CONFLUENT

THE TALES OF EVENT-DRIVEN TOLD BY A SPRING DEVELOPER

September 2020 / SpringOne


Streams

PREFACE

Photo by Khai Sze Ong on Unsplash


Photo by Rohier Ridao on Unsplash


Photo by Rohit Tandon on Unsplash

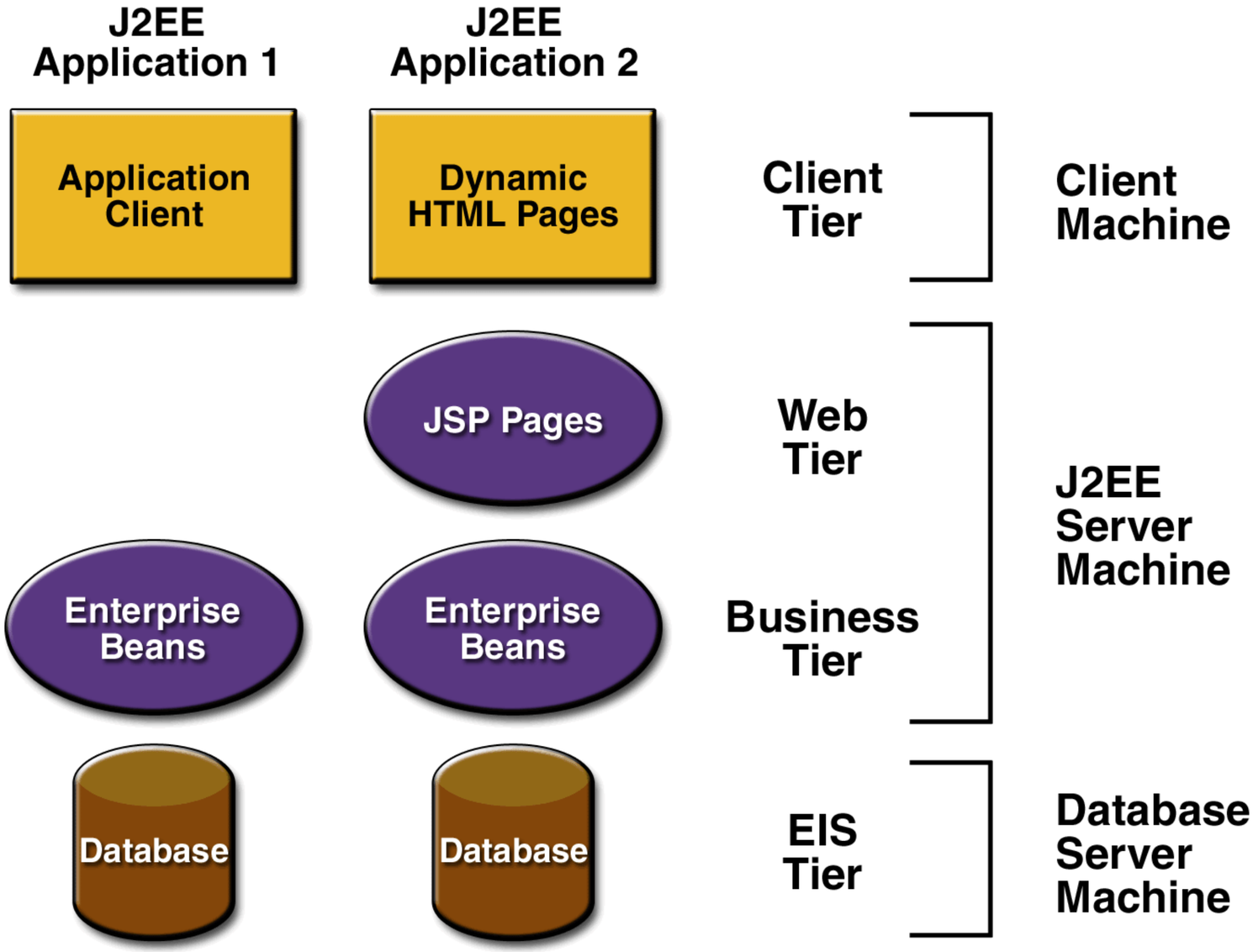

Photo by Theodore Moore on Unsplash

Photo by Cristian Grecu on Unsplash


PETE
DEAD


REFACTORING TO
MICROSERVICES


IT USED TO BE SO SIMPLE...


J2EE Application 1
J2EE Application 2
Application Client
Dynamic HTML Pages
Client Tier
Client Machine
JSP Pages
Web Tier
J2EE Server Machine
Enterprise Beans
Enterprise Beans
Business Tier
Database
Database
EIS Tier
Database Server Machine


CONFLUENT
NORMAL APPLICATIONS
(I.E., MONOLITHS)
@gamussa | #springone | @confluentinc


CONFLUENT
MONOLITHS
ARE HARD TO THINK ABOUT
@gamussa | #springone | @confluentinc


