Apache Kafka,what is it good for?

Leotis Buchanan

May 5, 2017

Why I luv Kafka

This was linkedin architecture before kafka

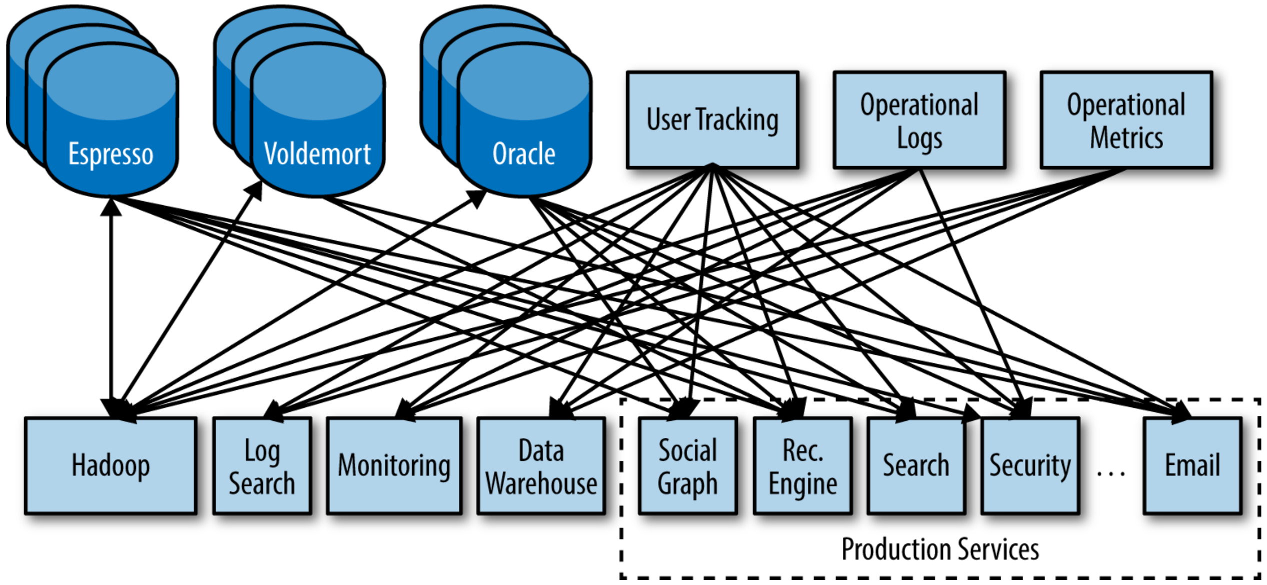


image borrowed from: "Jay Kreps why I love logs book"

why I luv Kafka

look what happened after they decided to use kafka

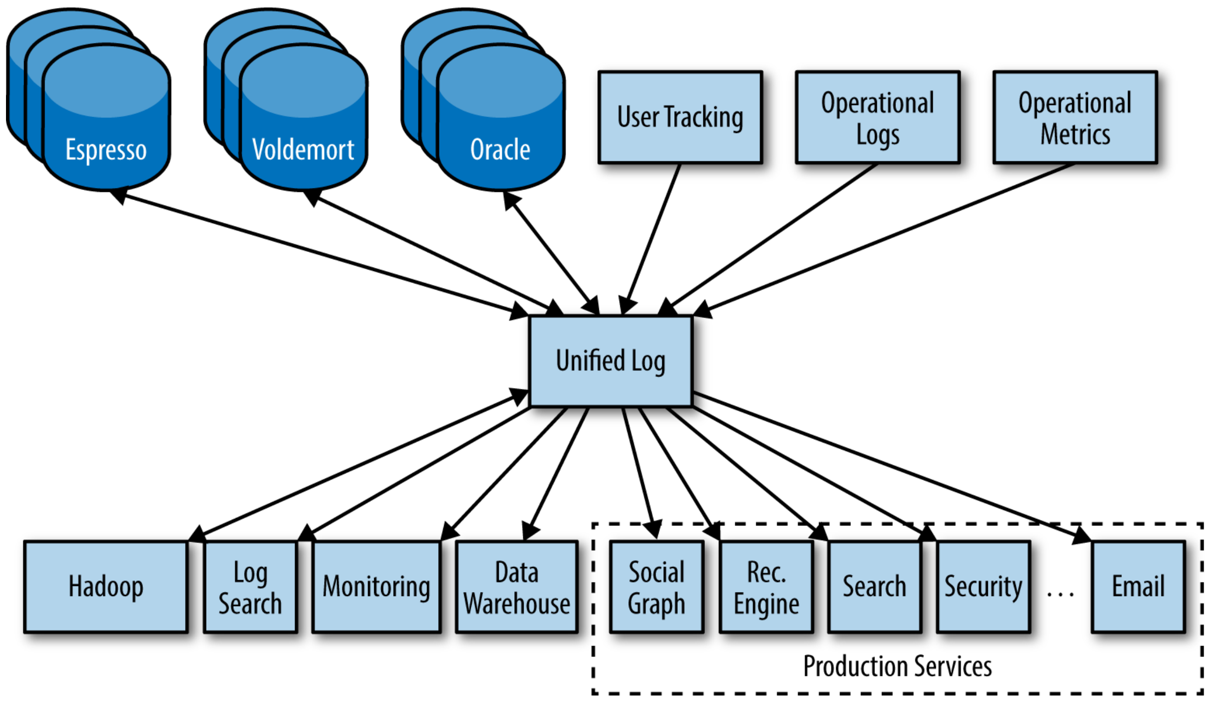


image borrowed from: "Jay Kreps why I love logs book"

why I luv Kafka

They have been able to[1]:

