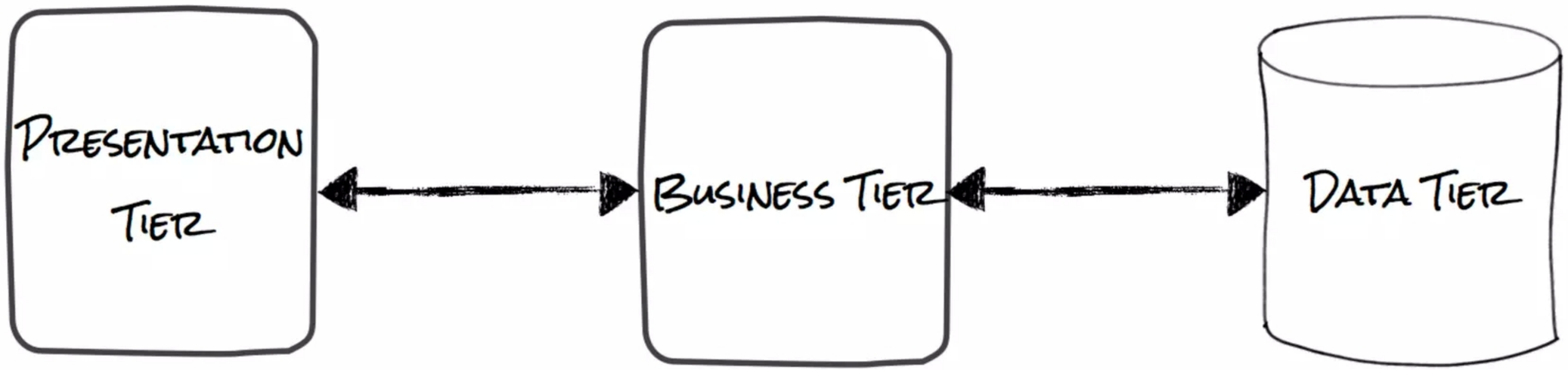
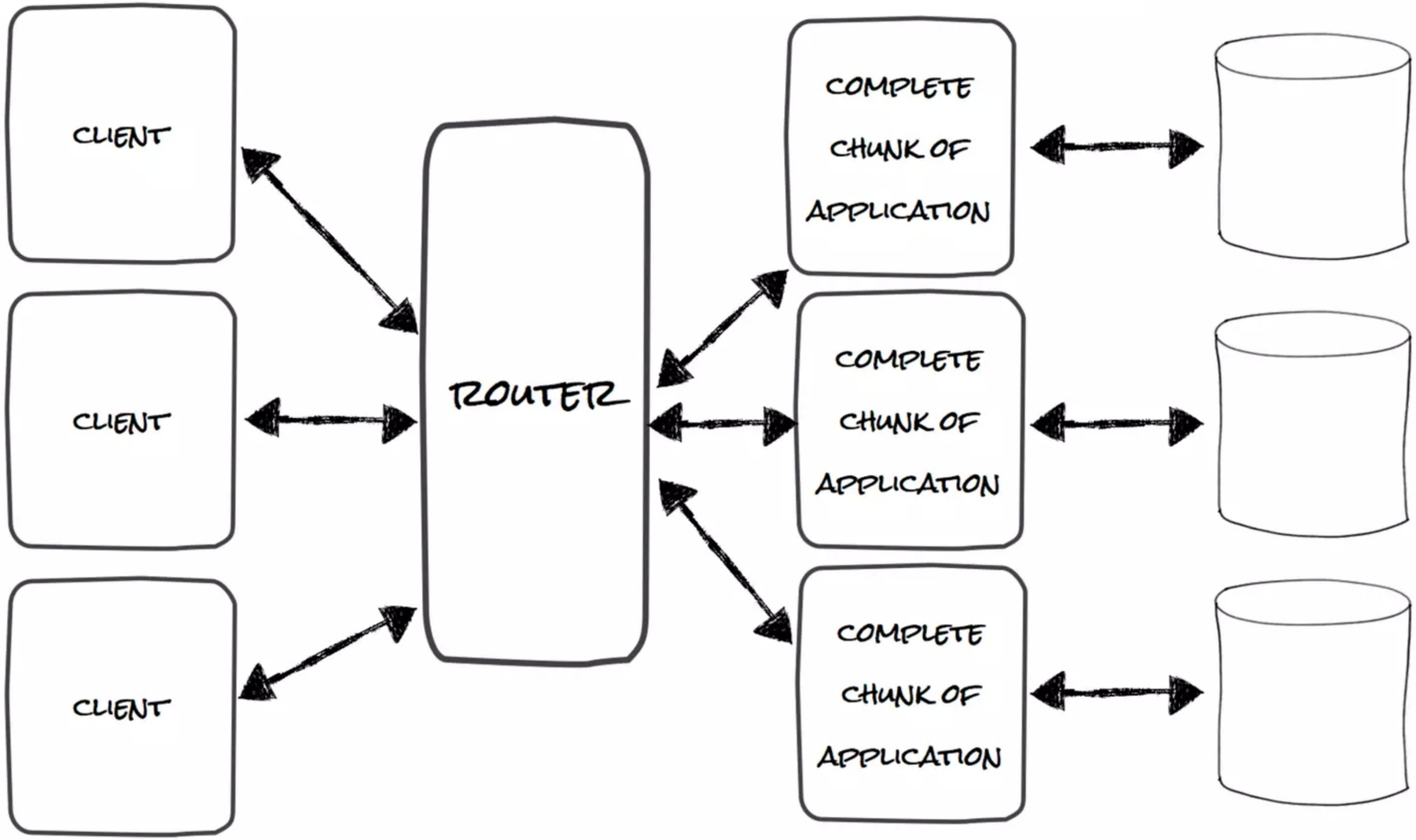
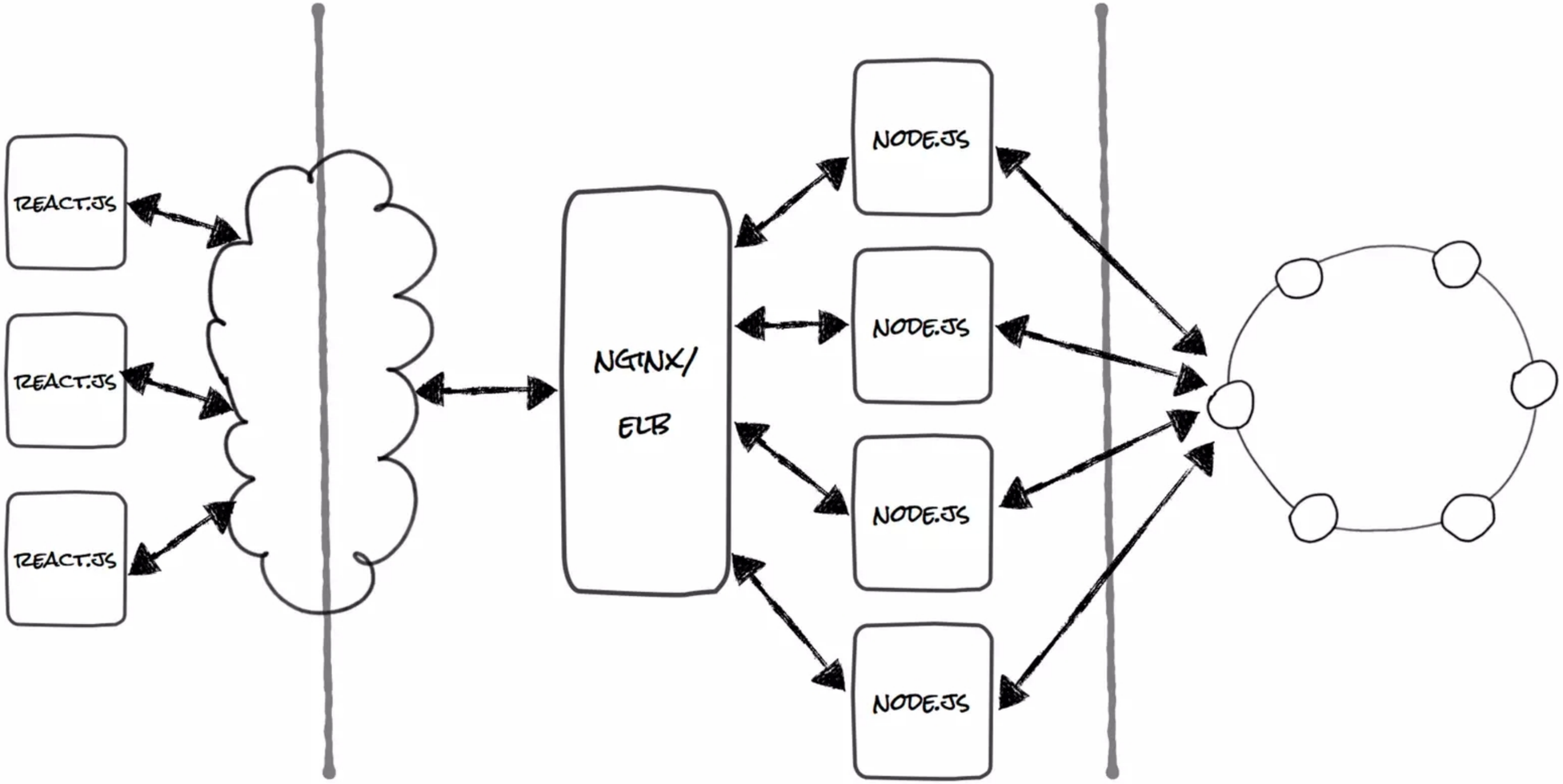
Four Streaming Data Architectures

