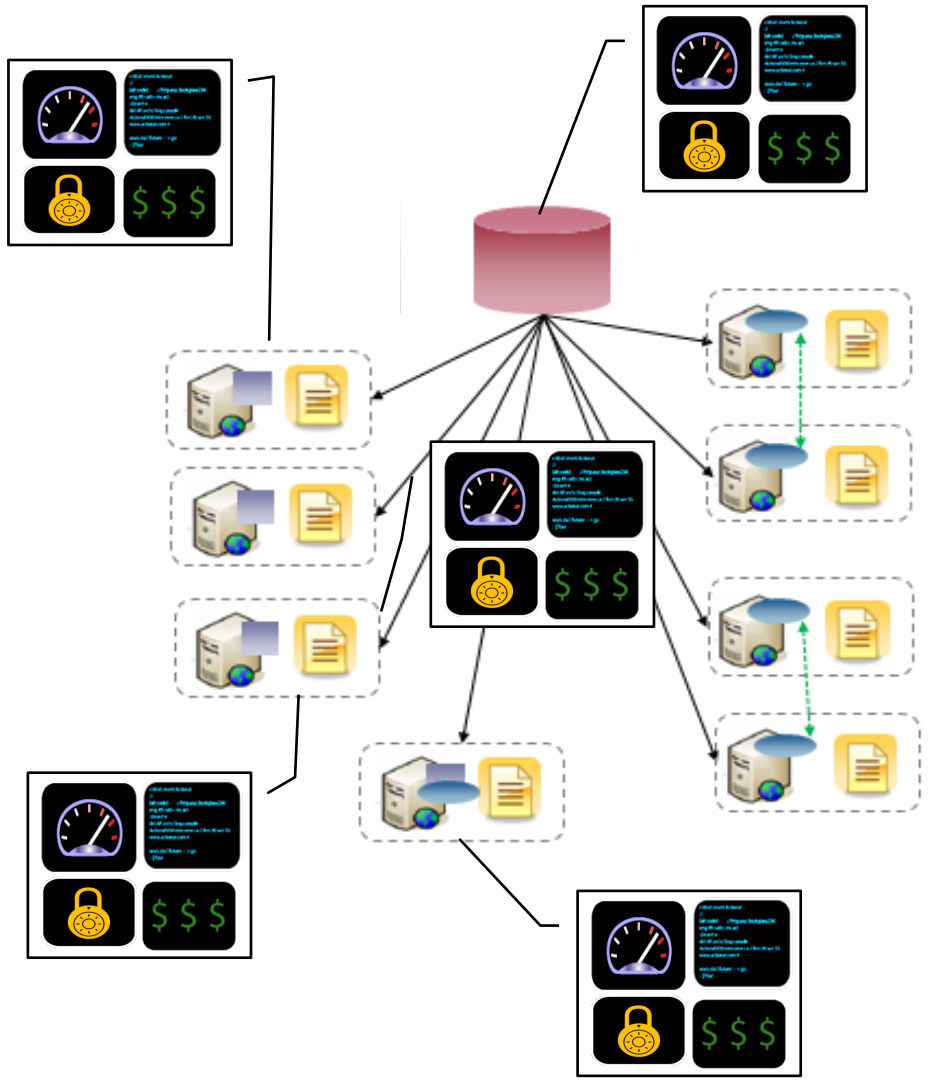

Photo by Rohit Tandon on Unsplash


Photo by Theodore Moore on Unsplash

Photo by Cristian Grecu on Unsplash

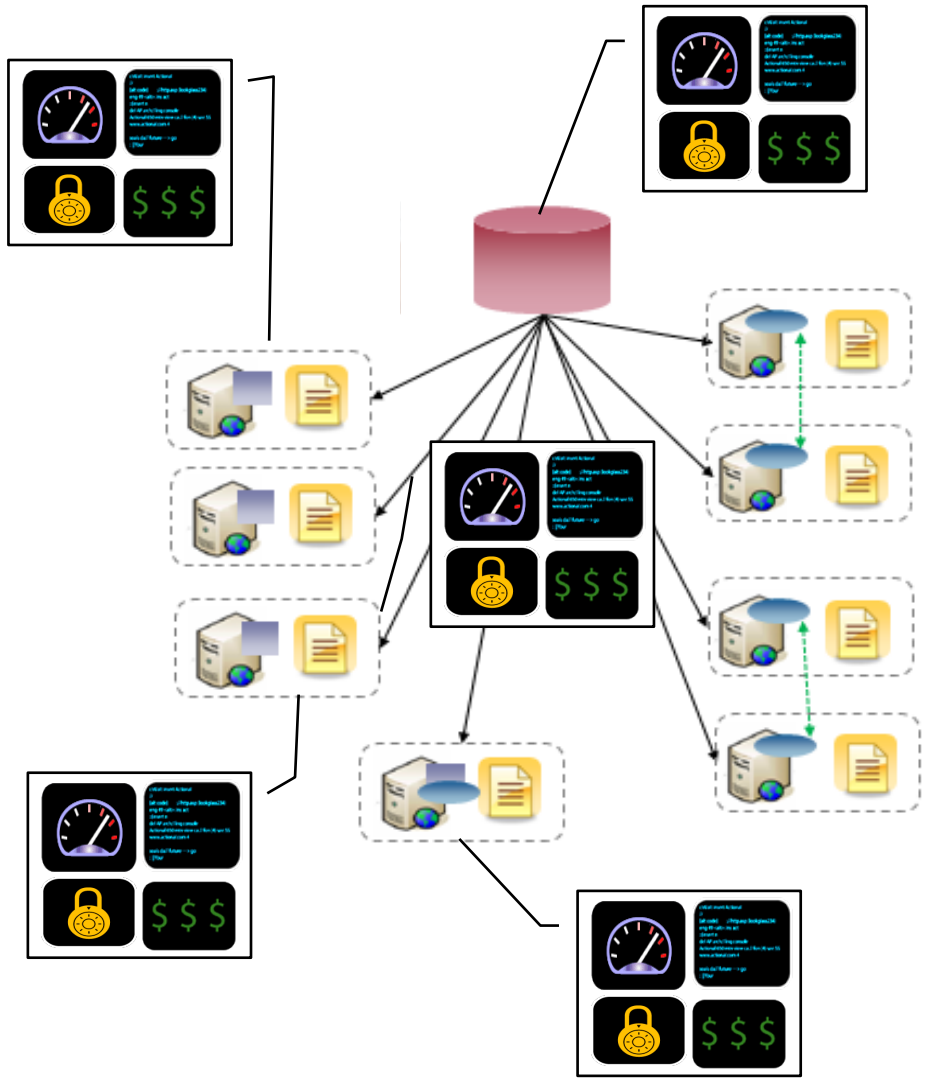

PETE
SILVA

confluent



HOW DID WE DO MONITORING?

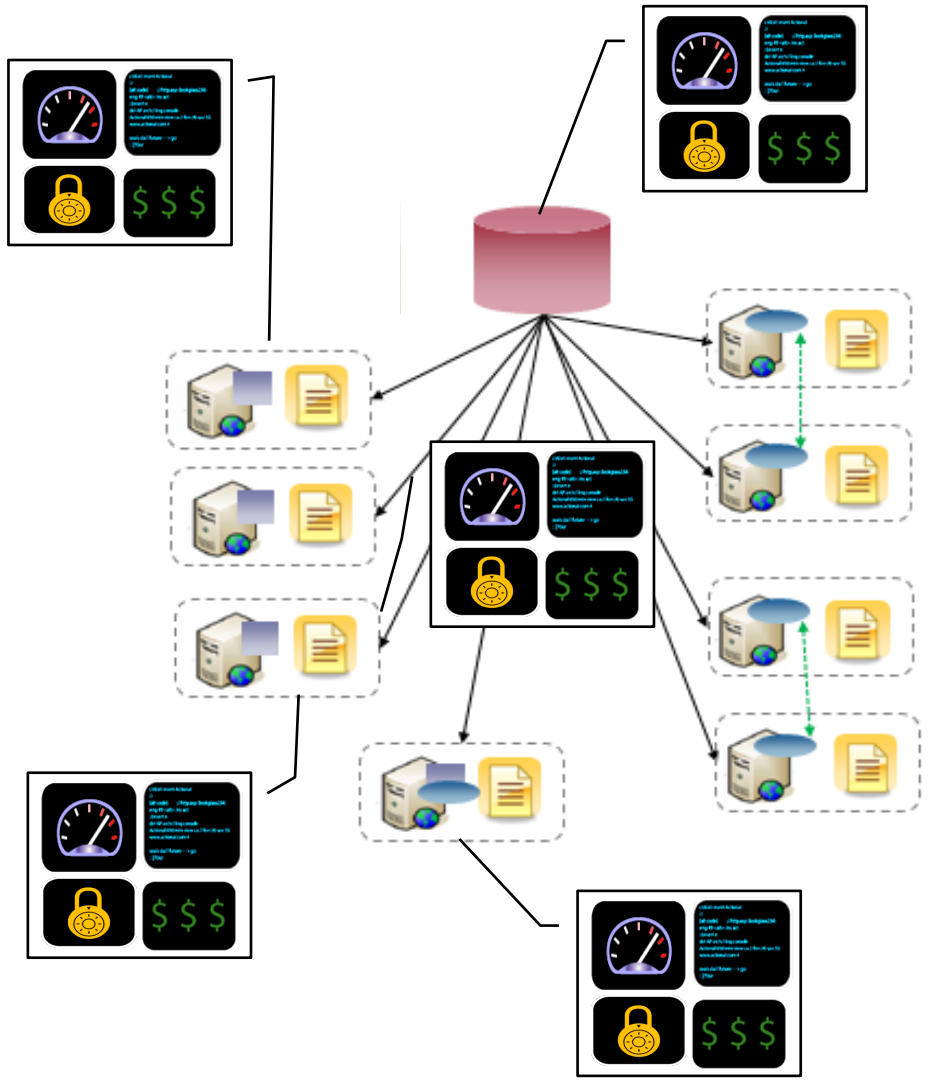
confluent



How did we do monitoring?

- Each system would have their **very own** monitoring system.

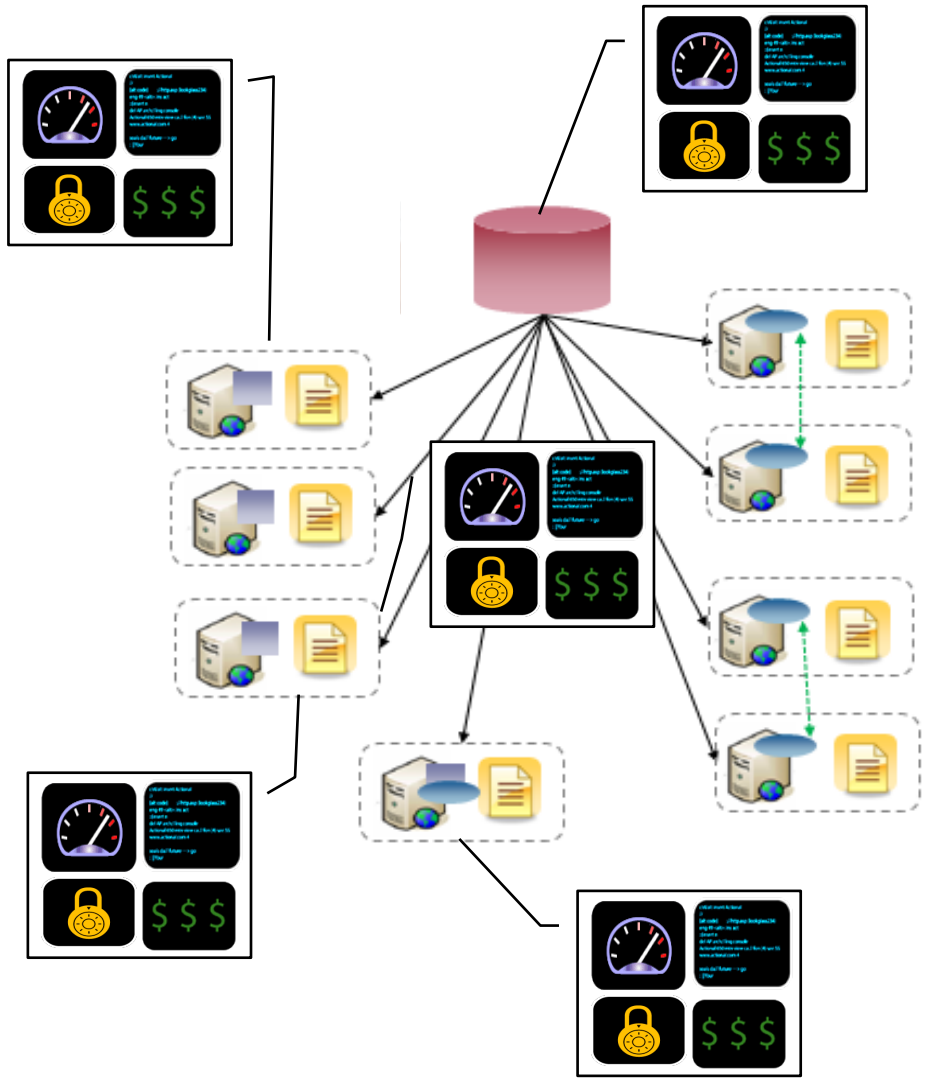
confluent



How did we do monitoring?

- Each system would have their **very own** monitoring system.
- Developers would not worry about monitoring. This was supposed to be **“IT stuff”**.

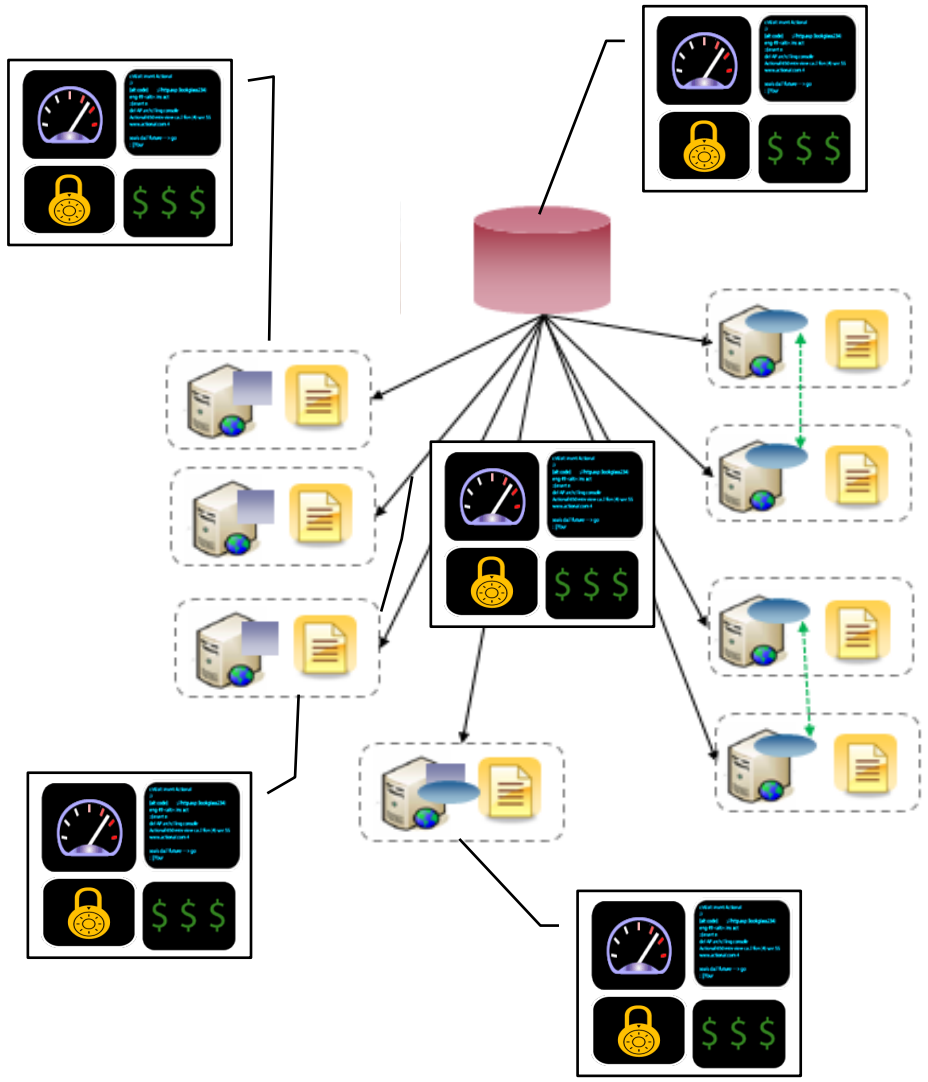
confluent



How did we do monitoring?

- Each system would have their **very own** monitoring system.
- Developers would not worry about monitoring. This was supposed to be **“IT stuff”**.
- Application was deemed OK if all systems were **showing green**.

confluent

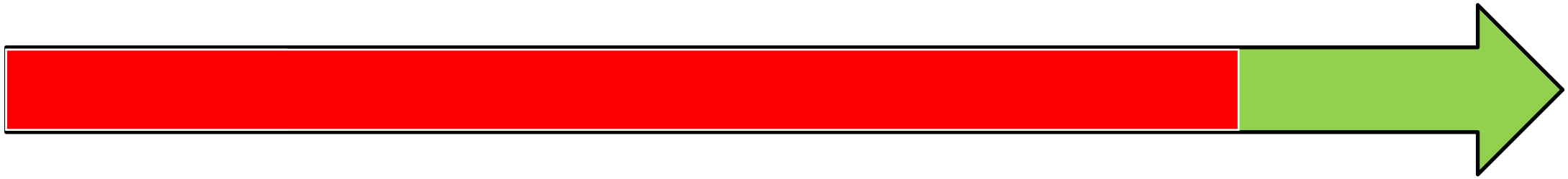


HOW DID WE DO MONITORING?

- Each system would have their **very own** monitoring system.
- Developers would not worry about monitoring. This was supposed to be **“IT stuff”**.
- Application was deemed OK if all systems were **showing green**.
- Different monitoring approaches were used throughout the entire IT stack portfolio, **requiring many teams** to be involved.

WTF

WTF 



WTF 🤬

TROUBLESHOOTING. DAYS, WEEKS, MONTHS!

WTF 🤬

SYSTEM IS WORKING!

TROUBLESHOOTING. DAYS, WEEKS, MONTHS!


Nope. Don't like that.


Nope. Don't like that.

INTRO TO OBSERVABILITY

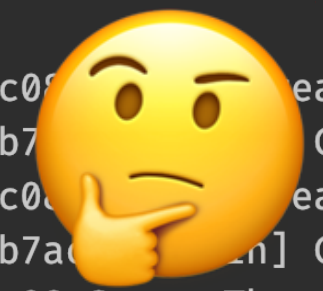
Cindy Sridharan
@copyconstruct

OH - "Observability - because devs don't like to do "monitoring" we need to package it in new nomenclature to make it palatable and trendy."