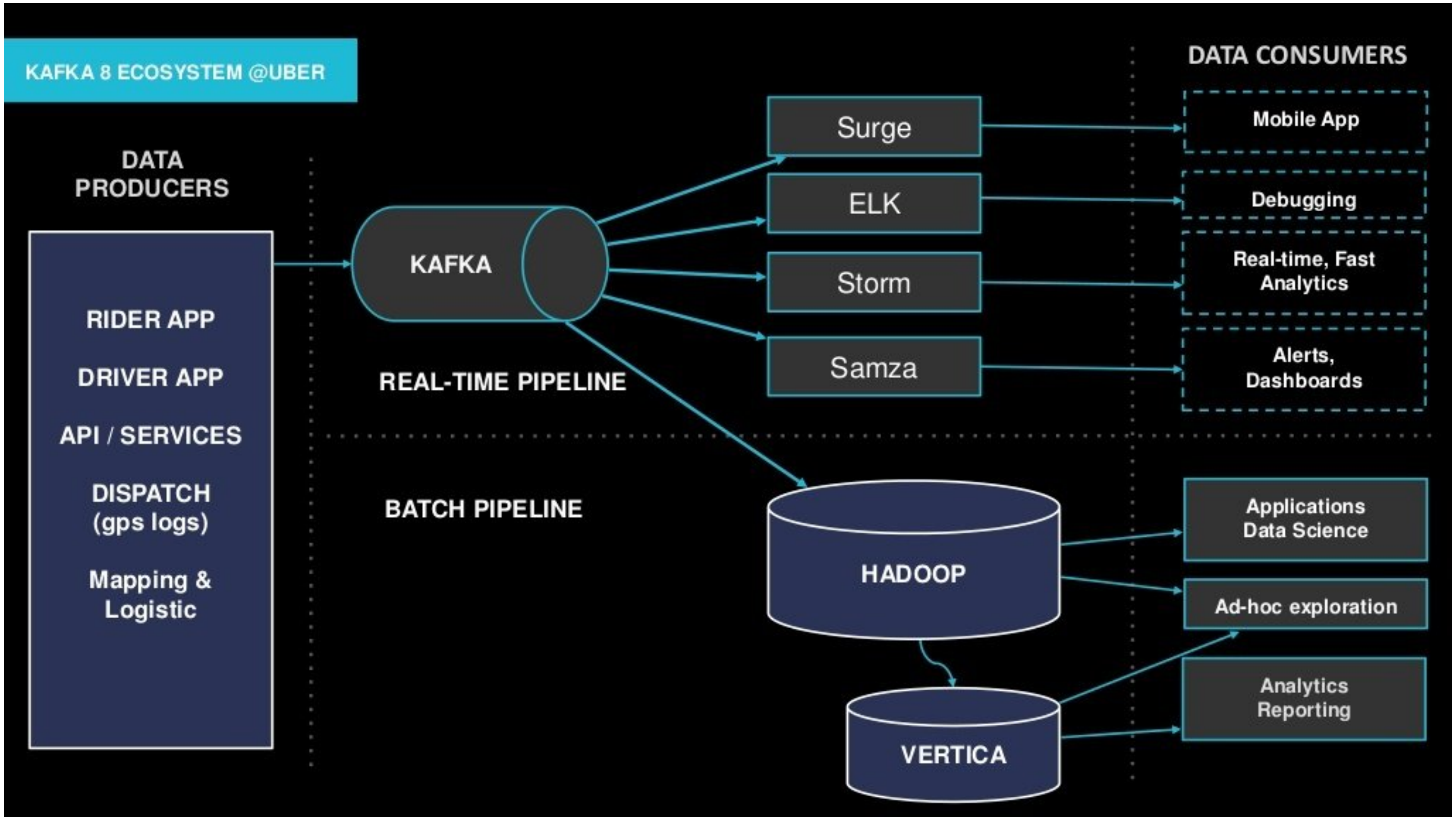
- capture user activity in real time
- change from a batch oriented system to real-time data processing system
- handle 10 billion message writes each day with a sustained peak of over 172,000 messages per second
- support dozens of subscribing systems and delivers more than 55 billion messages to these consumer processing each day

why I luv Kafka

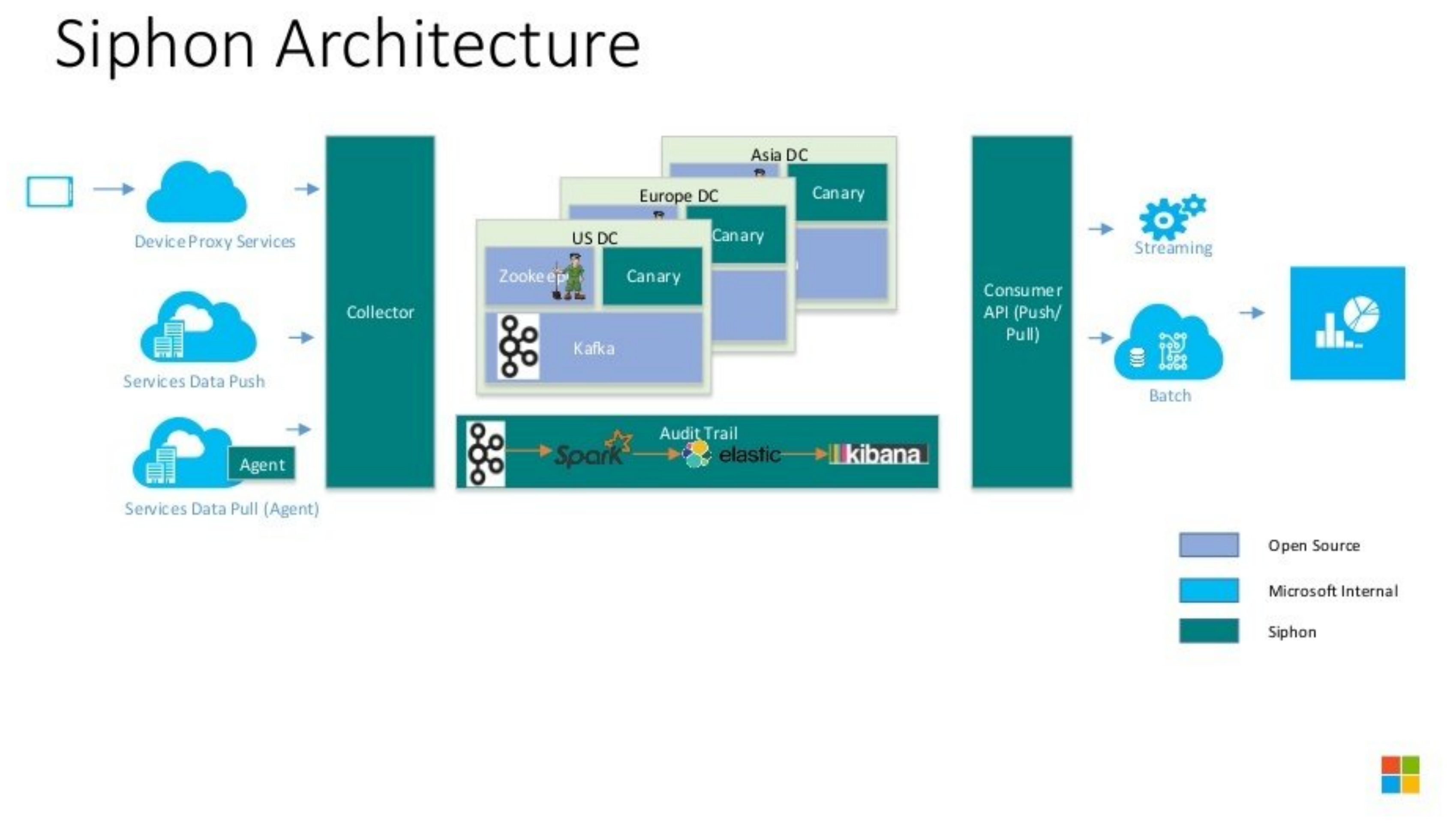
Who else is using kafka to do some really cool things

