Apache Kafka

Scalable Message Processing and more!

Guido Schmutz - 24.4.2017

@gschmutz

guidoschmutz.wordpress.com

Guido Schmutz

Working at Trivadis for more than 20 years

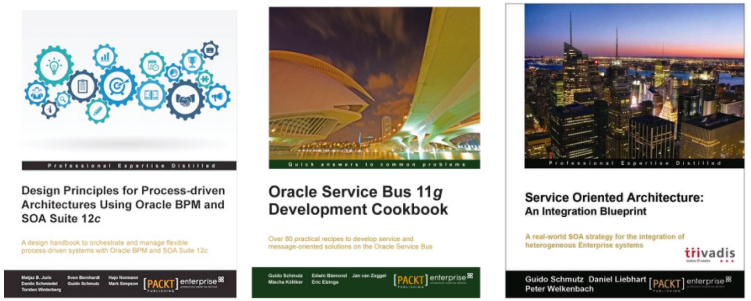
Oracle ACE Director for Fusion Middleware and SOA

Consultant, Trainer Software Architect for Java, Oracle, SOA and Big Data / Fast Data

Member of Trivadis Architecture Board

Technology Manager @ Trivadis

More than 30 years of software development experience



Contact: guido.schmutz@trivadis.com

Blog: http://guidoschmutz.wordpress.com

Slideshare: http://www.slideshare.net/gschmutz

Twitter: gschmutz

trivadis
makes IT easier.

Agenda

1. Introduction & Motivation
2. Kafka Core
3. Kafka Connect
4. Kafka Streams
5. Kafka and "Big Data" / "Fast Data" Ecosystem
6. Kafka in Enterprise Architecture
7. Confluent Data Platform
8. Summary

Introduction & Motivation

Apache Kafka - Overview

Distributed publish-subscribe messaging system

Designed for processing of real time activity stream data (logs, metrics collections, social media streams, …)

Initially developed at LinkedIn, now part of Apache

Does not use JMS API and standards

Kafka maintains feeds of messages in topics

Apache Kafka - Motivation

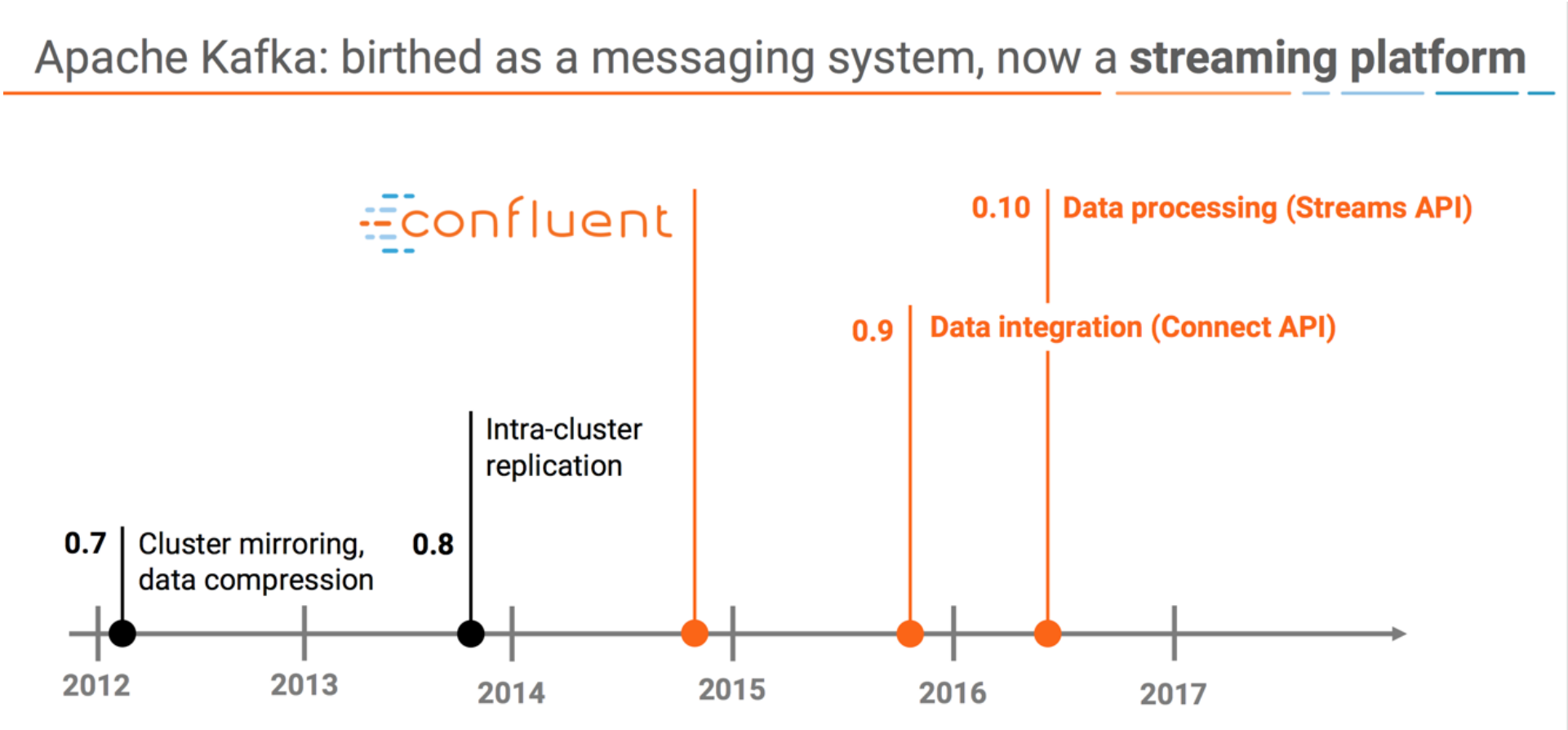
LinkedIn's motivation for Kafka was:

- "A unified platform for handling all the real-time data feeds a large company might have."

Must haves

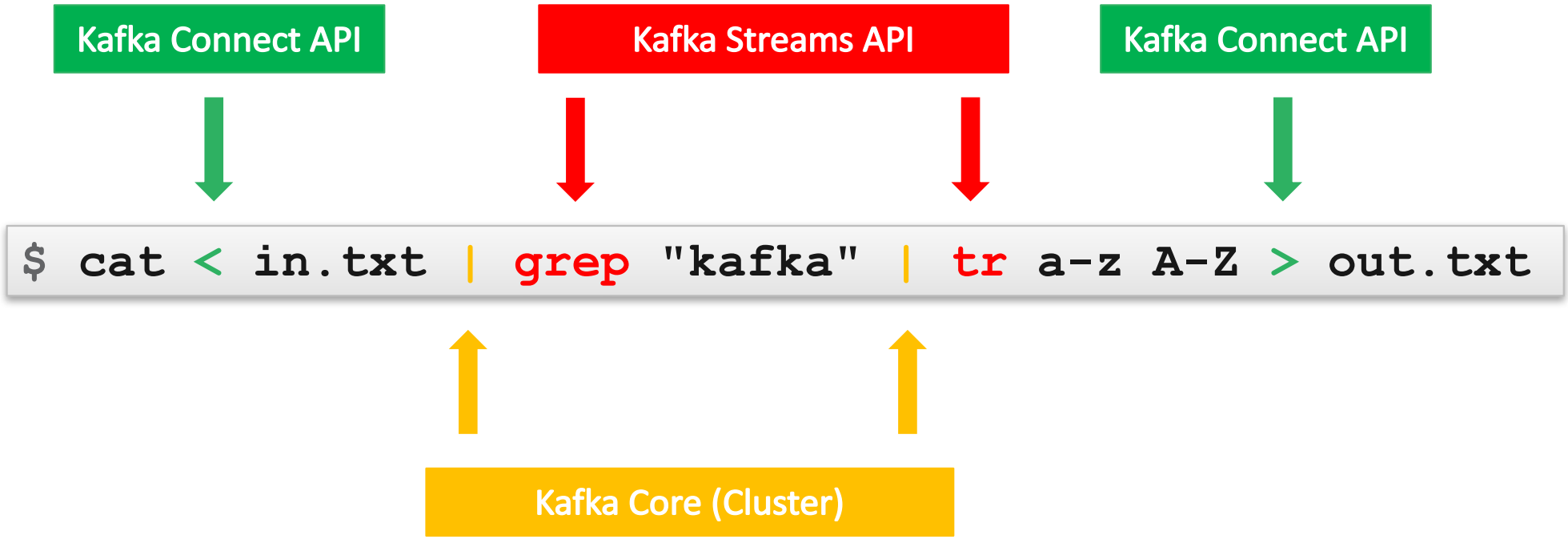
- High throughput to support **high volume event feeds**
- Support real-time processing of these feeds to create **new, derived feeds**.
- Support large data backlogs to handle periodic ingestion from **offline systems**
- Support low-latency delivery to handle more traditional **messaging use cases**
- Guarantee **fault-tolerance** in the presence of machine failures