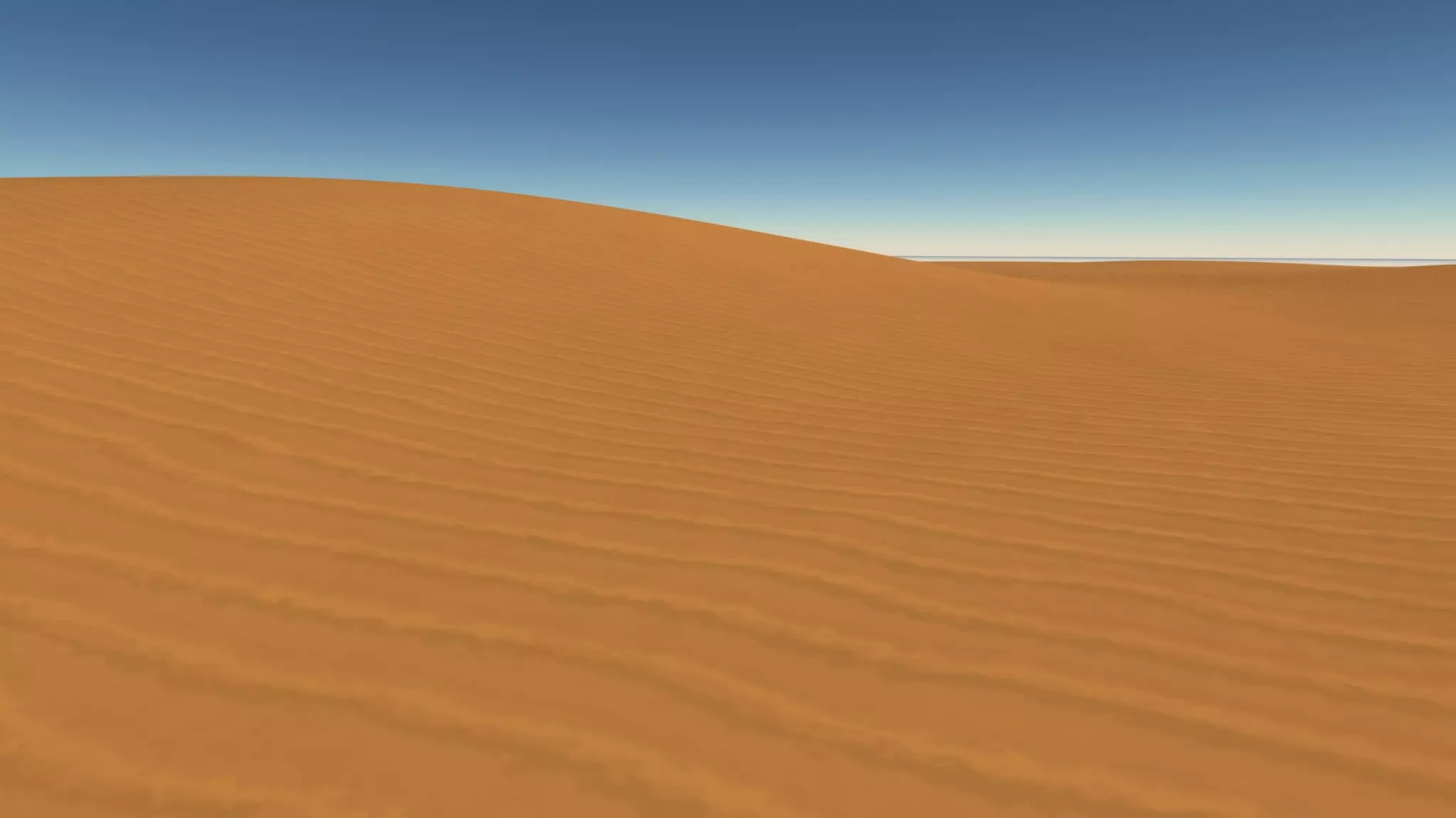
@tlberglund

The Old Story


Presentation Tier
Business Tier
Data Tier


CLIENT
CLIENT
CLIENT
ROUTER
COMPLETE CHUNK OF APPLICATION
COMPLETE CHUNK OF APPLICATION
COMPLETE CHUNK OF APPLICATION


REACT.JS
REACT.JS
REACT.JS
NGINX/
ELB
NODE.JS
NODE.JS
NODE.JS
NODE.JS



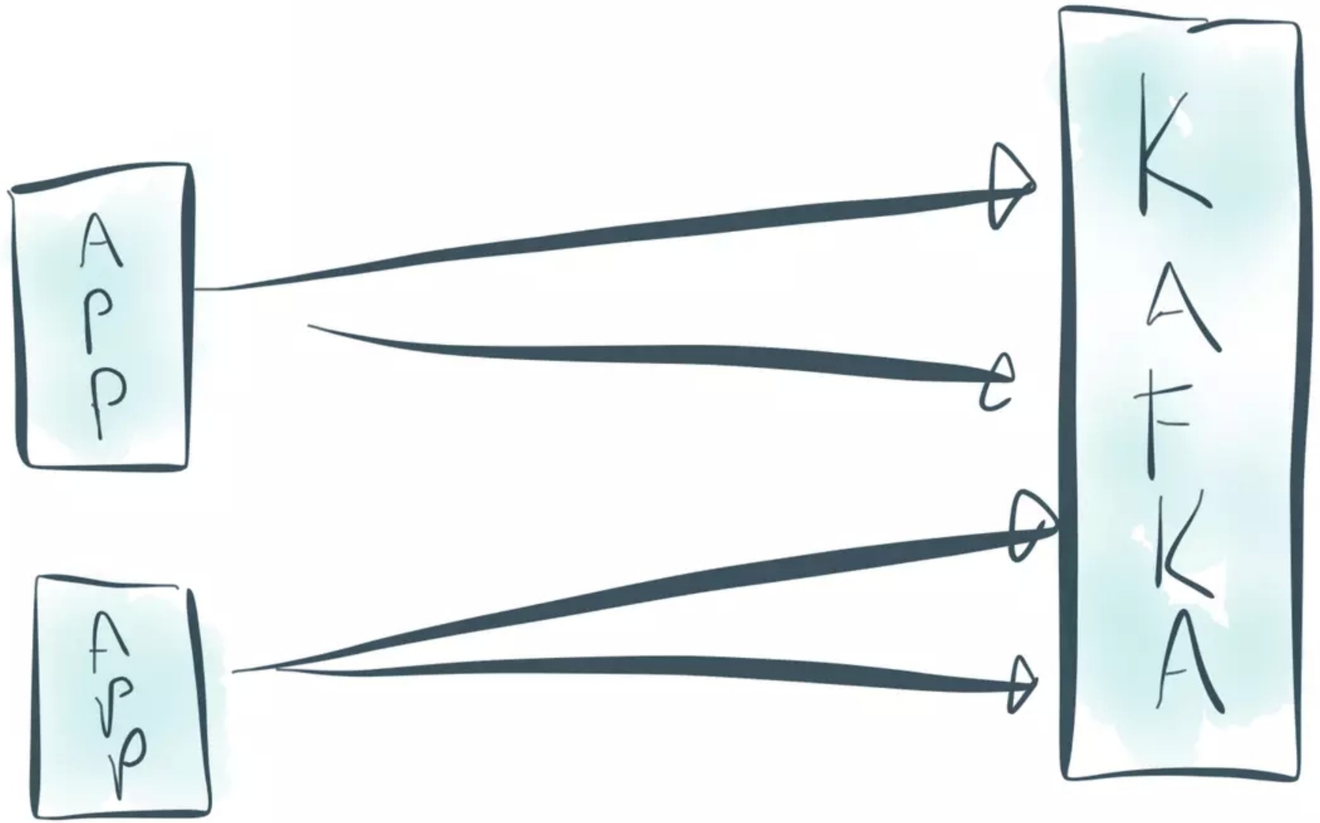
The Streaming Story

Maria-José Viñas, NASA Earth Science News Team.

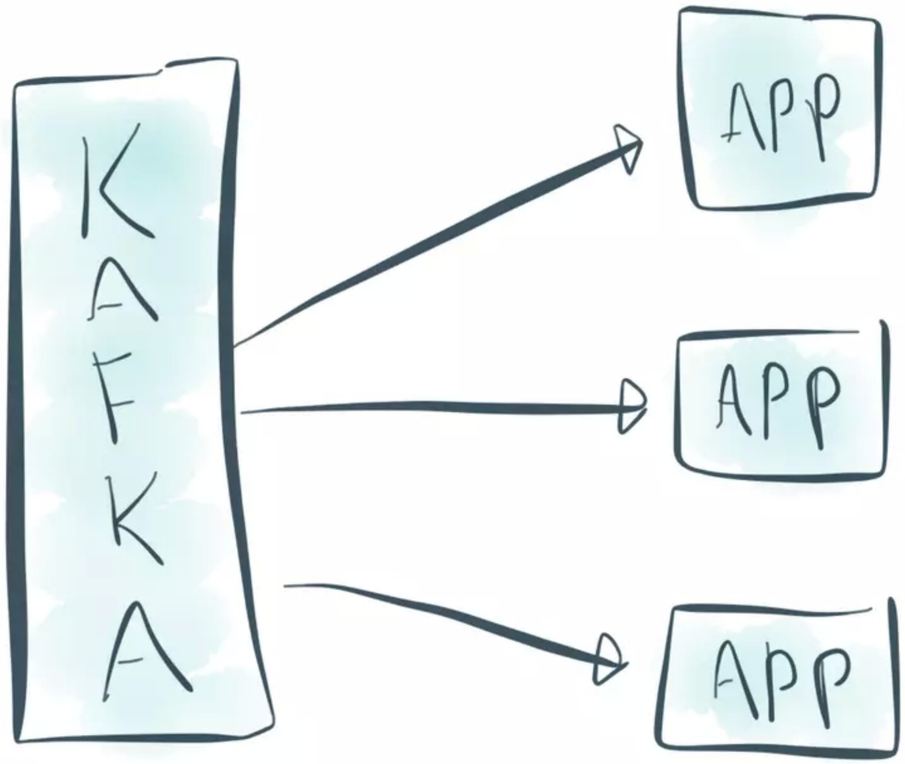
Image credit: NASA/JPL/UCSD/JSC

Kafka

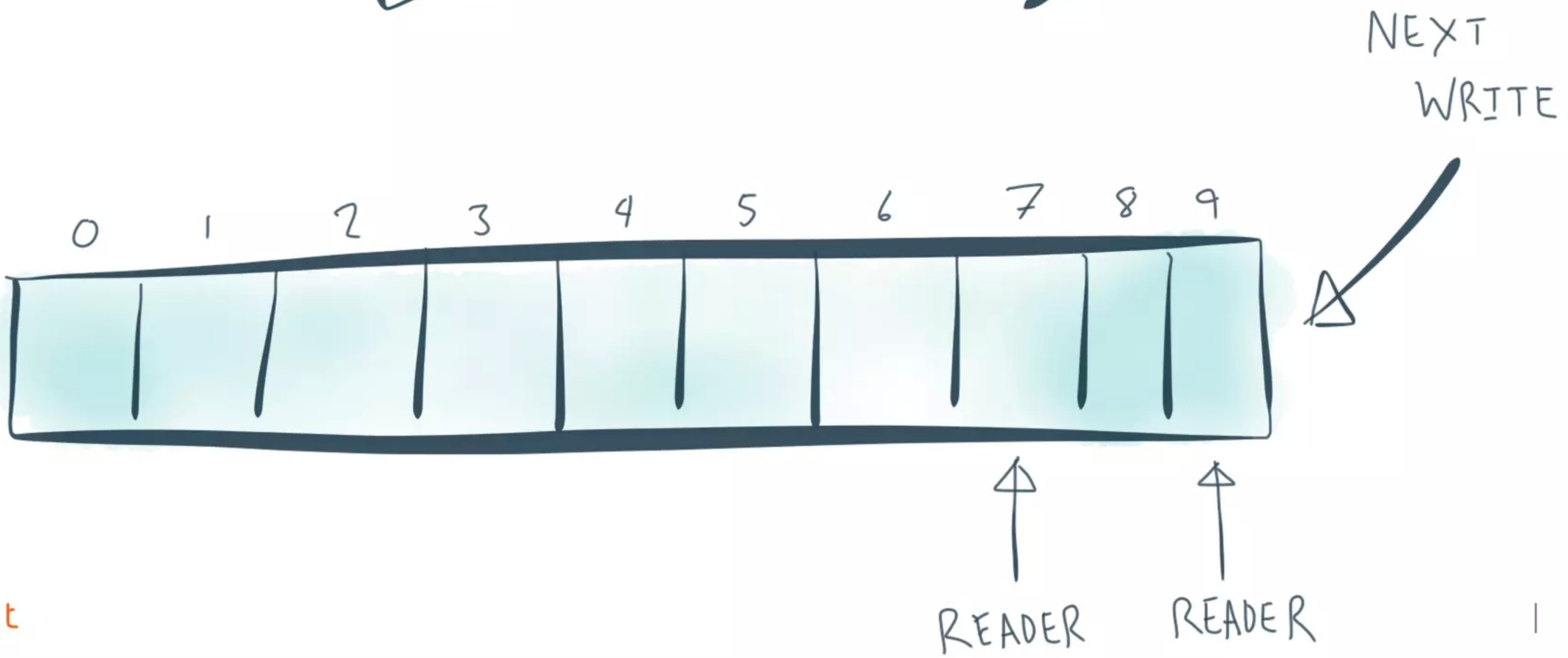
PRODUCER



CONSUMER



LOGS



Examples

Yelp

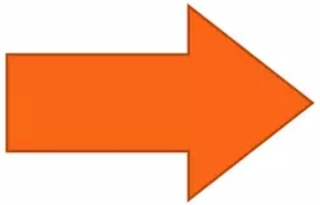
https://engineeringblog.yelp.com/2016/07/billions-of-messages-a-day-yelps-real-time-data-pipeline.html