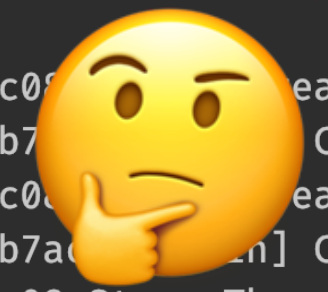
6:41 PM · Jul 28, 2017 from San Francisco, CA · Twitter for iPhone

THE PILLARS OF OBSERVABILITY

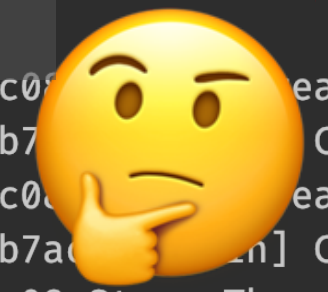
```
on to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.
[2019-05-14 11:05:26,947] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:27,726] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:27,970] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:28,089] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:28,779] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:28,852] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:29,595] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:29,685] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:29,955] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:30,810] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:31,072] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:31,098] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:31,797] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:32,199] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:32,602] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:32,727] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:33,016] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:33,424] INFO [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Metad
org.apache.kafka.common.errors.TimeoutException: Timed out waiting for a node assignment.
[2019-05-14 11:05:33,764] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac0
[2019-05-14 11:05:33,928] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7
[2019-05-14 11:05:33,945] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac0
[2019-05-14 11:05:34,933] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7a
[2019-05-14 11:05:34,957] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:34,964] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:35,886] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
```

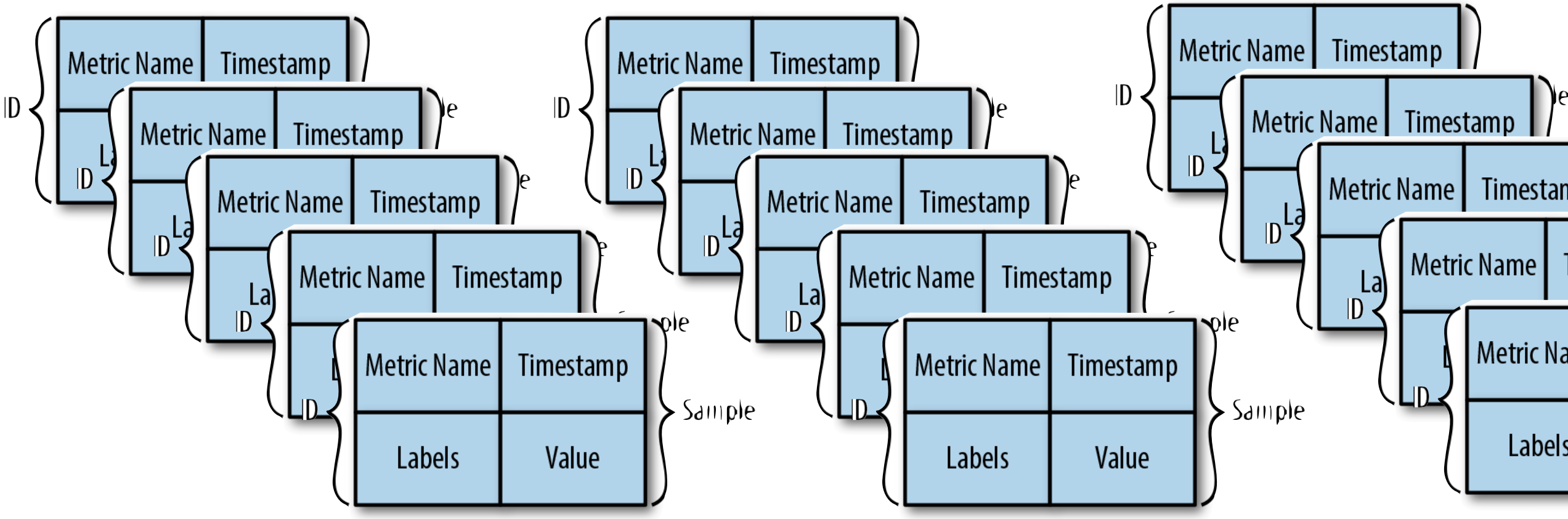


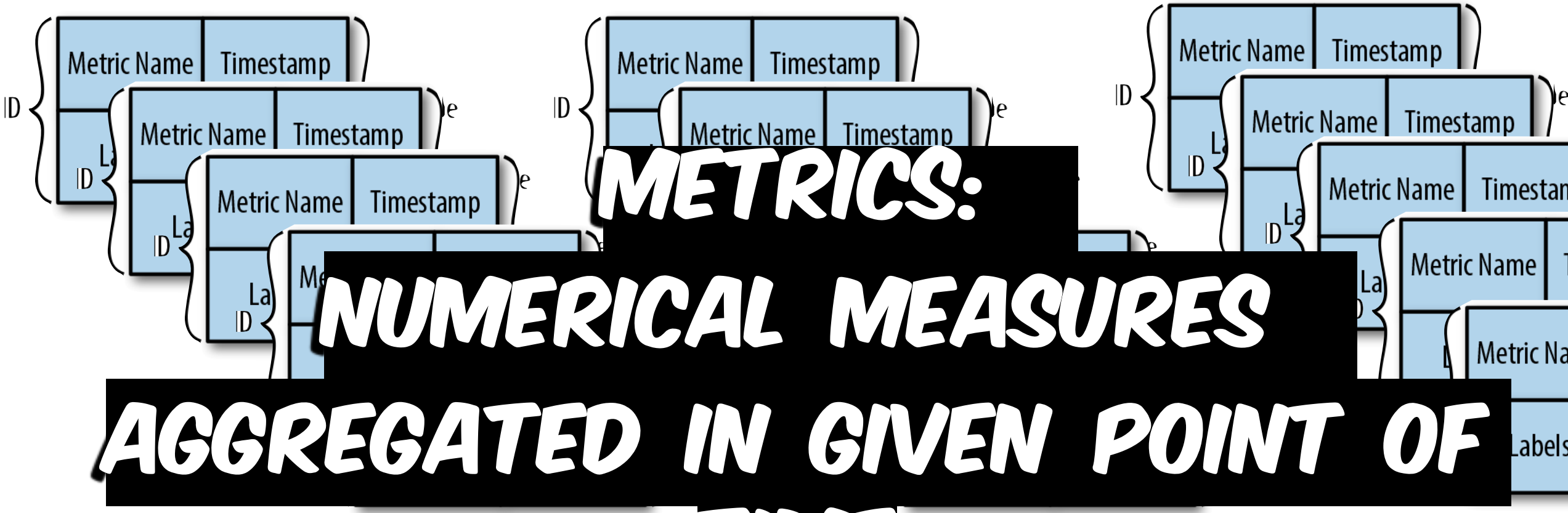
```
on to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.
[2019-05-14 11:05:26,947] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:27,726] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:27,970] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:28,089] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:28,779] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:28,852] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:29,595] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:29,685] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:29,955] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:30,810] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:31,072] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:31,098] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:31,797] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:32,199] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:32,602] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:32,727] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:33,016] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:33,424] INFO [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Metad
org.apache.kafka.common.errors.TimeoutException: Timed out waiting for a node assignment.
[2019-05-14 11:05:33,764] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:33,928] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:33,945] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:34,933] WARN [AdminClient clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-admin] Conne
[2019-05-14 11:05:34,957] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:34,964] WARN [Consumer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
[2019-05-14 11:05:35,886] WARN [Producer clientId=kafka-films-7d330361-673d-4ee5-ac93-291ab2b7ac08-StreamThread-1-
```



A RAW SEQUENCE OF EVENTS FROM A SINGLE INSTANCE OF SERVICE



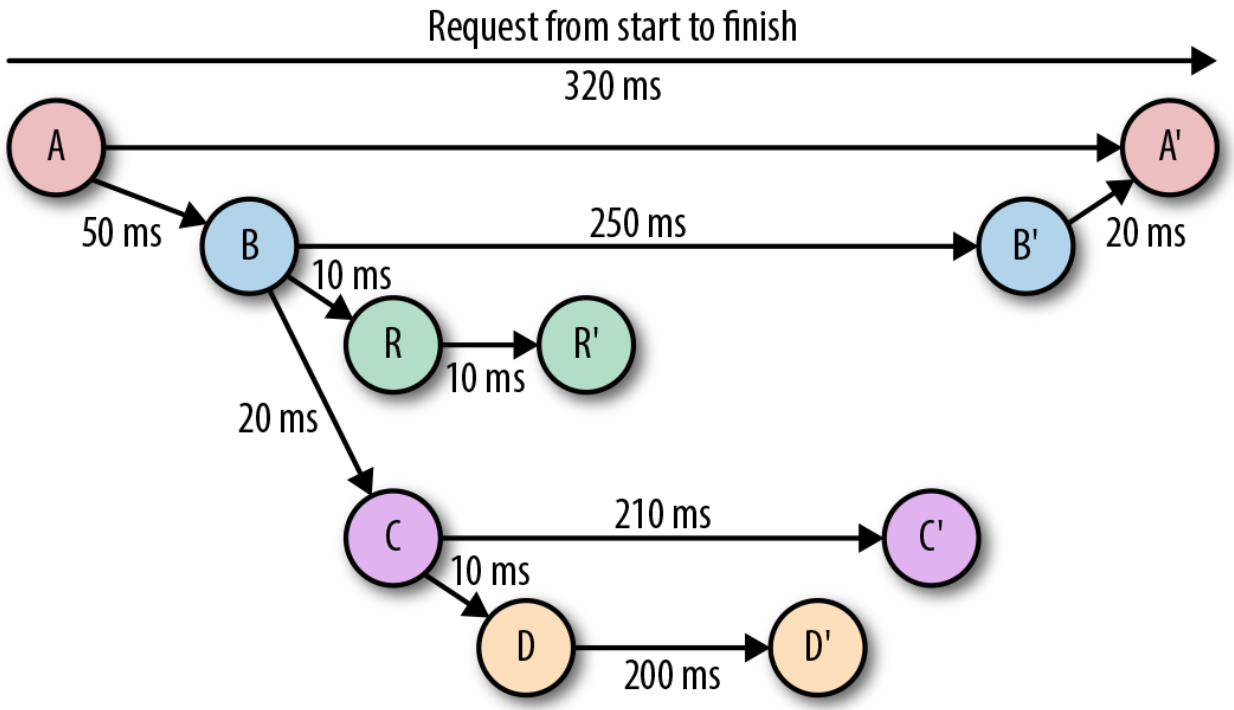

Metric Name
Timestamp
Labels
Value
Sample



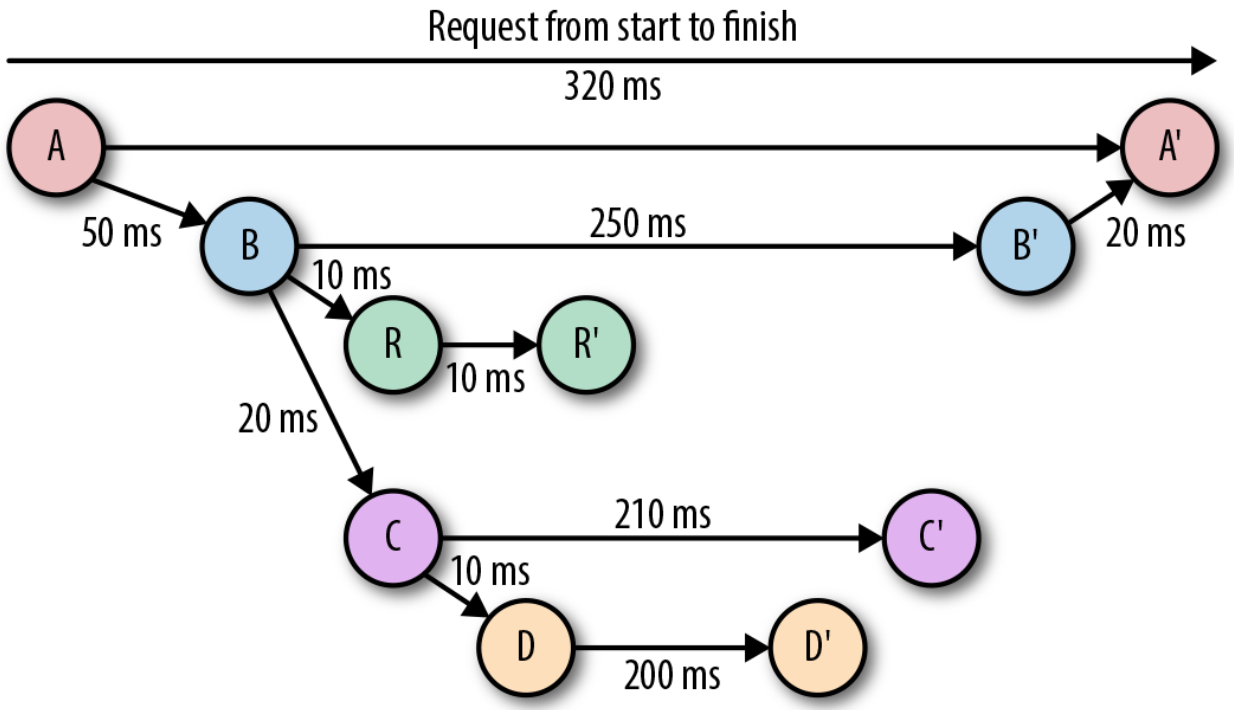
METRICS: NUMERICAL MEASURES AGGREGATED IN GIVEN POINT OF TIME

confluent

PILLARS OF OBSERVABILITY



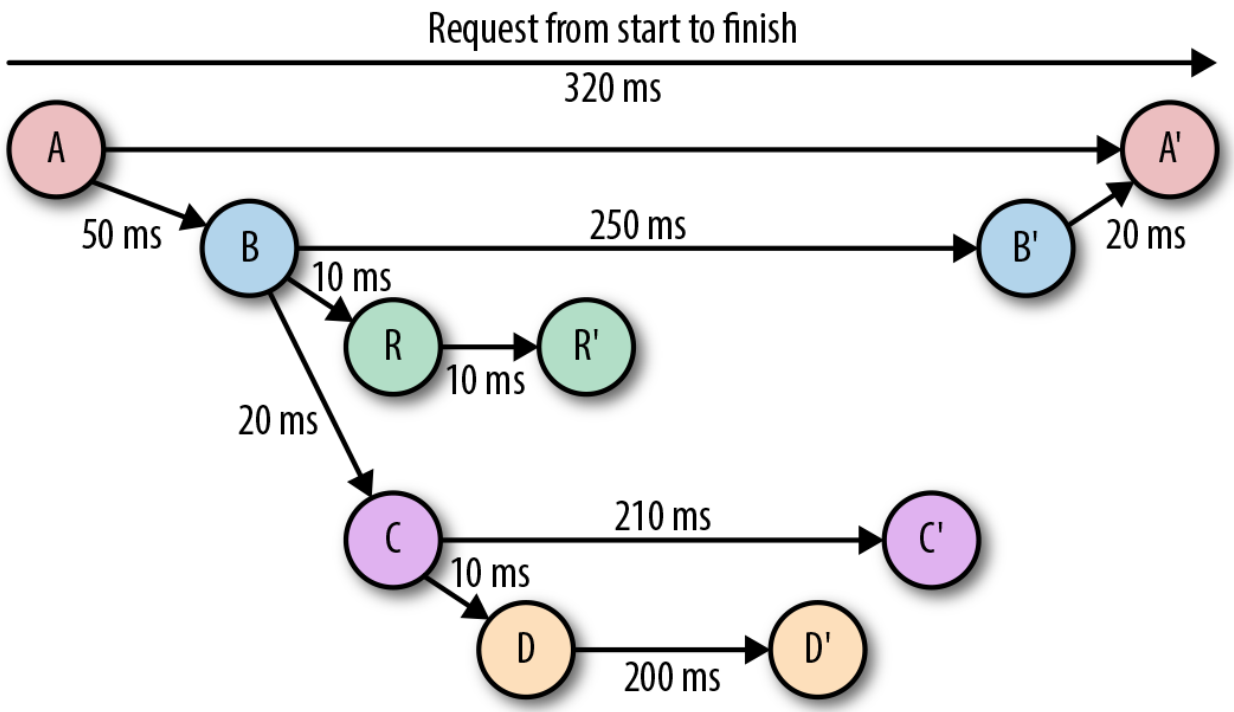
confluent



PILLARS OF OBSERVABILITY

- Distributed Tracing

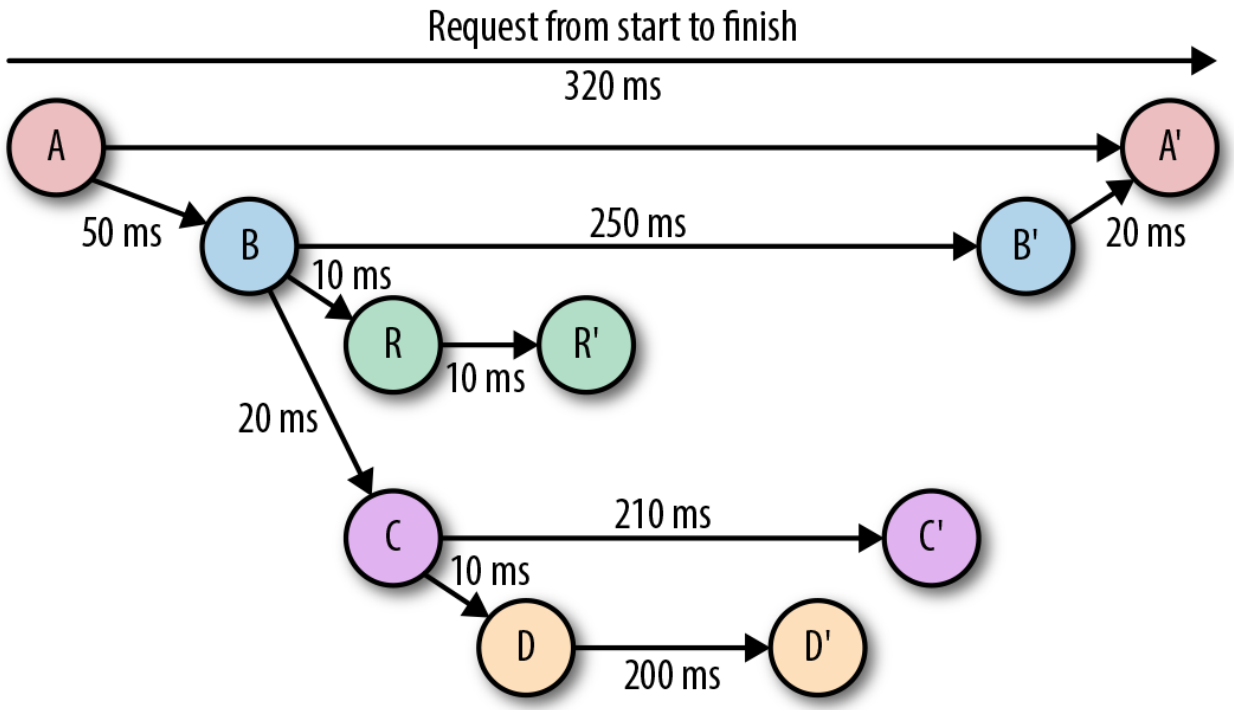
confluent



PILLARS OF OBSERVABILITY

- Distributed Tracing
  - detailed execution of the causality-related activities performed by a given transaction. It answers:

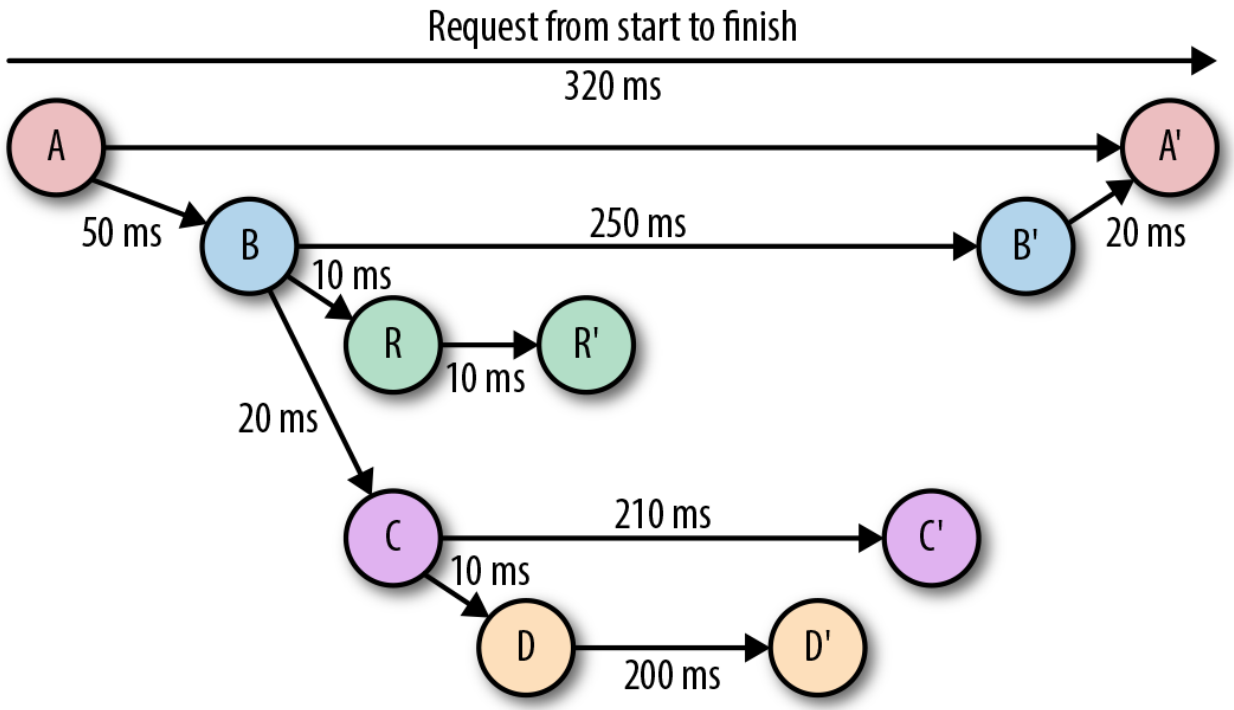
confluent



PILLARS OF OBSERVABILITY

- Distributed Tracing
  - detailed execution of the causality-related activities performed by a given transaction. It answers:
    - Which services were involved?