CONFLUENT
MONOLITHS ARE HARD TO CHANGE
@gamussa | #springone | @confluentinc

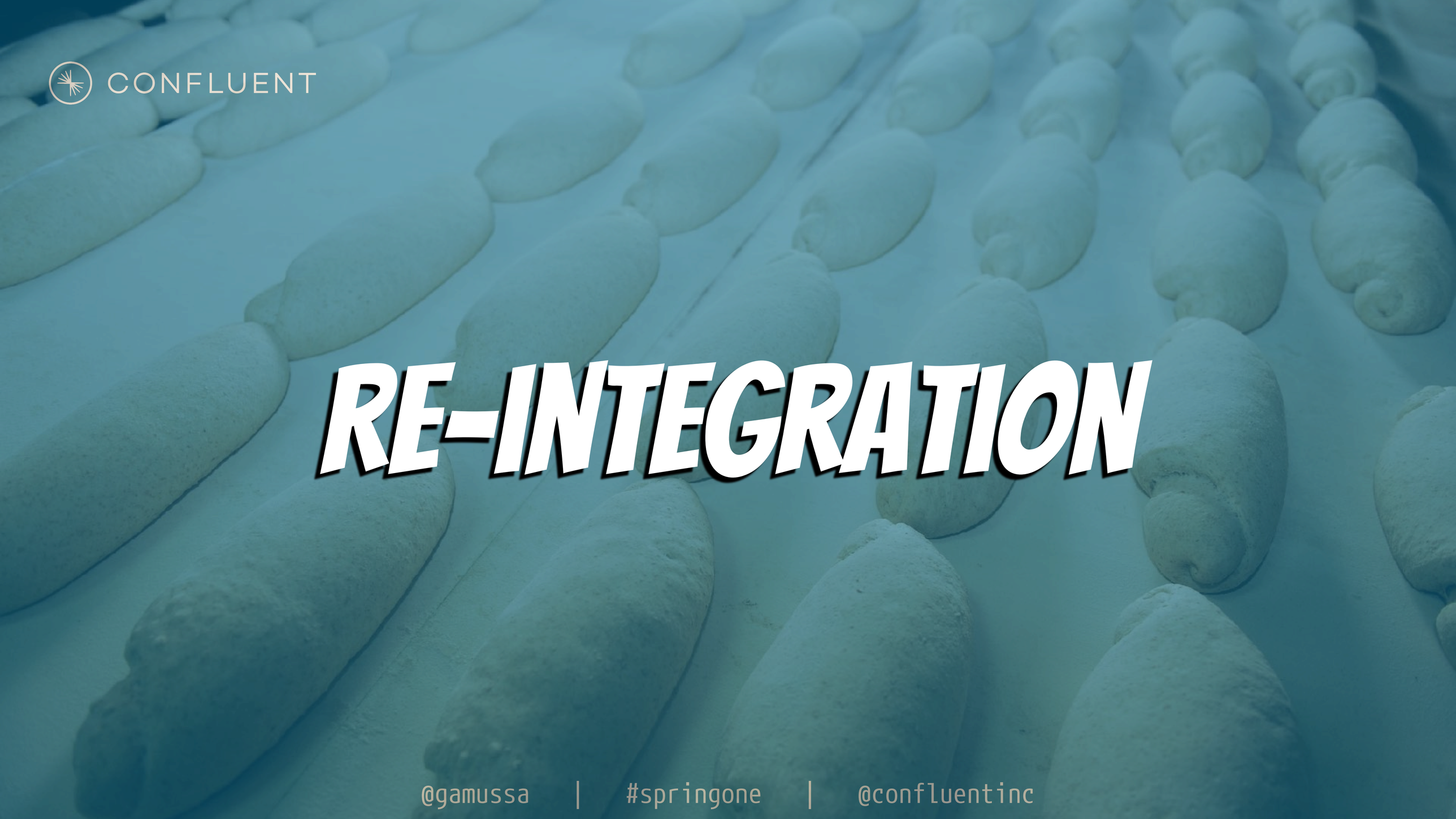

CONFLUENT
RE-INTEGRATION
@gamussa | #springone | @confluentinc


Photo by Deva Darshan from Pexels


THERE ARE NO GOOD WAYS
TO INTEGRATE
MICROSERVICES

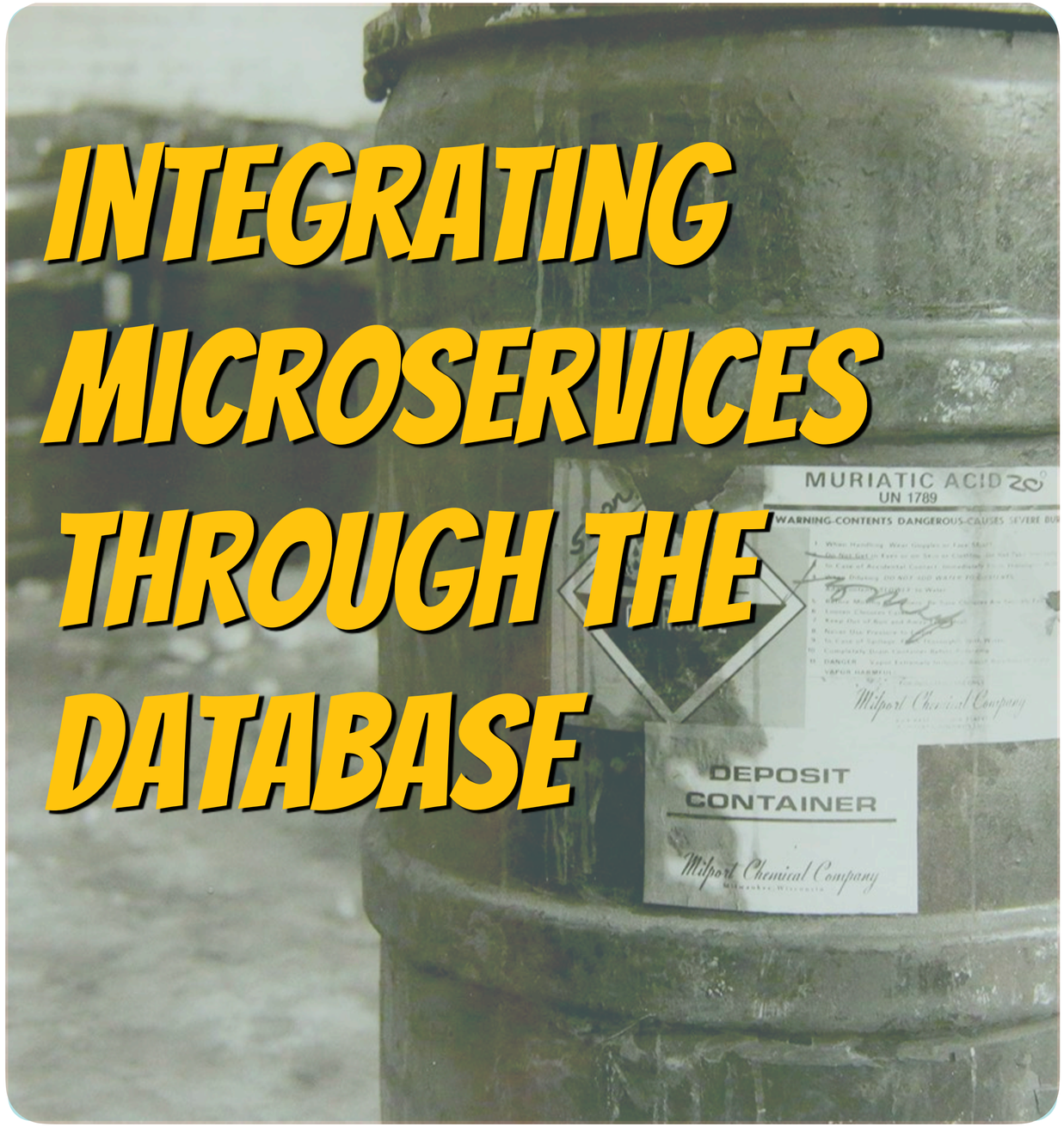

THERE ARE NO-GOOD WAYS
TO INTEGRATE
MICROSERVICES


CONFLUENT
FILESYSTEM
@gamussa | #springone | @confluentinc

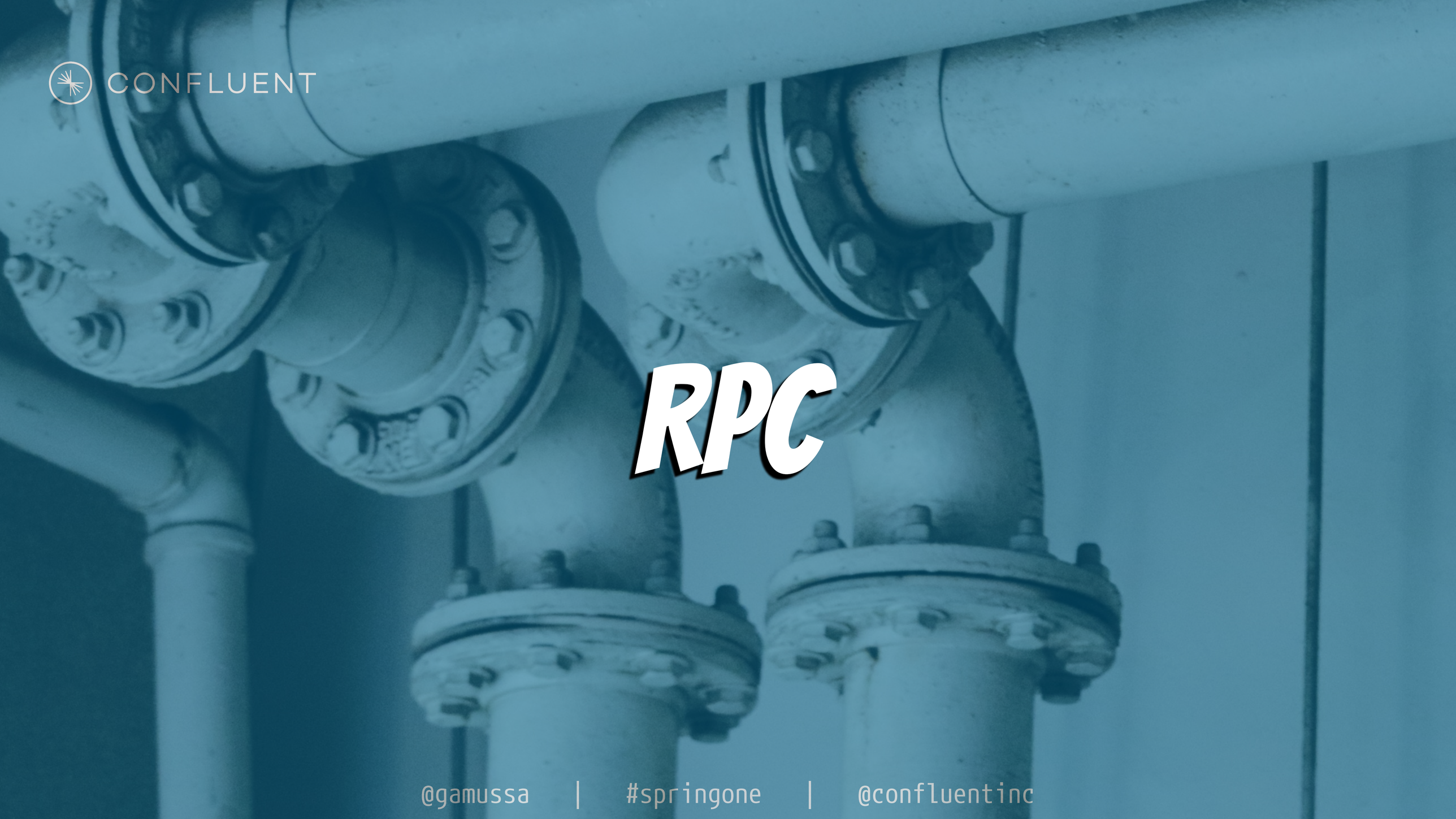
CONFLUENT

DATABASE

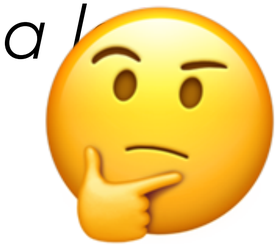

Nope. Don't like that.



- “I have a database and I know how to use it.”
- Eventually causes services to co-mingle.
- Violation the «bounded context»
- Great to use **inside** a service boundary!
- Terrible for **sharing data** or **negotiating change**.

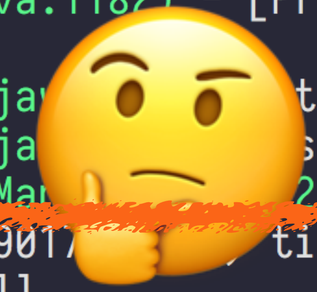

CONFLUENT
RPC
@gamussa | #springone | @confluentinc

- Avoids problems of database integration
- Feels natural
- Aligns with the request/response paradigm
- Problem: cascading failures
- Question: how do you **debug** this system?
- Answer: *you build a l*



```
DemoApplicationTests > contextLoads() PASSED

TransactionTransformerTest > shouldHaveEnoughFunds() PASSED

TransactionTransformerTest > shouldHaveInsufficientFunds() PASSED

TransactionTransformerTest > shouldStoreTransaction() PASSED

KStreamConfigTest > balanceShouldBe300() PASSED

KStreamConfigTest > shouldCreateSuccessfulTransaction() PASSED

KStreamConfigTest > testDriverShouldNotBeNull() PASSED