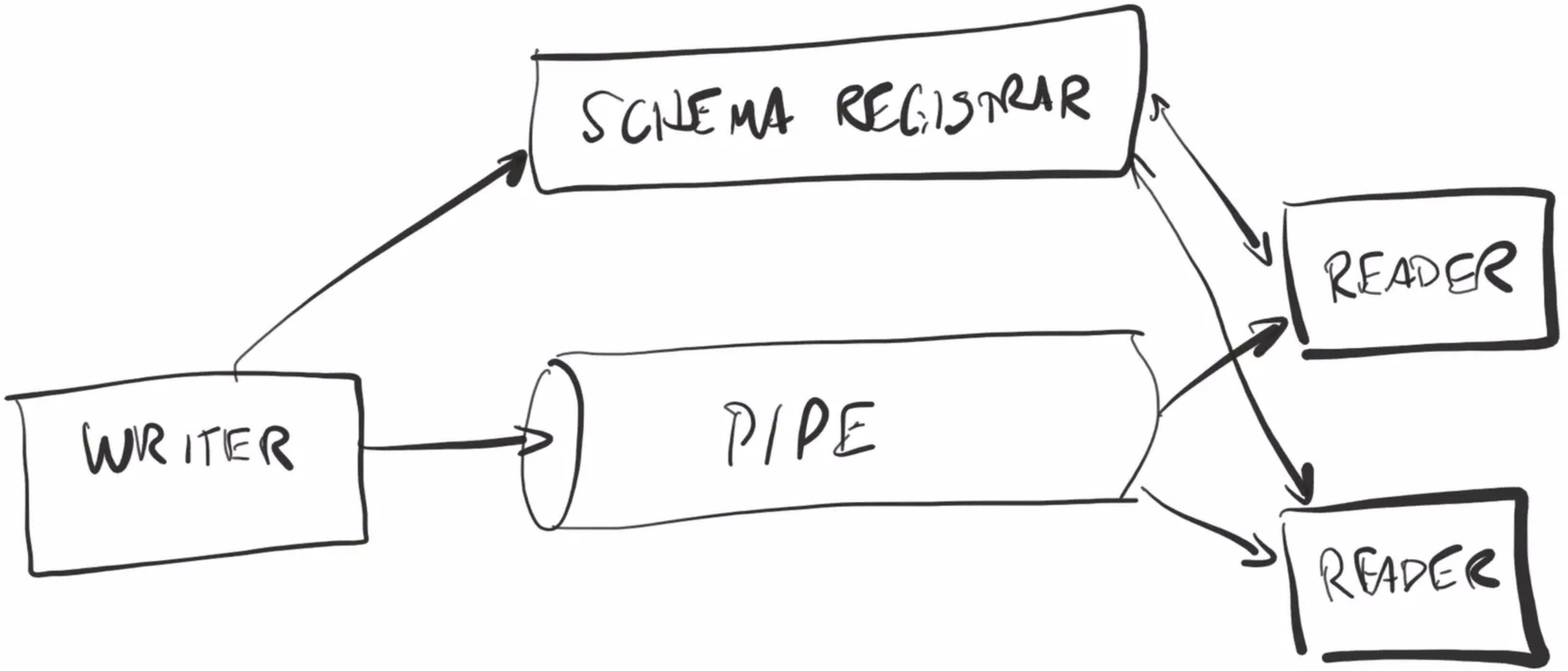
Justin C., Software Engineer

(May, 2016)

A Streaming Backbone

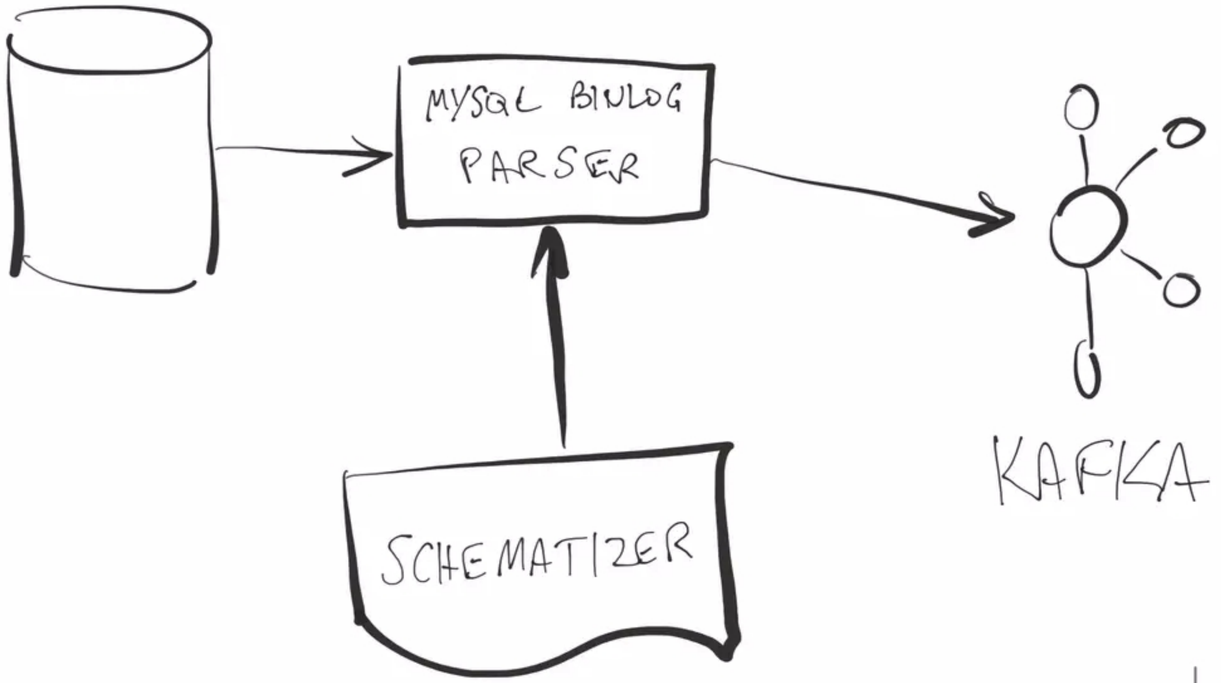
- 150 production services
- Metcalf’s Law
- REST scales poorly
- N+1 problem and bulk data services




SCHEMA REGISTRAR
WRITER
PIPE
READER
READER

Streaming Databases

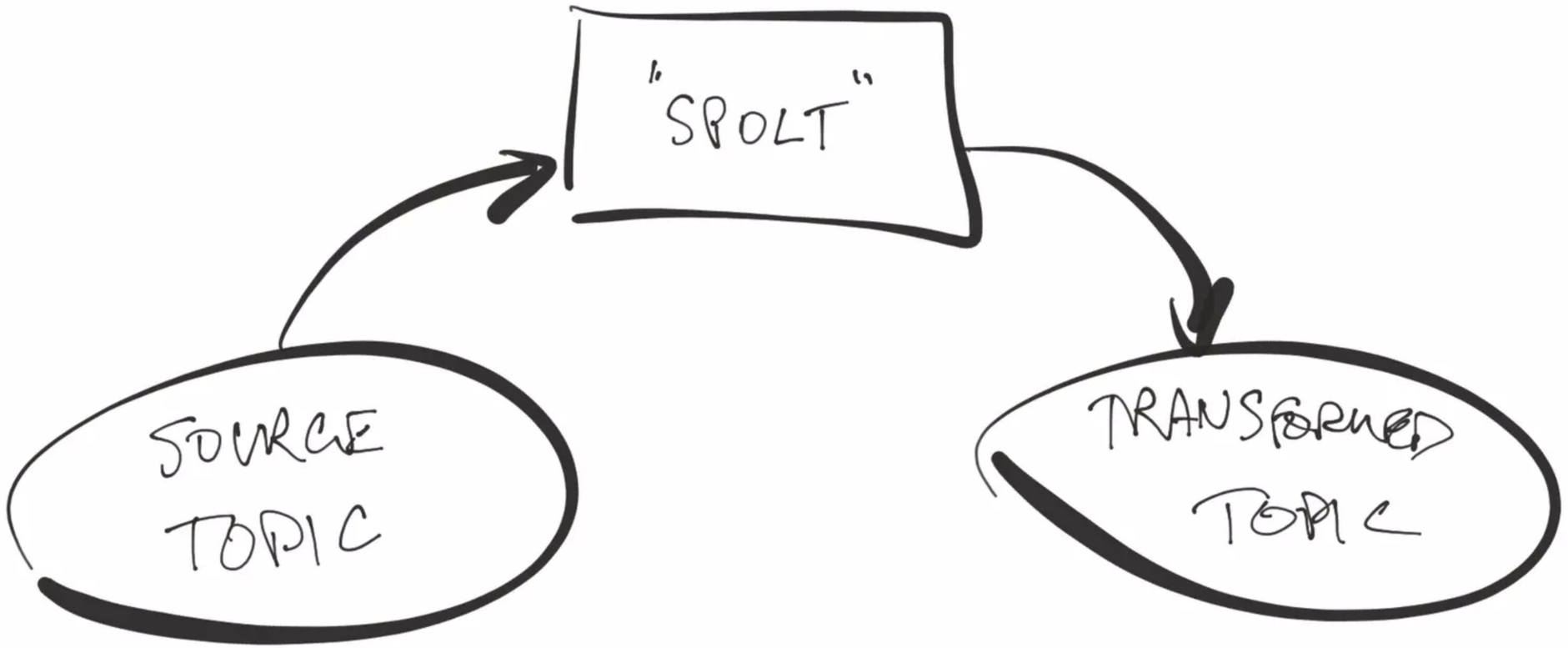
- Stream-table duality
- But some services still want databases
- Individual services use MySQL
- Millions of rows across services



Computing Streams

- In 2010, commitment to MapReduce was strong
- That fun was soon over
- Want to capitalize on schema translation tools
- Want near-real-time results
- Storm didn’t support enough Python
- Wrote their own!

PaaStorm Instance



Yelp Conclusions

- Streaming pipeline ensures evolvability
- Stream/Table Duality
- Schema matters
- Yelp often ahead of the game!

LINE Corp

https://engineering.linecorp.com/en/blog/detail/80

Goals

- Unified data delivery pipeline between systems
- Replace a legacy, bespoke system for processing background tasks
- Had to pick a framework

Option: Apache Samza™

Option: Apache Kafka™ Streams

https://www.confluent.io/blog/introducing-kafka-streams-stream-processing-made-simple

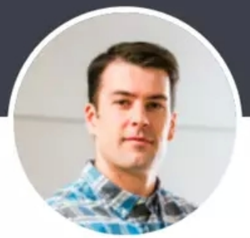
English | Deutsch

confluent

Product Solutions Services Resources About Blog Download

STREAM PROCESSING

Introducing Kafka Streams: Stream Processing Made Simple



Jay Kreps

Streaming API for Kafka

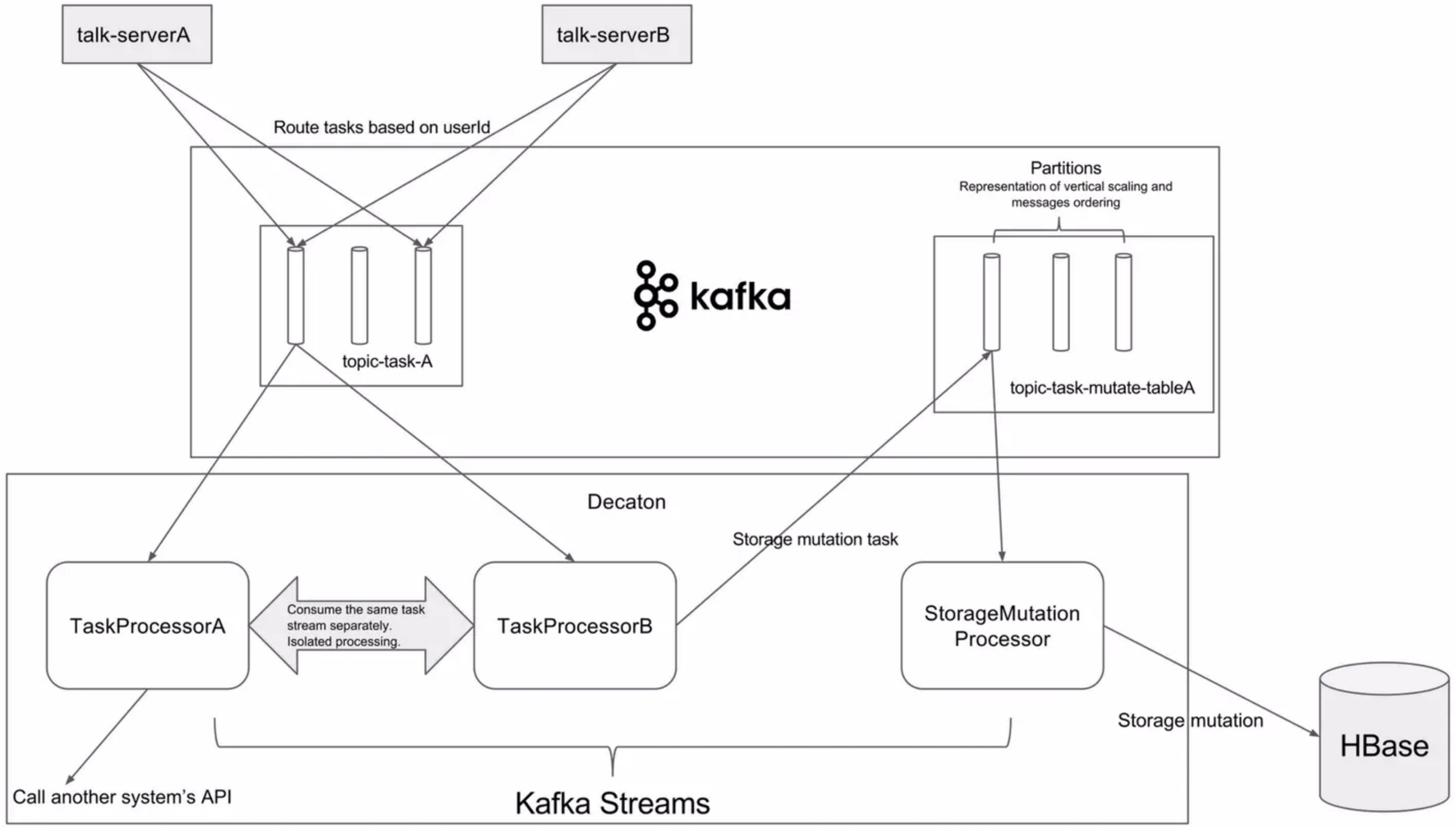
```
KStreamBuilder builder = new KStreamBuilder();

// Read the input Kafka topic into a KStream instance.
KStream<byte[], String> textLines = builder.stream("TextLinesTopic");

// Convert to upper case (:: is Java 8 syntax)
KStream<byte[], String> uppercasedWithMapValues = textLines.mapValues(String::toUpperCase);

// Write the results to a new Kafka topic called "UppercasedTextLinesTopic".
uppercasedWithMapValues.to("UppercasedTextLinesTopic");
KafkaStreams streams = new KafkaStreams(builder, streamsConfiguration);
streams.start();
```

