Introduction to Apache Kafka

- Overview
- Architecture
- Kafka Connect
- Kafka Stream
- KSQL
- Cluster Installation

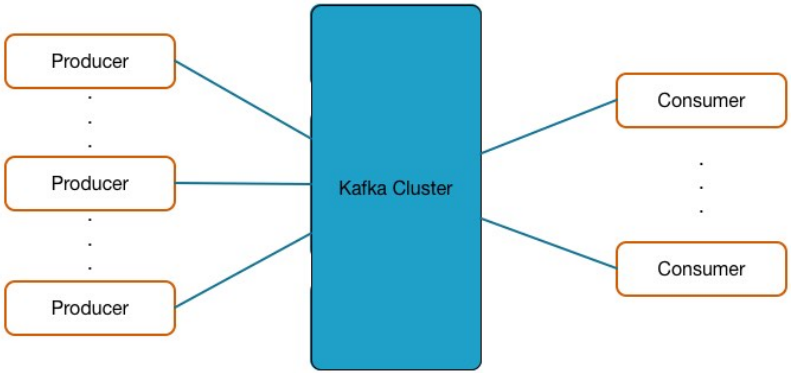
Apache Kafka - Overview

- Kafka was designed to solve both problems
  - Simplifying data pipelines
  - Handling streaming data
- Created by Linkedin in 2010
- Apache open source project since 2012

https://www.quora.com/What-is-the-relation-between-Kafka-the-writer-and-Apache-Kafka-the-distributed-messaging-system

Apache Kafka - Overview

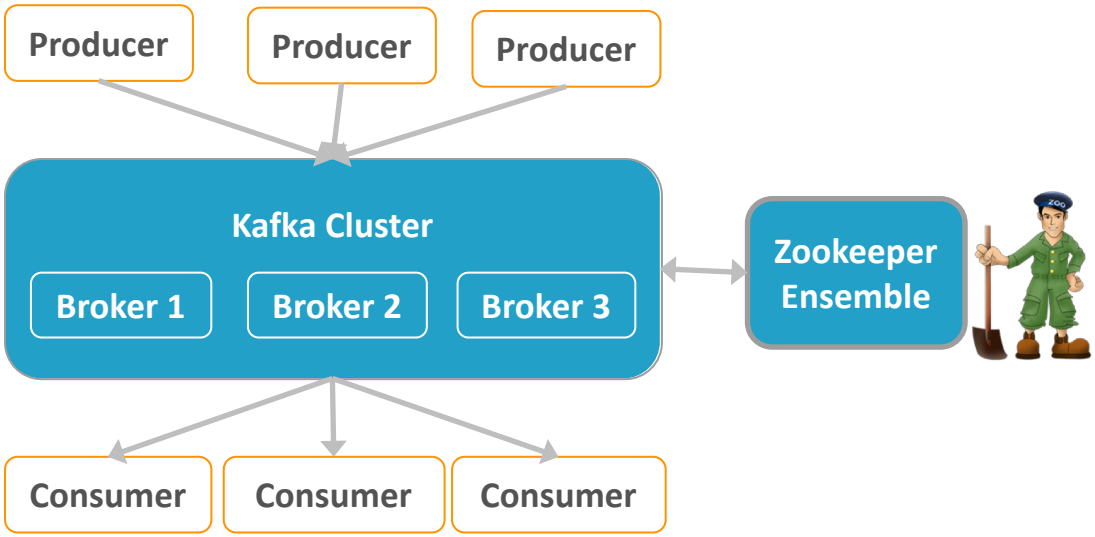
- Kafka decouples data source and destination systems
  - Publish/subscribe architecture
- All data sources write their data to the Kafka cluster
- All systems wishing to use the data read from Kafka