KStreamConfigTest sh[illegible]
2020-09-01 23:27:[illegible] [INFO] (AbstractStreams.java:987) - stream-client [trans
2020-09-01 23:27:[illegible] [INFO] [illegible] (StreamThread.java:1116) - stream
2020-09-01 23:27:30 o.a.kafka.streams.processor.internals.StreamThread [INFO] (StreamThread.java:221) - stream-
2020-09-01 23:27:30 o.a.kafka.streams.processor.internals.StreamThread [INFO] (StreamThread.java:1130) - stream
2020-09-01 23:27:30 org.apache.kafka.clients.consumer.KafkaConsumer [INFO] (KafkaConsumer.java:1082) - [Consume
atterns and assigned partitions
2020-09-01 23:27:30 org.apache.kafka.clients.producer.KafkaProducer [INFO] (KafkaProducer.java:1182) - [Produce
233720368547758O7 ms.
2020-09-01 23:27:30 o.a.kafka.streams.processor.internals.StreamThread [INFO] (StreamThread.ja[illegible]tream-
2020-09-01 23:27:30 o.a.kafka.streams.processor.internals.StreamThread [INFO] (StreamThread.ja[illegible]stream
2020-09-01 23:27:30 o.a.k.clients.admin.internals.AdminMetadataManager [INFO] (AdminMetadataMa[illegible]235) -
org.apache.kafka.common.errors.TimeoutException: Call(callName=fetchMetadata, deadlineMs=159901[illegible], timed
Caused by: org.apache.kafka.common.errors.TimeoutException: Timed out waiting to send the call.
2020-09-01 23:27:30 org.apache.kafka.streams.KafkaStreams [INFO] (KafkaStreams.java:989) - stream-client [trans
```

LOG4J TYPE OF LOG?



CONFLUENT

EVENTS?

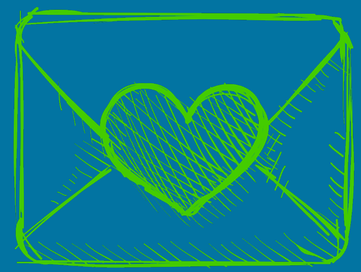
WHAT'S AN EVENT?