- uber is using it to build something they all "The World’s Realtime Transit Infrastructure"[2]

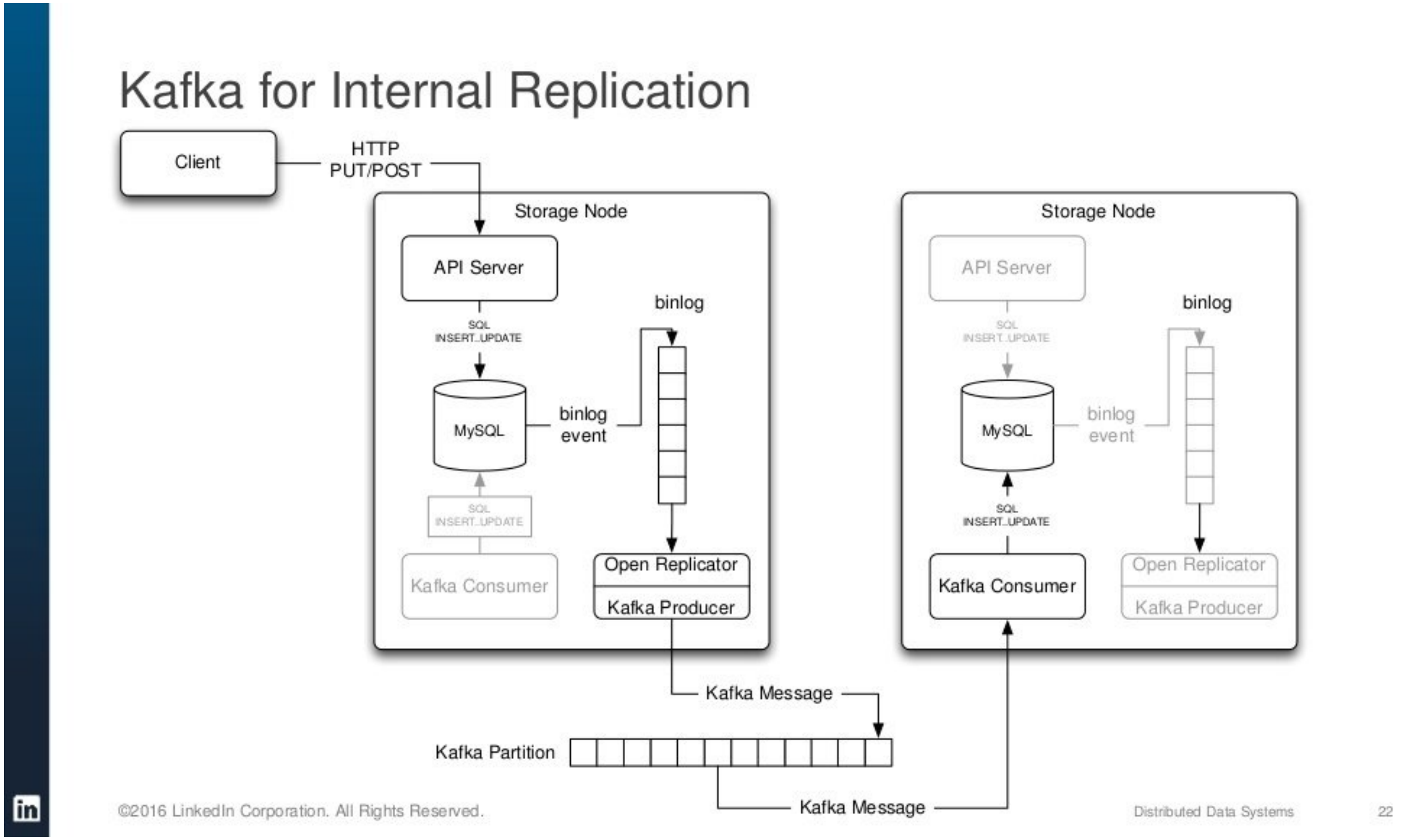
Uber "The World’s Realtime Transit Infrastructure"[2]



Microsoft "Siphon"[3]