Stateful Transformations

Loopback Replicator

- One big input topic of all TalkOperations (1M messages/sec)
- Some services want a limited view of these
- Simple filtering

Decaton

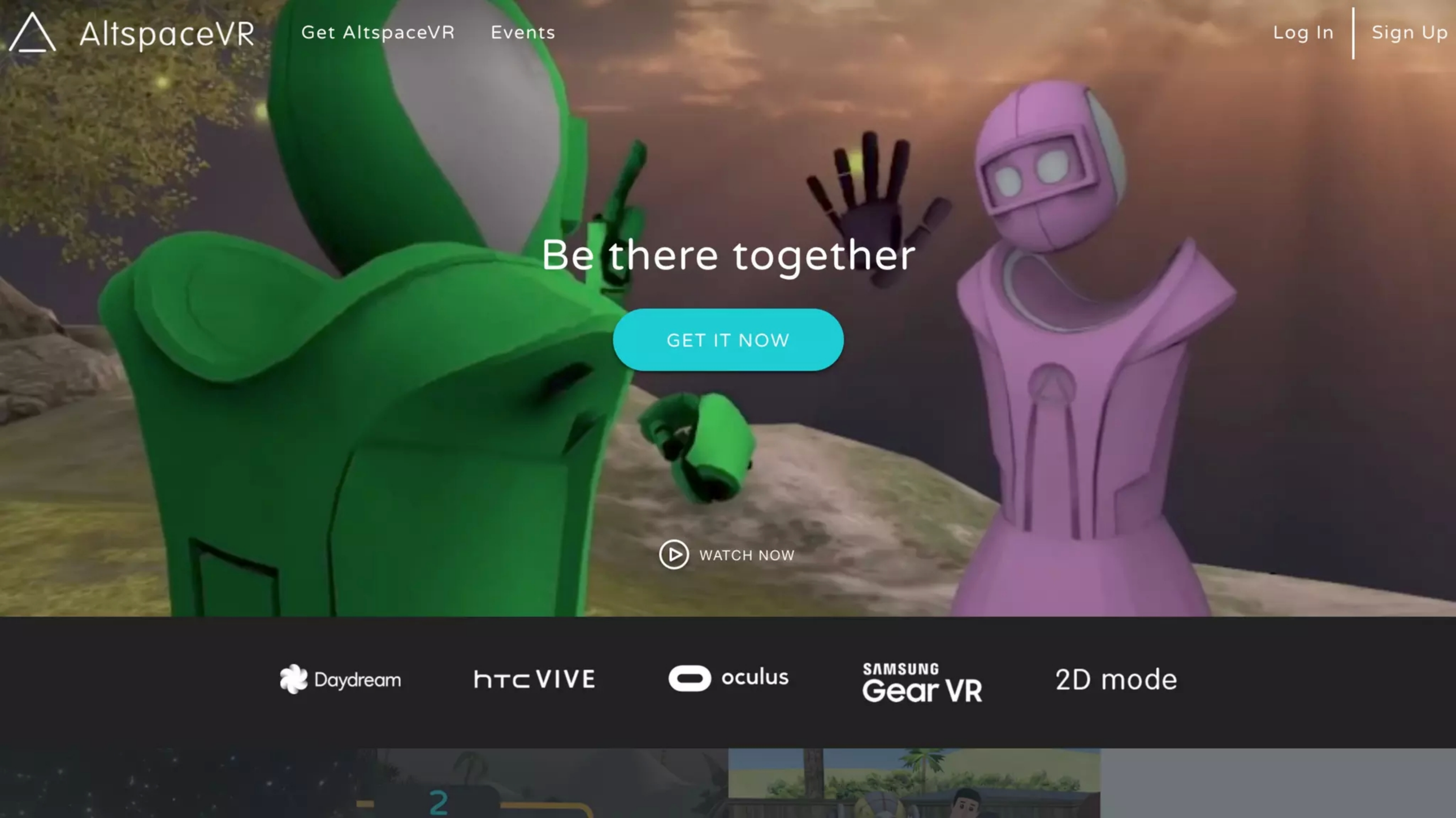


Source: LINE Corp

LINE Conclusions

- Classical Kafka Streams implementation
- Better to “buy” than build

Altspace VR

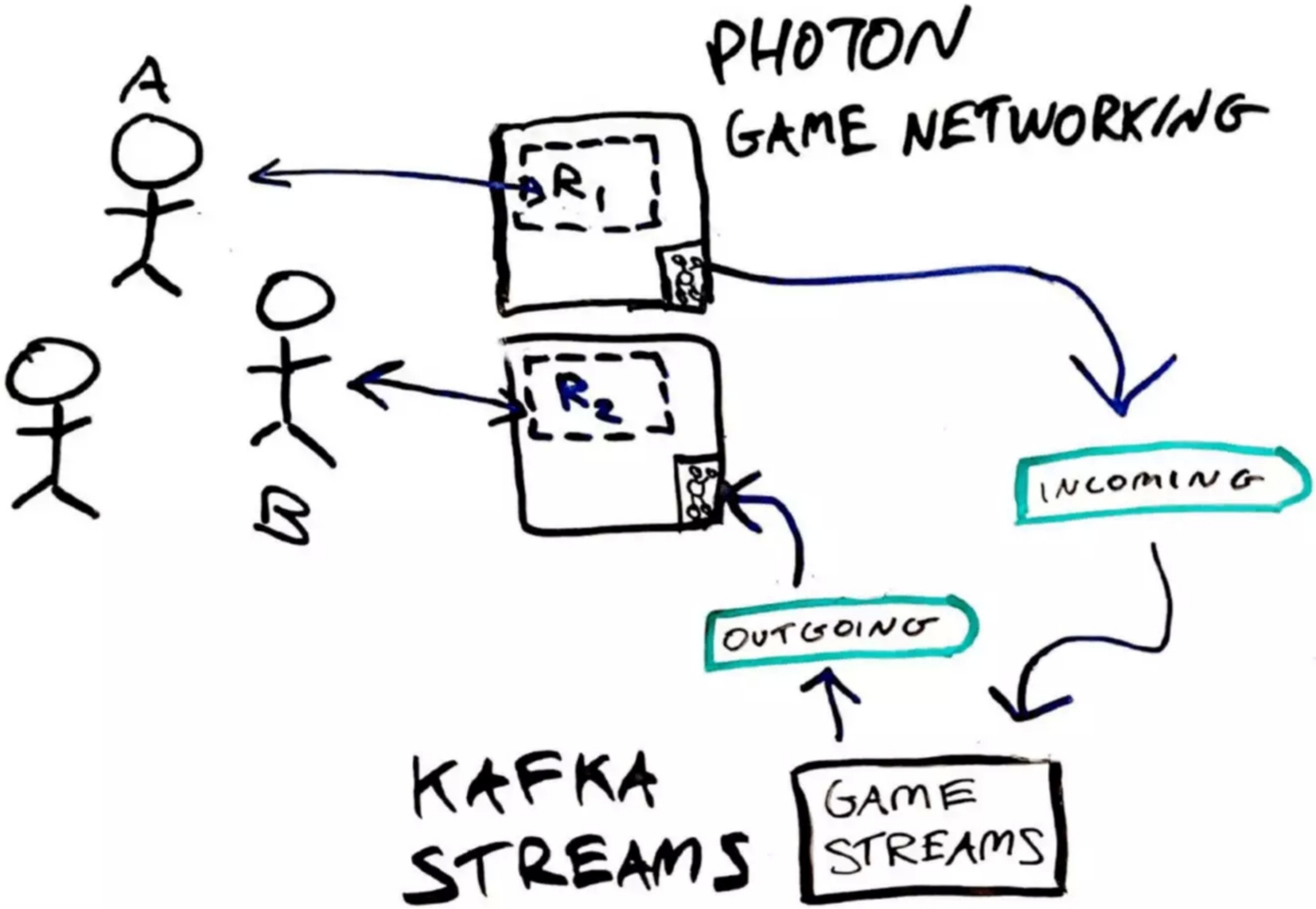

AltspaceVR
Get AltspaceVR
Events
Log In
Sign Up
Be there together
GET IT NOW
WATCH NOW
Daydream
htc VIVE
oculus
SAMSUNG
Gear VR
2D mode