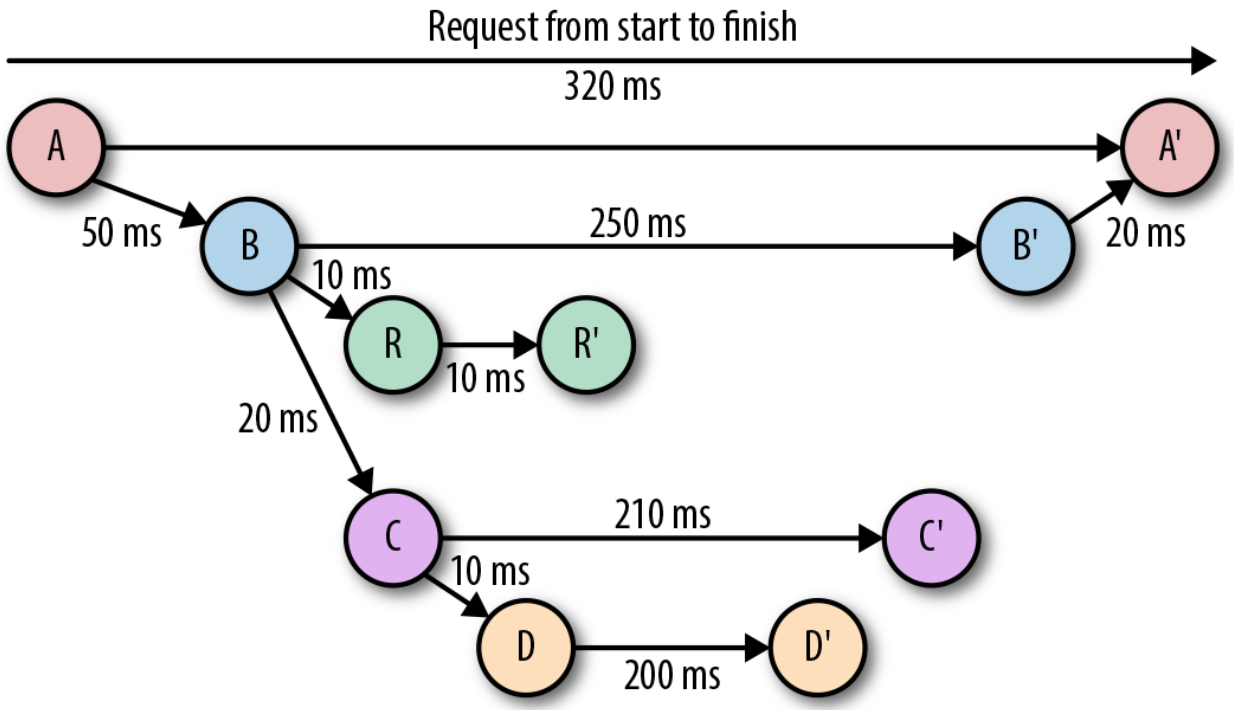
confluent



PILLARS OF OBSERVABILITY

- Distributed Tracing
  - detailed execution of the causality-related activities performed by a given transaction. It answers:
    - Which services were involved?
    - If it was slow, who caused that?

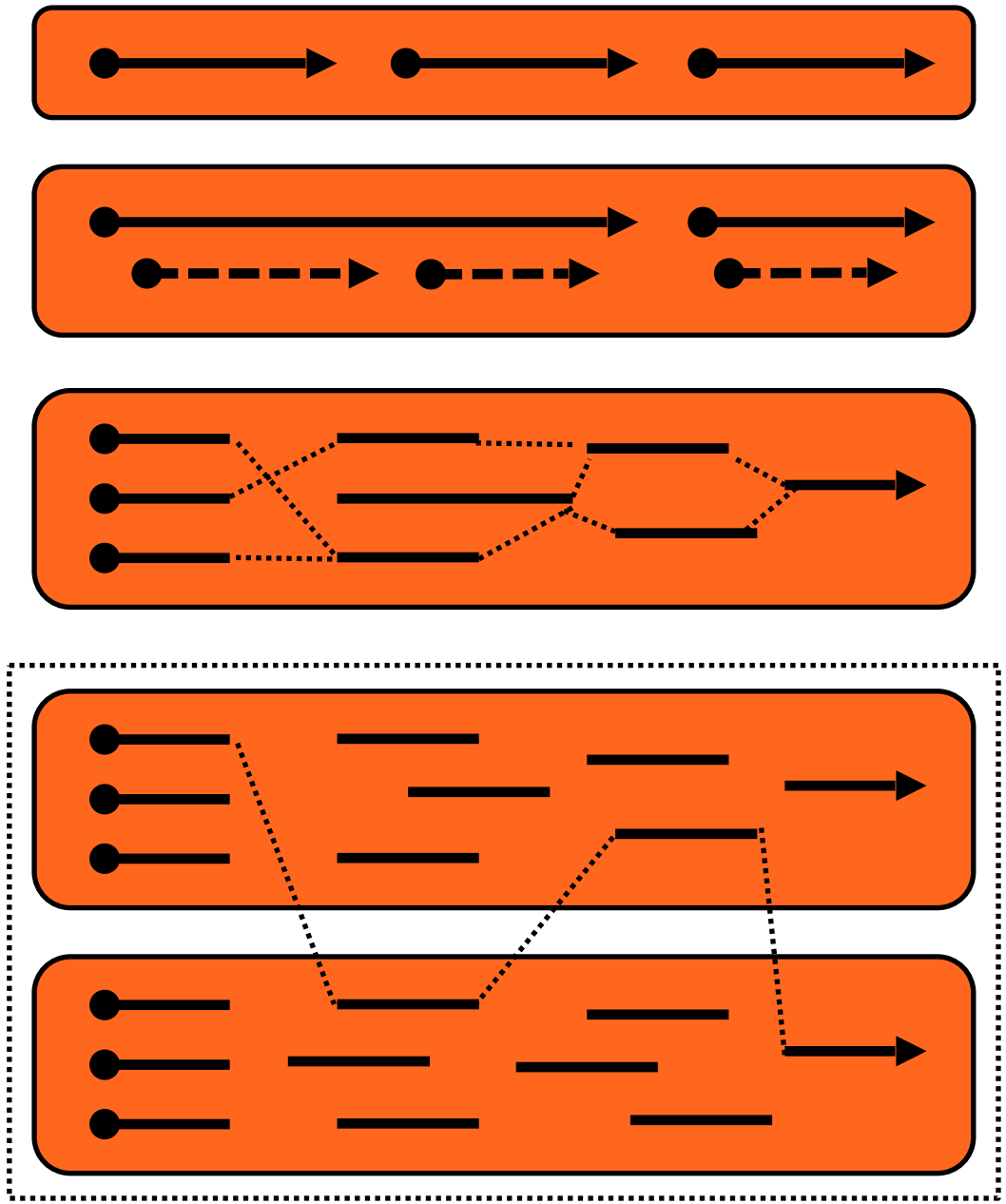
confluent



PILLARS OF OBSERVABILITY

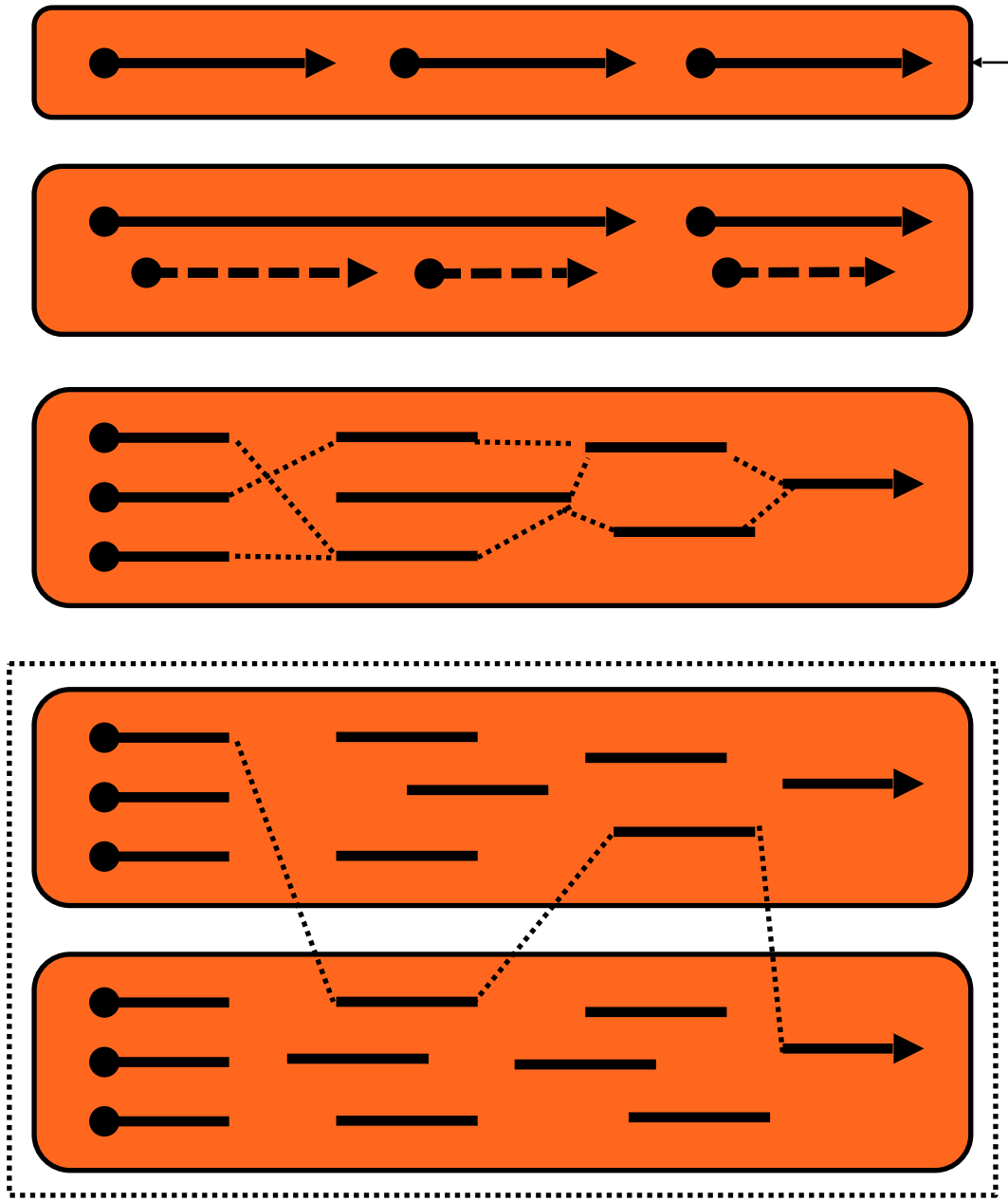
- Distributed Tracing
  - detailed execution of the causality-related activities performed by a given transaction. It answers:
    - Which services were involved?
    - If it was slow, who caused that?
    - If failed, who actually failed?

confluent



EVOLUTION OF CONCURRENCY

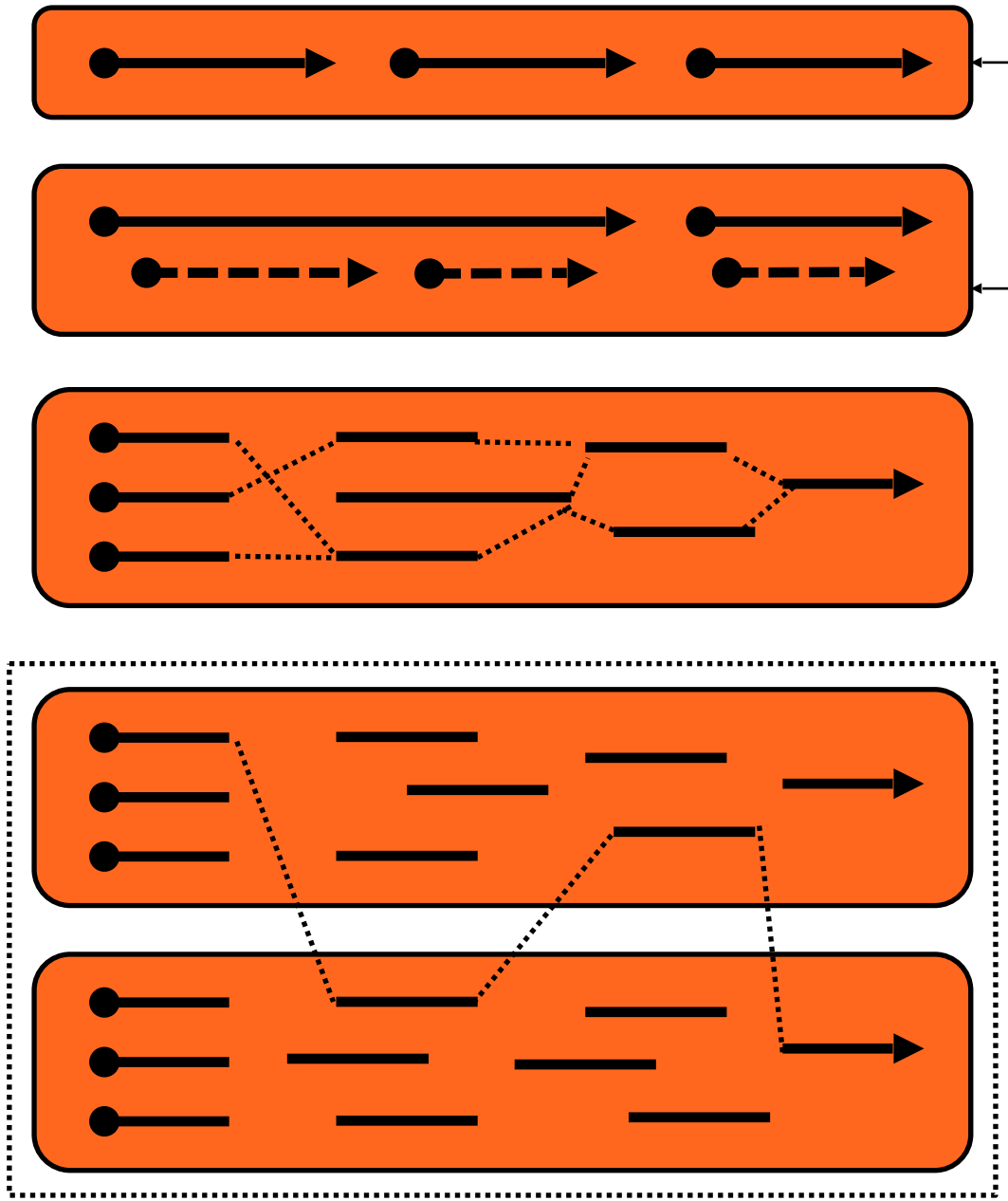
confluent



EVOLUTION OF CONCURRENCY

- No Concurrency at all
  - Ex: Apache HTTP Server

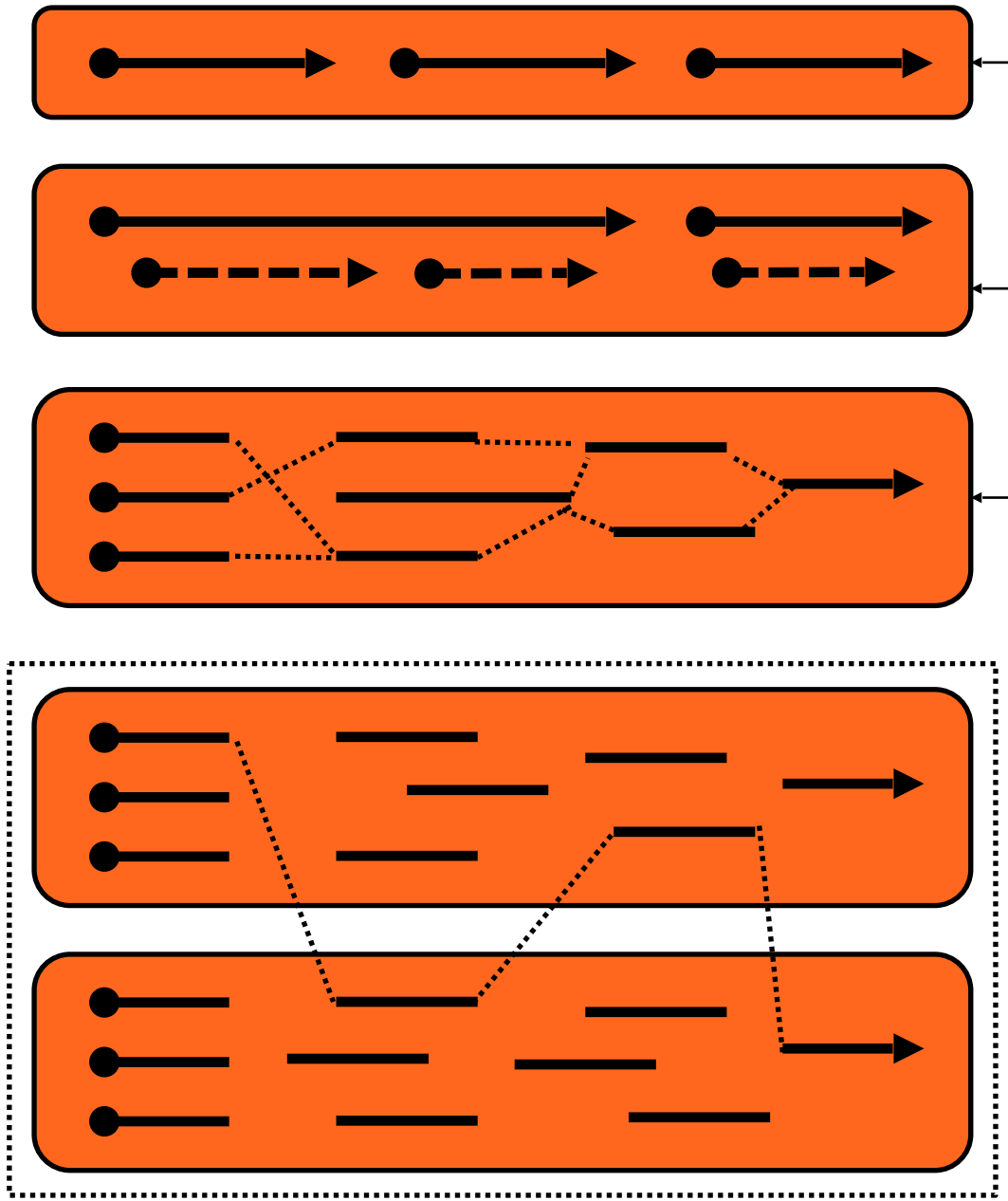
confluent



EVOLUTION OF CONCURRENCY

- No Concurrency at all
  - Ex: Apache HTTP Server
- Basic Concurrency
  - Ex: Multi-threaded Applications