Architecture

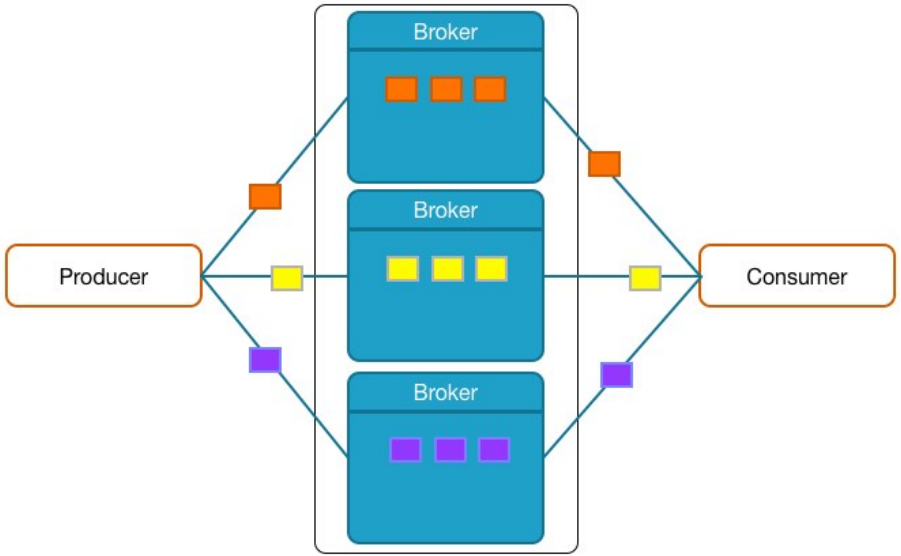
Apache Kafka - Architecture

- **Producer** send data to the cluster
- **Consumers** read data
- **Brokers** are the main component for storage and messaging

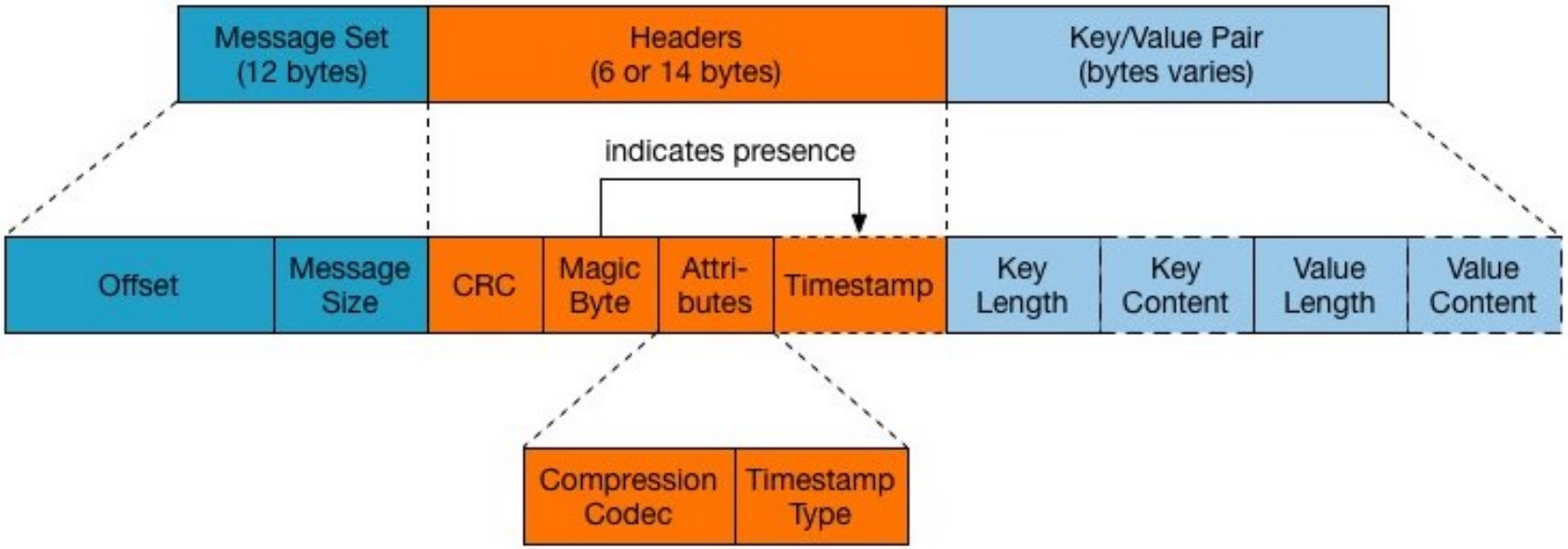


Apache Kafka - Architecture

- Basic unit in Kafka is a **message**
- **Producers** write **messages** to **Brokers**
- **Consumers** read **messages** from **Brokers**
- A **message** is a key-value pair