NFL
1ST & 10
SEA 7
2nd :49 :19
1st & 10


JASH VR
REGGIE WATTS
LIVE IN VIRTUAL REALITY

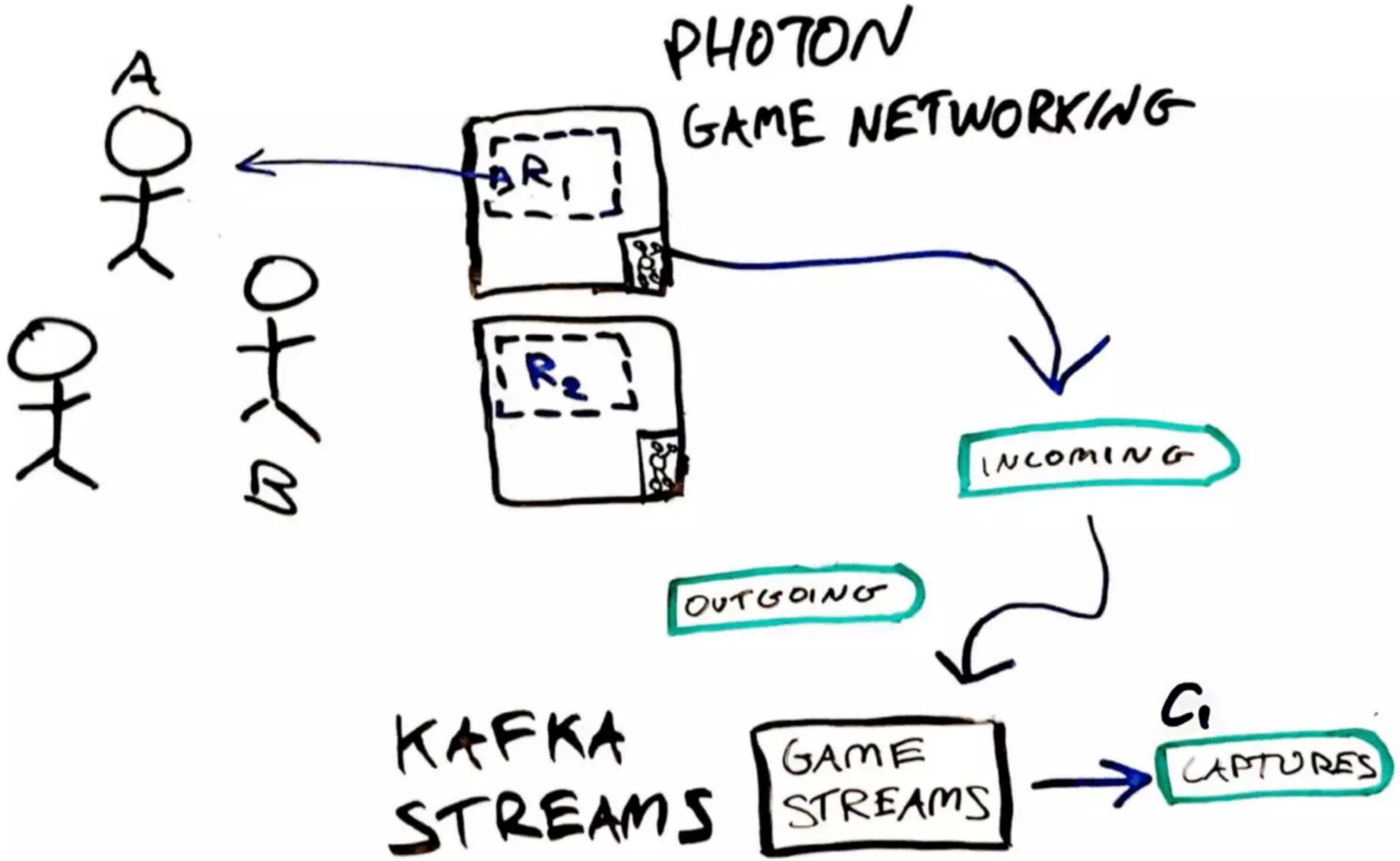
Altspace Architecture

- Photon game engine
- Real-time play
- Capture for playback
- An entirely classical stream processing system



"Mirror User A to room R2"

Source: Altspace

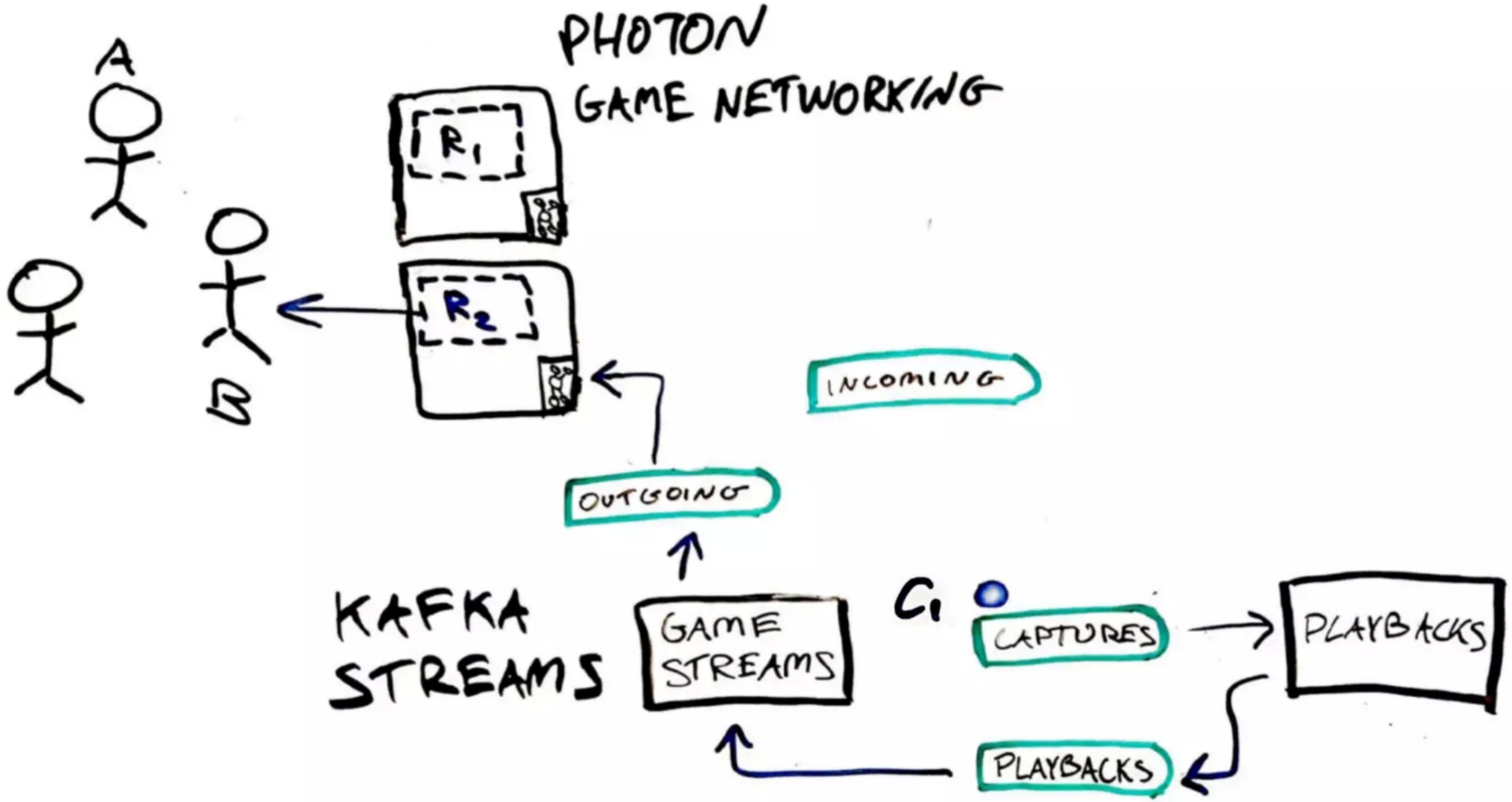


6 months later: “Capture User A”

Source: Altspace

Playbacks

- Source capture data from Kafka or S3
- “Produce” it in a timed fashion



"Playback capture to room R_2"