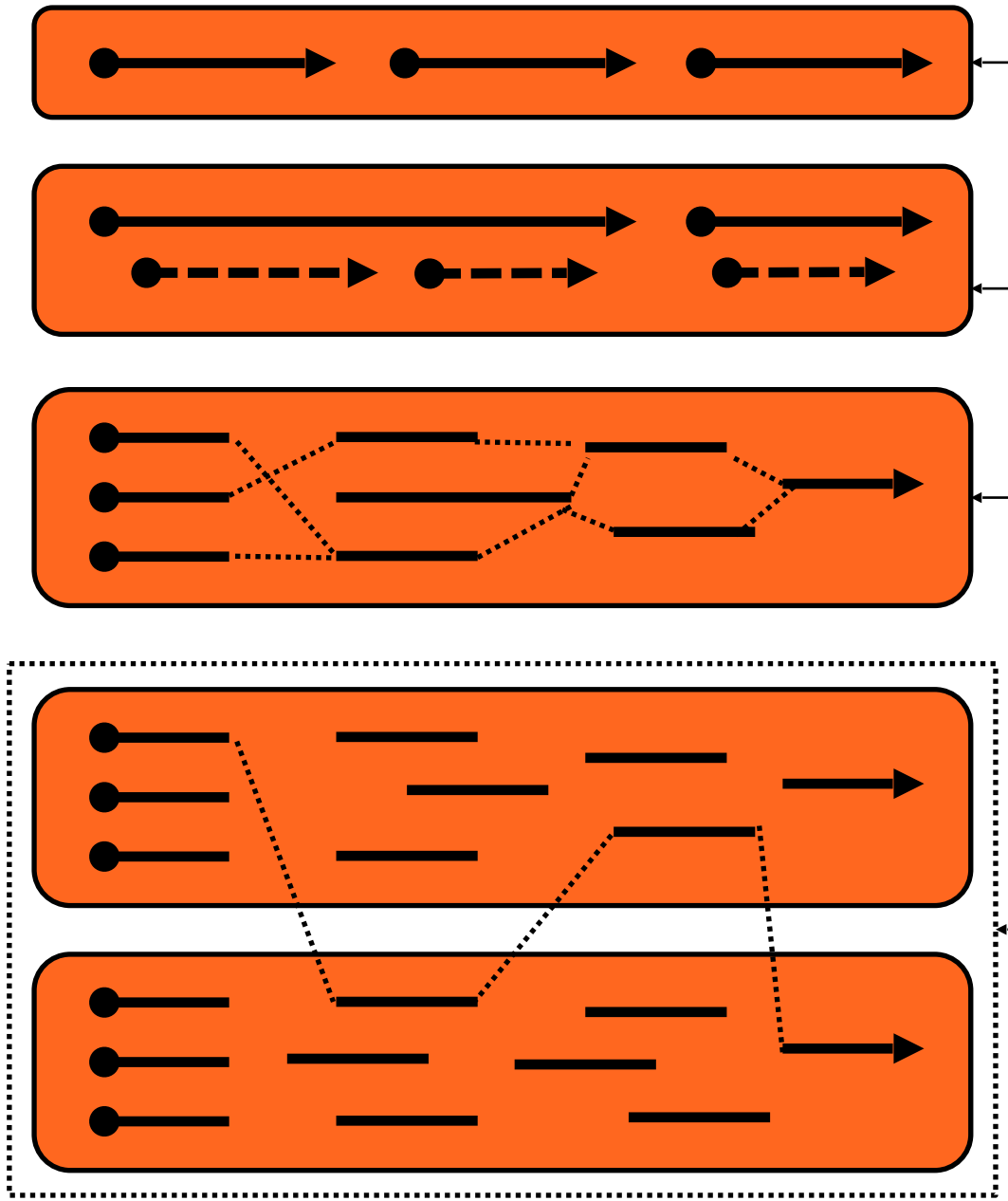
confluent



EVOLUTION OF CONCURRENCY

- No Concurrency at all
  - Ex: Apache HTTP Server
- Basic Concurrency
  - Ex: Multi-threaded Applications
- Async Concurrency
  - Ex: Actor-based Programming

confluent



EVOLUTION OF CONCURRENCY

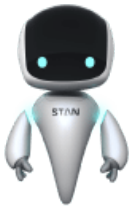
- No Concurrency at all
  - Ex: Apache HTTP Server
- Basic Concurrency
  - Ex: Multi-threaded Applications
- Async Concurrency
  - Ex: Actor-based Programming
- Distributed Concurrency
  - Ex: µServices Architecture Style

confluent

ZIPKIN

OPENTRACING

INSTANA



DATADOG

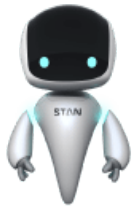
DISTRIBUTED TRACING TODAY

confluent

ZIPKIN

OPENTRACING

INSTANA

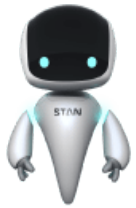


DATADOG

kubernetes

DISTRIBUTED TRACING TODAY

- There are **many distributed tracing** technologies available.



DISTRIBUTED TRACING TODAY

- There are **many distributed tracing** technologies available.
- Standards are getting created to ensure a **single programming model** for each µService.

DISTRIBUTED TRACING TODAY

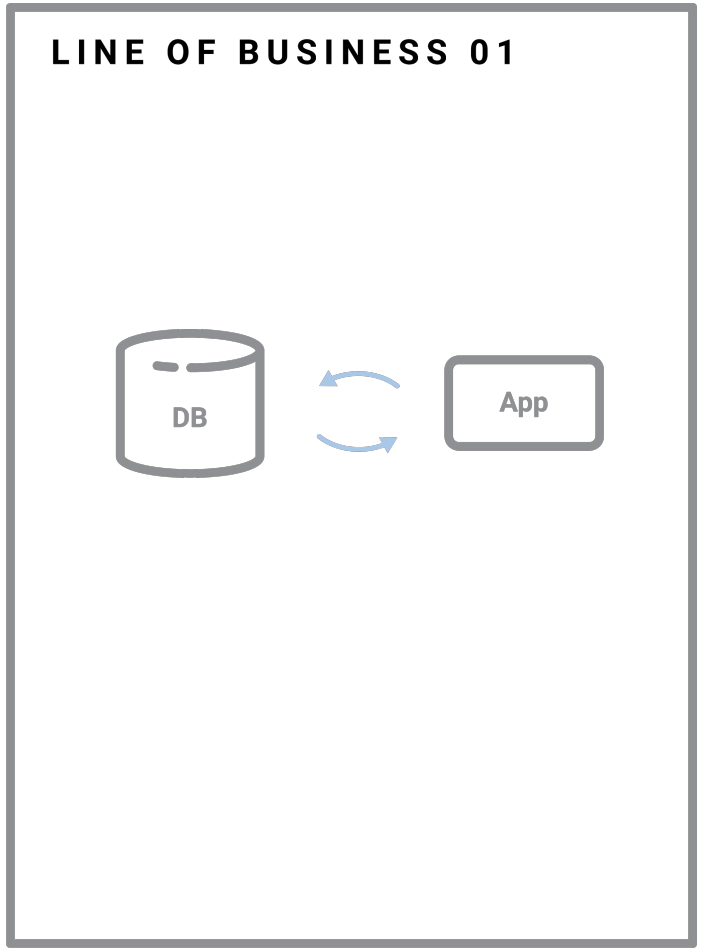
- There are **many distributed tracing** technologies available.
- Standards are getting created to ensure a **single programming model** for each µService.
- Deployment mechanisms such as **Kubernetes** are also taking care of that automatically.

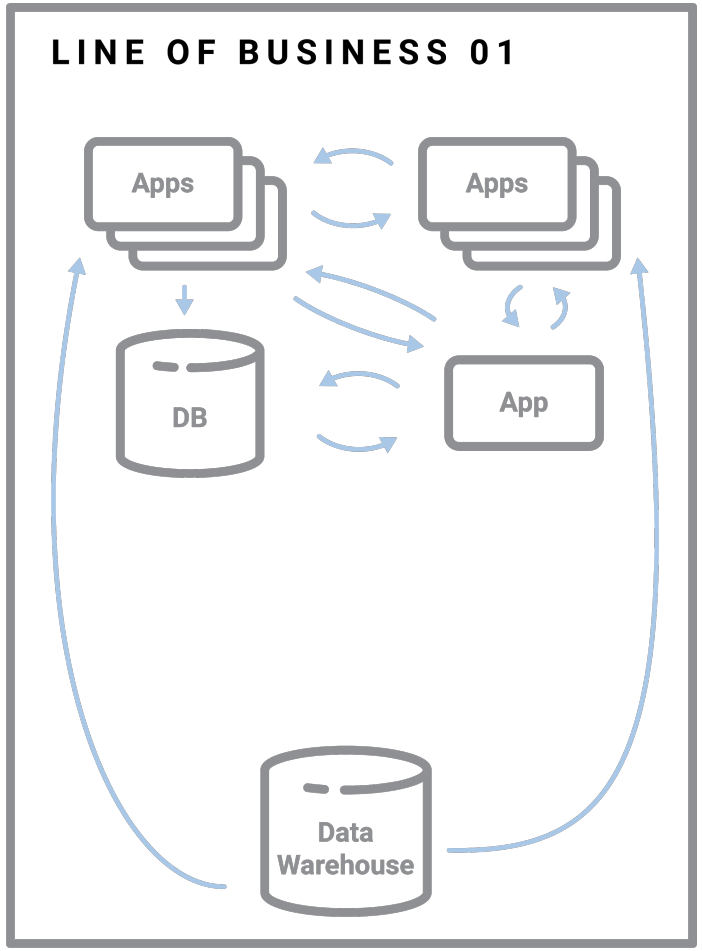
DISTRIBUTED TRACING TODAY

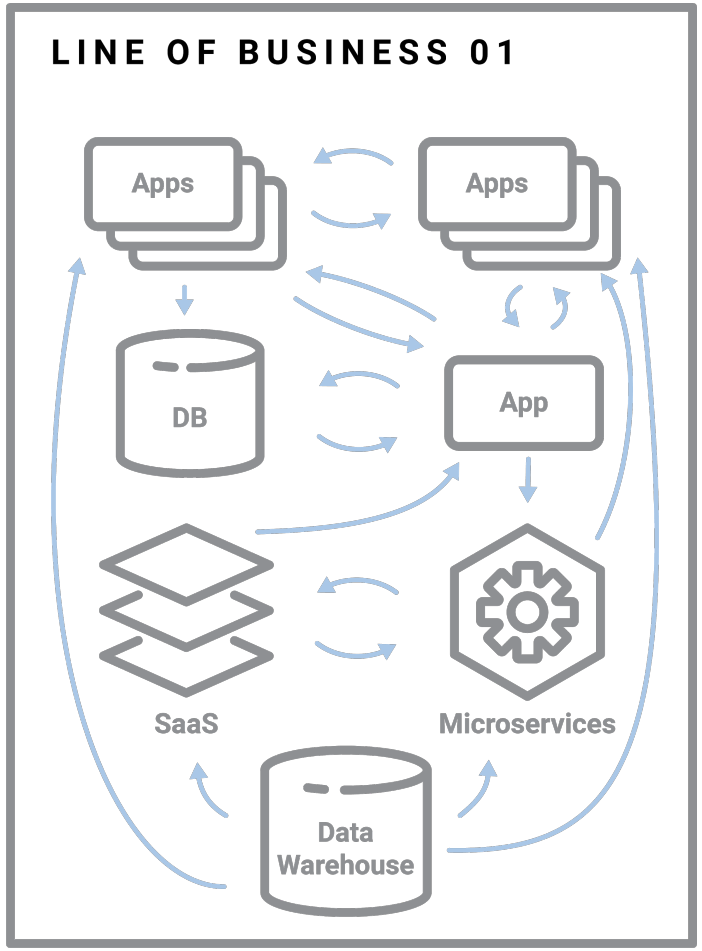
- There are **many distributed tracing** technologies available.
- Standards are getting created to ensure a **single programming model** for each µService.
- Deployment mechanisms such as **Kubernetes** are also taking care of that automatically.
- Network proxies such as **Service Meshes** are also handling this.

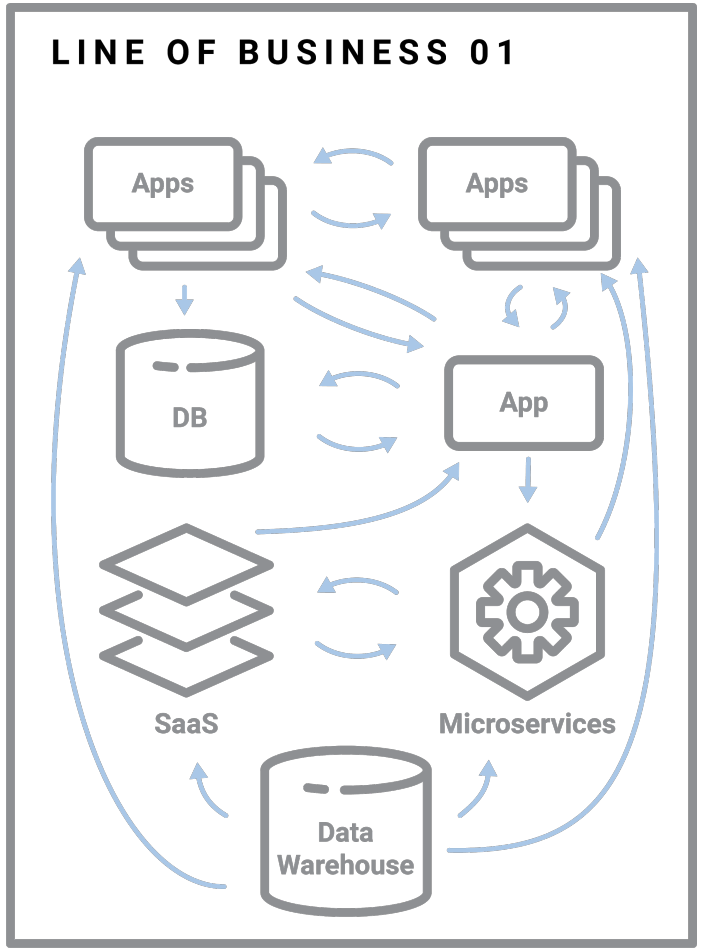
DISTRIBUTED TRACING TODAY

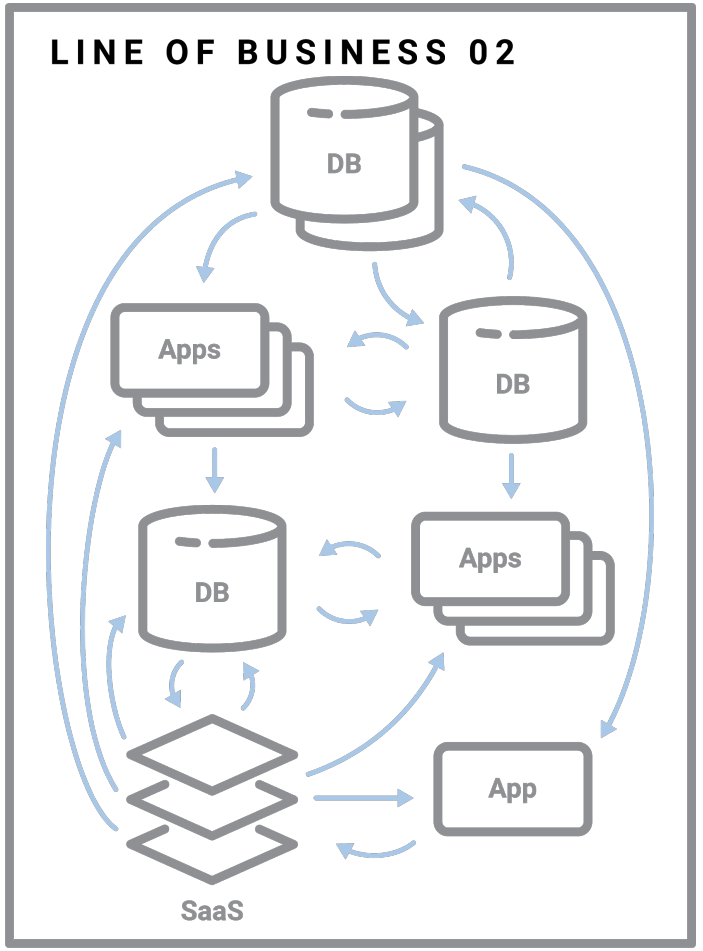
- There are **many distributed tracing** technologies available.
- Standards are getting created to ensure a **single programming model** for each µService.
- Deployment mechanisms such as **Kubernetes** are also taking care of that automatically.
- Network proxies such as **Service Meshes** are also handling this.
- OSS and proprietary options.