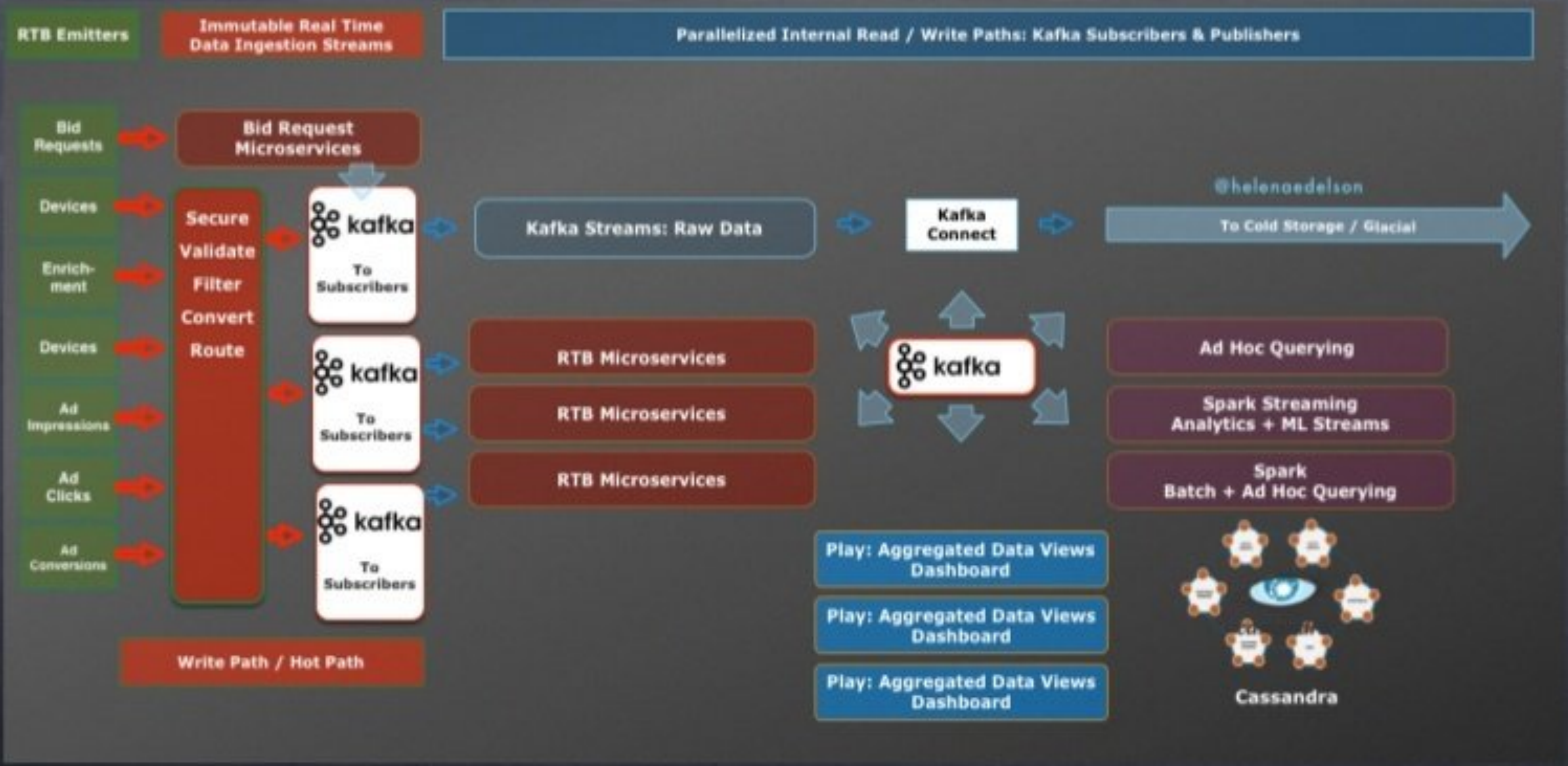
Linkedin "espresso database replication with kafka"[4]



Kafka for Internal Replication

TupleJump "Leveraging Kafka for Big Data in Real Time Bidding, Analytics, Machine Learning and Campaign Management for Globally Distributed Data Flows"[5]