Source: Altspace

The Dream Stream

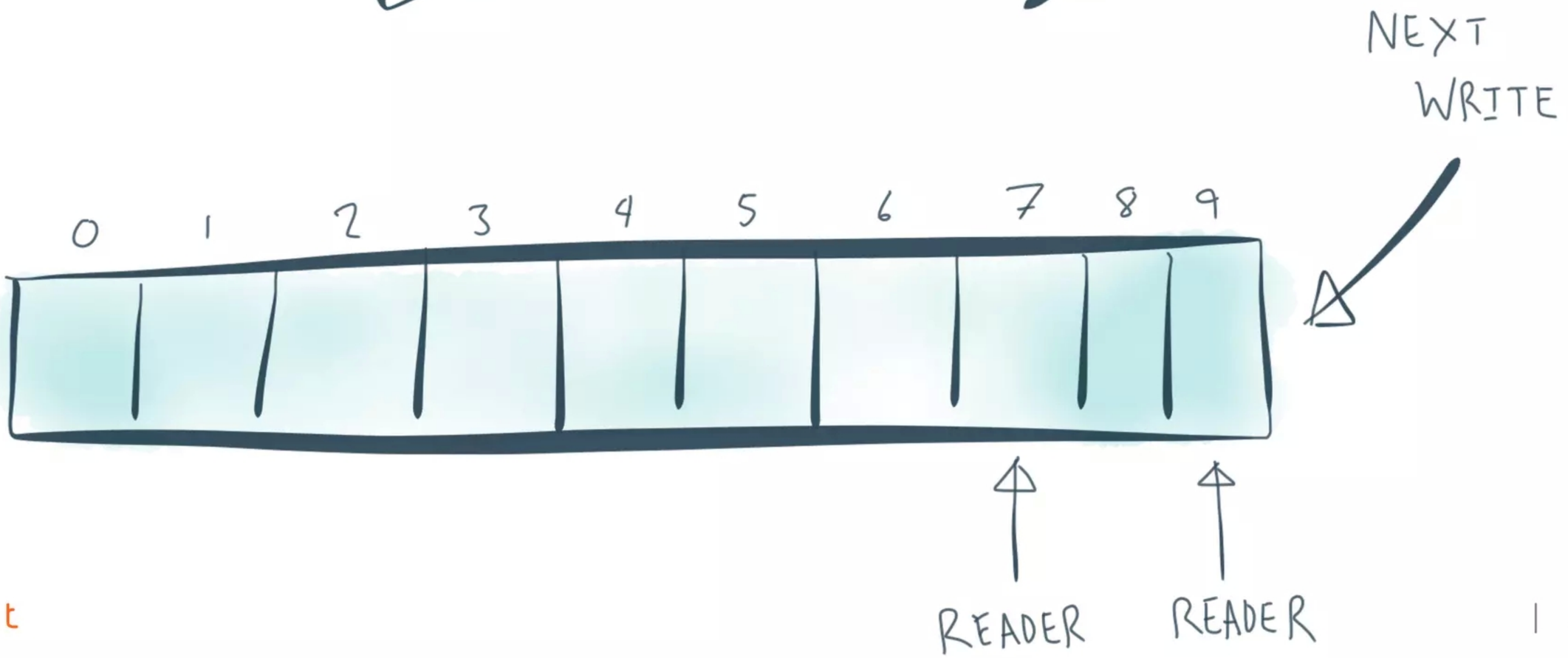
STREAMING PLATFORM

1 PUB/SUB

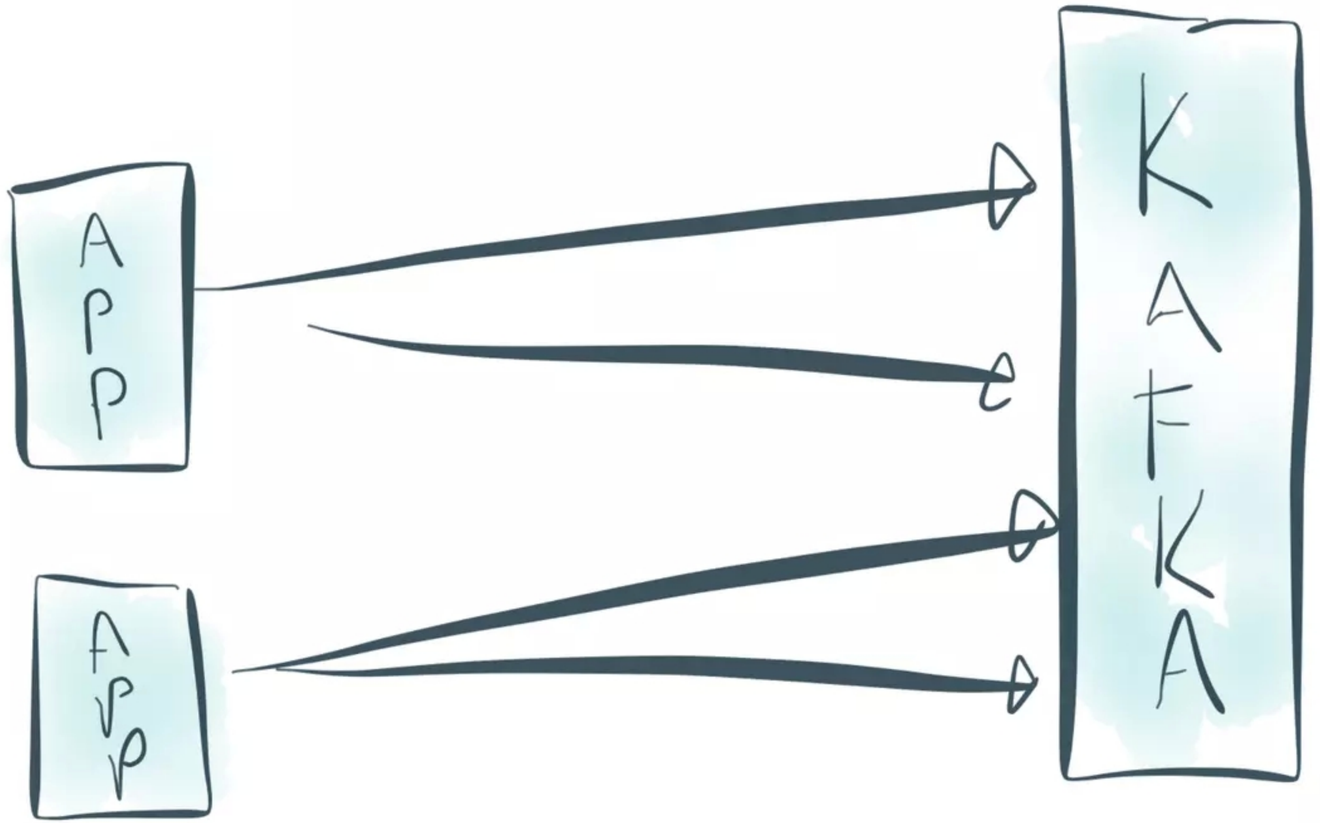
2 STORE

3 PROCESS

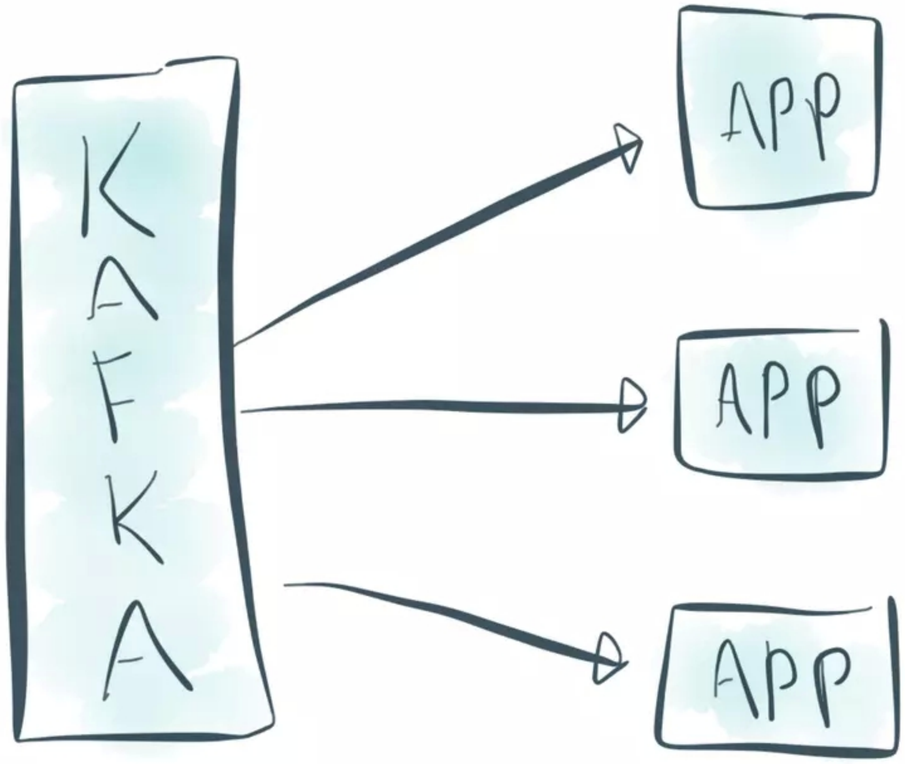
LOGS



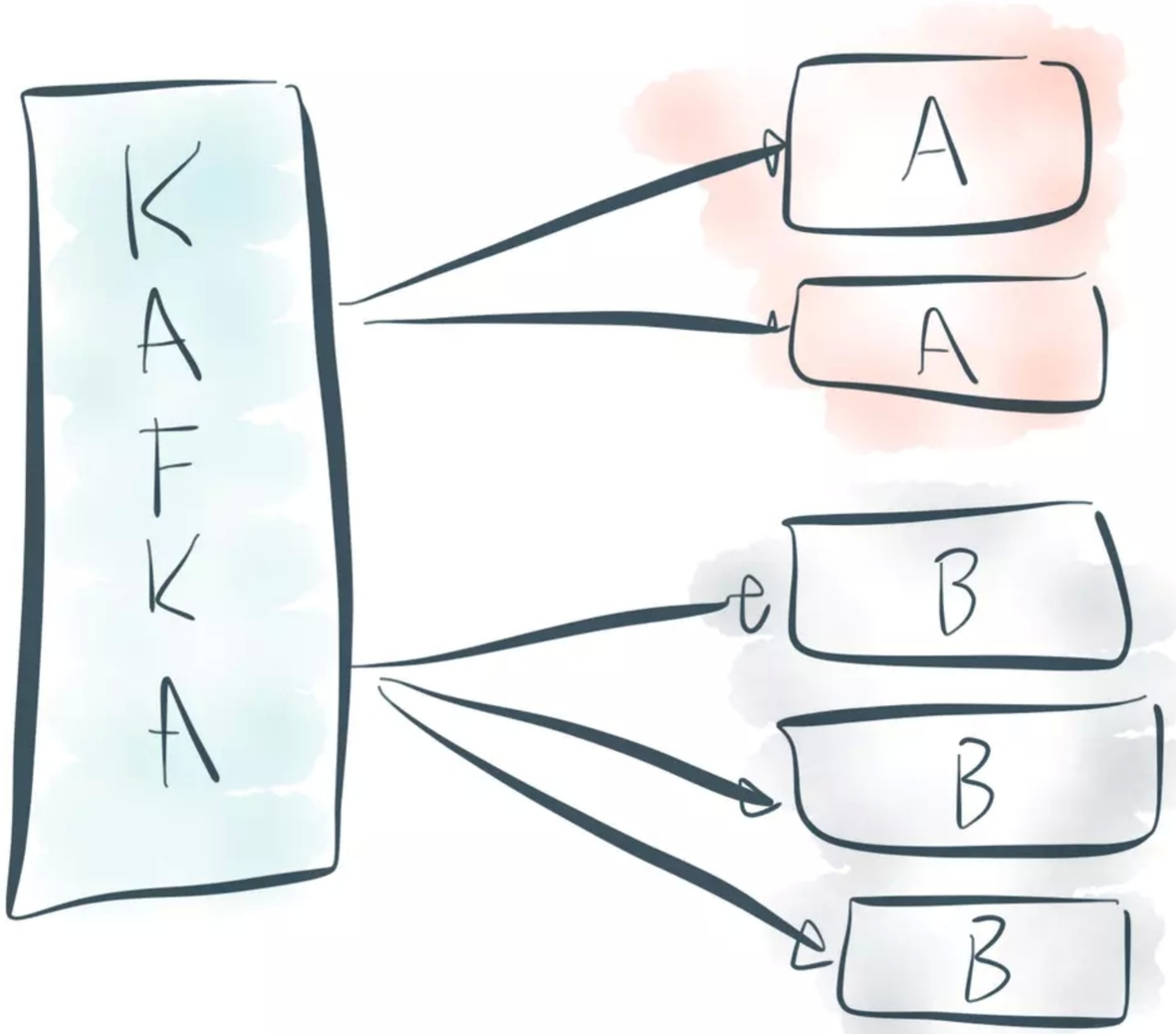
PRODUCER



CONSUMER



GROUPS

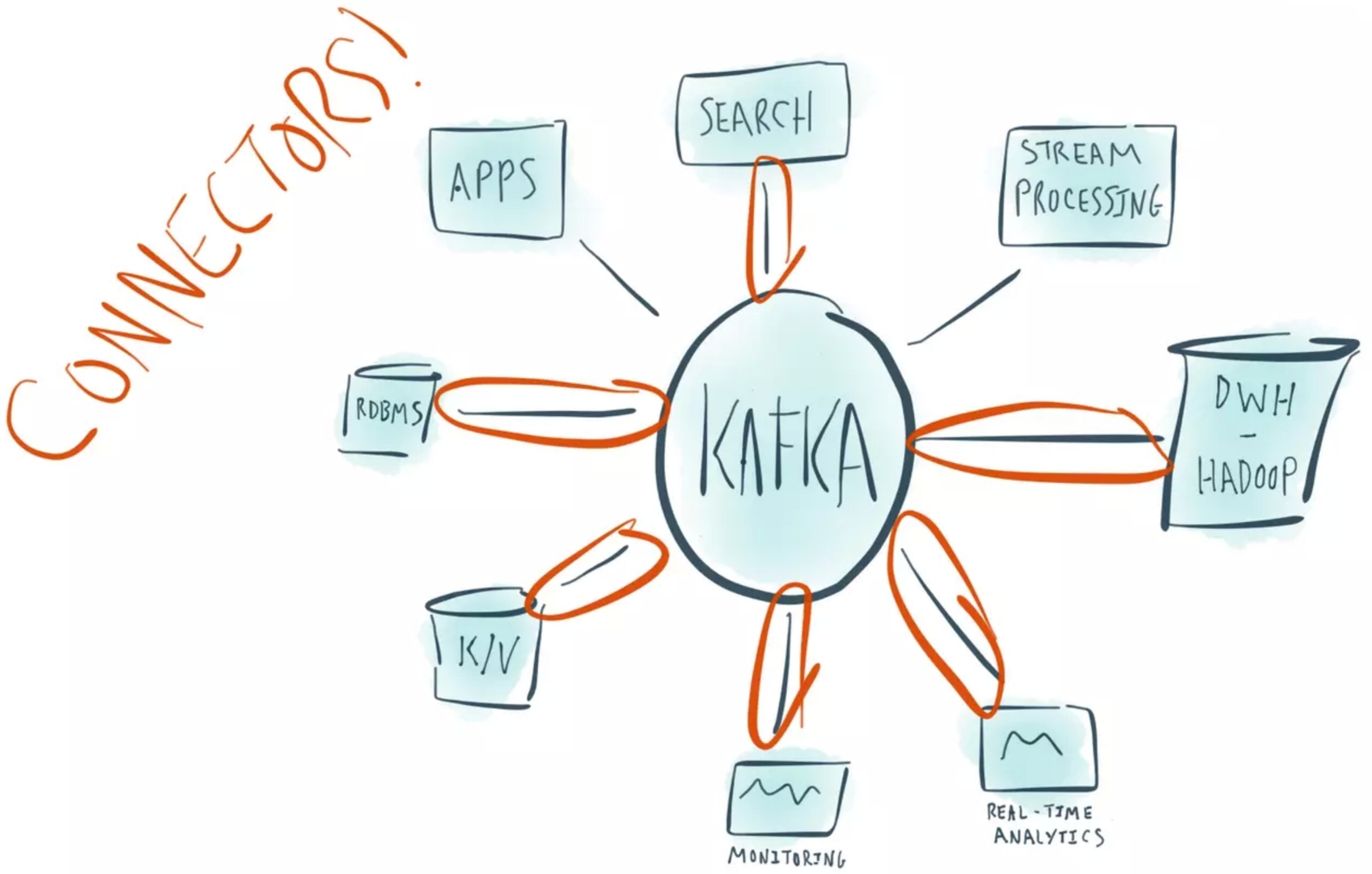


KAFKA CONNECT

KAFKA CONNECT DOES THE HARD WORK

1. SCALE OUT
2. FAULT TOLERANCE
3. CENTRAL MANAGEMENT
4. SCHEMA PROPAGATION

STREAMING PLATFORM



STREAMING PLATFORM

1 PUB/SUB

2 STORE

3 PROCESS

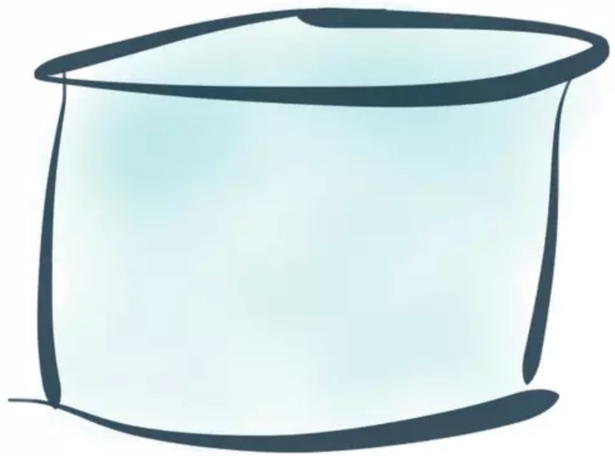

WHY
STORE?