Apache Kafka - Architecture



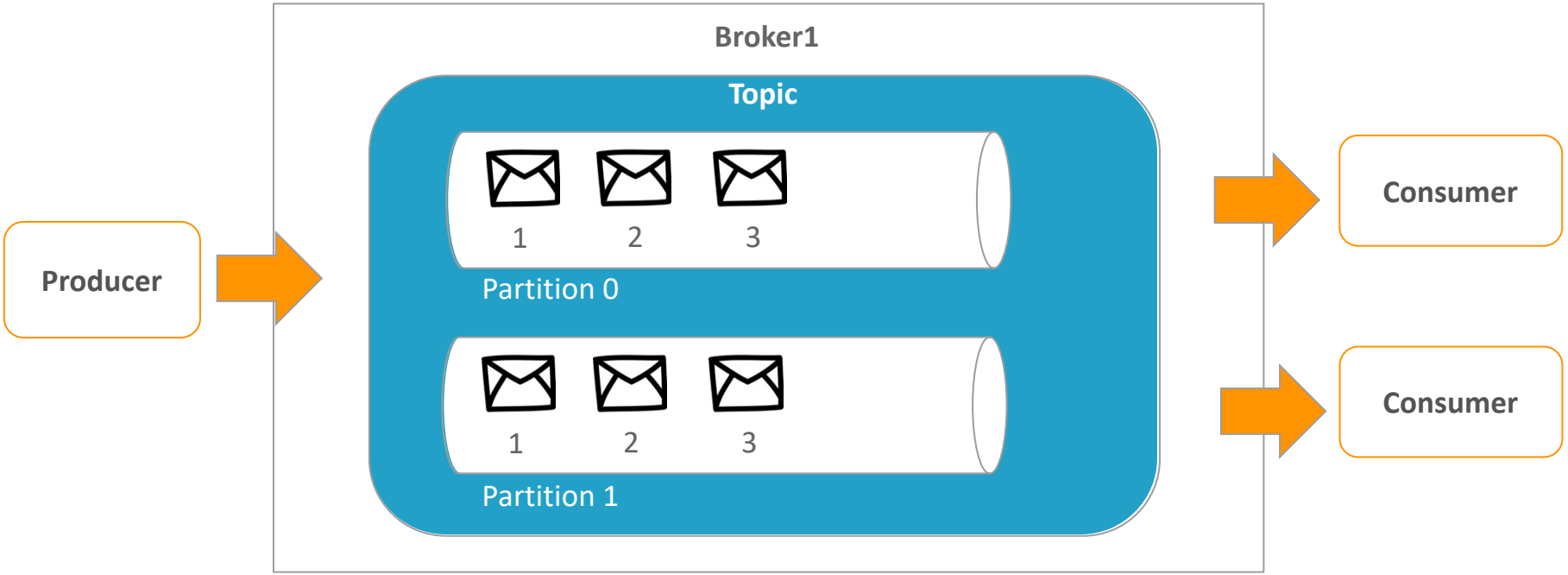
http://kafka.apache.org/documentation.html#messageformat