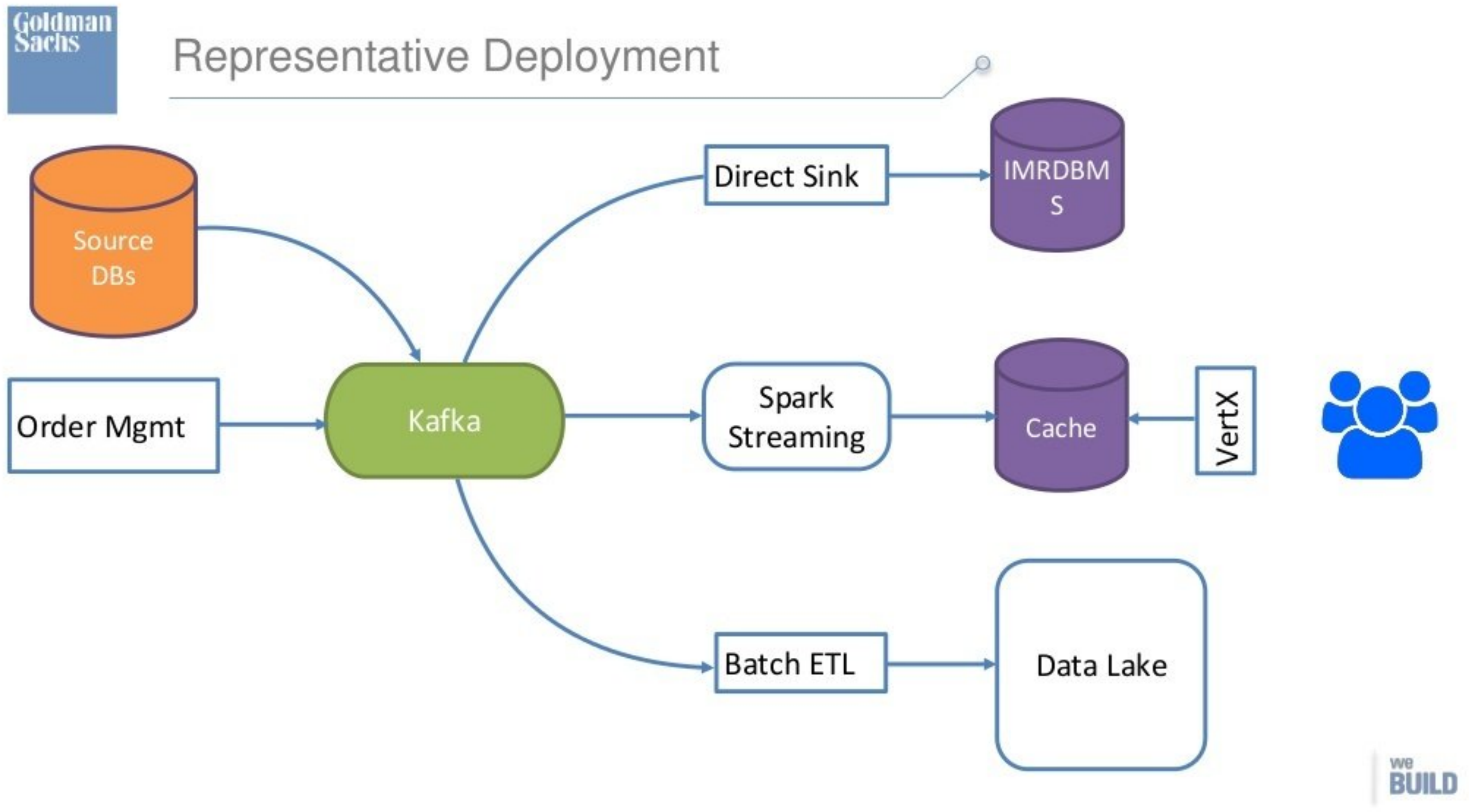
Kafka as Platform Fabric



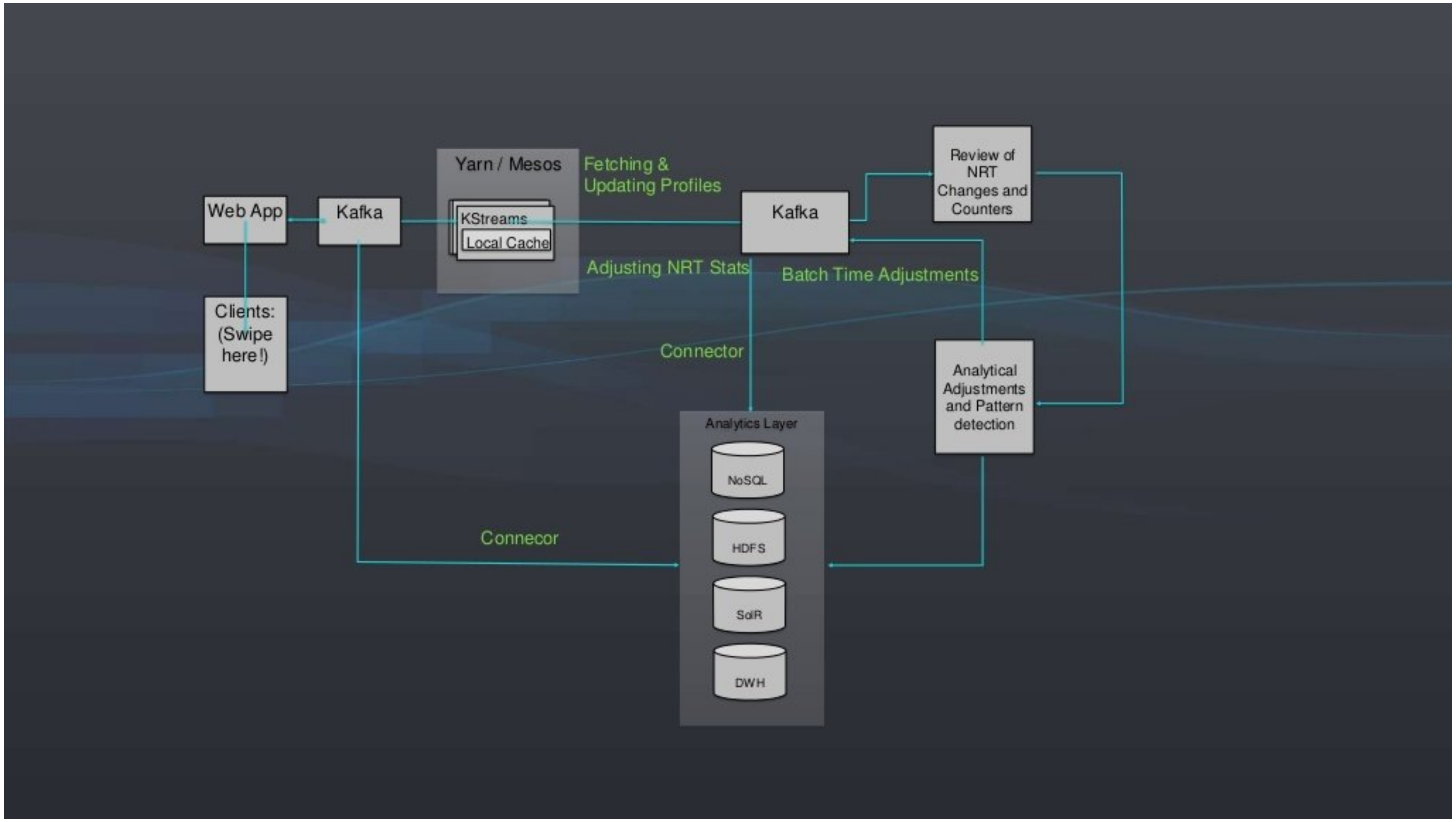
RTB Emitters
Immutable Real Time Data Ingestion Streams
Parallelized Internal Read / Write Paths: Kafka Subscribers & Publishers
Bid Requests
Devices
Enrich-ment
Devices
Ad Impressions
Ad Clicks
Ad Conversions
Bid Request Microservices
Secure
Validate
Filter
Convert
Route
kafka
To Subscribers
kafka
To Subscribers
kafka
To Subscribers
Write Path / Hot Path
Kafka Streams: Raw Data
RTB Microservices
RTB Microservices
RTB Microservices
Kafka Connect
kafka
@helenaedelson
To Cold Storage / Glacial
Ad Hoc Querying
Spark Streaming Analytics + ML Streams
Spark Batch + Ad Hoc Querying
Play: Aggregated Data Views Dashboard
Play: Aggregated Data Views Dashboard
Play: Aggregated Data Views Dashboard
Cassandra