SCALABILITY OF A FILESYSTEM

- HUNDREDS OF MB/SEC THROUGHPUT
- MANY TB PER SERVER
- COMMODITY HARDWARE
- O(1) WRITES

GUARANTEES OF A DATABASE

- STRICT ORDERING
- PERSISTENCE



DISTRIBUTED BY DESIGN

- REPLICATION
- FAULT TOLERANCE
- PARTITIONING
- ELASTIC SCALING

STREAMING PLATFORM

1 PUB/SUB

2 STORE

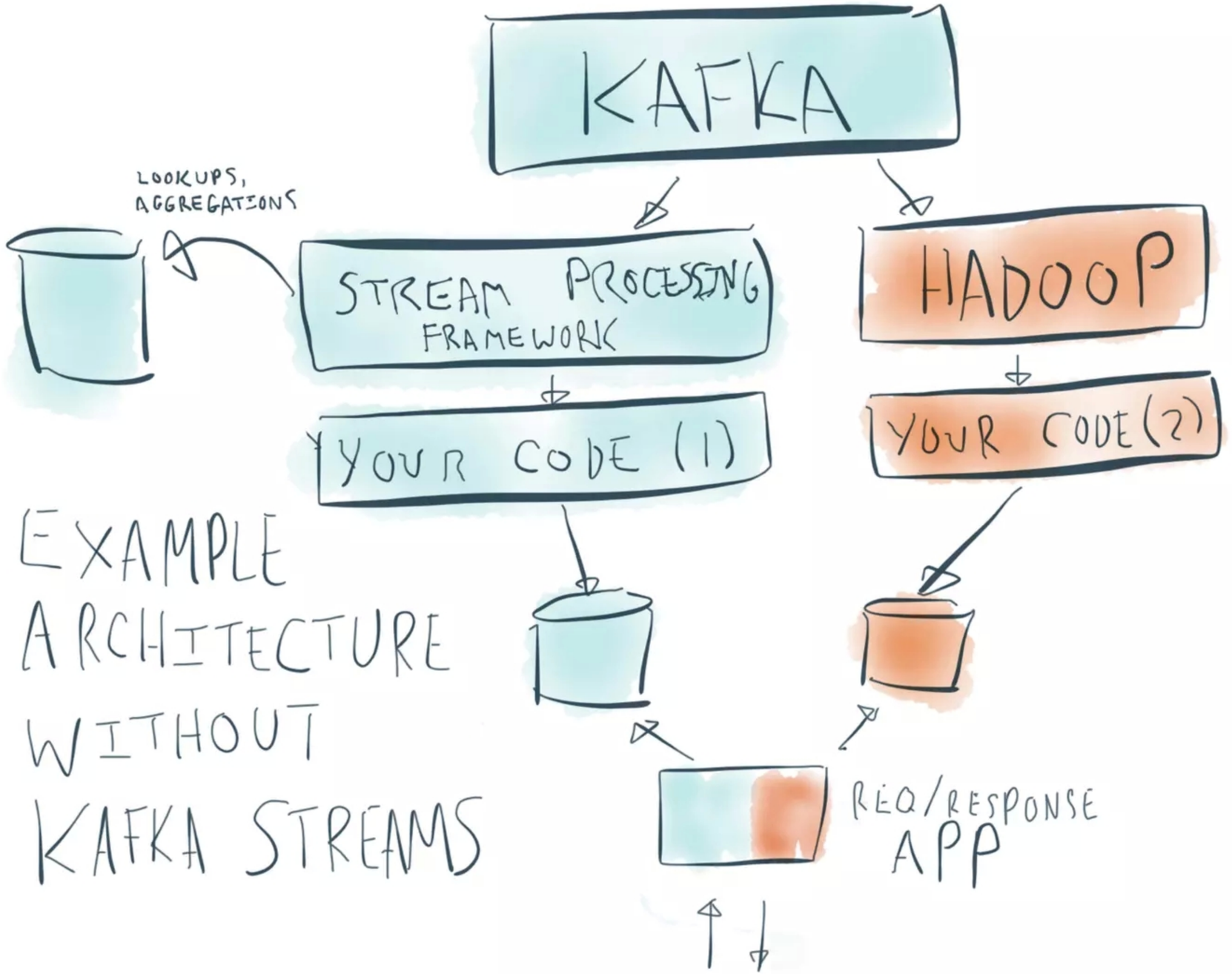
3 PROCESS

KAFKA STREAMS

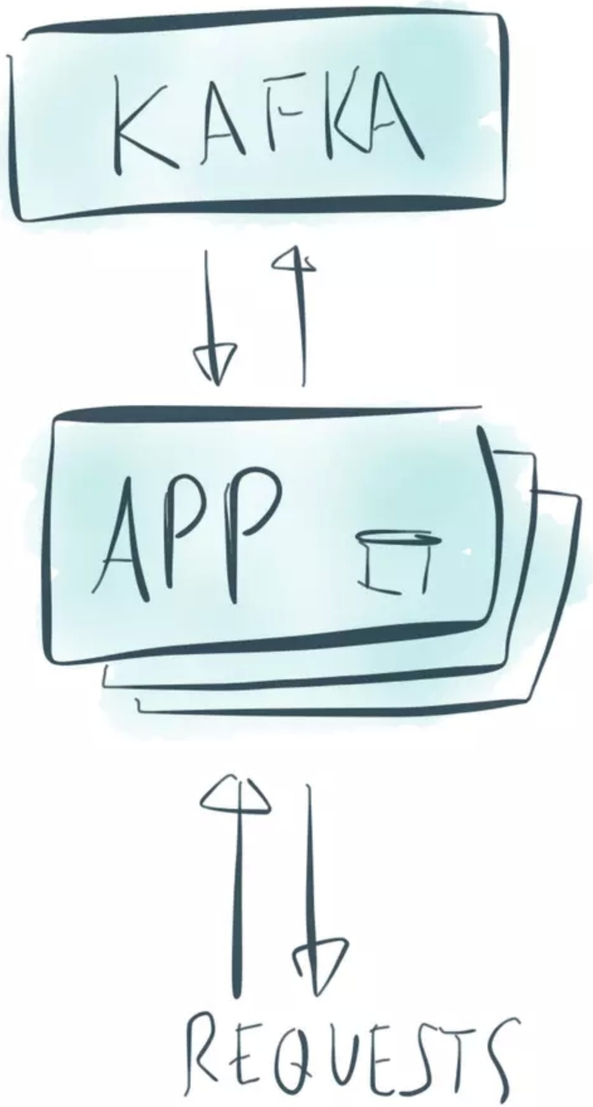
- SIMPLE LIBRARY
- CONVENIENT DSL
- DATAFLOW STYLE WINDOWING
- REPROCESSING
- NO MICROBATCH
- LOCAL STATE

KEY OPERATIONS

- MAP
- FILTER
- AGGREGATE (COUNT, SUM, ETC)
- JOIN

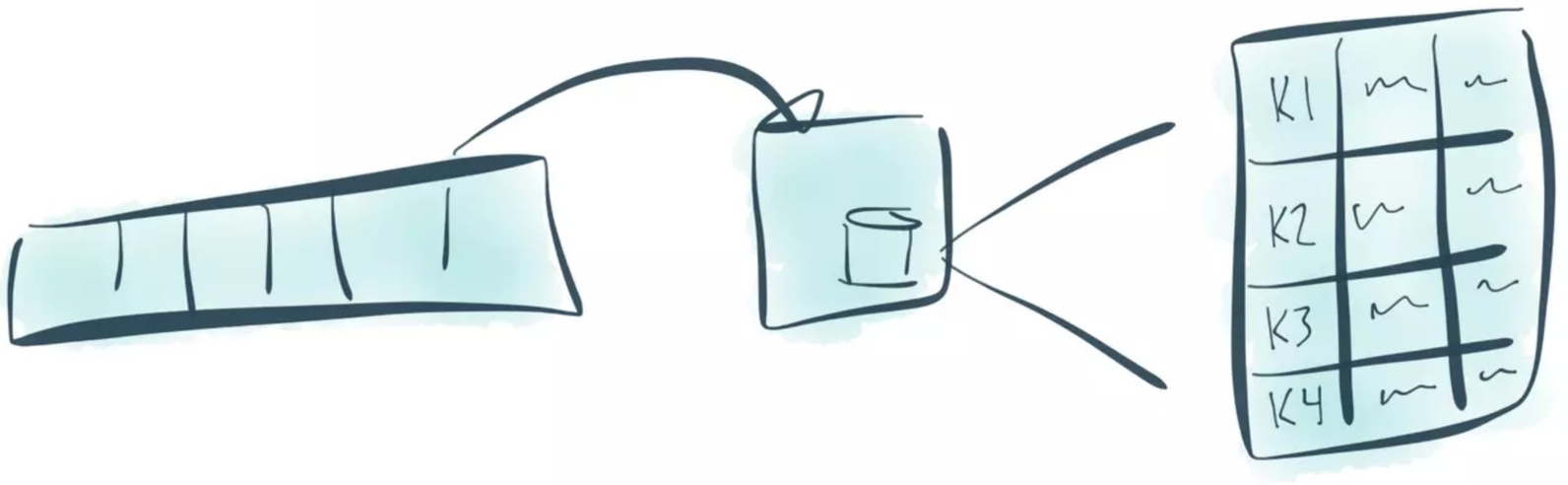

KAFKA
LOOKUPS,
AGGREGATIONS
STREAM PROCESSING
FRAMEWORK
HADOOP
YOUR CODE (1)
YOUR CODE (2)
EXAMPLE
ARCHITECTURE
WITHOUT
KAFKA STREAMS
REQ/RESPONSE
APP