Apache Kafka History



Source: Confluent

Apache Kafka - Unix Analogy



Source: Confluent

Kafka Core

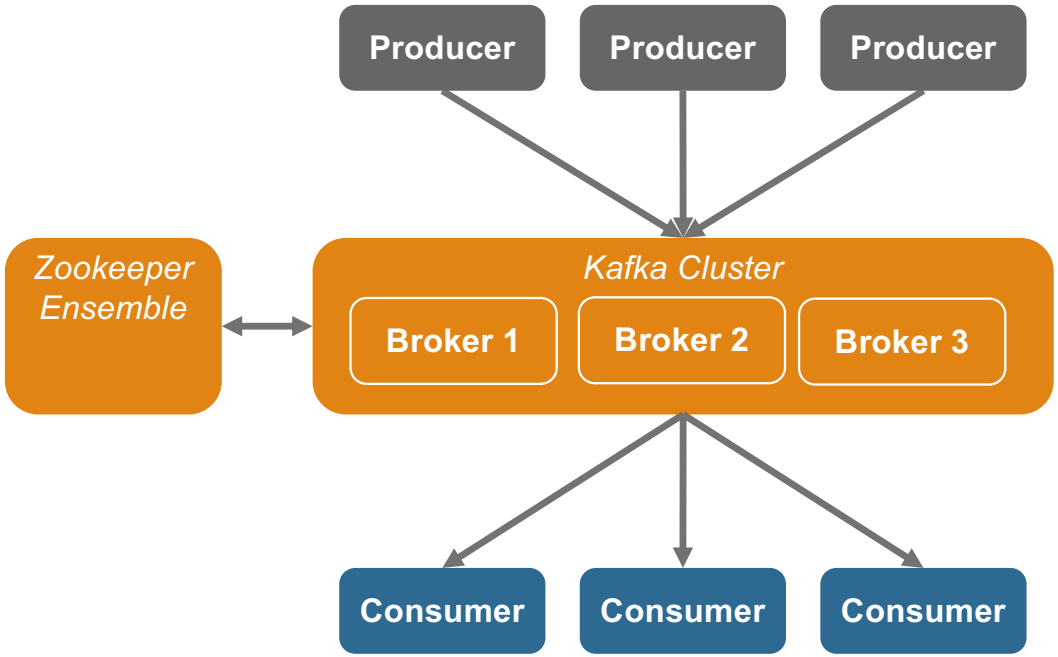
Kafka High Level Architecture

The who is who

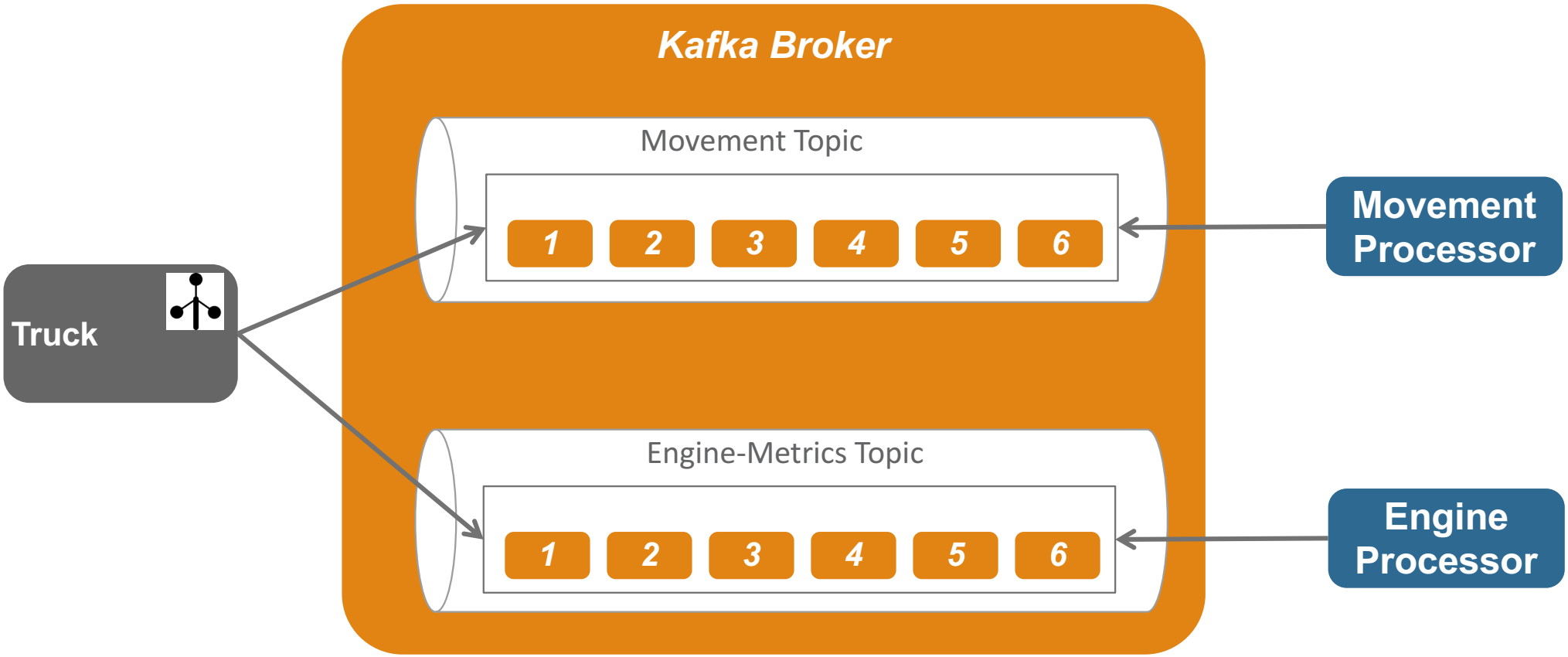
- **Producers** write data to **brokers**.
- **Consumers** read data from **brokers**.
- All this is distributed.

The data

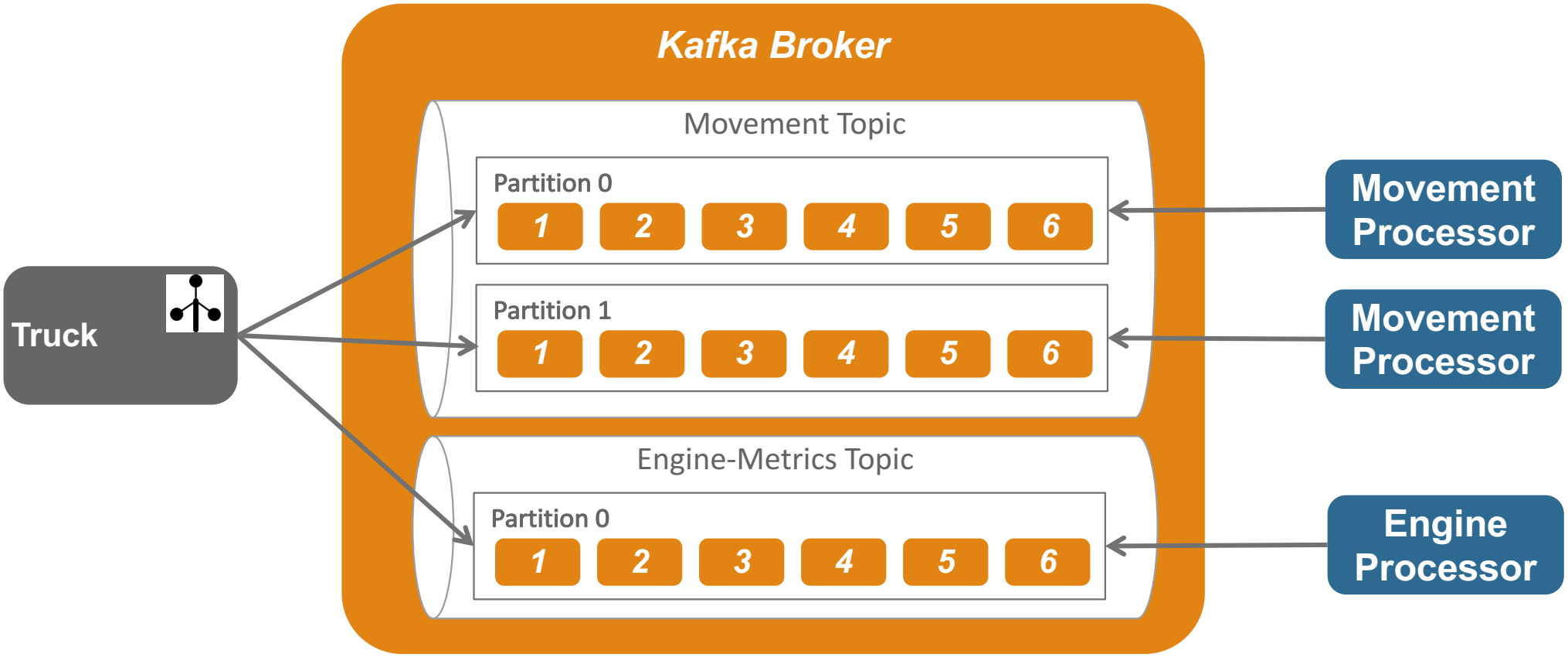
- Data is stored in **topics**.
- **Topics** are split into **partitions**, which are **replicated**.