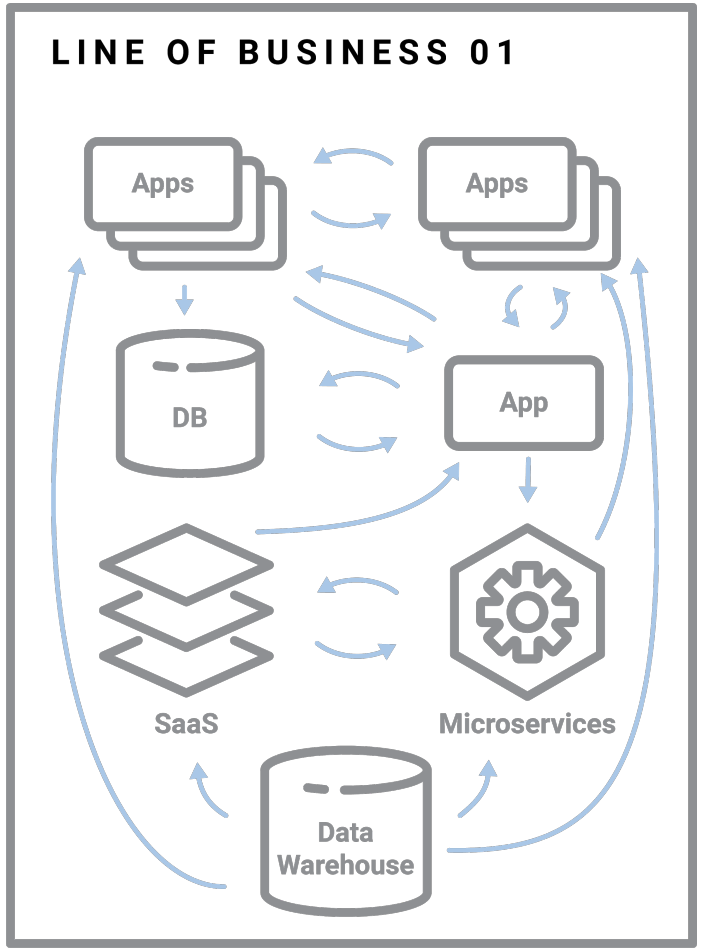

LINE OF BUSINESS 01
DB
App

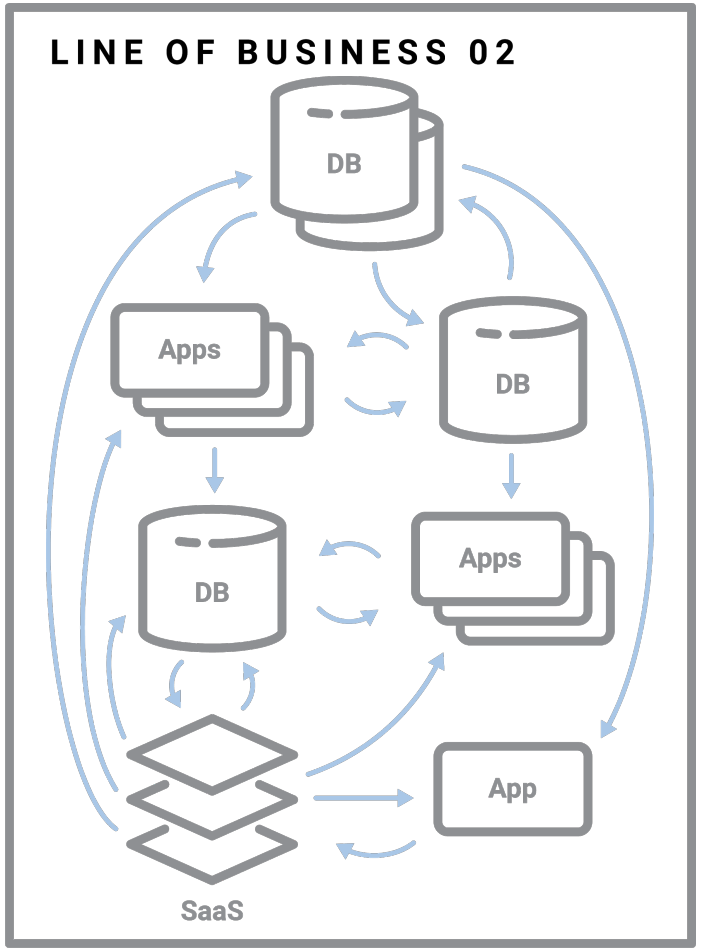

LINE OF BUSINESS 01
Apps
Apps
DB
App
Data
Warehouse

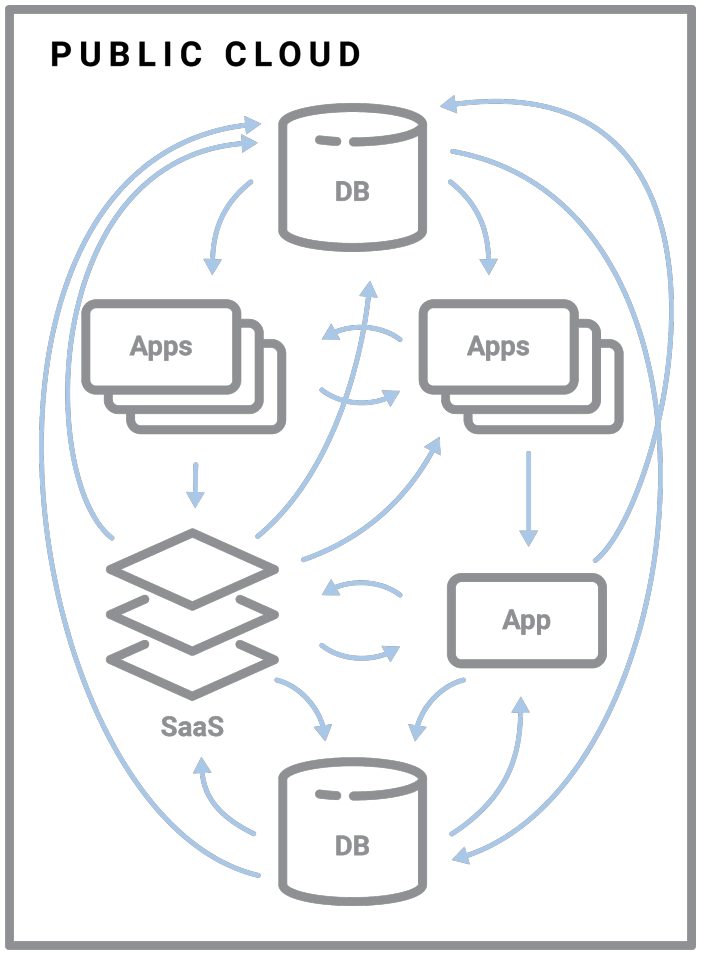

LINE OF BUSINESS 01
Apps
Apps
DB
App
SaaS
Microservices
Data Warehouse

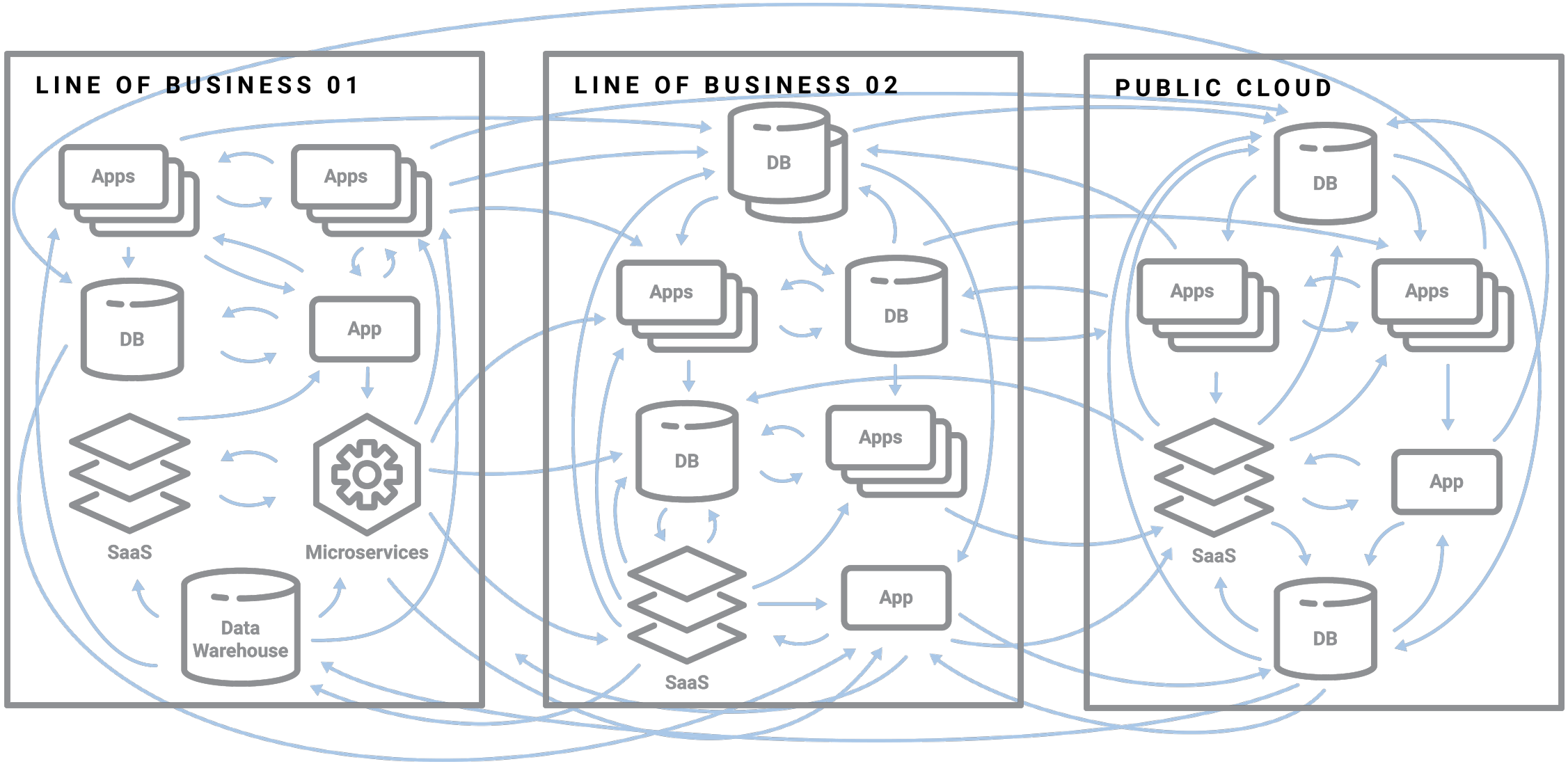

LINE OF BUSINESS 01
Apps
Apps
DB
App
SaaS
Microservices
Data Warehouse


LINE OF BUSINESS 02
DB
Apps
DB
DB
Apps
App
SaaS

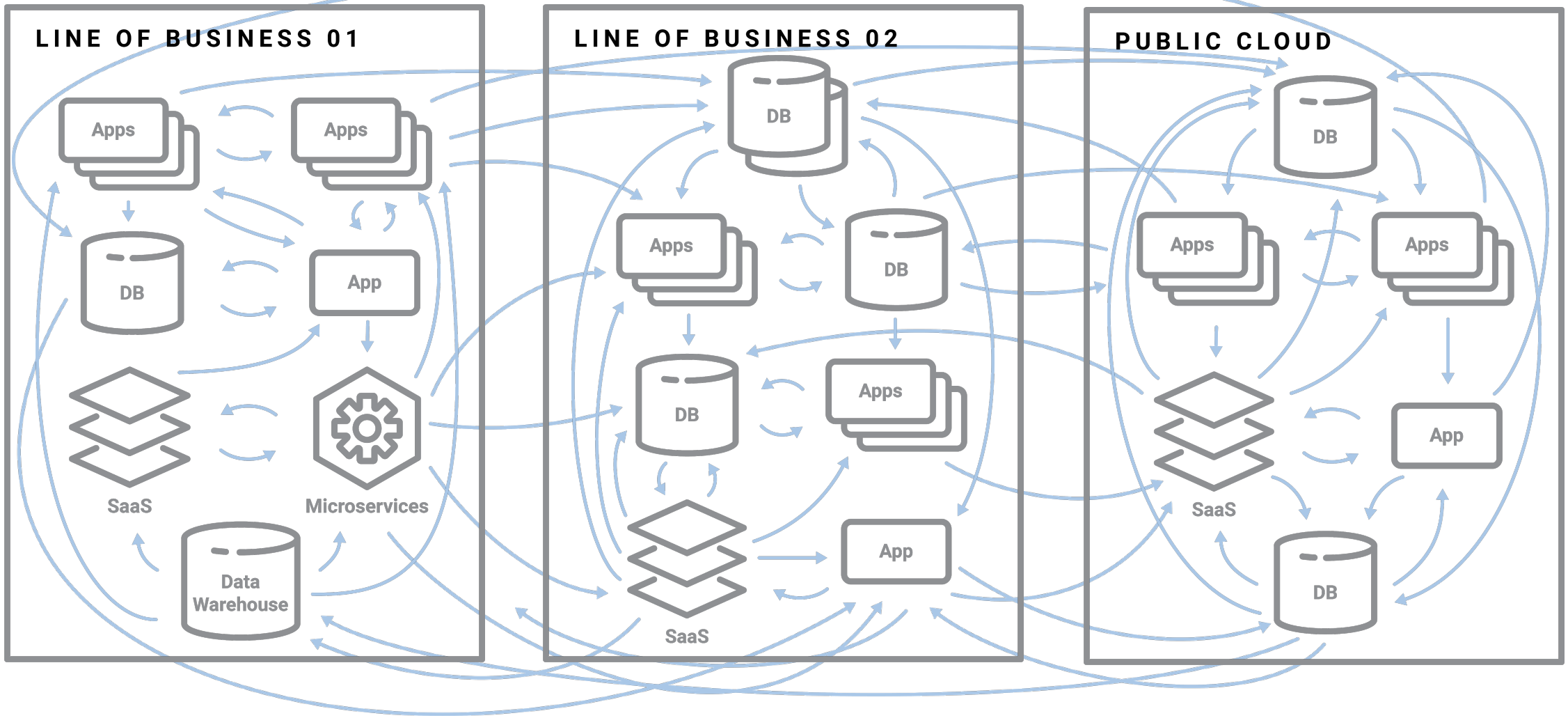

LINE OF BUSINESS 01
Apps
Apps
DB
App
SaaS
Microservices
Data Warehouse

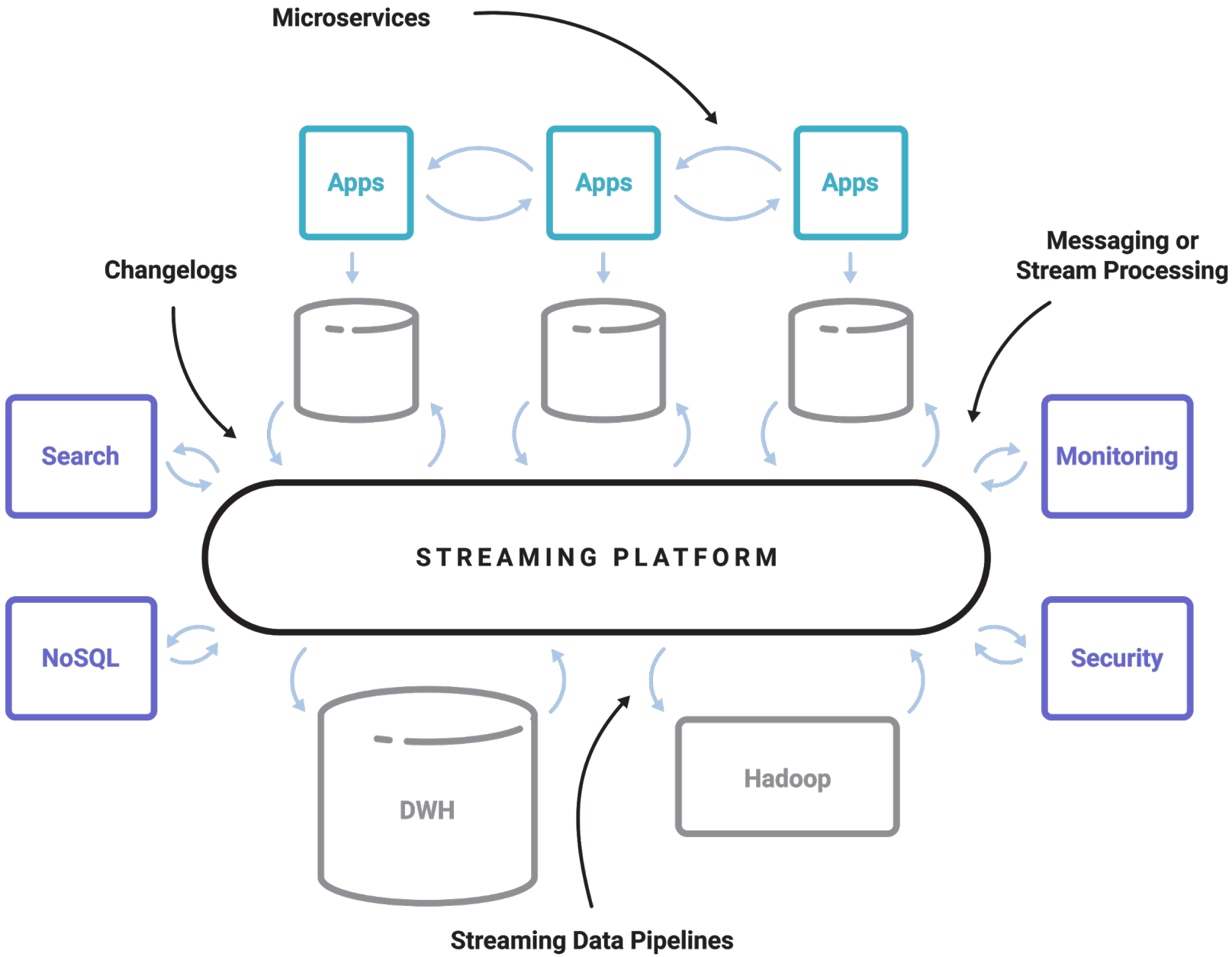

LINE OF BUSINESS 02
DB
Apps
DB
DB
Apps
App
SaaS


PUBLIC CLOUD
DB
Apps
Apps
SaaS
App
DB

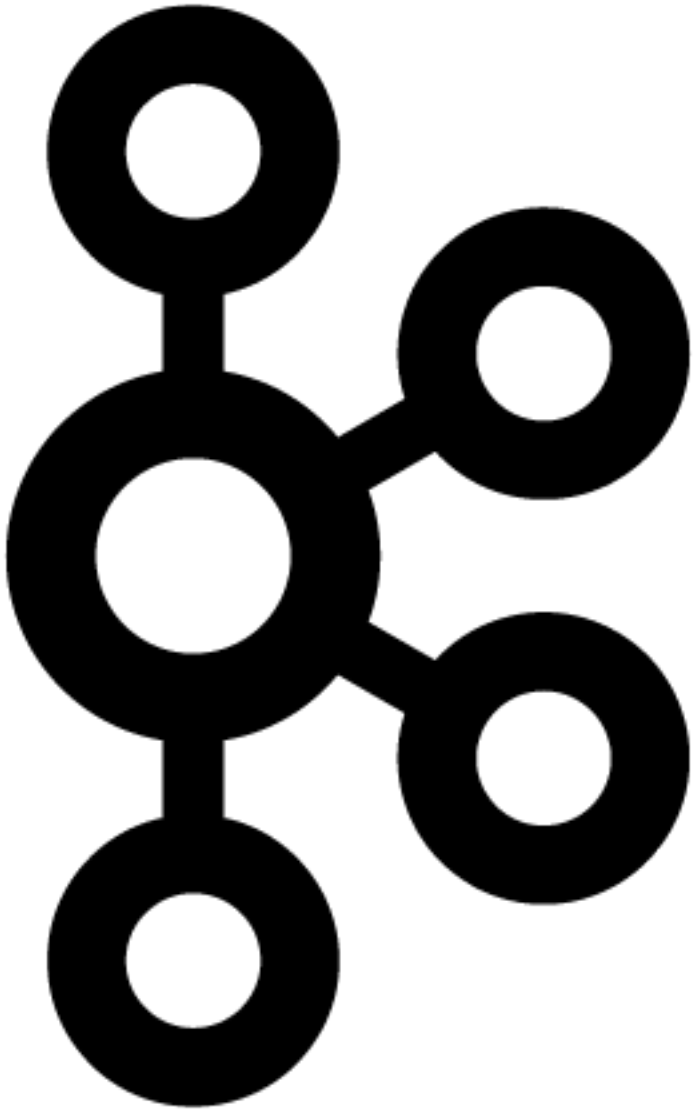

LINE OF BUSINESS 01
Apps
Apps
DB
App
SaaS
Microservices
Data Warehouse
LINE OF BUSINESS 02
DB
Apps
DB
DB
Apps
App
SaaS
PUBLIC CLOUD
DB
Apps
Apps
SaaS
App
DB

TRACE THAT!


LINE OF BUSINESS 01
Apps
Apps
DB
App
SaaS
Microservices
Data Warehouse
LINE OF BUSINESS 02
DB
Apps
DB
DB
Apps
App
SaaS
PUBLIC CLOUD
DB
Apps
Apps
App
SaaS
DB

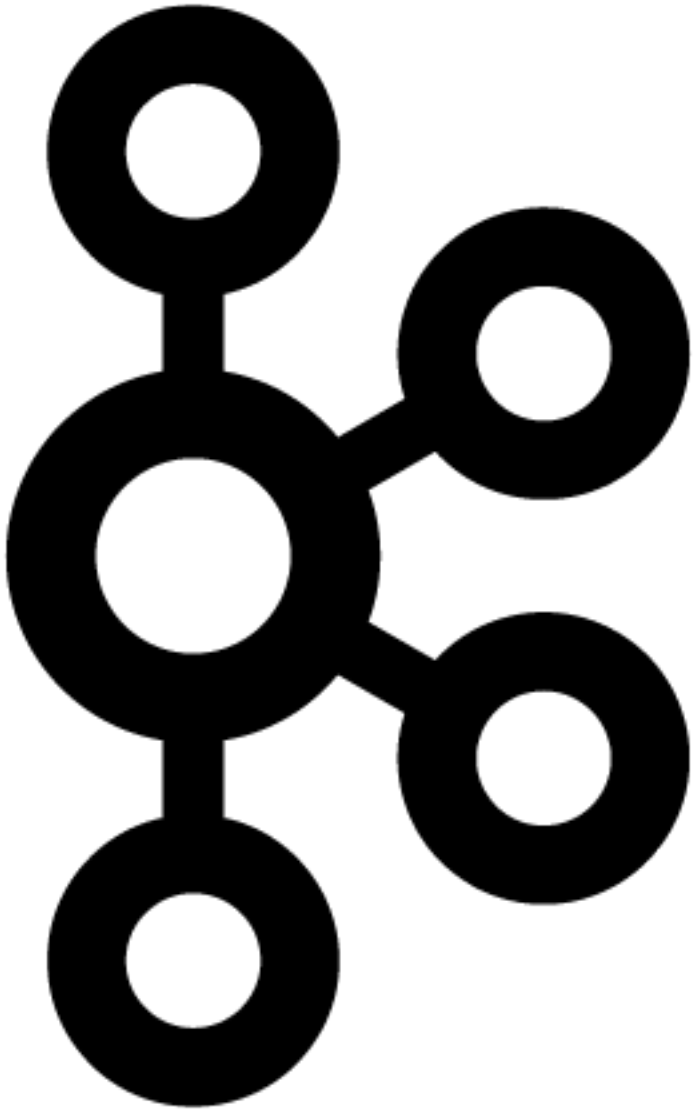

Microservices
Apps
Apps
Apps
Changelogs
Messaging or
Stream Processing
Search
Monitoring
STREAMING PLATFORM
NoSQL
Security
DWH
Hadoop
Streaming Data Pipelines

confluent



WHY TRACE APACHE KAFKA?