@helenaedelson #kafkasummit

Goldman Sacs "Real-Time Analytics Visualized w/ Kafka + Streamliner + MemSQL + ZoomData" **[6]**



Fraud detection "Gwen Shapira**" [7]**



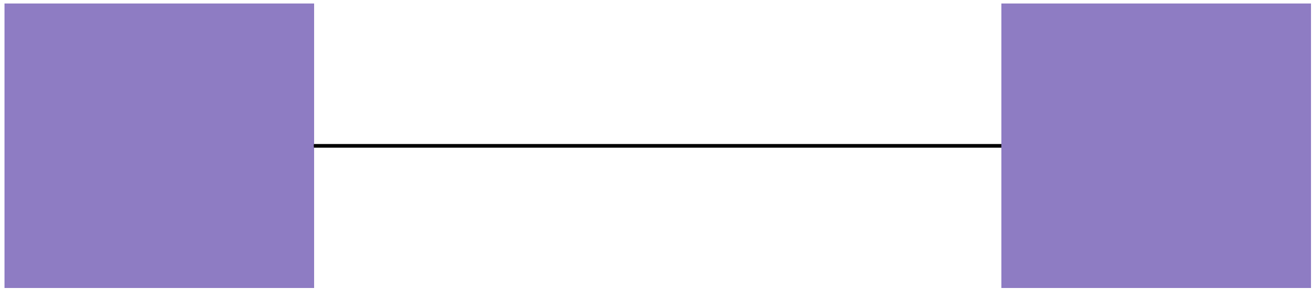
What is kafka?

- Originally created at LinkedIn in 2010
- Designed to support batch and real-time analytics
- Performs extremely well at very large scale
- LinkedIn’s installation of Kafka processes over 1.4 *trillion* messages per day
- Made open source in 2011, became a top-level Apache project in 2012
- In use at many not very successful organizations:
- Twitter, Netflix, Goldman Sachs, Hotels.com, IBM, Spotify, Uber, Square, Cisco…

Motivation for kafka?

You start out with a simple requirement, so you decide to build a simple pipeline

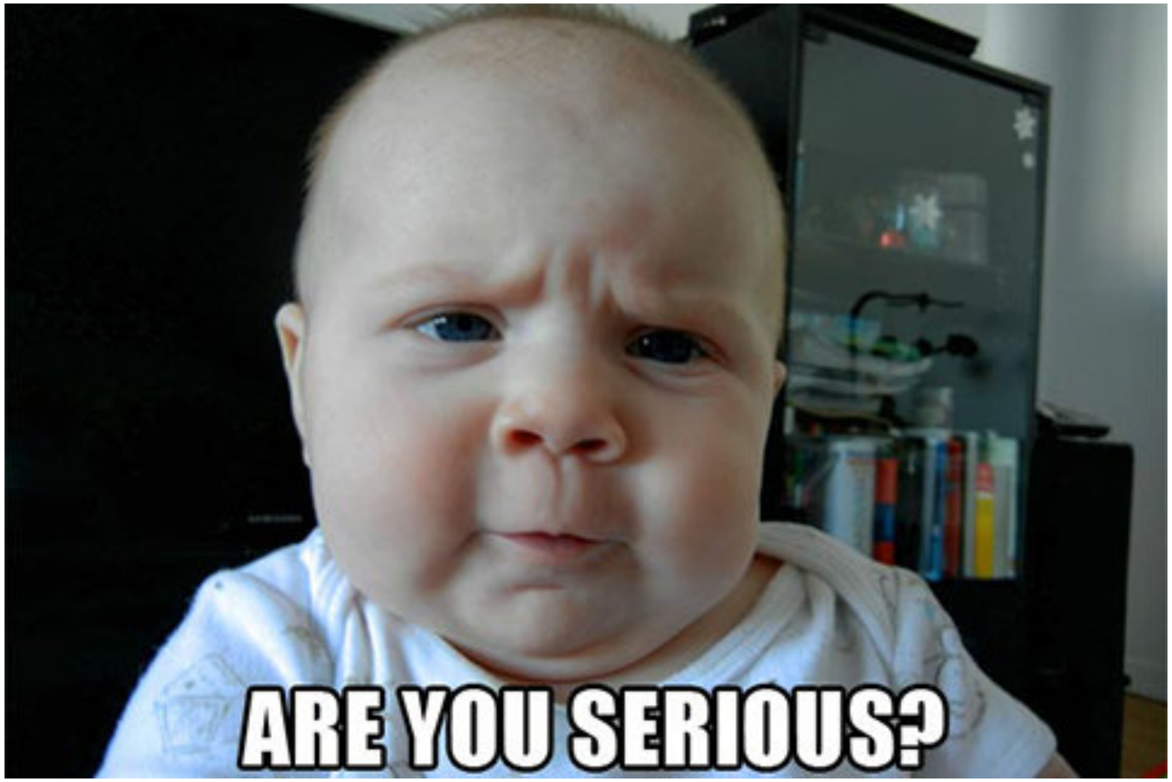
- A single place where all data resides
- A single ETL (Extract, Transform, Load) process to move data to that location



Motivation for kafka?