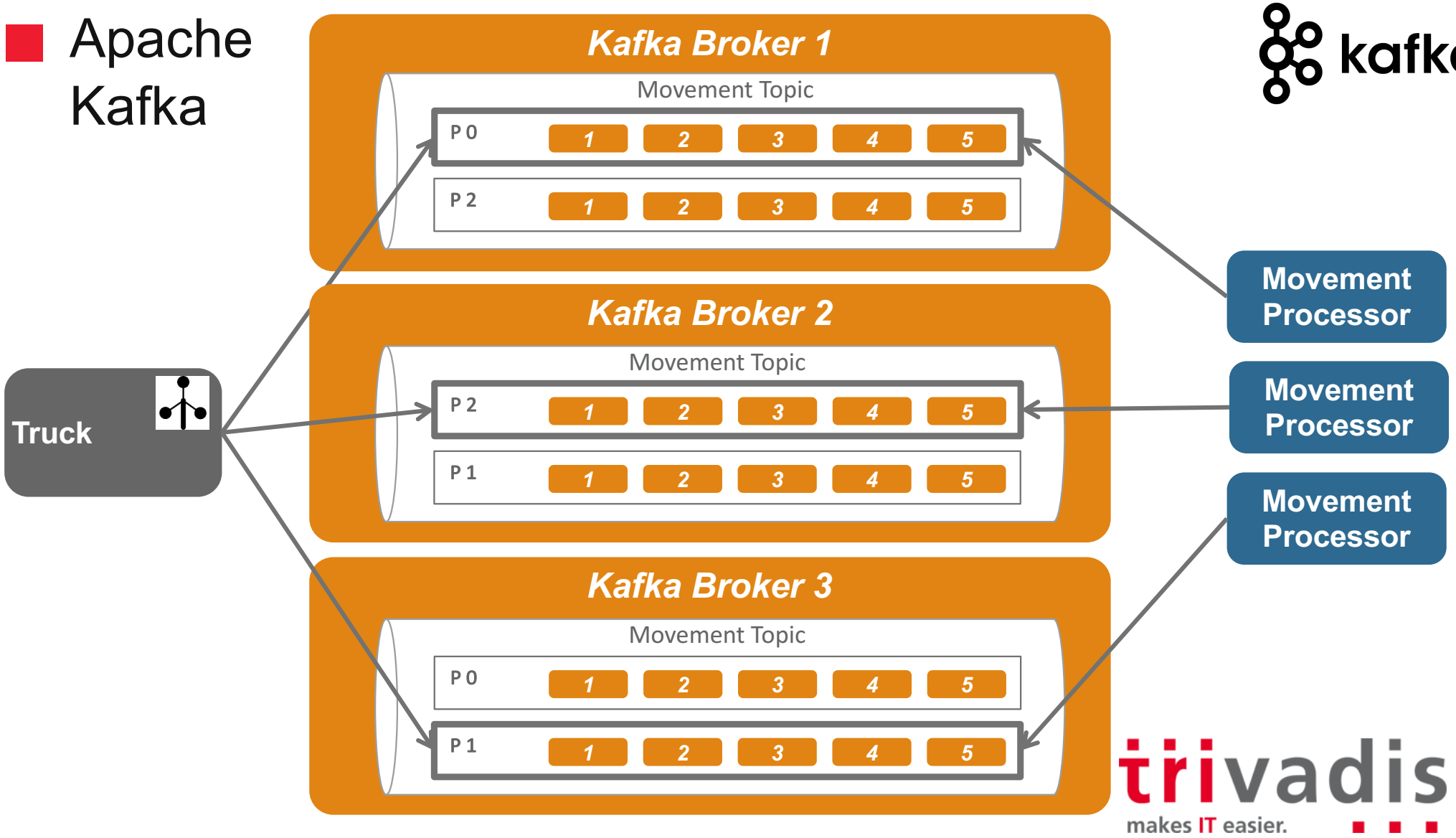
Apache Kafka - Architecture



Apache Kafka - Architecture



Apache Kafka

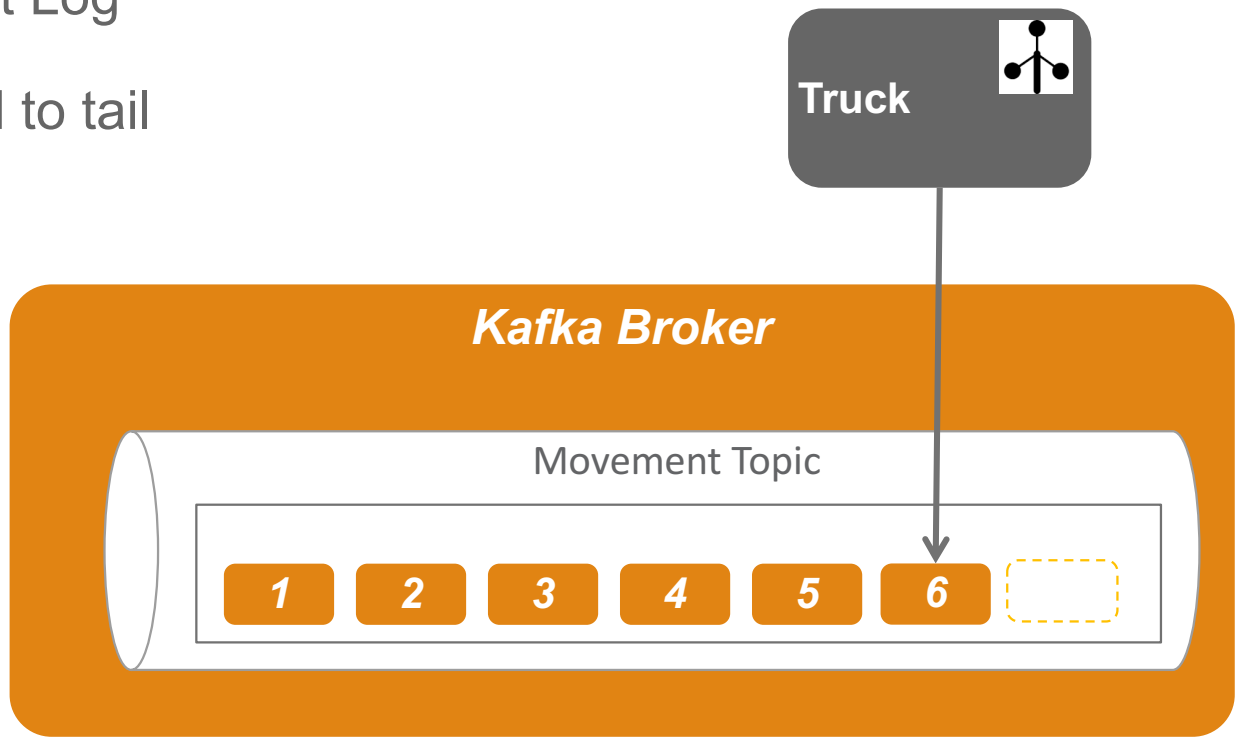


kafka

trivadis
makes IT easier.

Apache Kafka - Architecture

- Write Ahead Log / Commit Log
- Producers always append to tail
- think append to file



trivadis
makes IT easier.

Kafka Topics

Creating a topic

- Command line interface

```
$ kafka-topics.sh -zookeeper zk1:2181 --create \
   --topic my.topic --partitions 3 \
   --replication-factor 2 --config x=y
```

- Using `AdminUtils.createTopic` method
- Auto-create via `auto.create.topics.enable = true`

Modifying a topic

https://kafka.apache.org/documentation.html#basic_ops_modify_topic

Deleting a topic

- Command Line interface

Kafka Producer

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

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

```
ProducerRecord<String, String> record =
   new ProducerRecord<>("topicName", "Key", "Value");
try {
   producer.send(record);
} catch (Exception e) {}
```

Durability Guarantees

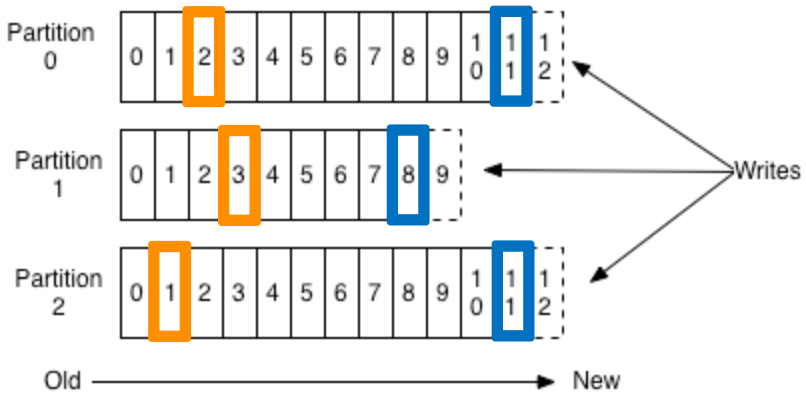
Producer can configure acknowledgements

Value	Description	Throughput	Latency	Durability
0	• Producer doesn't wait for leader	high	low	low (no guarantee)
1 (default)	• Producer waits for leader • Leader sends ack when message written to log • No wait for followers	medium	medium	medium (leader)
all (-1)	• Producer waits for leader • Leader sends ack when all In-Sync Replica have acknowledged	low	high	high (ISR)