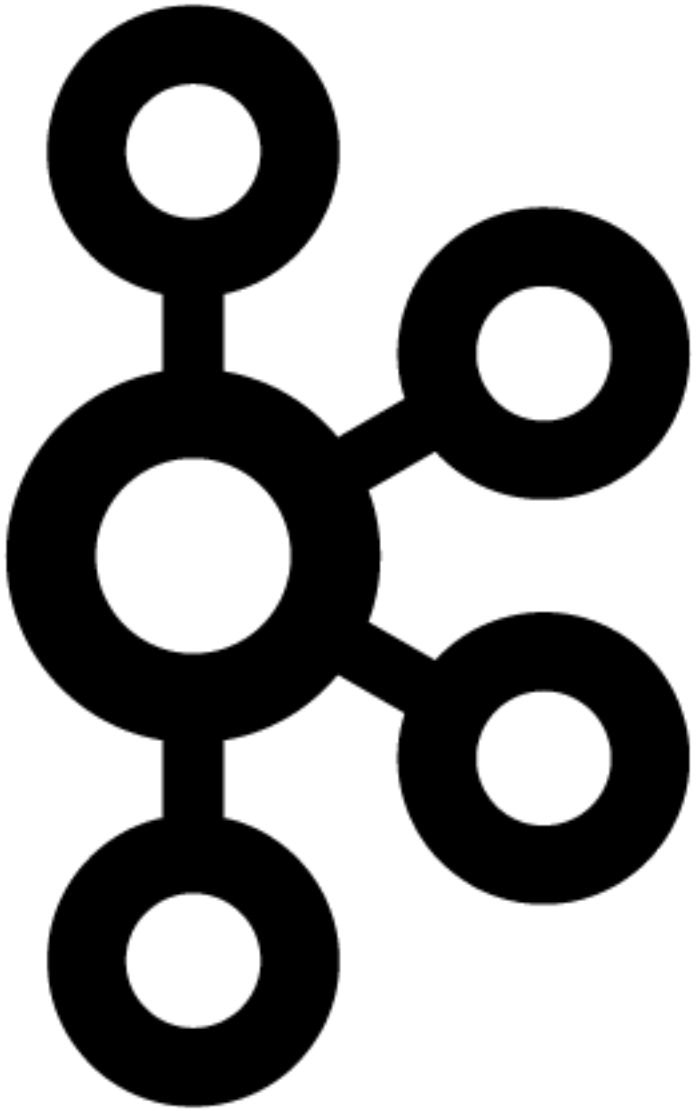
confluent



WHY TRACE APACHE KAFKA?

- Apache Kafka is becoming the de-facto standard to handle data.

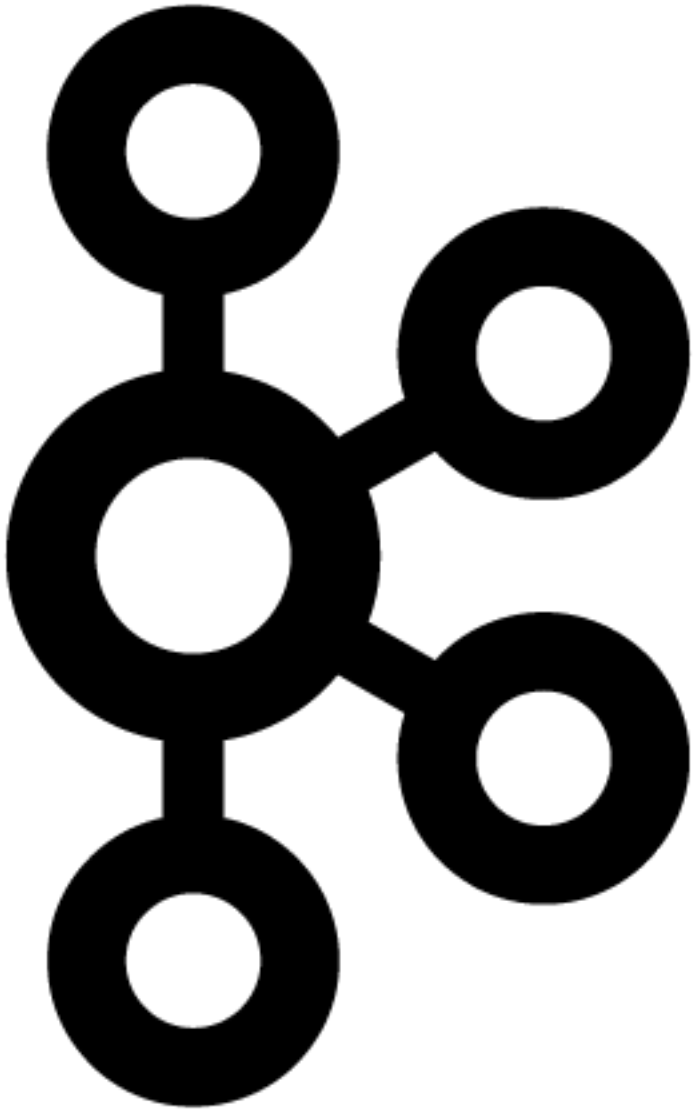
confluent



WHY TRACE APACHE KAFKA?

- Apache Kafka is becoming the de-facto standard to handle data.
- Prediction? It will be the central nervous system of any company.

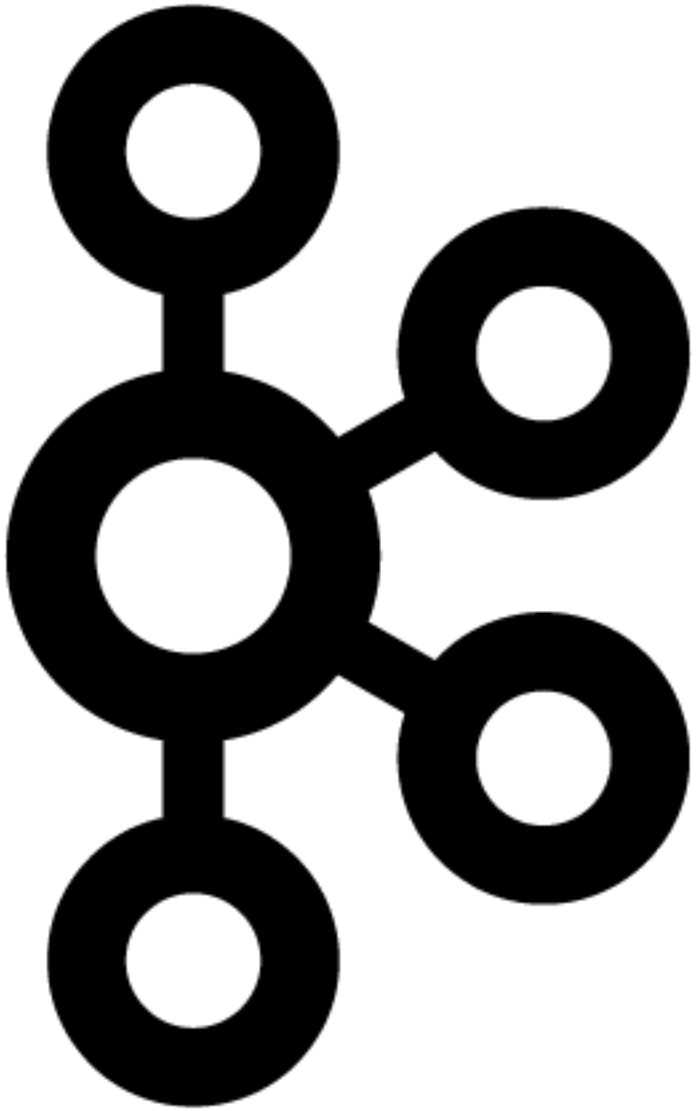
confluent



WHY TRACE APACHE KAFKA?

- Apache Kafka is becoming the de-facto standard to handle data.
- Prediction? It will be the central nervous system of any company.
- µServices in general already use Kafka to exchange messages and keep their data stores in-sync.

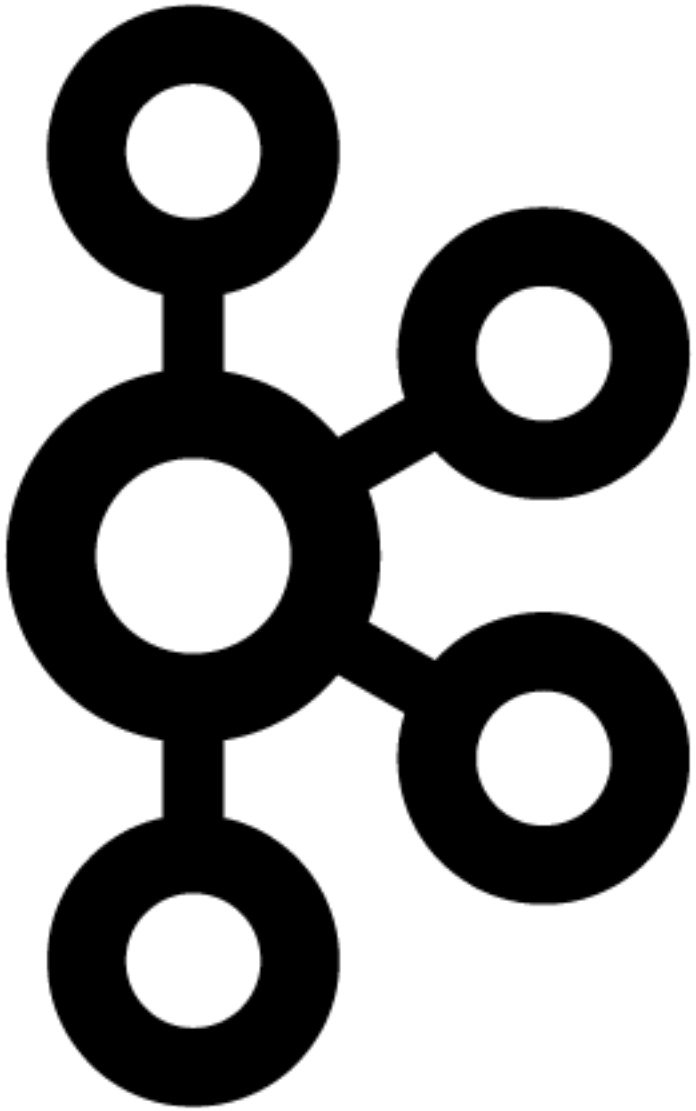
confluent



WHY TRACE APACHE KAFKA?

- Apache Kafka is becoming the de-facto standard to handle data.
- Prediction? It will be the central nervous system of any company.
- µServices in general already use Kafka to exchange messages and keep their data stores in-sync.
- With event streaming becoming even more popular, the Kafka adoption tend to grow even more.

confluent



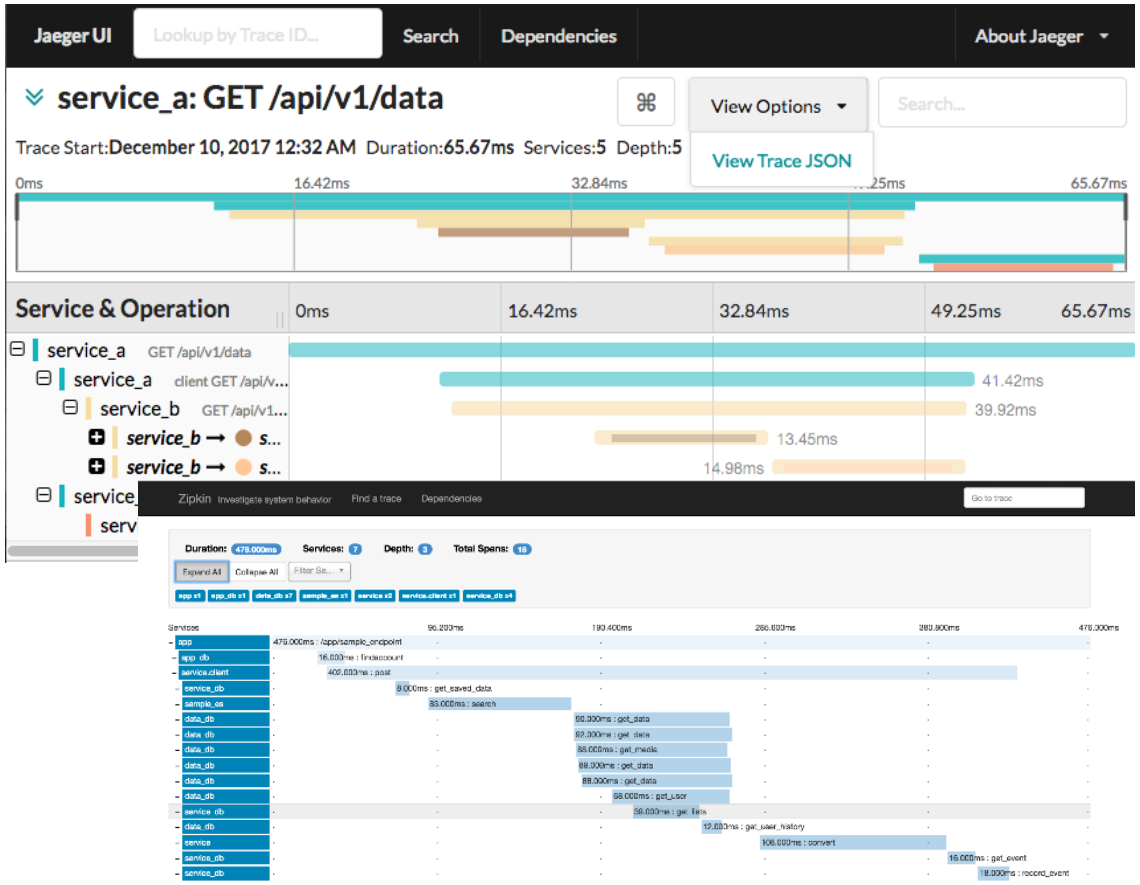
WHY TRACE APACHE KAFKA?

- Apache Kafka is becoming the de-facto standard to handle data.
- Prediction? It will be the central nervous system of any company.
- µServices in general already use Kafka to exchange messages and keep their data stores in-sync.
- With event streaming becoming even more popular, the Kafka adoption tend to grow even more.
- Because it is so freaking cool! 🤗

DISTRIBUTED TRACING IN KAFKA

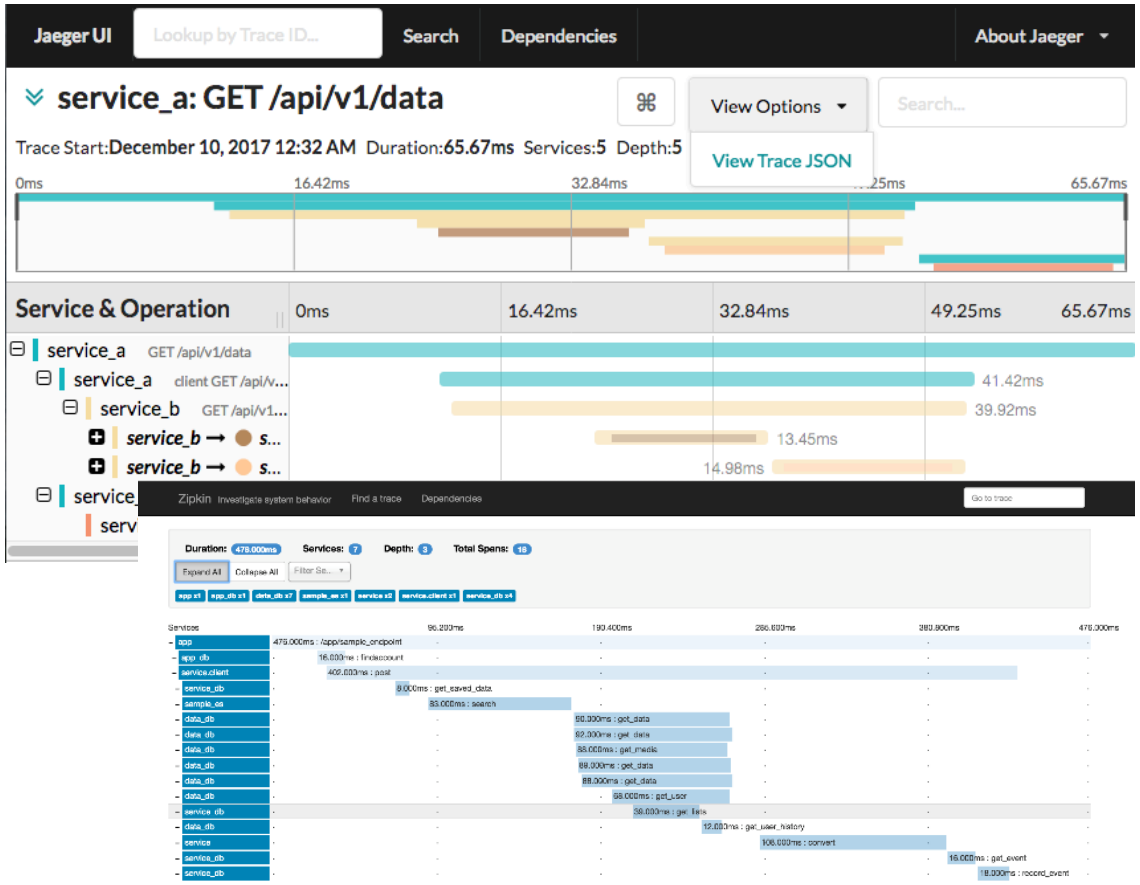
confluent

OPENTRACING JAVA API



https://github.com/opentracing-contrib/java-kafka-client

confluent

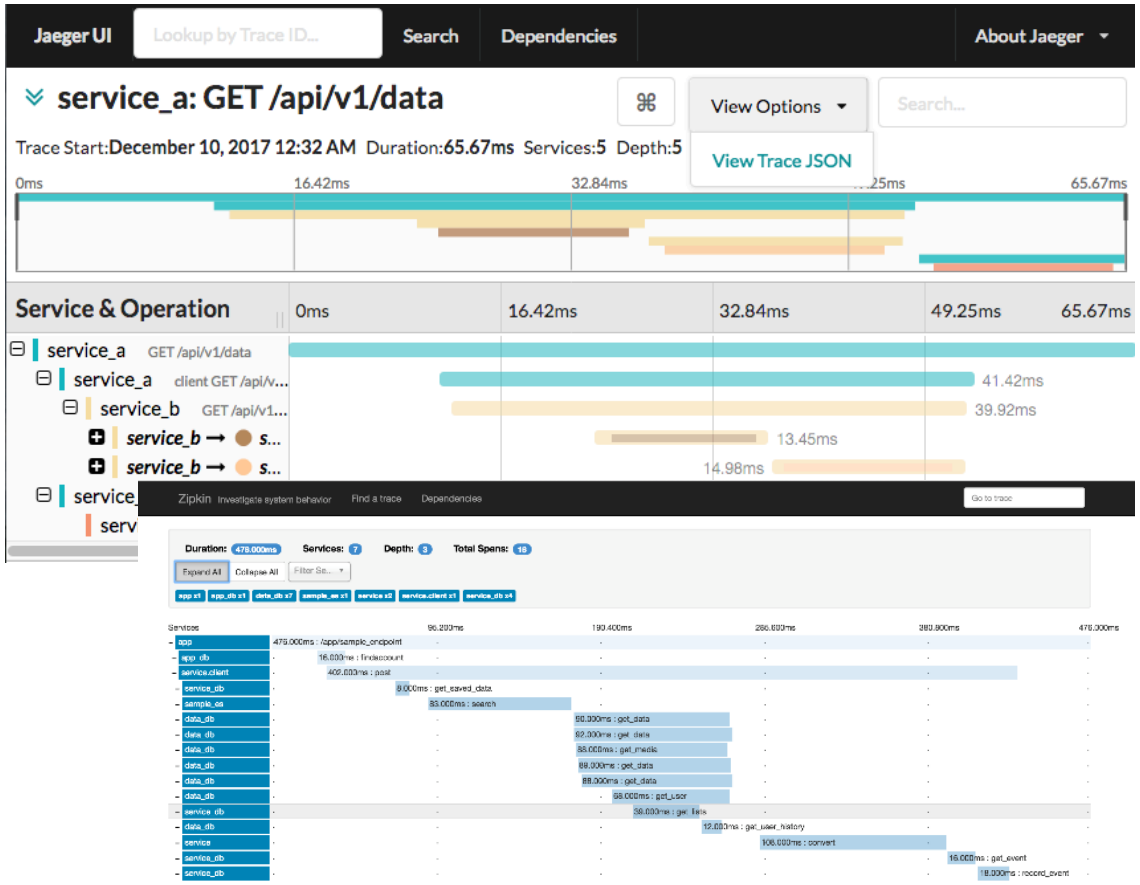


https://github.com/opentracing-contrib/java-kafka-client

OPENTRACING JAVA API

- Library written in Java to handle distributed tracing via OpenTracing compatible APIs.