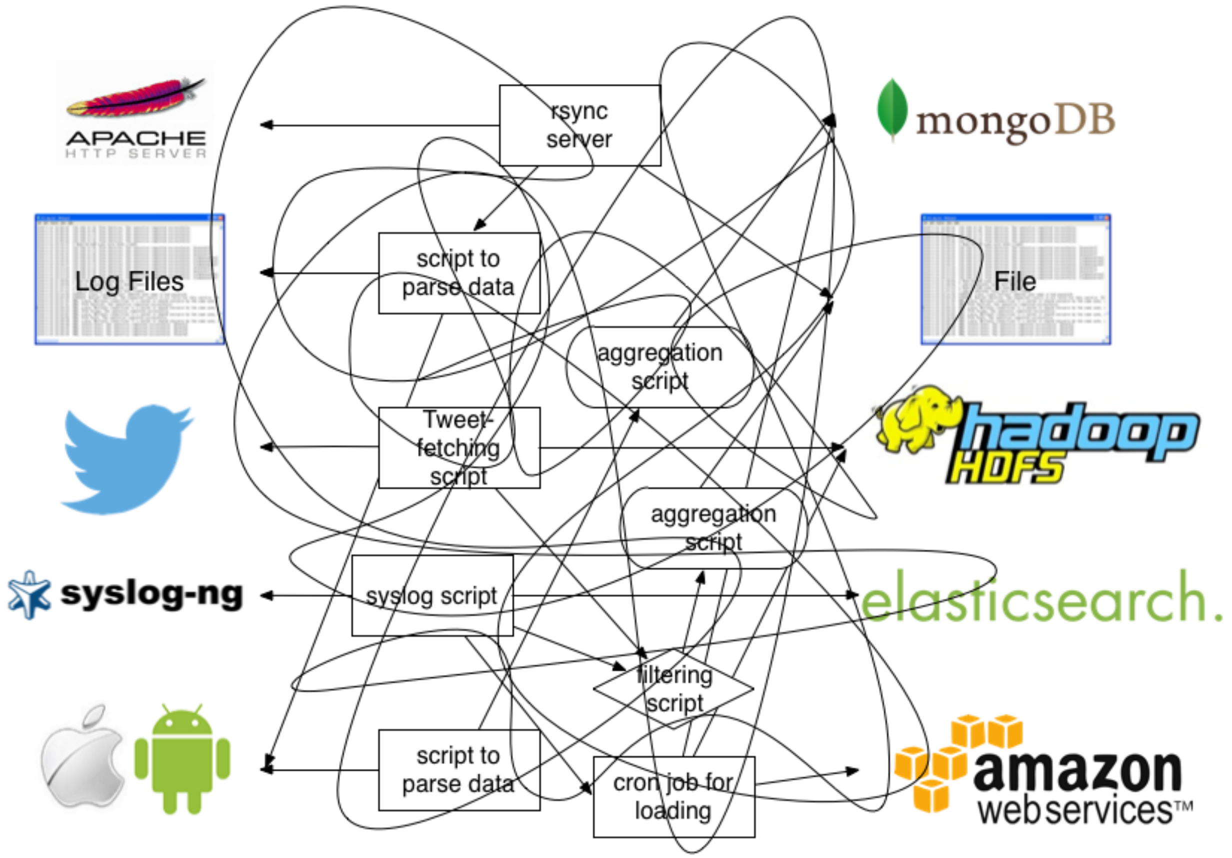
It is a beautiful success, now everybody knows about it and now they need you add more features:

1. every body wants to be notified if it fails
2. can we add more datasources?
3. they have a new vision for your hastily put together pipeline?
4. Lets make this available to everyone!!


ARE YOU SERIOUS?

Yes they are serious. So you reluctantly follow orders

and create a master piece



Then it fails a month after and they ask you to fix it

so back to the motivations for kafka

- Data pipelines typically start out simply
- Data pipelines inevitably grow over time
- Systems and ETL become increasingly hard to manage

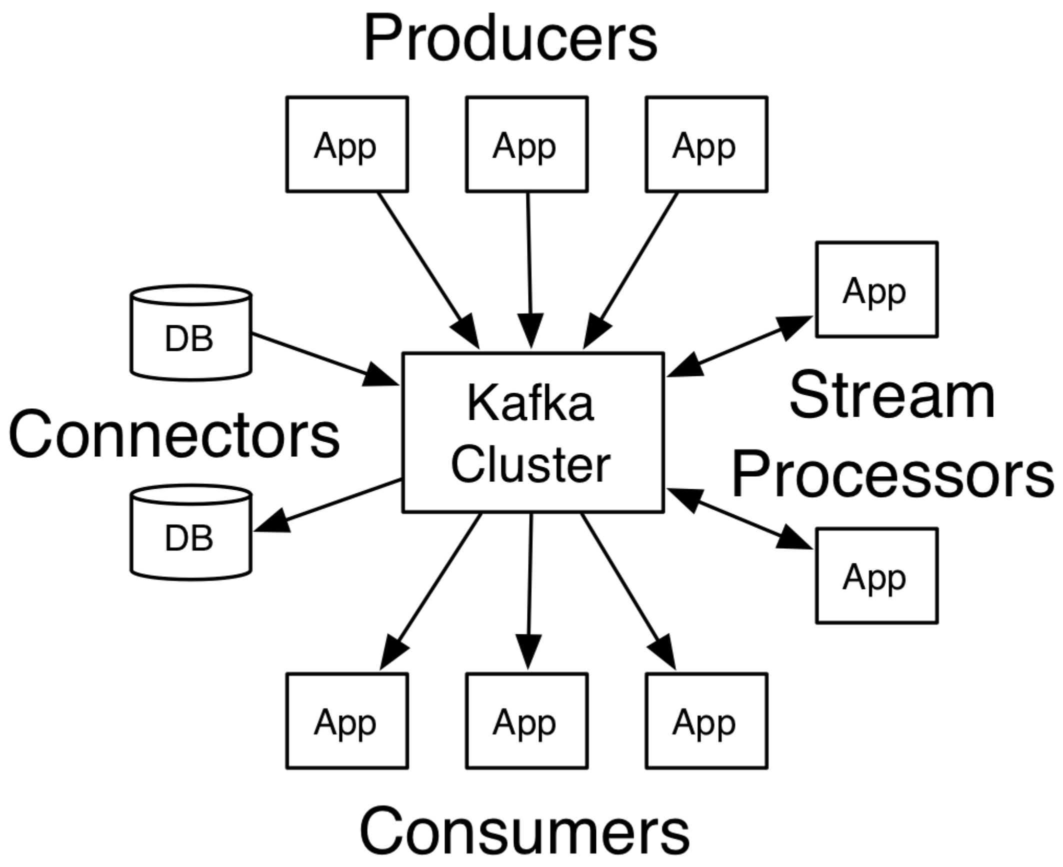
so back to the motivations for kafka

- Traditionally, almost all data processing was batch-oriented
- This is limiting
- If you process data as it is being generated you can make decisions now.