CONFLUENT

AN OCCURRENCE, ESPECIALLY ONE OF SOME IMPORTANCE.

A COMBINATION OF:

- NOTIFICATION

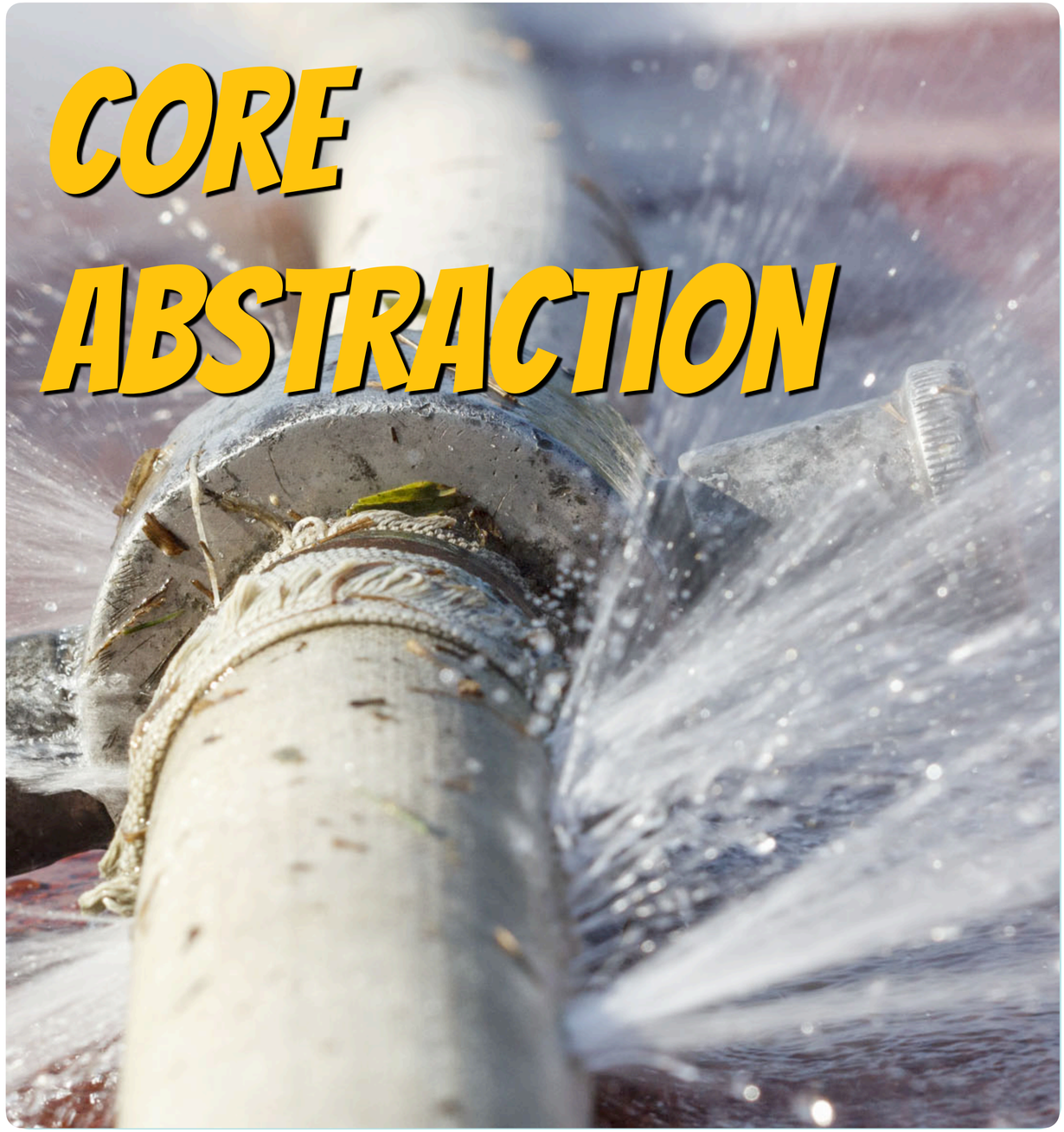


- STATE TRANSFER

ALSO,

EVENTS

ARE IMMUTABLE.



- DB - table
- Hadoop - file
- Kafka - ?