confluent

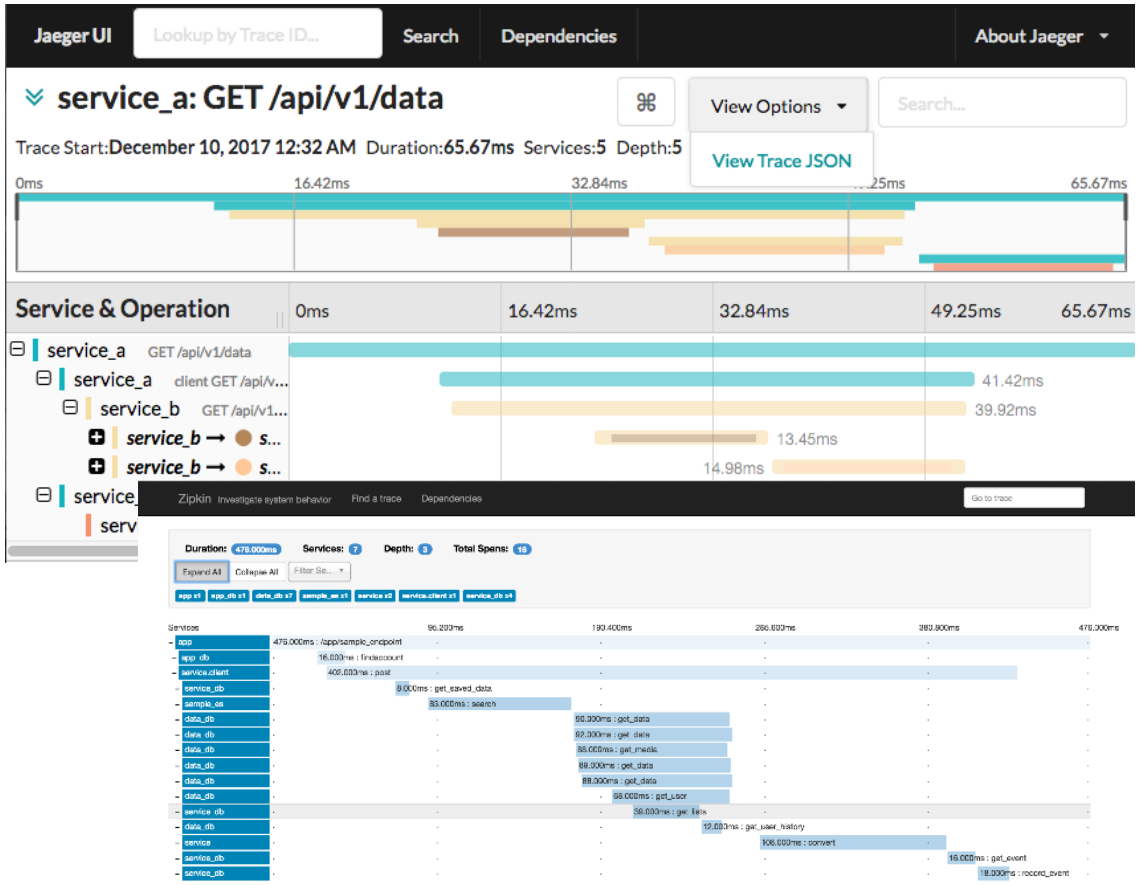


https://github.com/opentracing-contrib/java-kafka-client

OPENTRACING JAVA API

- Library written in Java to handle distributed tracing via OpenTracing compatible APIs.
- Requires the creation of specific tracer using the distributed tracing technology API.

confluent

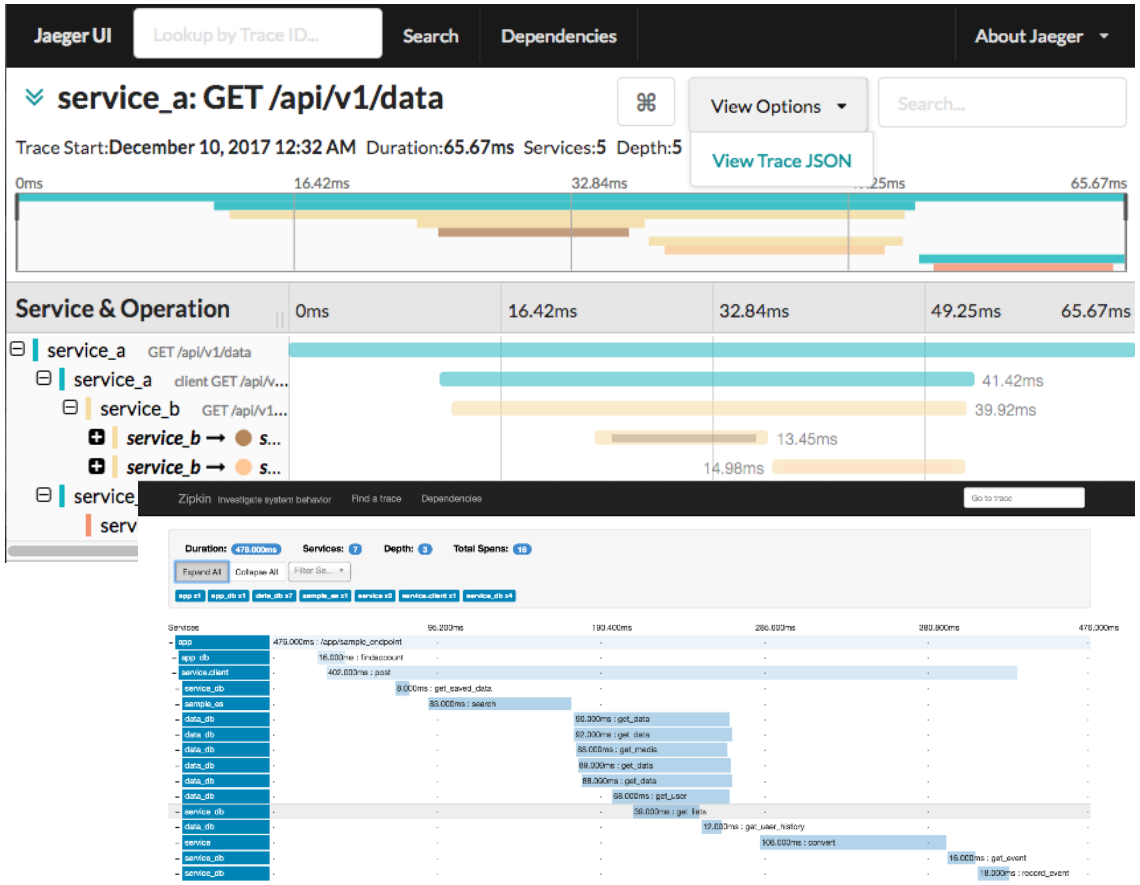


https://github.com/opentracing-contrib/java-kafka-client

OPENTRACING JAVA API

- Library written in Java to handle distributed tracing via OpenTracing compatible APIs.
- Requires the creation of specific tracer using the distributed tracing technology API.
- Uses the **GlobalTracer** utility class to handle the tracer throughout the JVM application.

confluent



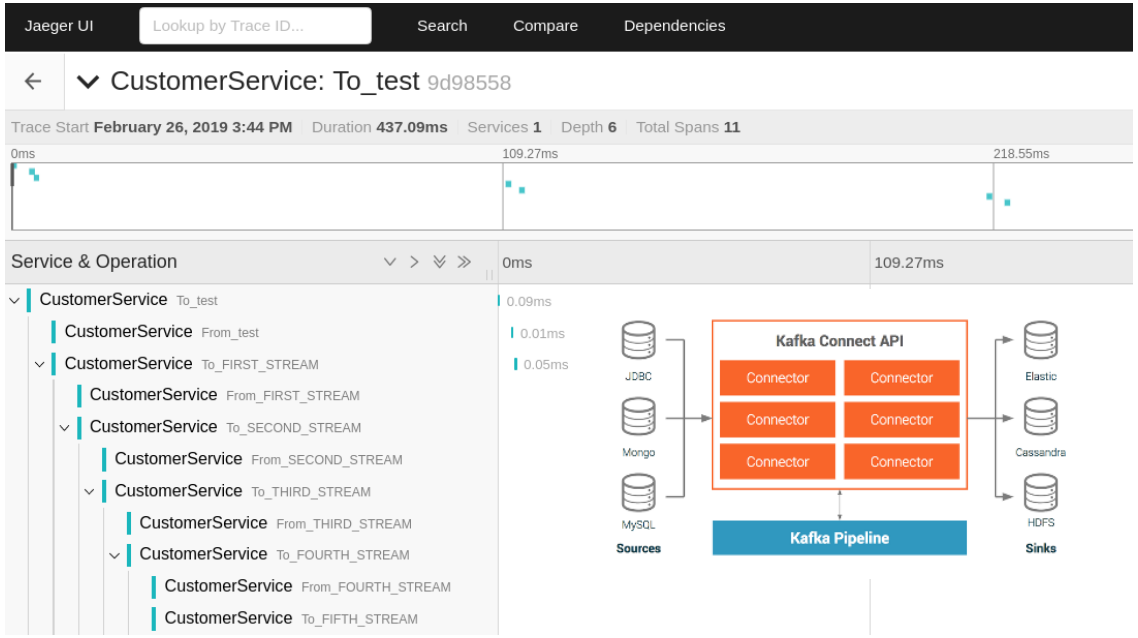
https://github.com/opentracing-contrib/java-kafka-client

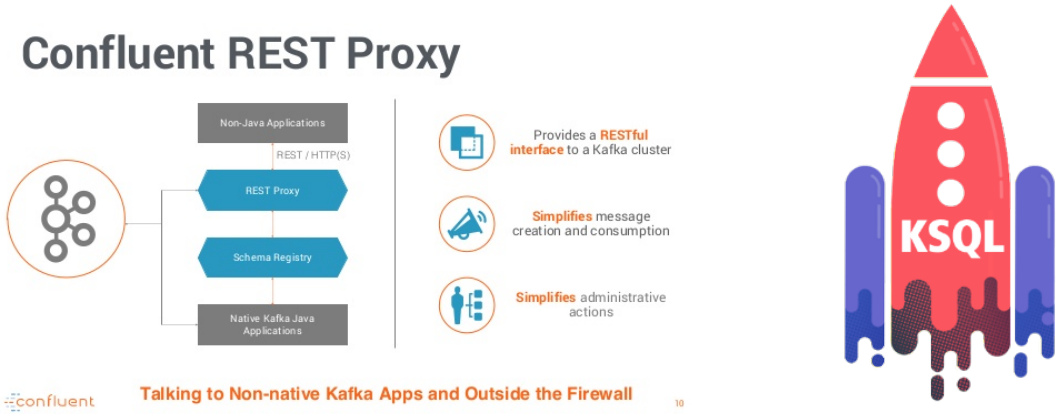
OPENTRACING JAVA API

- Library written in Java to handle distributed tracing via OpenTracing compatible APIs.
- Requires the creation of specific tracer using the distributed tracing technology API.
- Uses the **GlobalTracer** utility class to handle the tracer throughout the JVM application.
- Supports: Apache Kafka Clients, Kafka Streams, and Spring Kafka.

confluent

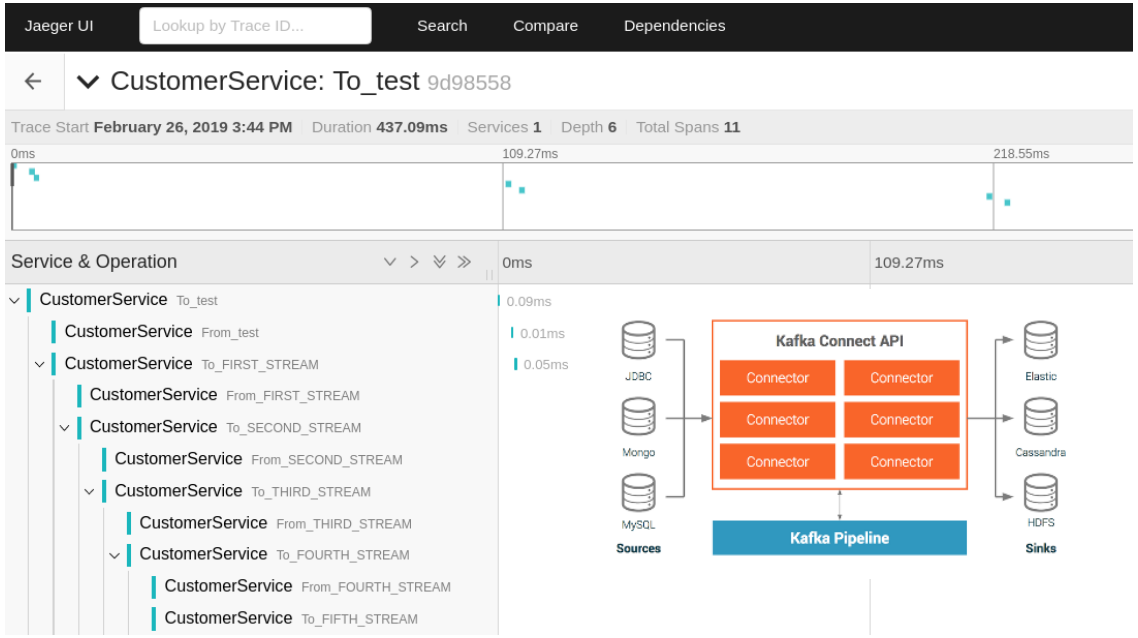
SUPPORT FOR BUNDLED JVMS

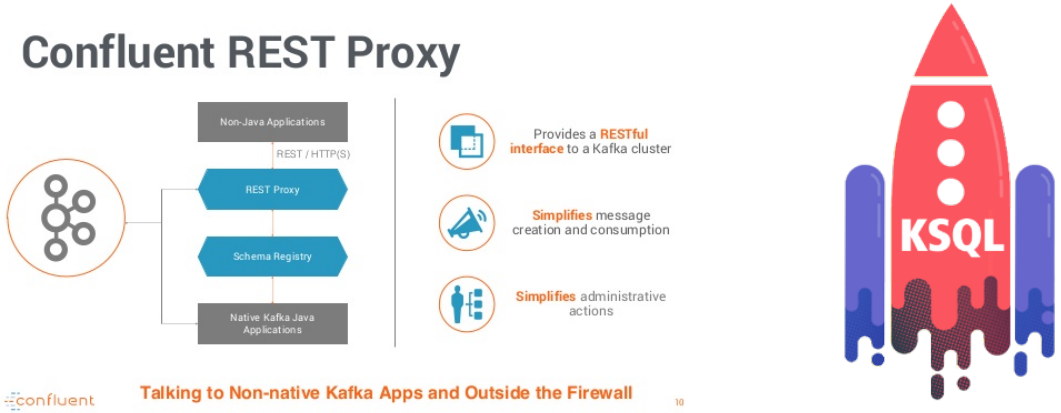




https://github.com/riferrei/kafka-tracing-support

confluent



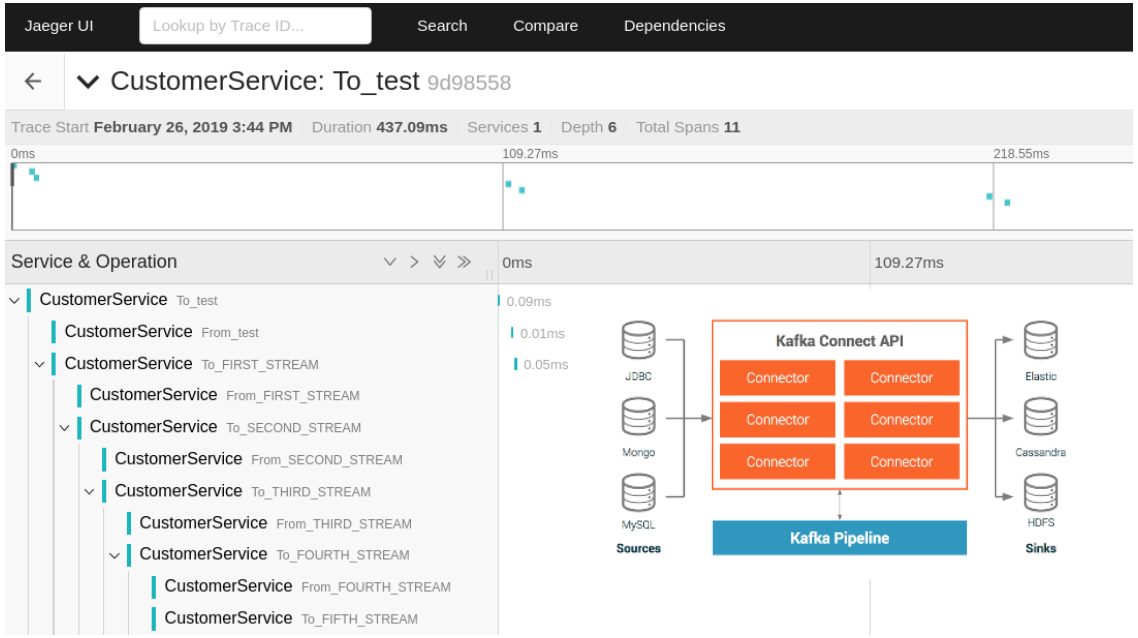


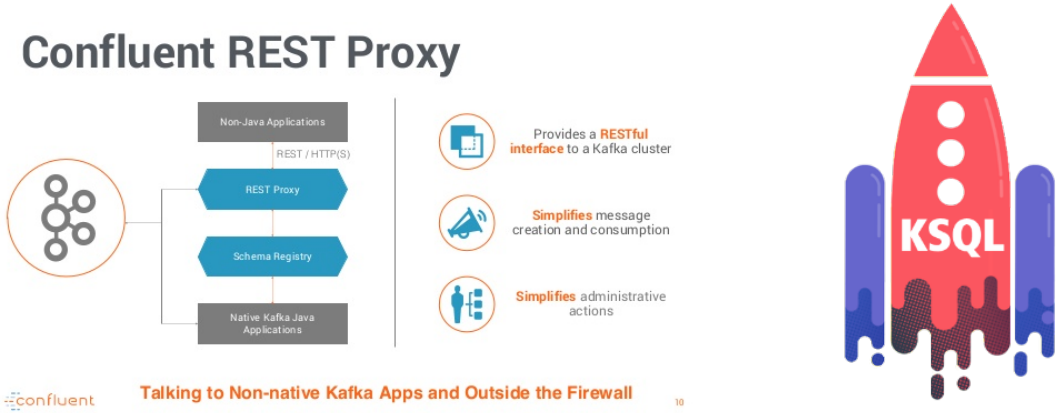
https://github.com/riferrei/kafka-tracing-support