Kafka is a universal pipeline

- Kafka decouples data source and destination systems – Via a *publish/subscribe* architecture
- Using Kafka, all data sources write their data to the Kafka cluster
- Any system wishing to use the data reads it from Kafka

Kafka is a universal pipeline



Kafka is a universal pipeline

- Once the data is in Kafka, it can be read by multiple different Consumers
- Increasing the number of Consumers does not add significant load to the system
- Adding a new Consumer does not require any modification to the Producer(s)

Kafka features

- Producers write data in the form of *messages* to the Kafka cluster
- Messages are written to *topics*
- Consumers read messages from one or more topics
- Data retention time in Kafka can be configured on a per-topic basis

Kafka features

- **Kafka is very scalable, and very resilient**

– Even a small cluster can process a large volume of messages

– Tests have shown that three low-end machines can easily deal with two million writes per second

– Messages are replicated on multiple machines for reliability

- **Consumers can be shut down temporarily**

– When they restart, they will continue to read from where they left off

Kafka Components

There are four key components in a Kafka system

- Producers
- Brokers
- Consumers
- ZooKeeper

Producers

- A Producer sends messages to the Kafka cluster
- Producers can be written in any language
- A command-line Producer tool exists to send messages to the cluster

Kafka messages

- A message is the basic unit of data in Kafka
- A message is a key-value pair
- Key and value can be any data type
  - You provide a serializer to turn the key and value into byte arrays
- Key is optional
  - Keys are used to determine which *Partition* (see later) a message will be sent to

Topics

- Each message belongs to a *Topic*
- Developers decide which topics exist
- Typically, different systems will write to different topics

Broker

- Brokers receive and store messages when they are sent by the Producers
- A Kafka cluster will typically have multiple Brokers
- Each Broker manages one or more Partitions

Brokers Manage Partitions

- Any given Partition is handled by a single Broker
- Each Partition is stored on the Broker’s disk as one or more log files
- Each message in the log is identified by its *offset*

Consumer Basics

- Consumers pull messages from the cluster
- Multiple Consumers can read data from the same topic

Why Pull messages

- Kafka consumers work by pulling messages
- The advantages of pulling, rather than pushing, data, include:

– The ability to add more Consumers to the system without reconfiguring the cluster

– The ability for a Consumer to go offline and return later, resuming from where it left off

– No problems with the Consumer being overwhelmed by data

– It can pull, and process, the data at whatever speed it needs to

Keeping Track of Position

- As messages are written to a topic, the Consumer will automatically retrieve them
- The *Consumer Offset* keeps track of the latest message read
- If necessary, the Consumer Offset can be changed
- The Consumer Offset is stored in a special Kafka topic

ZooKeeper

- Apache ZooKeeper is an Apache project
- It is “a centralized service for maintaining configuration information”
- Used by many projects

How Kafka Uses ZooKeeper

Kafka Brokers use ZooKeeper for a number of important internal features

- Leader election, failure detection

Decoupling Producers and Consumers

- A key feature of Kafka is that Producers and Consumers are decoupled
- A slow Consumer will not affect Producers
- More Consumers can be added without affecting Producers
- Failure of a Consumer will not affect the system
- Multiple brokers, multiple topics, and *Consumer Groups* provide very high scalability

Example code

Creating a producer

```
private Properties kafkaProps = new Properties();
kafkaProps.put("bootstrap.servers",
 "broker1:9092,broker2:9092");

kafkaProps.put("key.serializer",
"org.apache.kafka.common.serialization.String-
    Serializer");

kafkaProps.put("value.serializer",
 "org.apache.kafka.common.serializa-
    tion.StringSerializer");

producer = new KafkaProducer<String, String>(kafkaProps);
```

Sending a message

```
ProducerRecord<String, String> record =
            new ProducerRecord<>("CustomerCountry",
           "Precision Products",
    "France");
    try {
      producer.send(record);
    } catch (Exception e) {
}
```

Creating a consumer

```
Properties props = new Properties();

props.put("bootstrap.servers",
"broker1:9092,broker2:9092");
props.put("group.id", "CountryCounter");
props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDe-
    serializer");
props.put("value.deserializer",
 "org.apache.kafka.common.serialization.StringDe-
    serializer");
KafkaConsumer<String, String> consumer =
new KafkaConsumer<String,
    String>(props);
```

Subscribing to a topic

```
consumer.subscribe(
Collections.singletonList("customerCountries"));
```

Reading a message

```
 try {
      while (true) {
          ConsumerRecords<String, String> records = consumer.poll(100);
          for (ConsumerRecord<String, String> record : records)
          {
              log.debug("topic = %s, partition = %s, offset = %d, customer = %s,
   country = %s\n",
              record.topic(), record.partition(), record.offset(), record.key(),
   record.value());
              int updatedCount = 1;
              if (custCountryMap.countainsKey(record.value())) {
                  updatedCount = custCountryMap.get(record.value()) + 1;
              }
              custCountryMap.put(record.value(), updatedCount)
              JSONObject json = new JSONObject(custCountryMap);
              System.out.println(json.toString(4))
          }
}
} finally {
      consumer.close();
   }
```

Summary

This was an introductory presentation. Kafka is a really cool tool.

if you want to learn more you can:

- Do the confluent kafka course

  [https://www.confluent.io/training/]

- Get yourself a copy of the kafka definite guide book

  [http://shop.oreilly.com/product/0636920044123.do]

- Build something using kafka.

Thank you for listening

Easy Questions Please

. References

- 1. Building LinkedIn’s Real-time Activity Data Pipeline Ken Goodhope, Joel Koshy, Jay Kreps, Neha Narkhede, Richard Park, Jun Rao, Victor Yang Ye LinkedIn
- 2. Kafka + Uber- The World’s Realtime Transit Infrastructure, Aaron Schildkrout [https://www.confluent.io/kafka-summit-2016-keynote-kafka-and-uber-the-worlds-realtime-transit-infrastructure/]
- 3. Siphon: [https://www.confluent.io/kafka-summit-2016-users-siphon-near-rea-time-databus-using-kafka/]
- 4. Espresso Database Replication with Kafka [https://www.confluent.io/kafka-summit-2016-users-espresso-database-replication-with-kafka/]
- 5. Goldma sacs [https://www.confluent.io/kafka-summit-2016-users-real-time-analytics-visualized-with-kafka/]