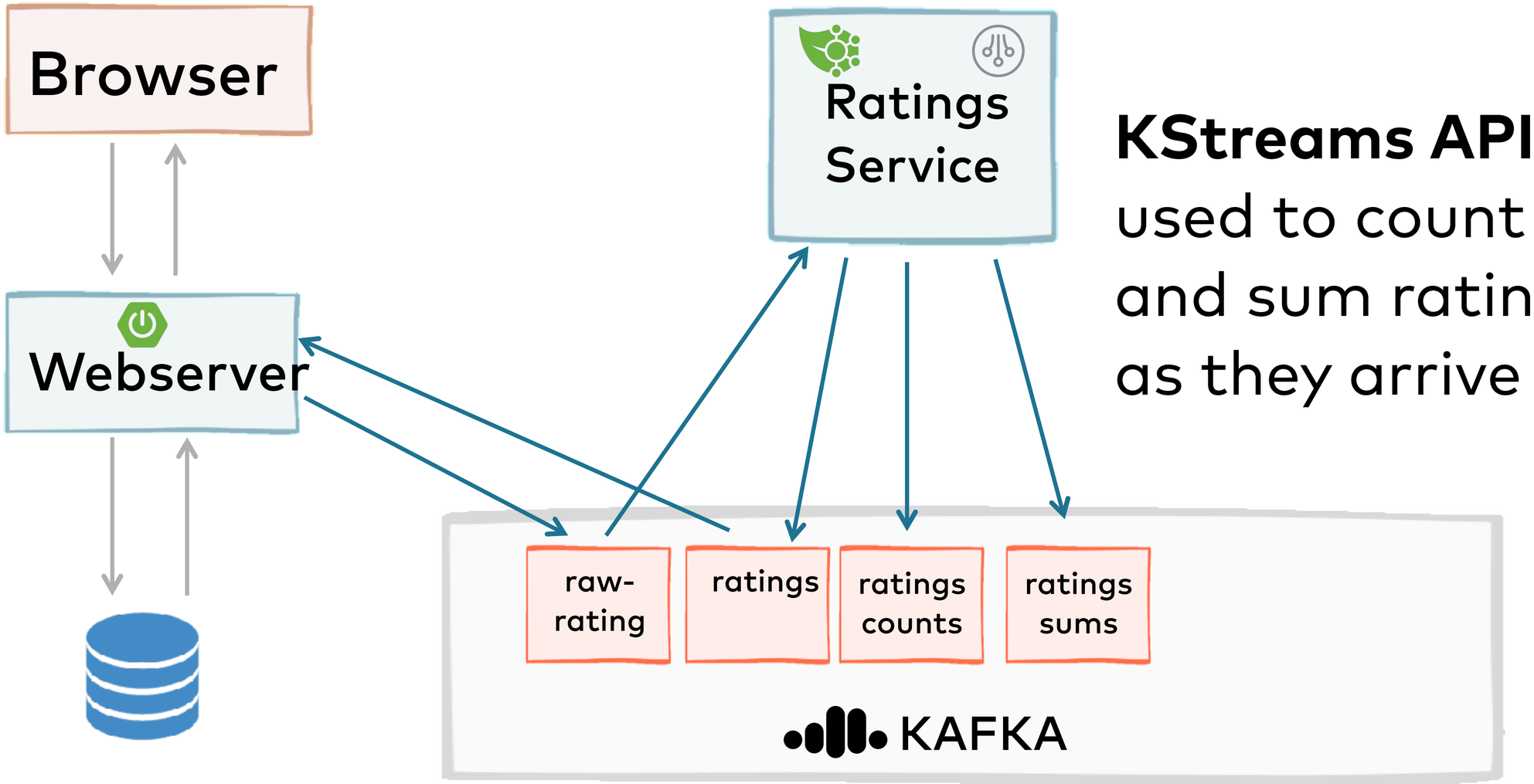
LOG

CONFLUENT

A RICHER MICROSERVICES APPLICATION

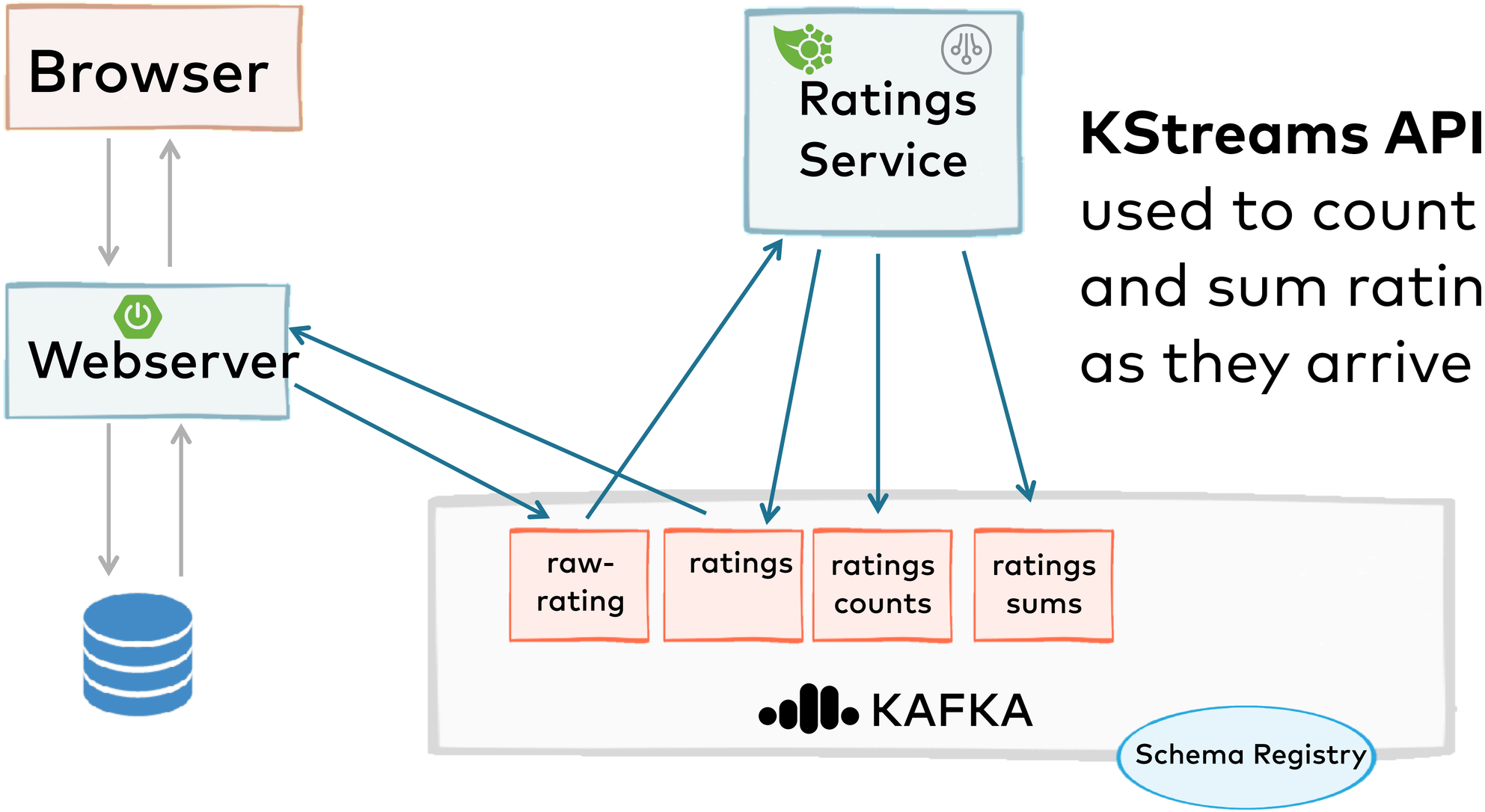

CERTIFIED
FRESH
Rotten
Tomatoes

RATINGS SERVICE OPERATING ON THE EVENT STREAM