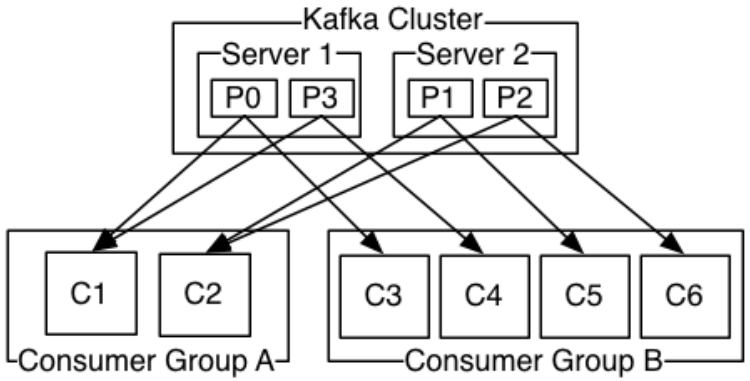
Apache Kafka - Partition offsets

Offset: messages in the partitions are each assigned a unique (per partition) and sequential id called the offset

- Consumers track their pointers via *(offset, partition, topic)* tuples



Consumer Group A Consumer Group B



Source: Apache Kafka

Data Retention – 3 options

1. Never
2. Time based (TTL)

```
log.retention.{ms | minutes | hours}
```

3. Size based

```
log.retention.bytes
```

4. Log compaction based (entries with same key are removed)

```
kafka-topics.sh --zookeeper localhost:2181 \
                --create --topic customers \
                --replication-factor 1 --partitions 1 \
                --config cleanup.policy=compact
```

Apache Kafka – Some numbers

Kafka at LinkedIn => over 1800+ broker machines / 79K+ Topics

https://engineering.linkedin.com/kafka/running-kafka-scale

1.3 Trillion messages per day	330 Terabytes in/day 1.2 Petabytes out/day	Peak load for a single cluster 2 million messages/sec 4.7 Gigabits/sec inbound 15 Gigabits/sec outbound

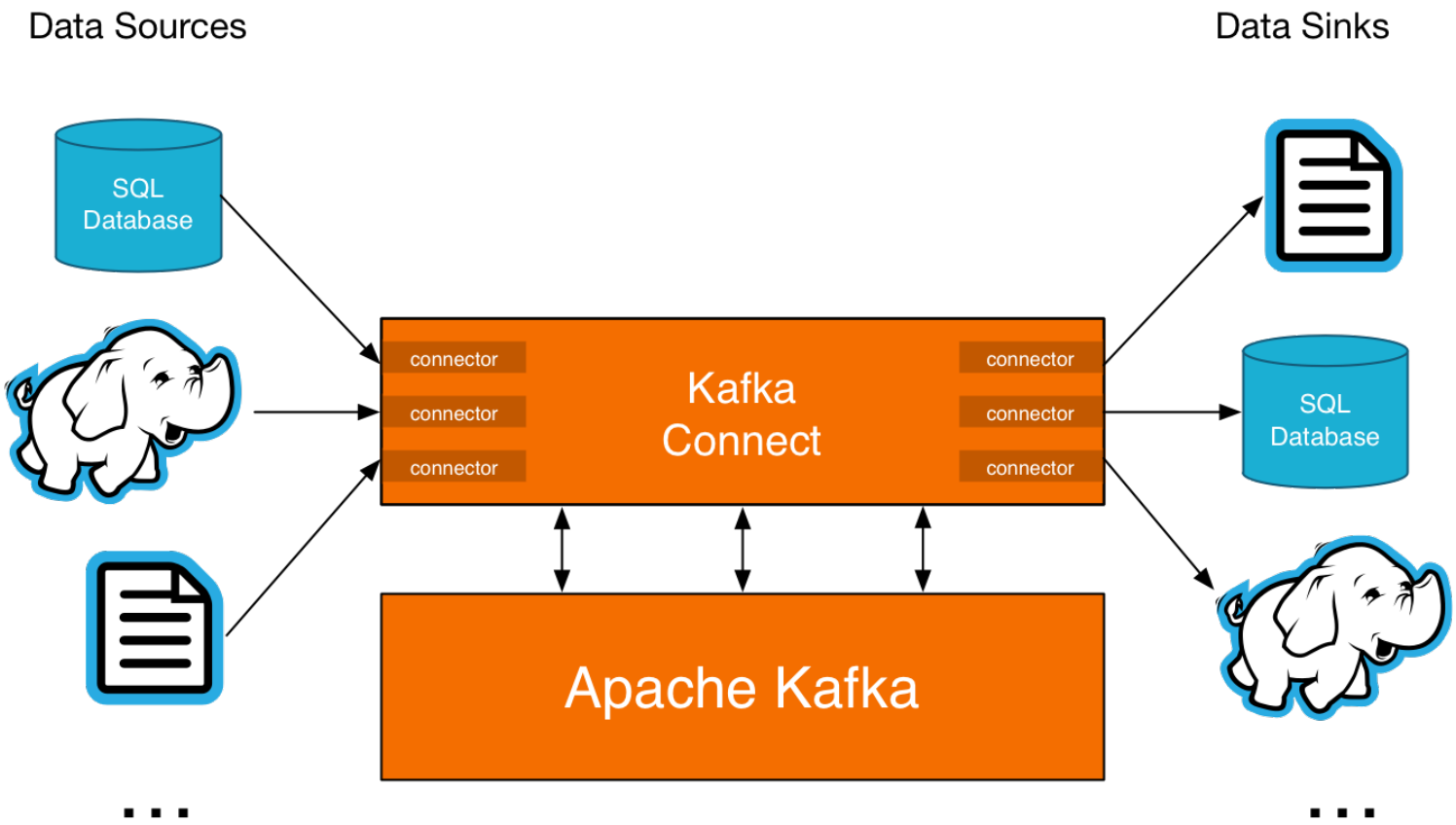
Kafka Performance at our own infrastructure => 6 brokers (VM) / 1 cluster

http://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines

- 445’622 messages/second
- 31 MB / second
- 3.0405 ms average latency between producer / consumer

Kafka Connect

Kafka Connect Architecture



Source: Confluent

Kafka Connector Hub – Certified Connectors

CONNECTOR	TAGS	DEVELOPER	SUPPORT
HDFS (Sink)	HDFS, Hadoop, Hive	Confluent	Confluent
JDBC (Source)	JDBC, MySQL	Confluent	Confluent
Attunity (Sink)	CDC, Oracle	Attunity	Attunity
Couchbase (Source)	Couchbase, NoSQL	Couchbase	Couchbase
JustOne (Sink)	Postgress	JustOne	JustOne
Striim (Source)	CDC, Oracle, MS SQLServer	Striim	Striim
Syncsort DMX (Source)	DB2, IMS, VSAM, CICS	Syncsort	Syncsort
Syncsort DMX (Sink)	DB2, IMS, VSAM, CICS	Syncsort	Syncsort
Vertica (Source)	Vertica	HP Enterprise	HP Enterprise
Vertica (Sink)	Vertica	HP Enterprise	HP Enterprise

Source: http://www.confluent.io/product/connectors

Kafka Connector Hub – Additional Connectors

CONNECTOR	TAGS	DEVELOPER	SUPPORT
Apache Ignite (Source)	File System	Community	Community
Apache Ignite (Sink)	File System	Community	Community
Bloomberg Ticker (Source)		Community	Community
Cassandra (Source)	Cassandra, Datastax	Community	Community 1 Community 2
Cassandra (Sink)	Cassandra, Datastax	Community	Community 1 Community 2
Elastic Search (Sink)	search, elastic, log, analytics	Community	Community 1 Community 2 Community 3
HBase Sink	HBase, NoSQL	Community	Community
Kudu (Sink)	Kudu	Community	Community
Mixpanel (Source)	analytics	Community	Community
MongoDB (Source)	Mongo, MongoDB, NoSQL	Community	Community
MQTT (Source)	MQTT, messaging	Community	Community
MySQL CDC - Debezium (Source)	MySQL, CDC, Oracle	Community	Community

Source: http://www.confluent.io/product/connectors

Kafka Connect – Twitter example

```
./connect-standalone.sh ../demo-config/connect-simple-source-standalone.properties
../demo-config/twitter-source.properties
```

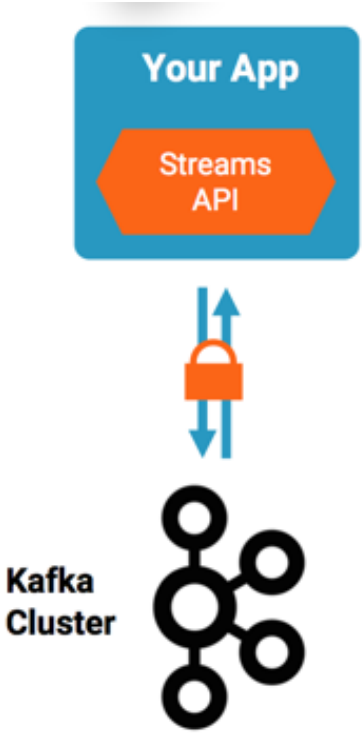
```
bootstrap.servers=localhost:9095,localhost:9096,localhost:9097

key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter
...
```

```
name=twitter-source
connector.class=com.eneco.trading.kafka.connect.twitter.TwitterSourceConnector
tasks.max=1
topic=tweets
twitter.consumerkey=<consumer-key>
twitter.consumersecret=<consumer-secret>
twitter.token=<token>
twitter.secret=<token-secret>
track.terms=bigdata
```