SUPPORT FOR BUNDLED JVMS

- Library written in Java that does the automatic creation of the tracer.

confluent



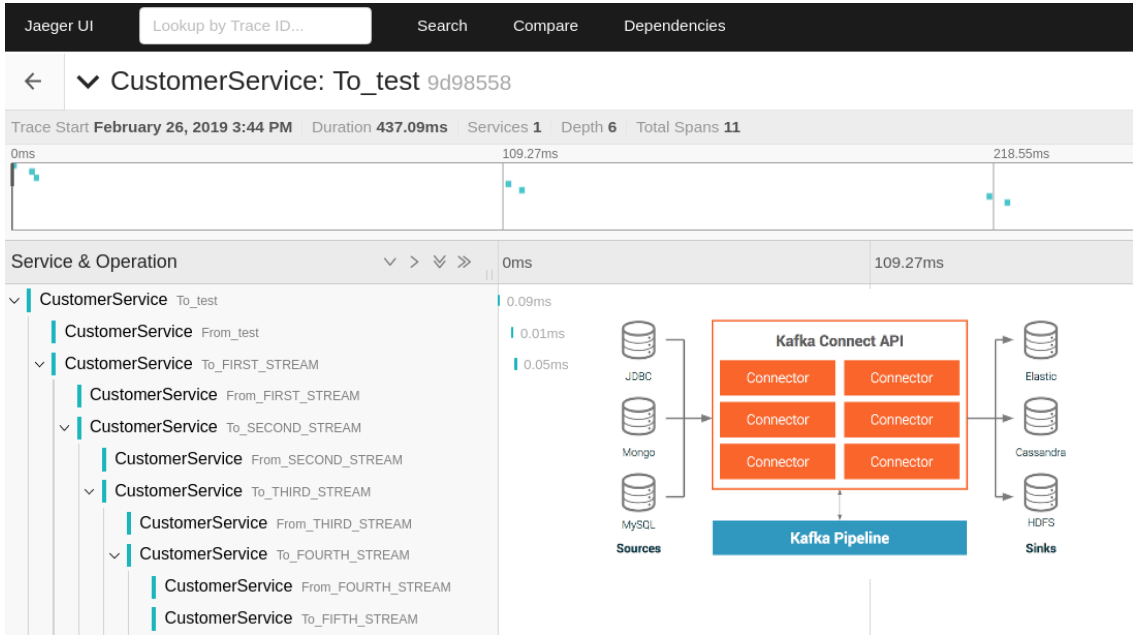


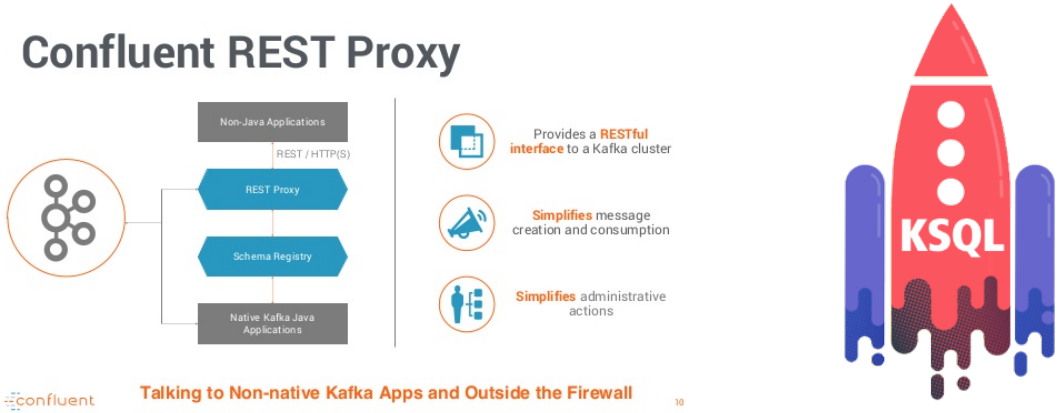
https://github.com/riferrei/kafka-tracing-support

SUPPORT FOR BUNDLED JVMS

- Library written in Java that does the automatic creation of the tracer.
- Implements the tracing logic using the Kafka Interceptors API.

confluent



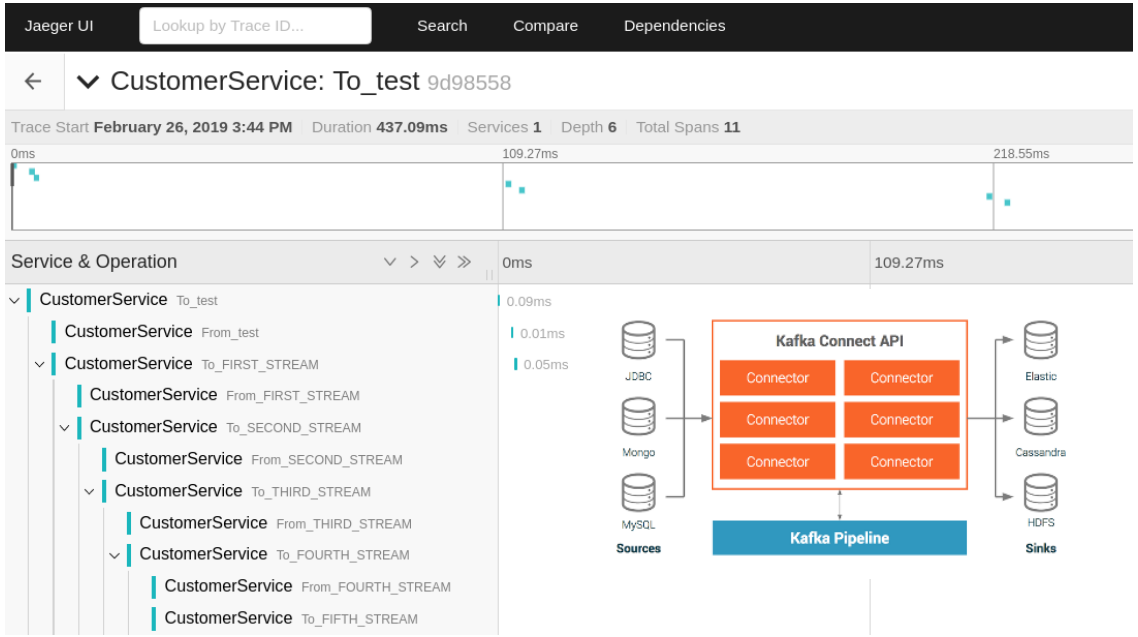


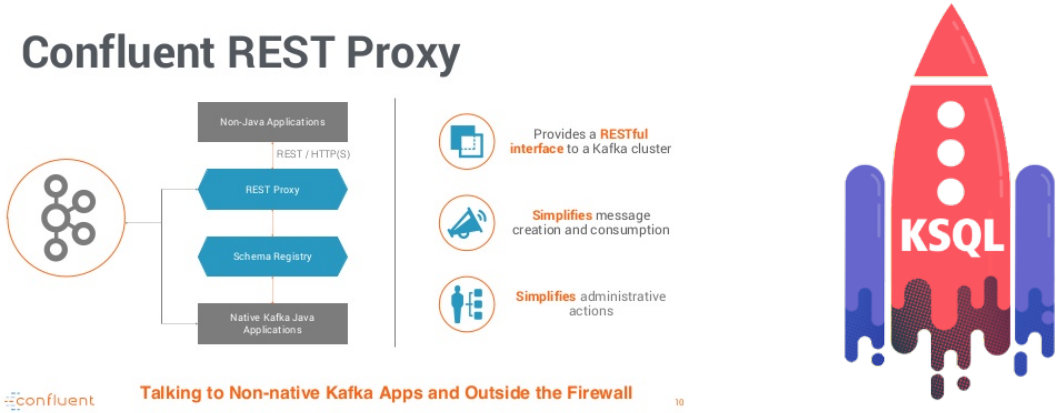
https://github.com/riferrei/kafka-tracing-support

SUPPORT FOR BUNDLED JVMS

- Library written in Java that does the automatic creation of the tracer.
- Implements the tracing logic using the Kafka Interceptors API.
- Allows different tracers to be used, by using the TracerResolver class.

confluent



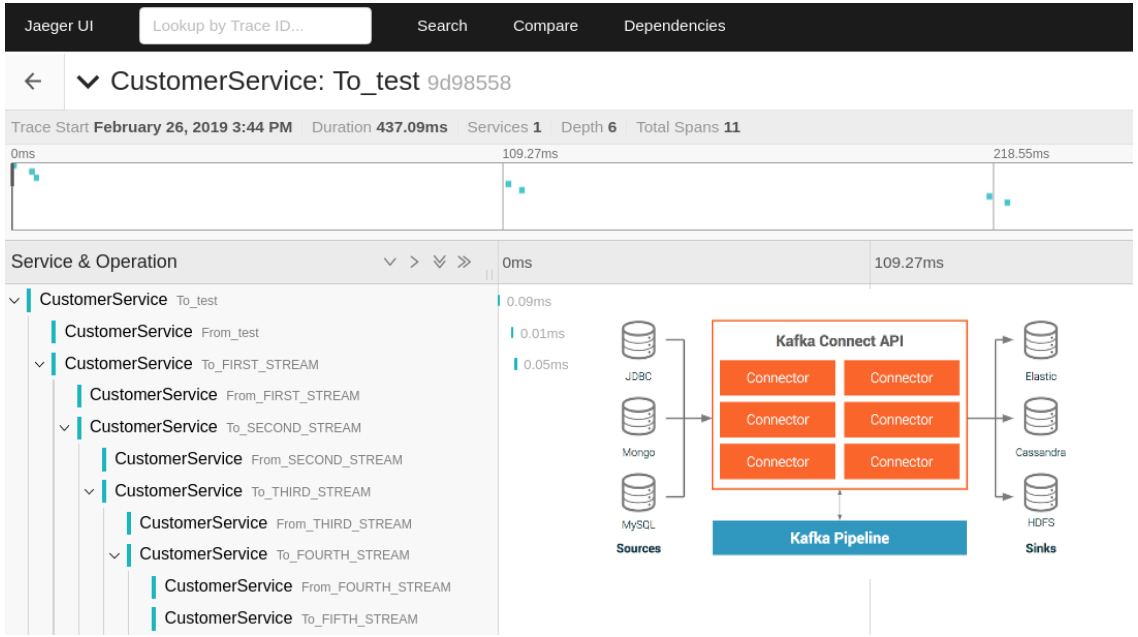


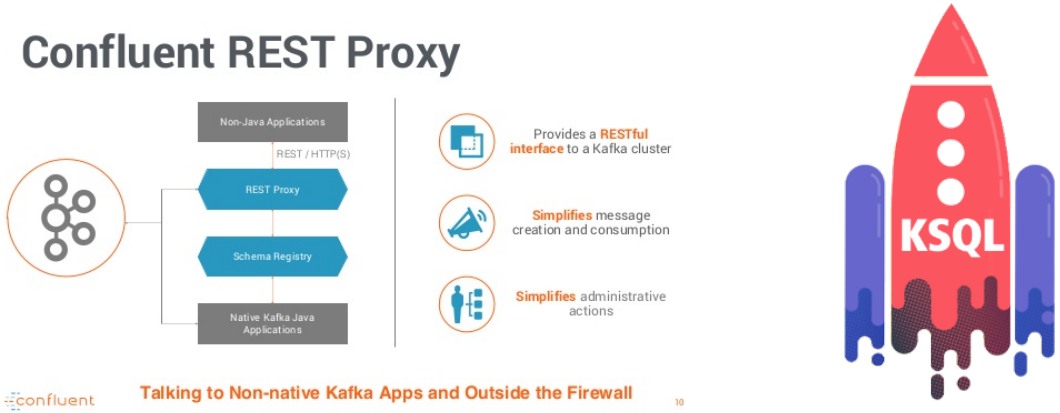
https://github.com/riferrei/kafka-tracing-support

SUPPORT FOR BUNDLED JVMS

- Library written in Java that does the automatic creation of the tracer.
- Implements the tracing logic using the Kafka Interceptors API.
- Allows different tracers to be used, by using the TracerResolver class.
- Provides OOTB support for Jaeger.

confluent



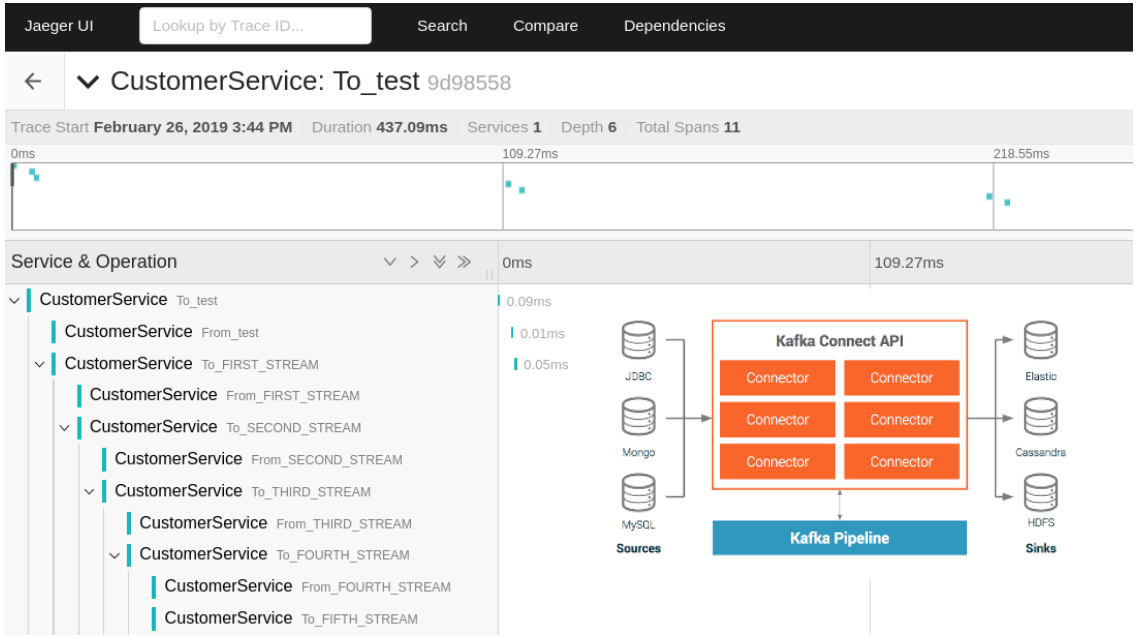


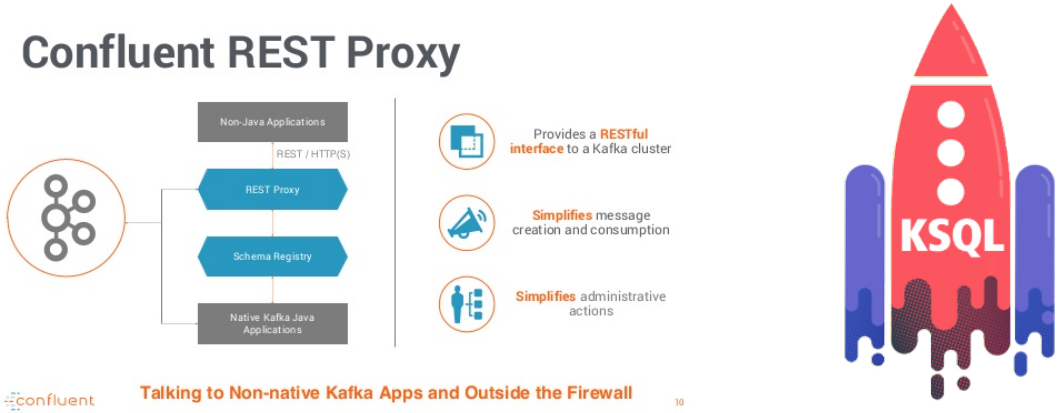
https://github.com/riferrei/kafka-tracing-support

SUPPORT FOR BUNDLED JVMS

- Library written in Java that does the automatic creation of the tracer.
- Implements the tracing logic using the Kafka Interceptors API.
- Allows different tracers to be used, by using the TracerResolver class.
- Provides OOTB support for Jaeger.
- Allows multiple services in the JVM use their own tracer by specifying a configuration properties file.

confluent





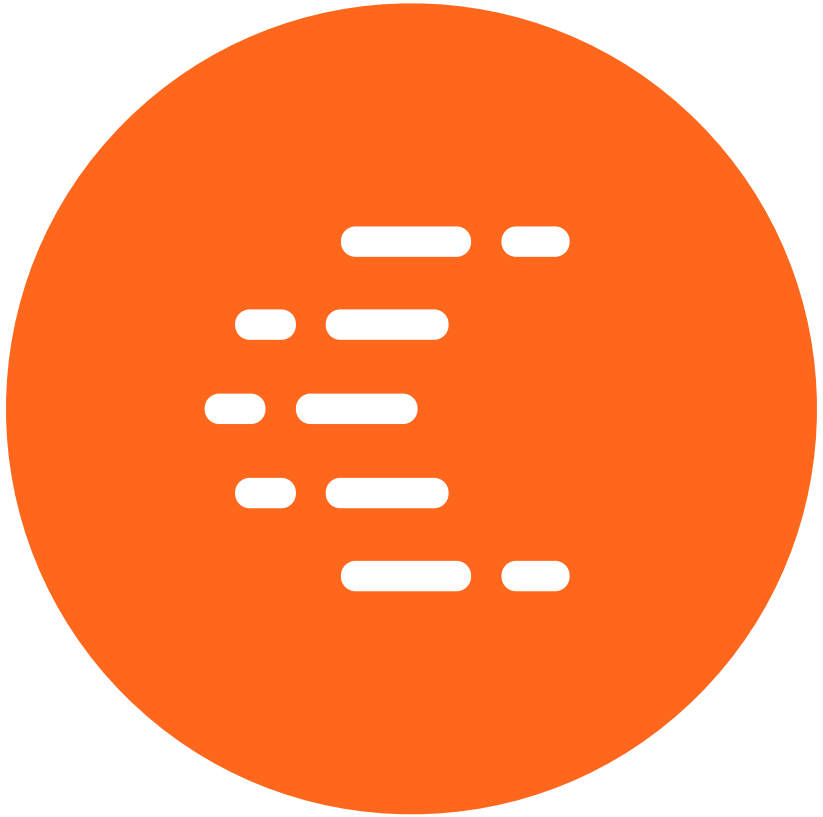
https://github.com/riferrei/kafka-tracing-support

SUPPORT FOR BUNDLED JVMS

- Library written in Java that does the automatic creation of the tracer.
- Implements the tracing logic using the Kafka Interceptors API.
- Allows different tracers to be used, by using the TracerResolver class.
- Provides OOTB support for Jaeger.
- Allows multiple services in the JVM use their own tracer by specifying a configuration properties file.
  - export INTERCEPTORS_CONFIG_FILE=

DEMO


NEXT STOP
SAN FRANCISCO
SEPT 30-OCT 1
GAMOV40
40% OFF*, duh!
kafka
summit
*Standard Priced Conference pass



THANKS!

@gamussa

viktor@confluent.io

https://slackpass.io/confluentcommunity


confluent