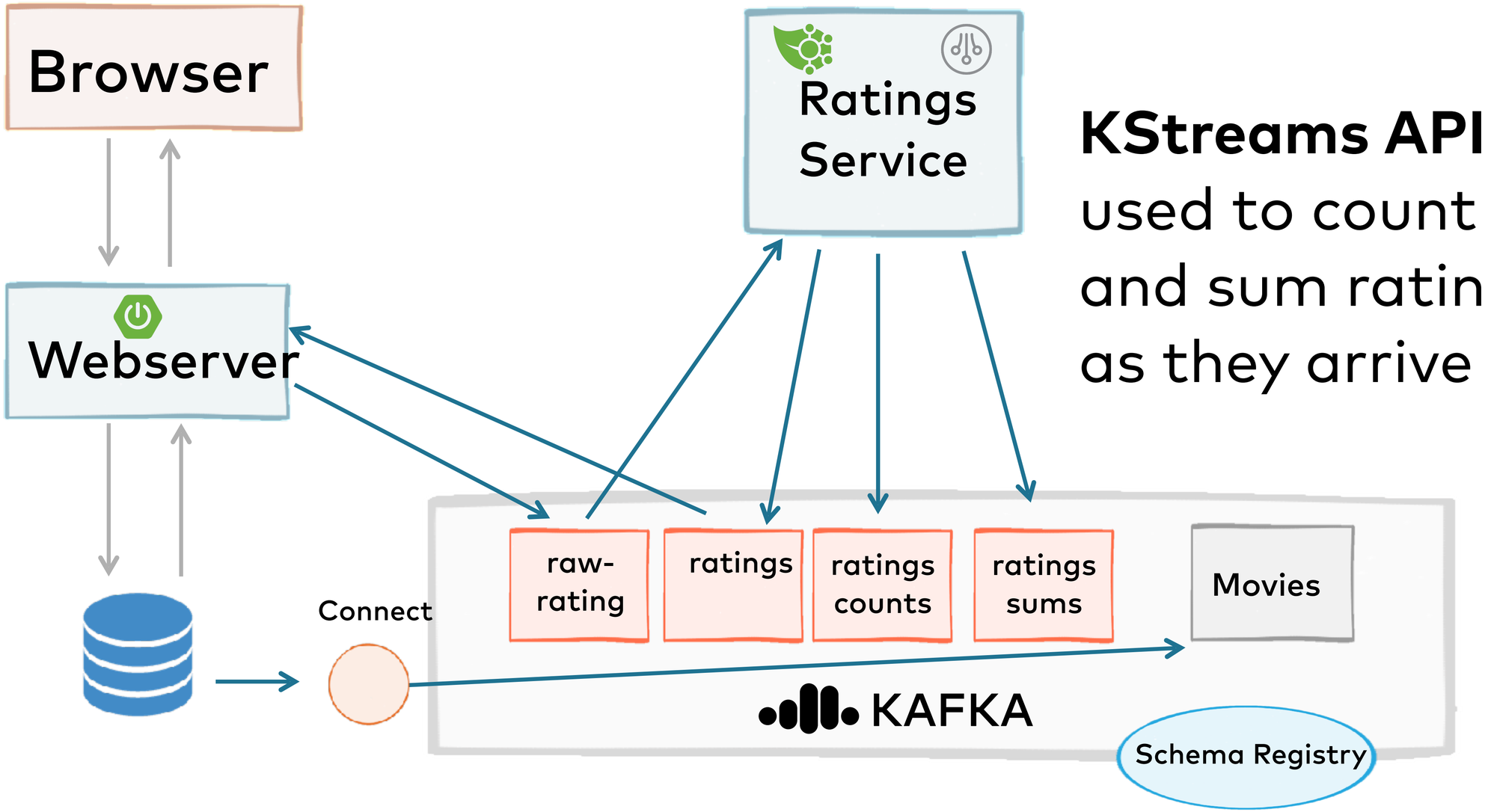
KStreams API used to count and sum ratings as they arrive

SCHEMA MANAGEMENT



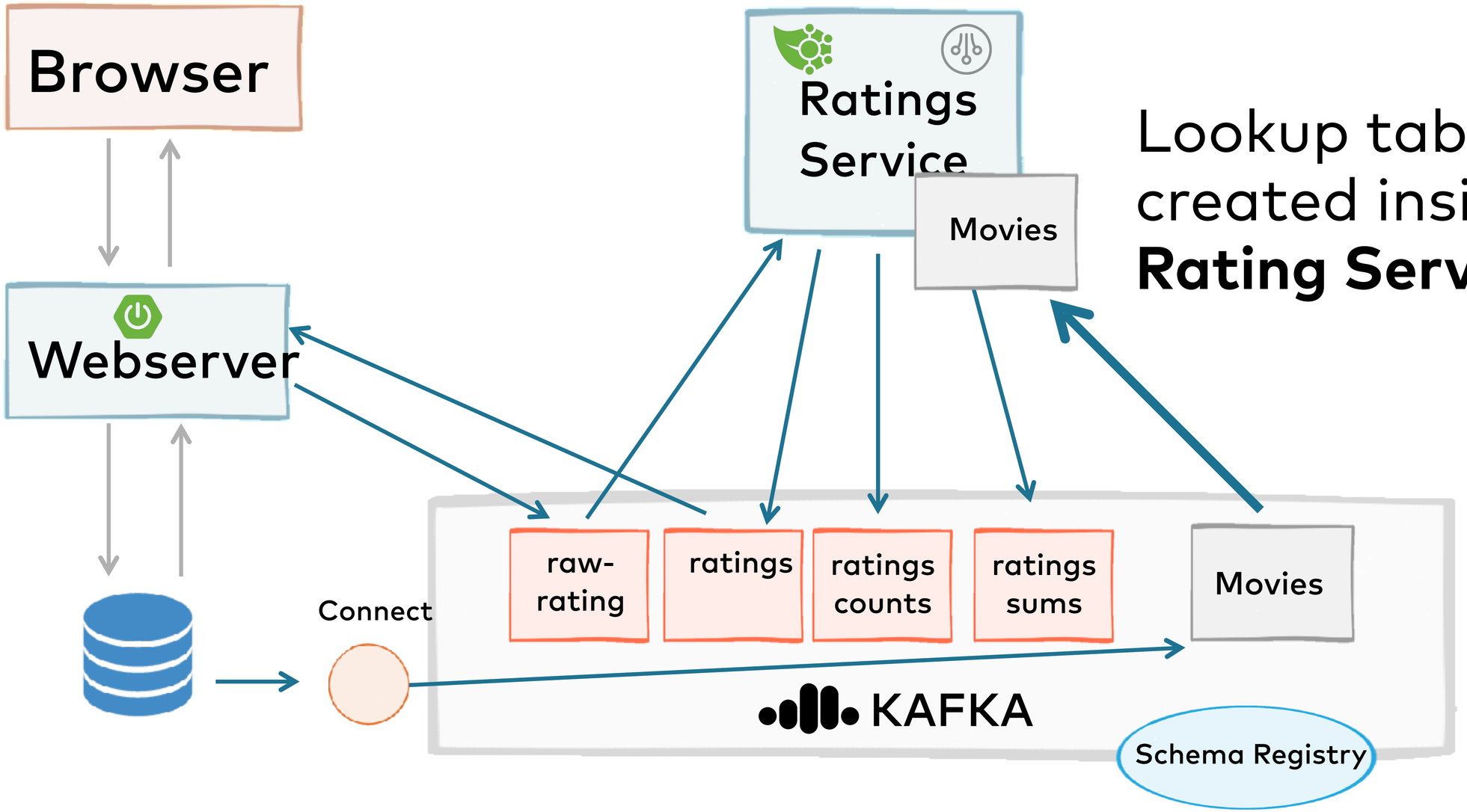
KStreams API used to count and sum ratings as they arrive