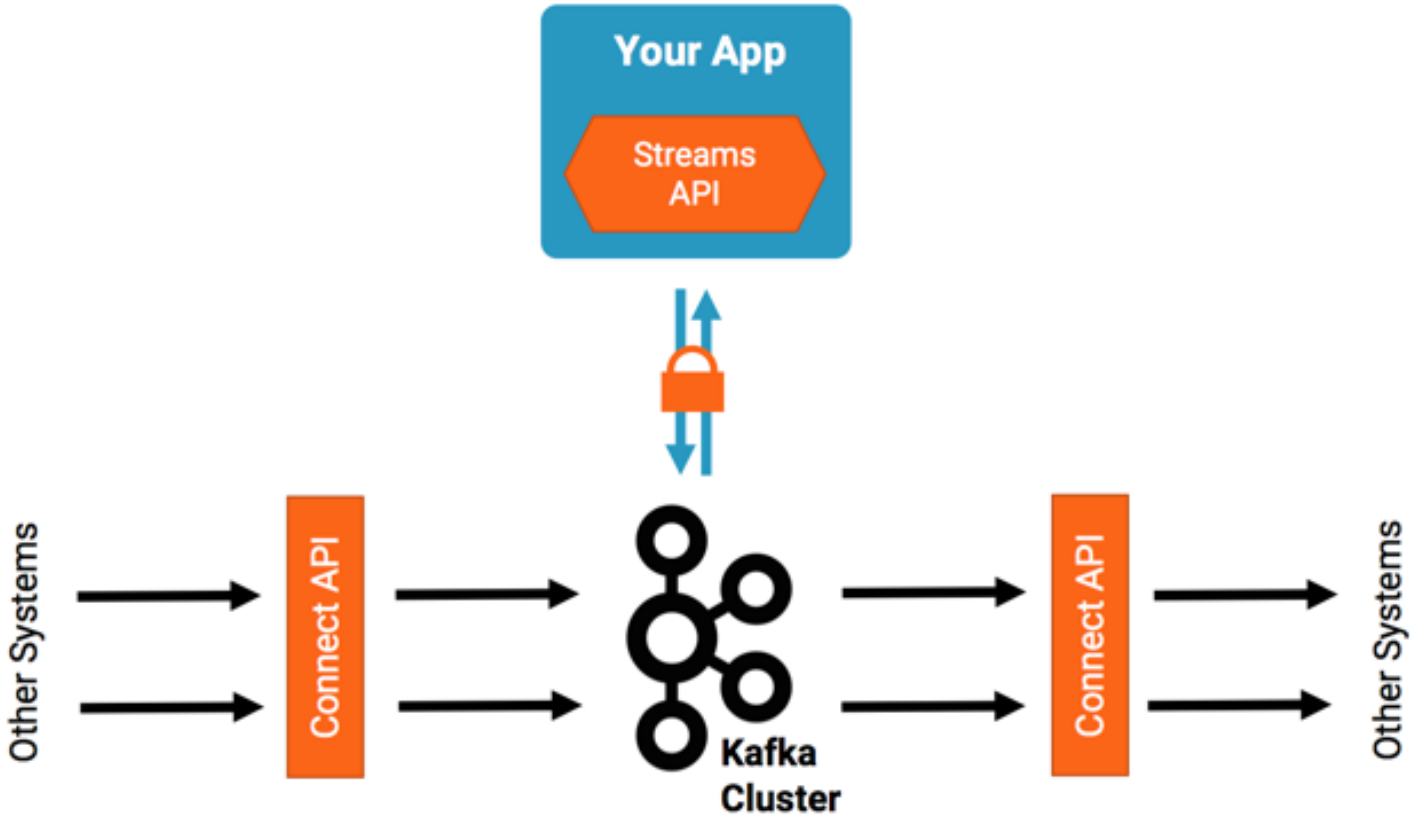
Kafka Streams

Kafka Streams

- Designed as a **simple and lightweight library in Apache Kafka**
- no external dependencies on systems other than Apache Kafka
- Part of open source Apache Kafka, introduced in 0.10+
- Leverages **Kafka as its internal messaging layer**
- agnostic to resource management and configuration tools
- Supports **fault-tolerant local state**
- Event-at-a-time processing (not microbatch) with millisecond latency
- Windowing with out-of-order data using a Google DataFlow-like model



Streams API in the context of Kafka



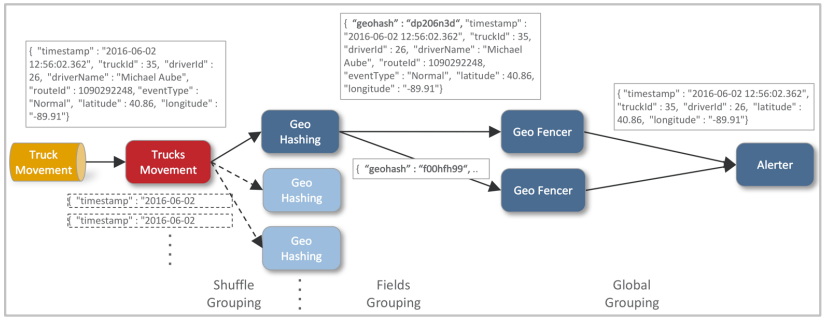
Source: Confluent

Kafka and "Big Data" / "Fast Data" Ecosystem

Kafka and the Big Data / Fast Data ecosystem

Kafka integrates with many popular products / frameworks

- Apache Spark Streaming
- Apache Flink
- Apache Storm



Storm built-in Kafka Spout to consume events from Kafka

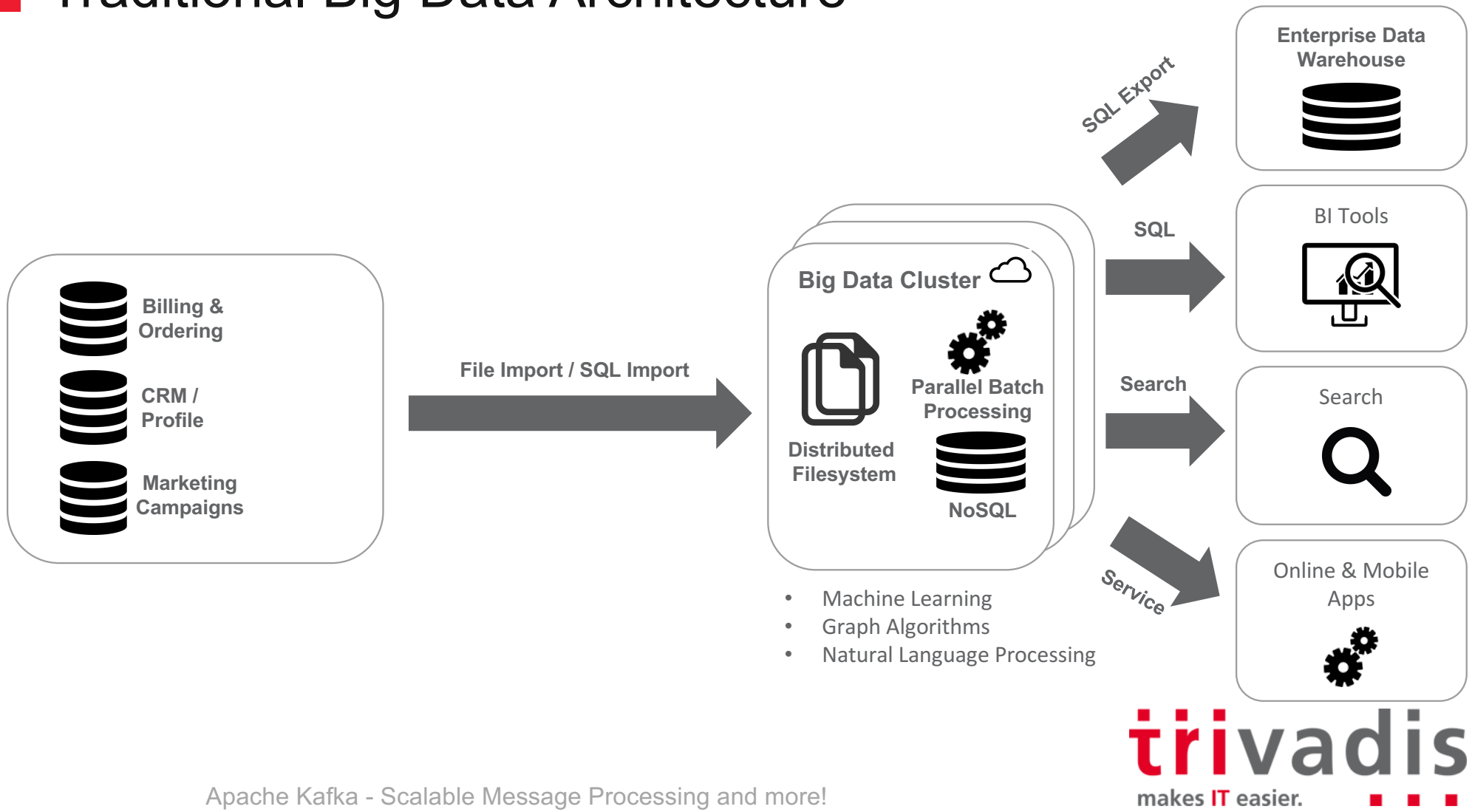
- Apache NiFi
- Streamsets
- Apache Flume
- Oracle Stream Analytics
- Oracle Service Bus
- Oracle GoldenGate
- Spring Integration Kafka Support
- …