EXAMPLE ARCHITECTURE WITH KAFKA STREAMS

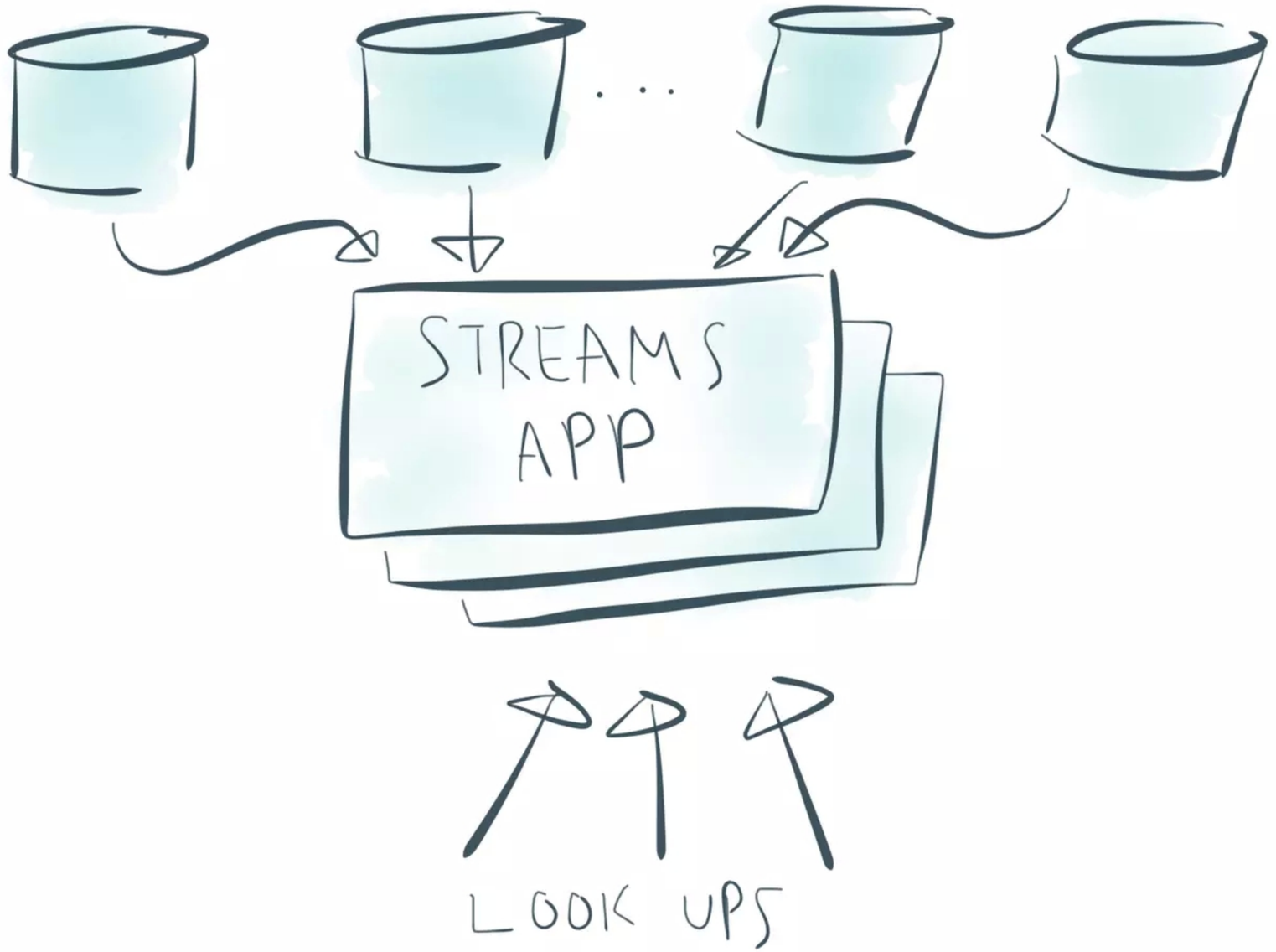


UNIFY

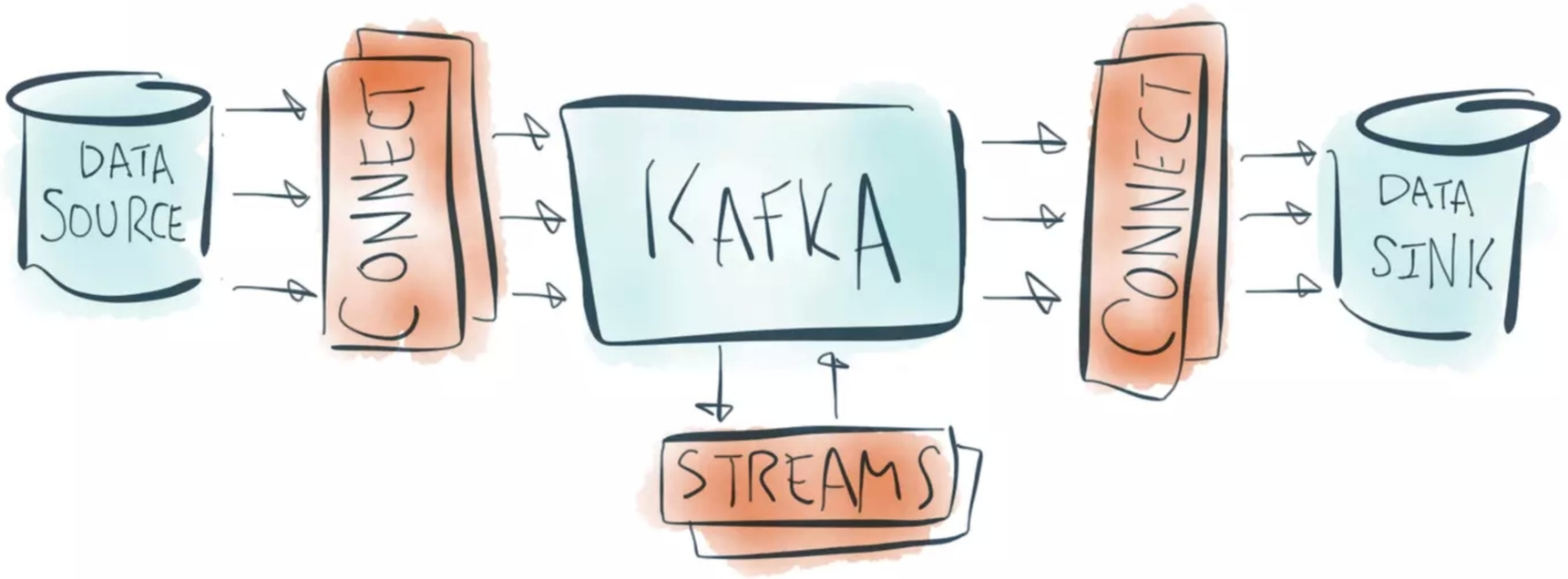
TABLES & STREAMS



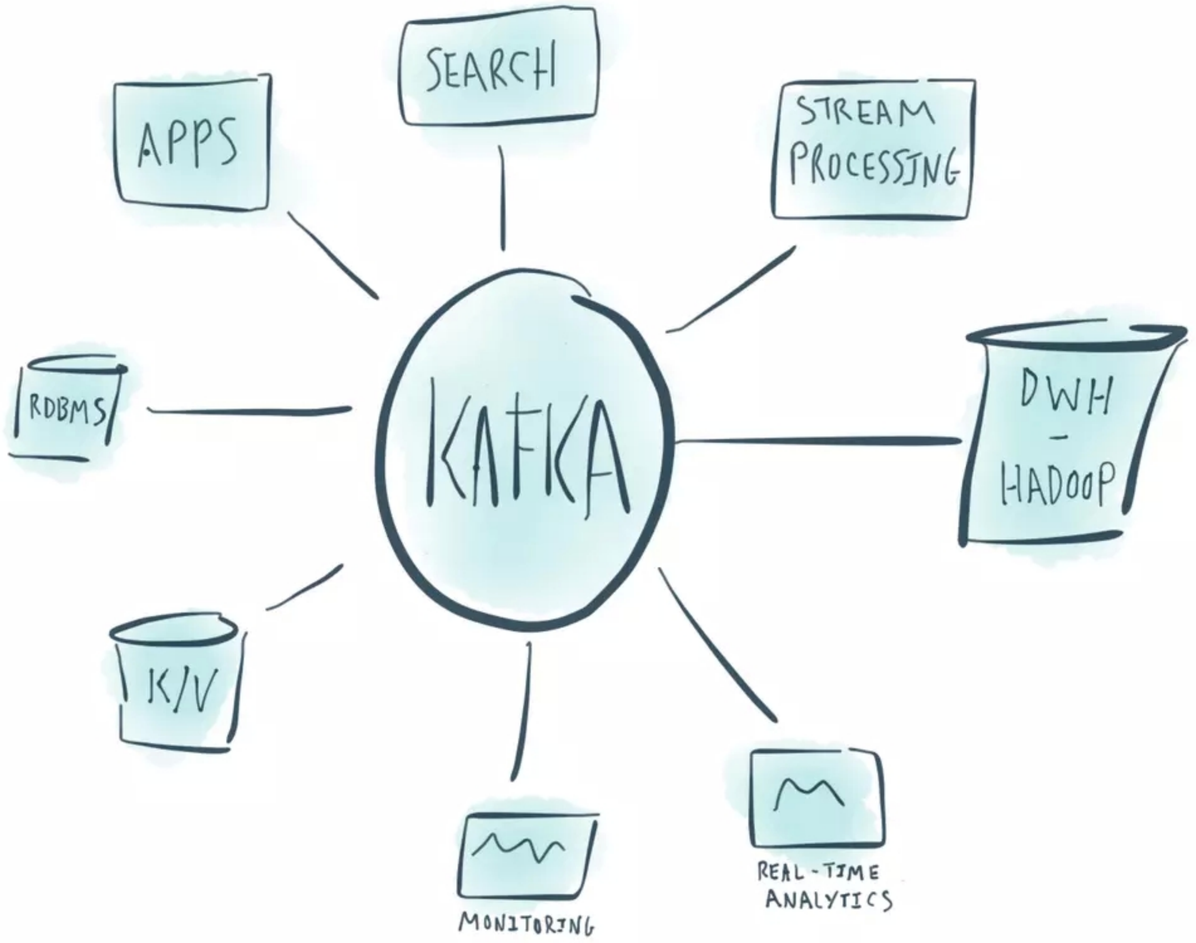
EXAMPLE: CUSTOMER 360



CONNECT + STREAMS

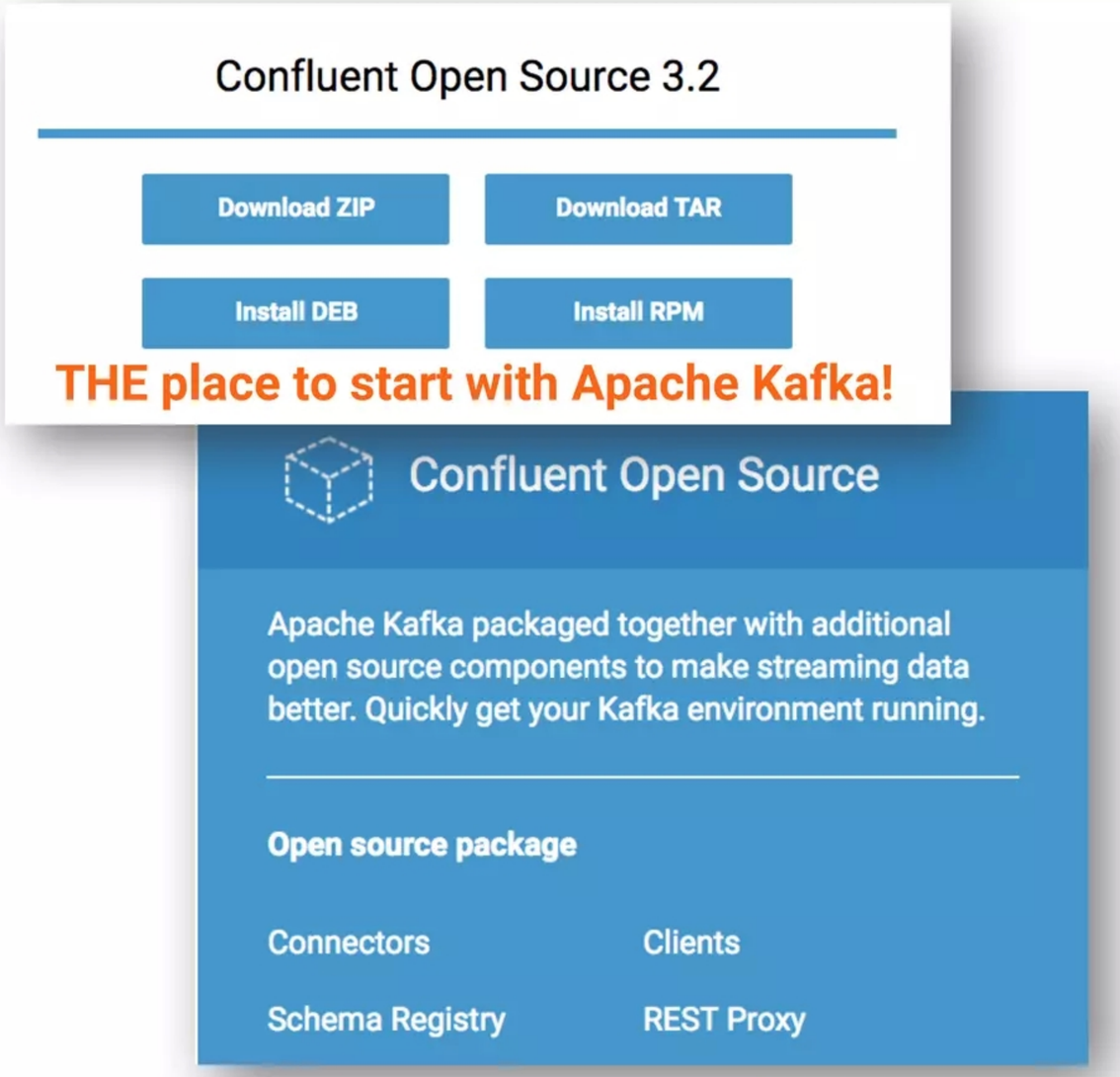


STREAMING PLATFORM



Streaming Platform Adoption

Get Started with Apache Kafka Today!



Thoroughly tested and quality assured

More extensible developer experience

Easy upgrade path to Confluent Enterprise

https://www.confluent.io/download/

Confluent Community

Find your local Meetup Group

https://www.confluent.io/apache-kafka-meetups/

Join us in Slack

https://slackpass.io/confluentcommunity

Thank you

@tlberglund