ACCESS LEGACY RELATION DATA



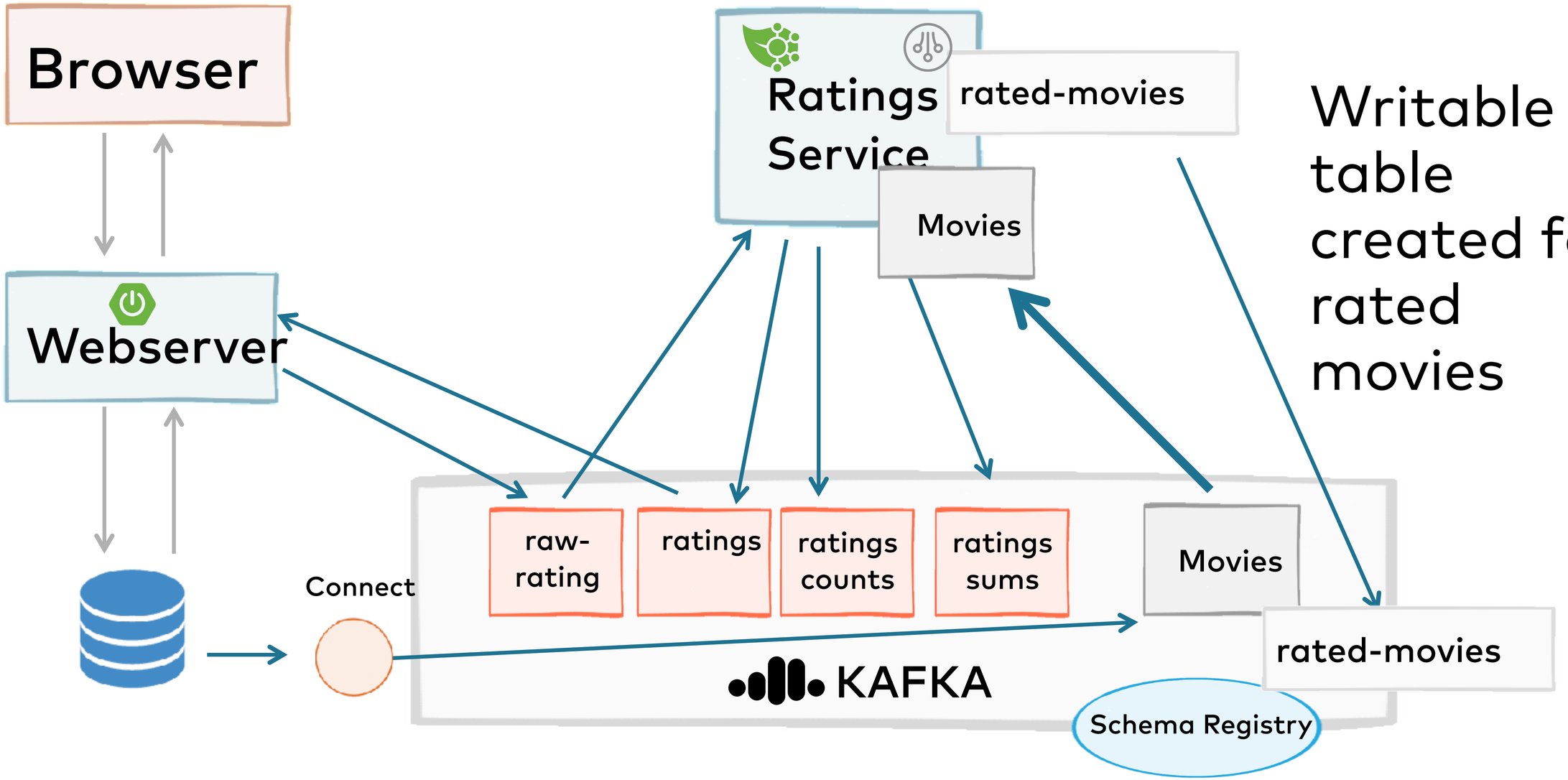
KStreams API used to count and sum ratings as they arrive