Kafka in “Enterprise Architecture”

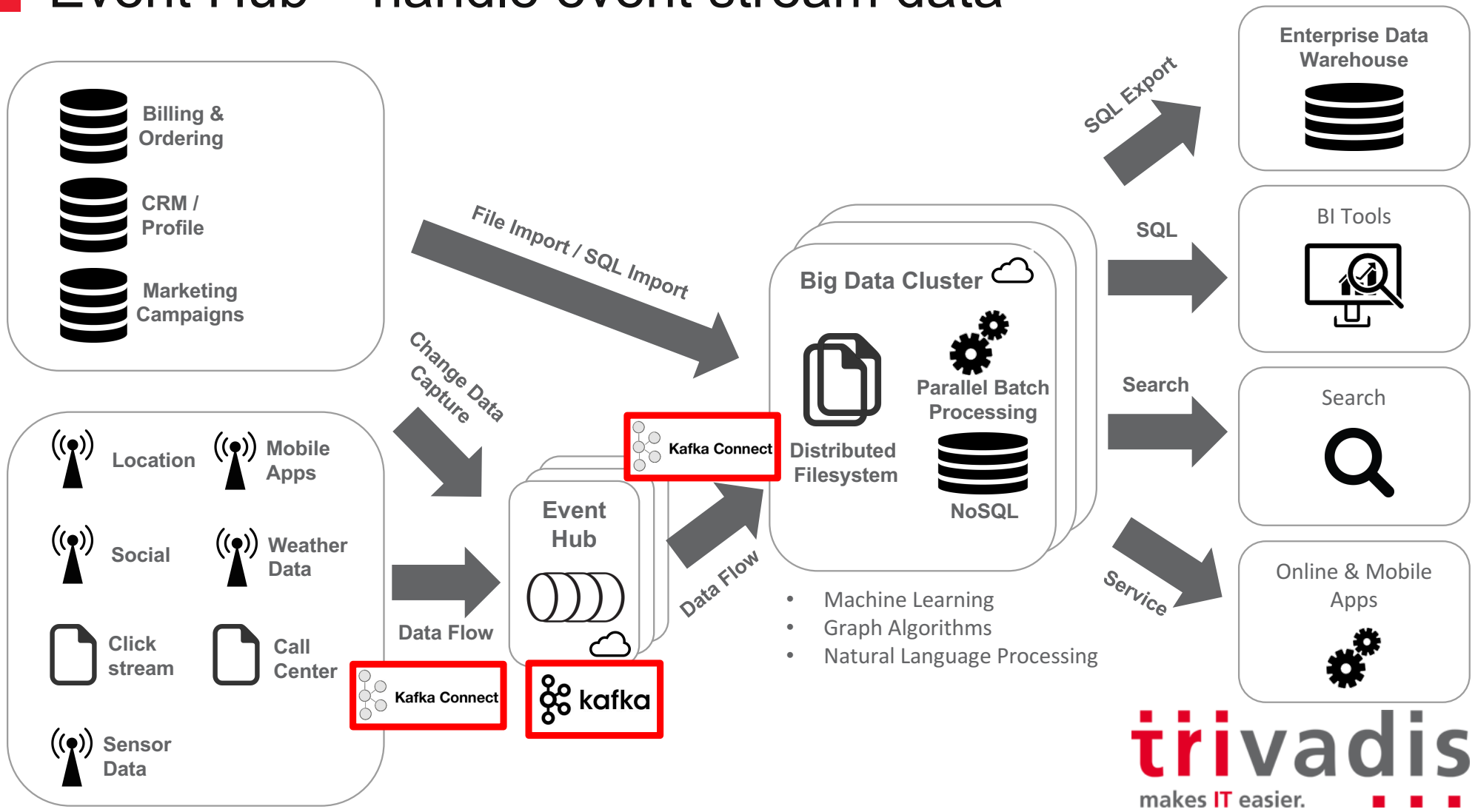
Apache Kafka - Scalable Message Processing and more!

trivadis
makes IT easier.

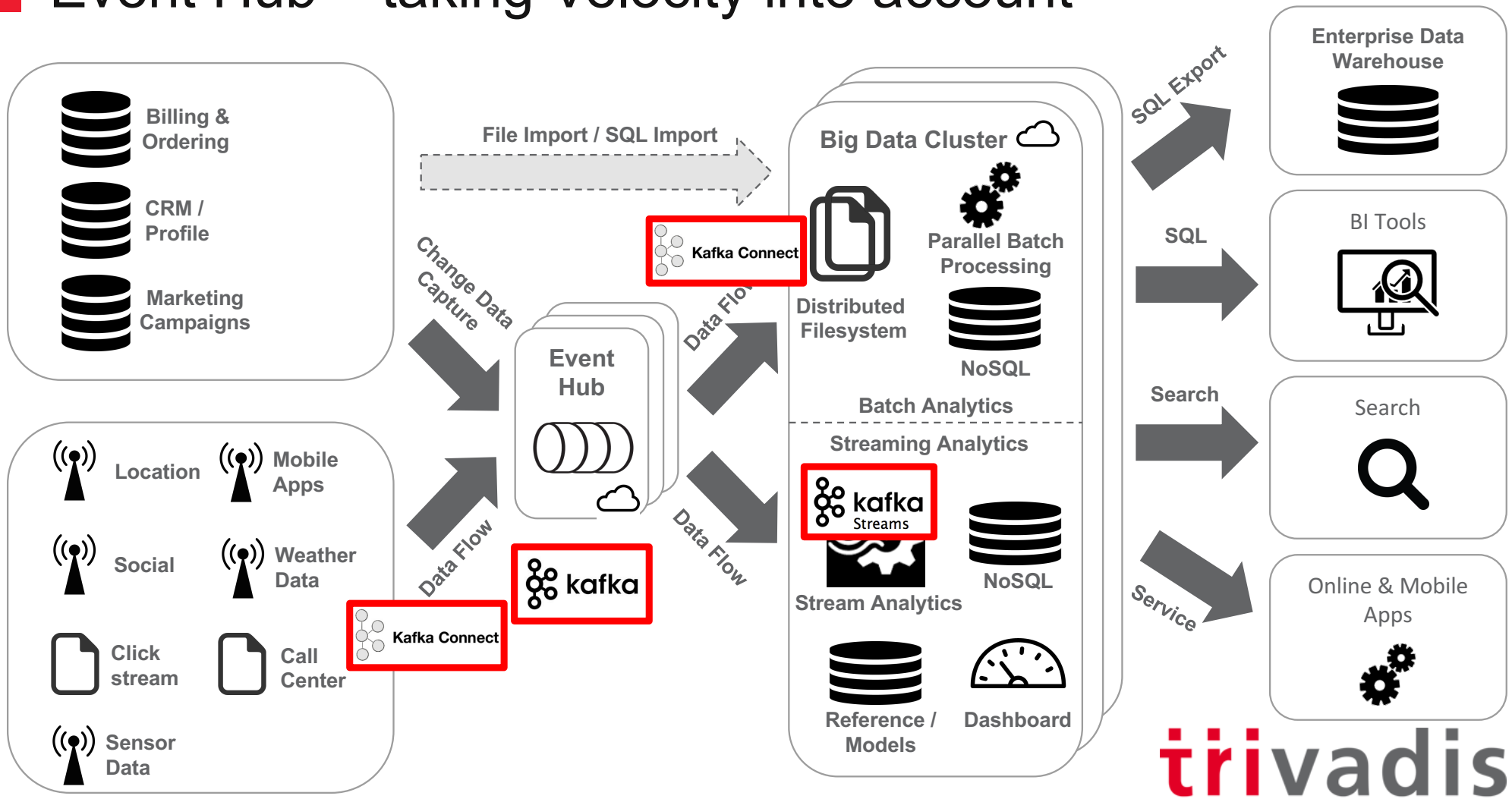
Traditional Big Data Architecture



Event Hub – handle event stream data

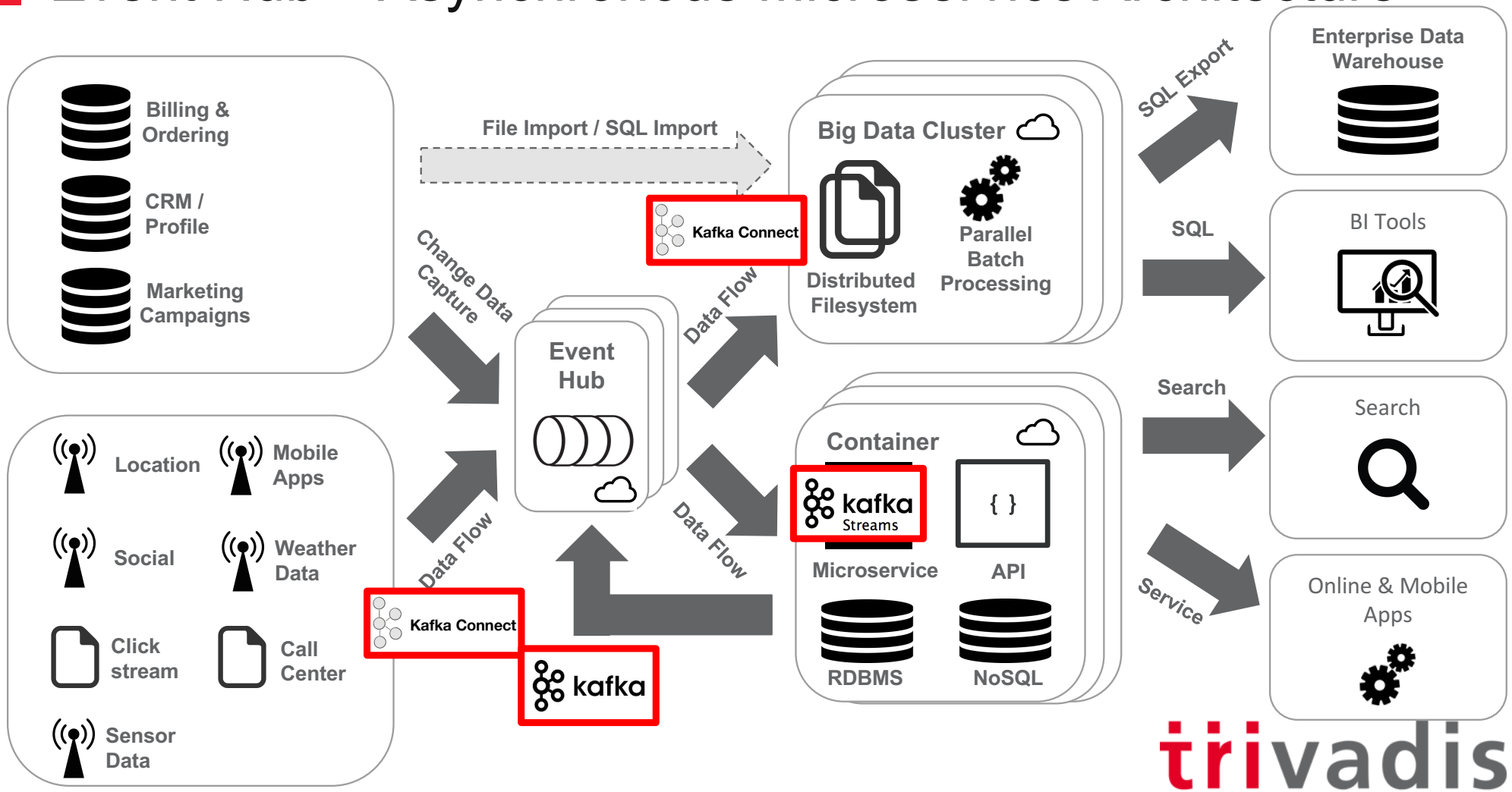


Event Hub – taking Velocity into account



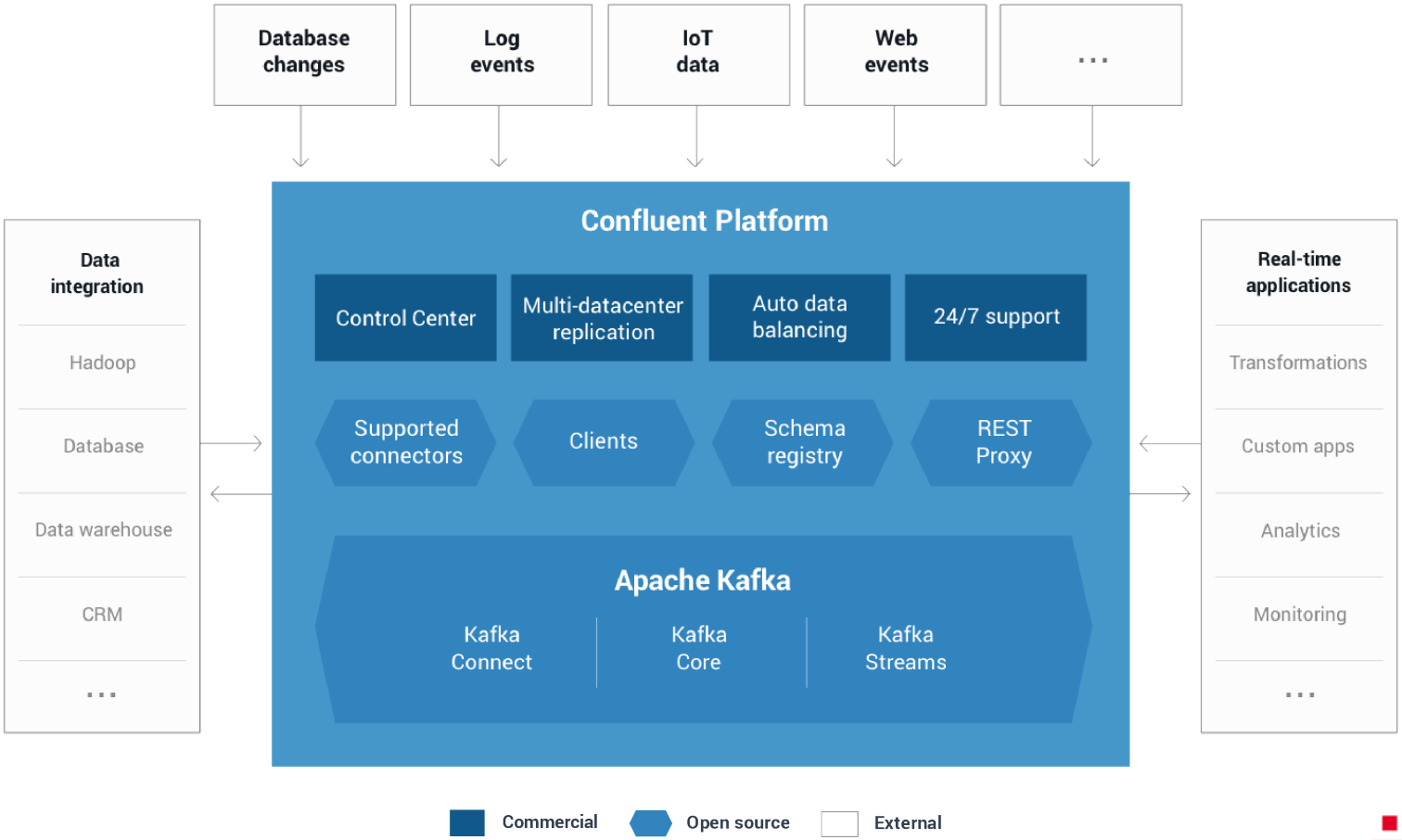
trivadis
makes IT easier.