MATERIALIZE TABLES INSIDE THE APP

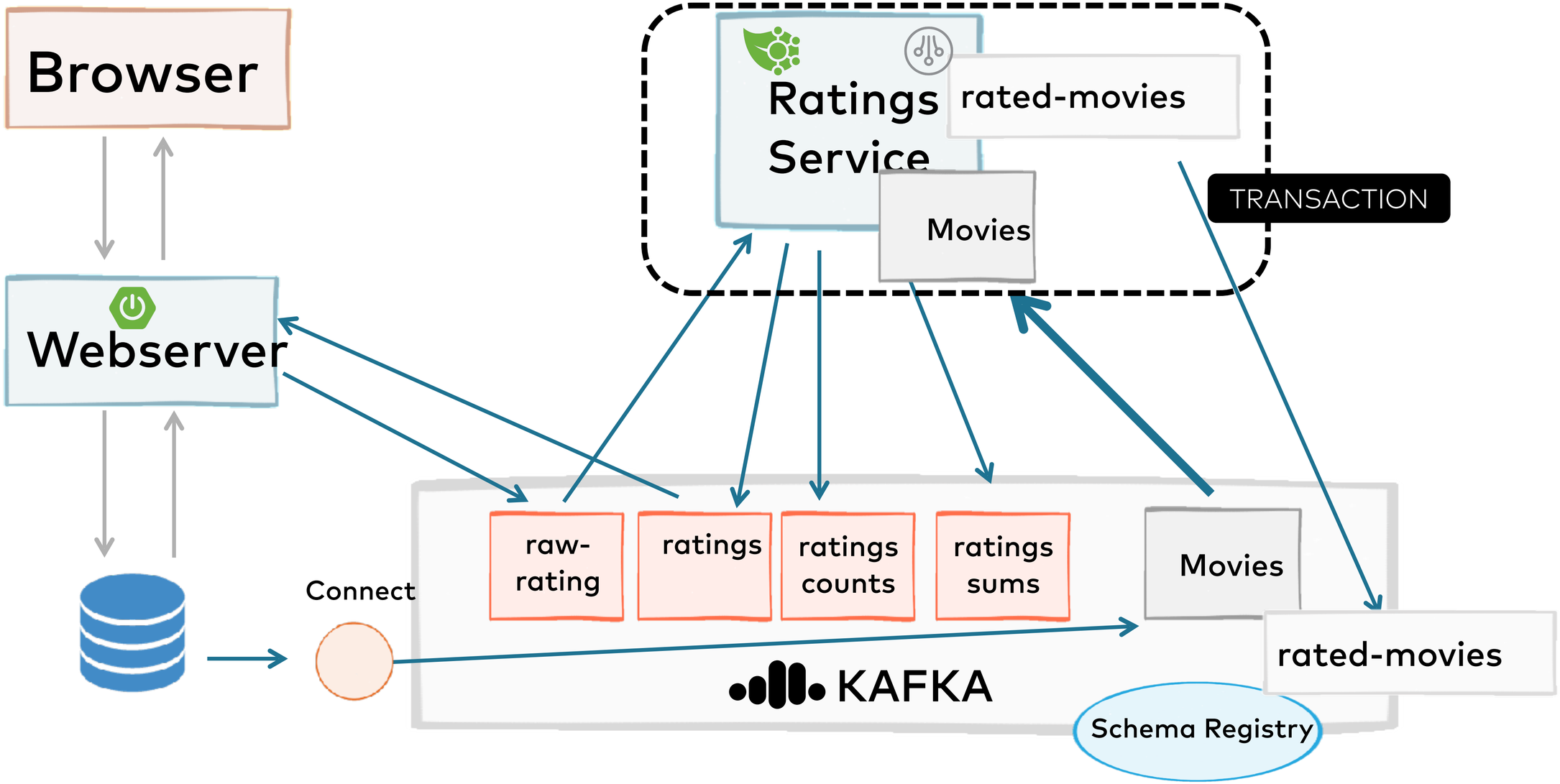


Lookup table created inside the **Rating Service**

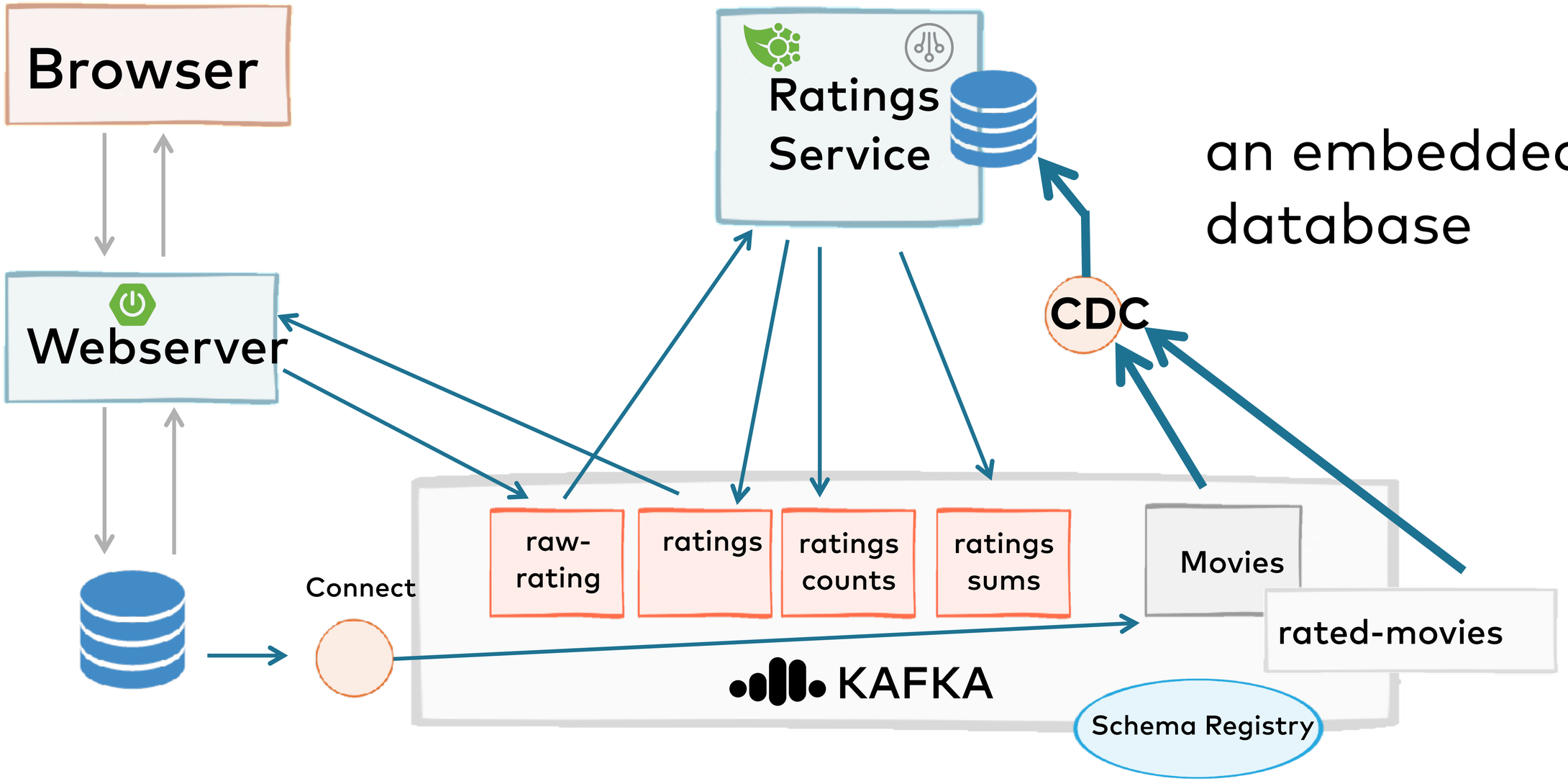
CREATE A NEW TABLE, PERSIST IT TO THE LOG



CREATE A NEW TABLE, PERSIST IT TO THE LOG



HYDRATE A MATERIALIZED VIEW



WHAT DID YOU LEARN TODAY?

- **Streaming services** improve on the traditional microservices pattern in a number of ways:
- Decouple ecosystems for **easier change**
- Evolve existing **databases into event streams** with Kafka Connect.
- Build on **asynchronicity first**. Add synchronicity where needed.

WHAT DID YOU LEARN TODAY?

- Remove the need for circuit breakers or other "overload" patterns.
- Quickly combine and operate on data from many different services.
- Embrace an immutable log that can be replayed.
- Keep many databases, in many independent services, in sync.
- Safely manage the evolution of data with the schema registry.

CONFLUENT

WANT TO LEARN MORE?

Workshops

Developing Event-Driven Microservices with Spring Boot, Confluent Cloud, Kotlin, and Java

Wed, Sep 2 at 10:00 AM EDT (90 minutes)

A microservices-based approach is now pretty universally recognized as the right way to develop modern applications. Technologies like Spring Boot and Spring Framework are ideal choices to develop and deploy your microservices (regardless of the language of choice, Java or Kotlin), with Apache Kafka® as the microservices communication layer. This approach provides highly scalable and robust communication between services that will allow you to unlock data in your existing monolithic and/or legacy systems via Kafka and e

These are the tools to help you achieve the very same [illegible] with well-known and well-thought architectural decisions with many successful projects behind them.

In this workshop, you will:

- Develop a small, functioning, microservices-based web application using Kafka Streams (using Java or Kotlin).
- Use Spring Boot to handle tedious and not-so-exciting tasks (like application configuration or container images generation).
- Use Confluent Cloud (that provides fully managed Apache Kafka as a service) to integrate the services.
- Use Spring for Kafka as the API between the two.

If you want to use Spring and Kafka together, apply the right patterns, avoid the wrong antipatterns, and generally get a good glimpse into a robust toolset for building modern Java web applications, look no further than this hands-on workshop.

This workshop is suited for Spring Boot (Java or Kotlin) developers who want to learn more about Apache Kafka®. We provide a code repository and Confluent Cloud coupon codes.

Sorry, registration is now closed.

If you've already reserved a seat, please [illegible] of the workshop to join.

HTTPS://GAMOV.DEV/SPRING20

Learn Kafka.

Start building with Apache Kafka at Confluent Developer.

Watch full version
https://gamov.dev/developer

developer.confluent.io

https://developer.confluent.io

CONFLUENT