Event Hub – Asynchronous Microservice Architecture



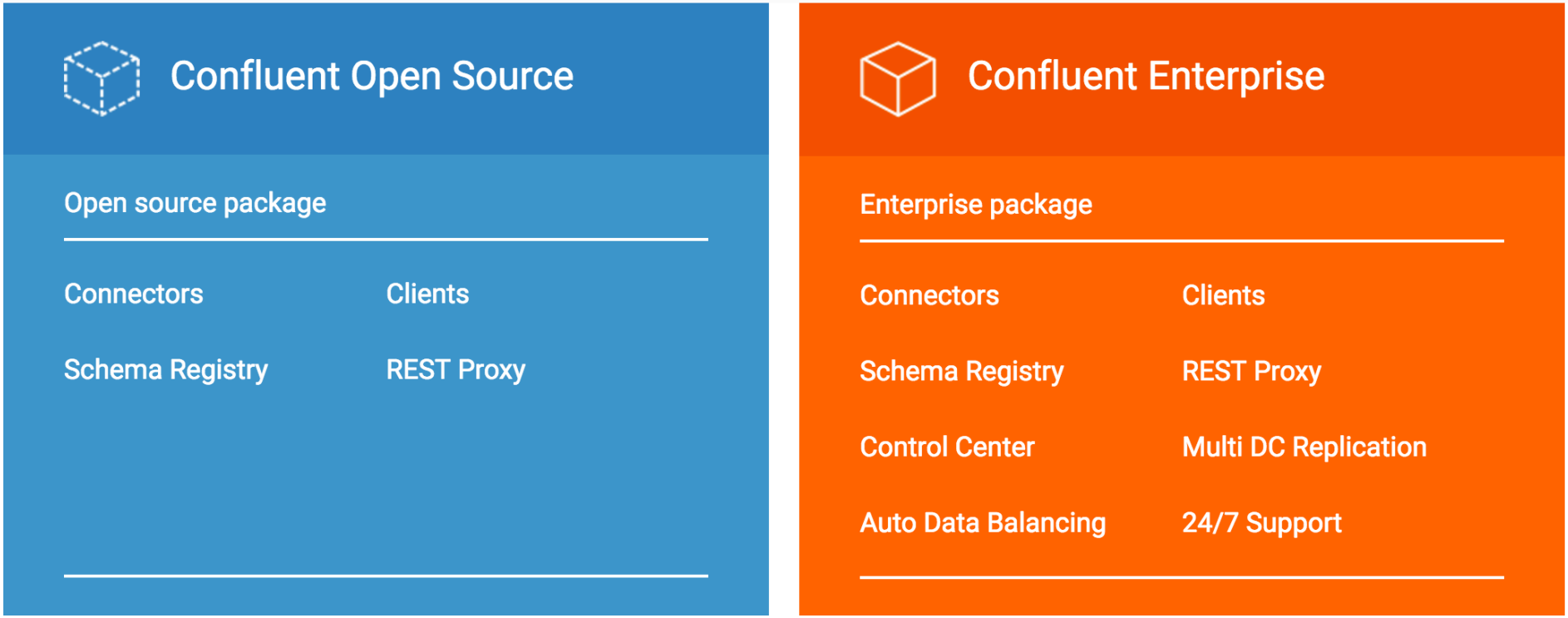
Confluent Platform

Confluent Data Platform 3.2



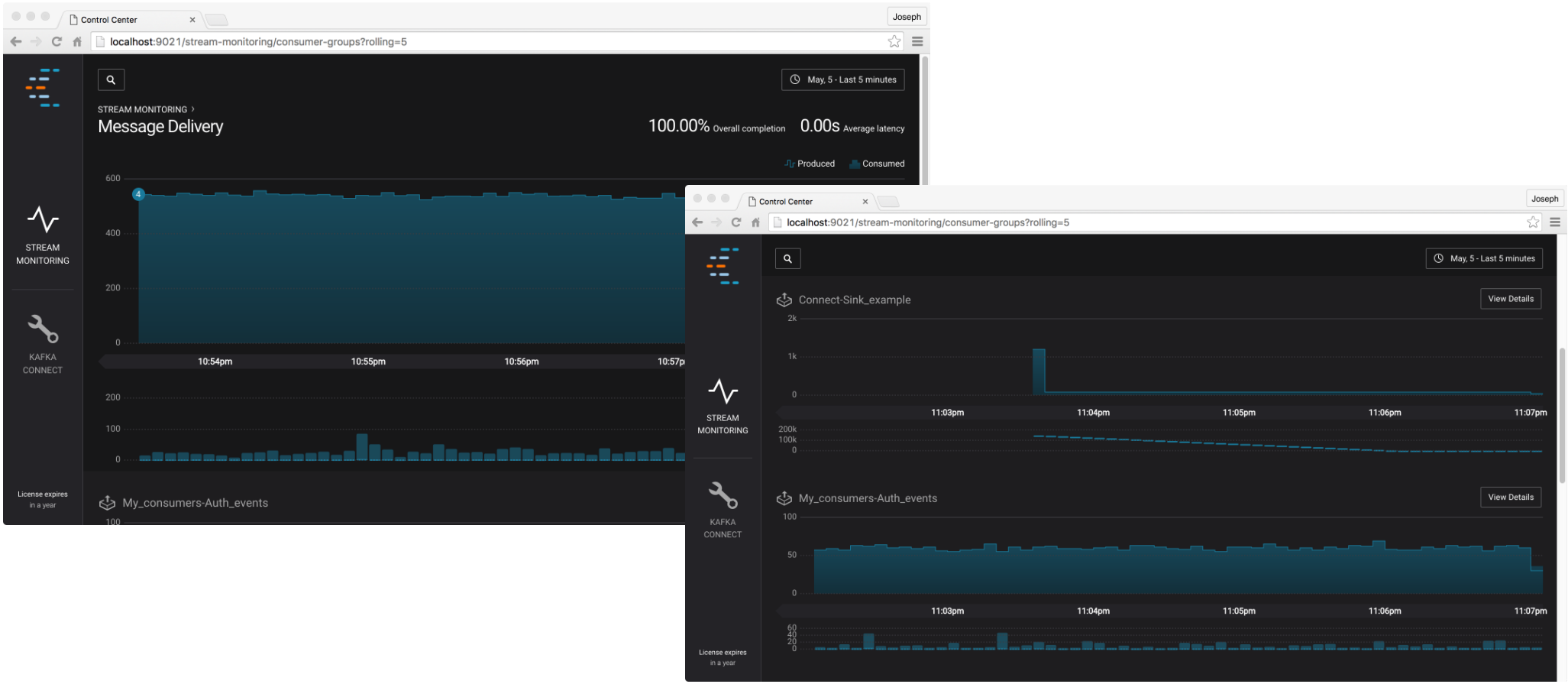
Source: Confluent

Confluent Data Platform 3.2



Source: Confluent

Confluent Enterprise – Control Center



Source: Confluent

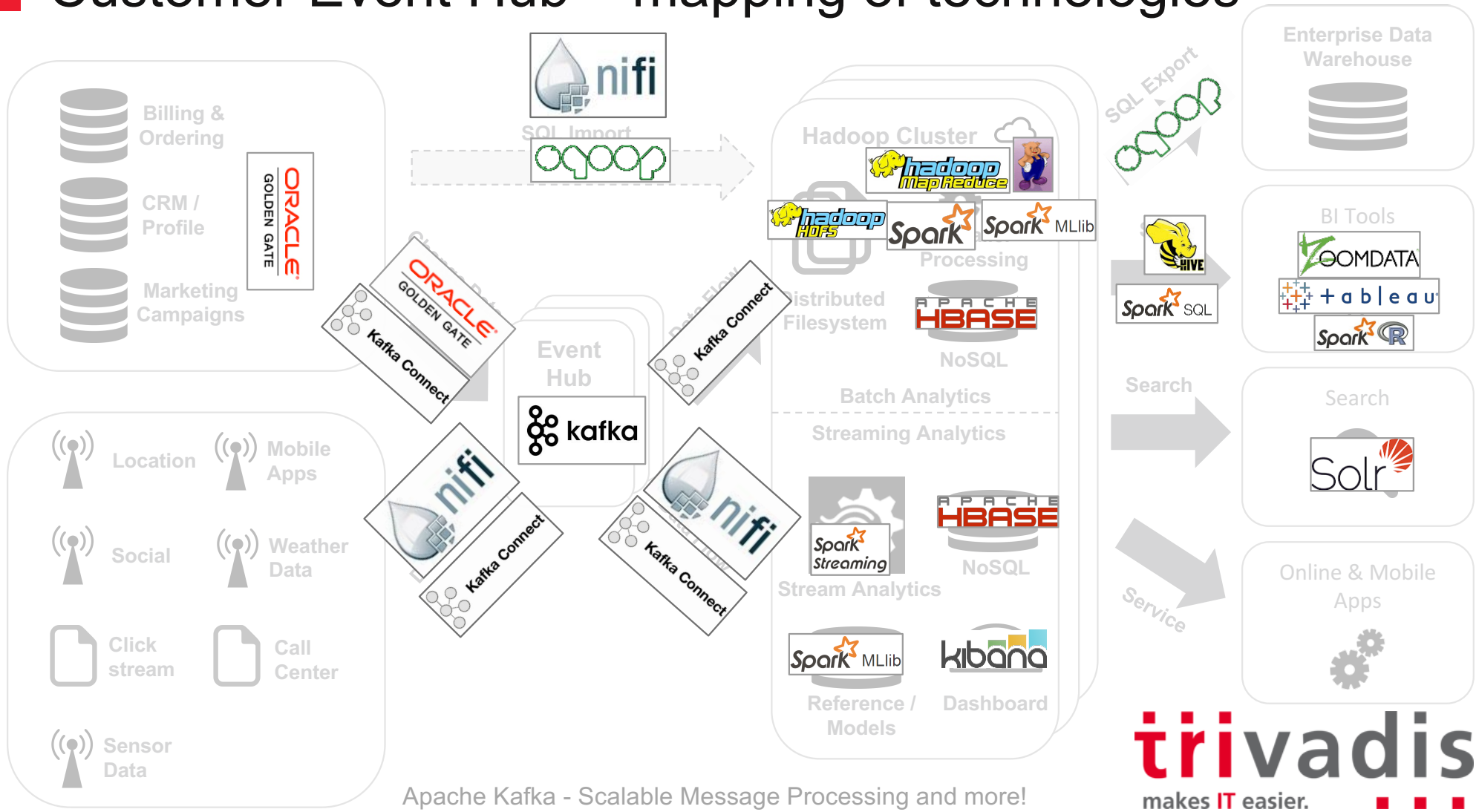
trivadis makes IT easier.

Summary

Summary

- Kafka can scale to millions of messages per second, and more
- Easy to start in a Proof of Concept (PoC), but more to invest to setup a production environment
- Monitoring is key
- Vibrant community and ecosystem
- Fast paced technology
- Confluent provides distribution and support for Apache Kafka
- Oracle Event Hub Service offers a Kafka Managed Service

Customer Event Hub – mapping of technologies



Apache Kafka - Scalable Message Processing and more!

trivadis
makes IT easier.

trivadis
makes IT easier.