Apache Kafka - Architecture

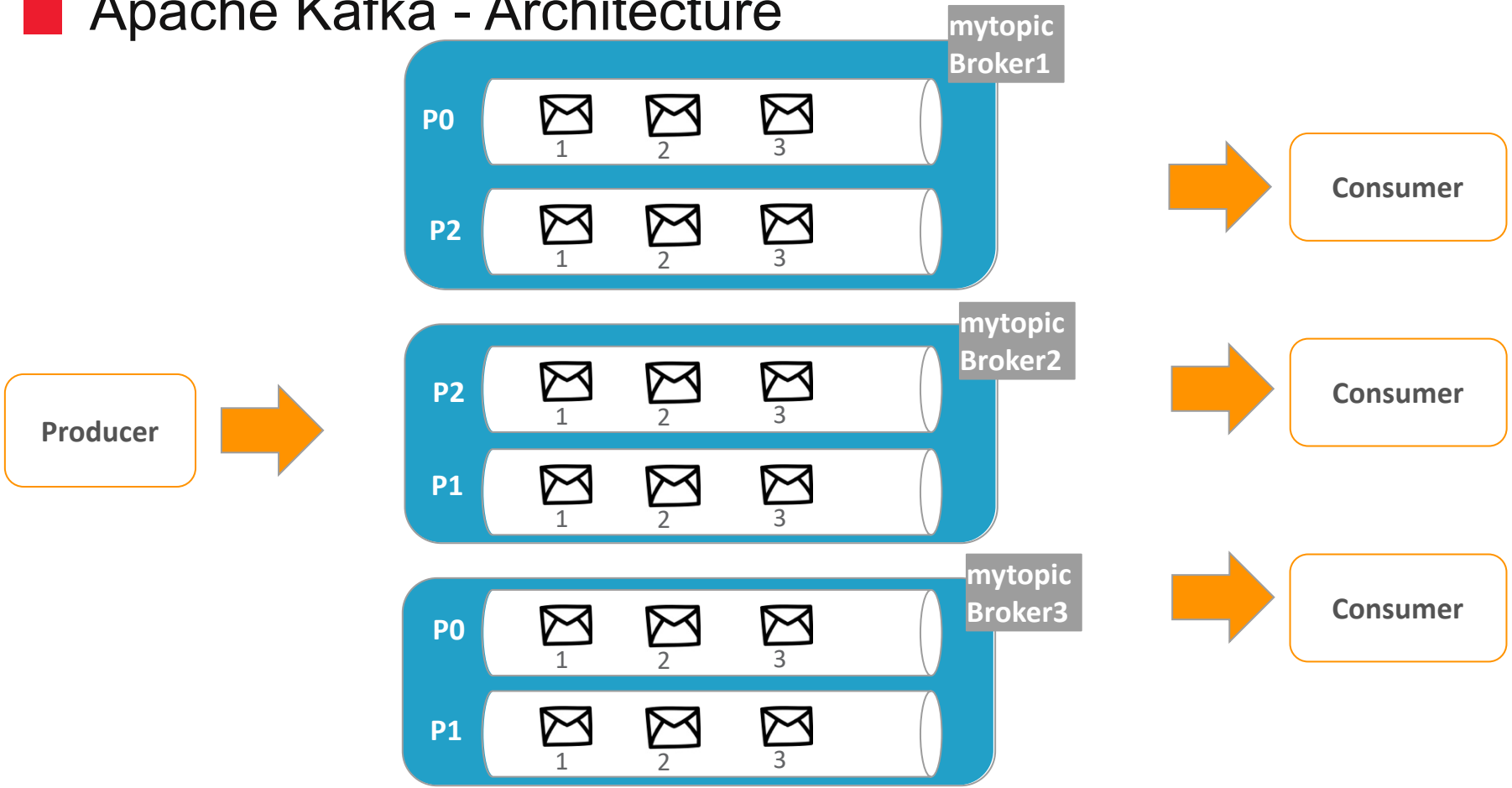
- Kafka maintains streams of **messages** in **Topics**
  - Logical representation
  - Categorize **messages** in groups
- Developer decides which **Topics** exist
  - **Topic** is auto created on first use
- No limit in number of **Topics**
- One or more **Producer** can write to the same **Topic**

Apache Kafka - Architecture

- **Topics** are split into **partitions**
- **Partitions** are distributed across the **Brokers**



Apache Kafka - Architecture



Apache Kafka - Producers

- Producers write data as messages to the Kafka cluster
- Can be written in any language
- A command line tool exists to send messages to the cluster
  - For testing, debugging, etc

```
kafka-console-producer --broker-list localhost:9092 --topic test
```

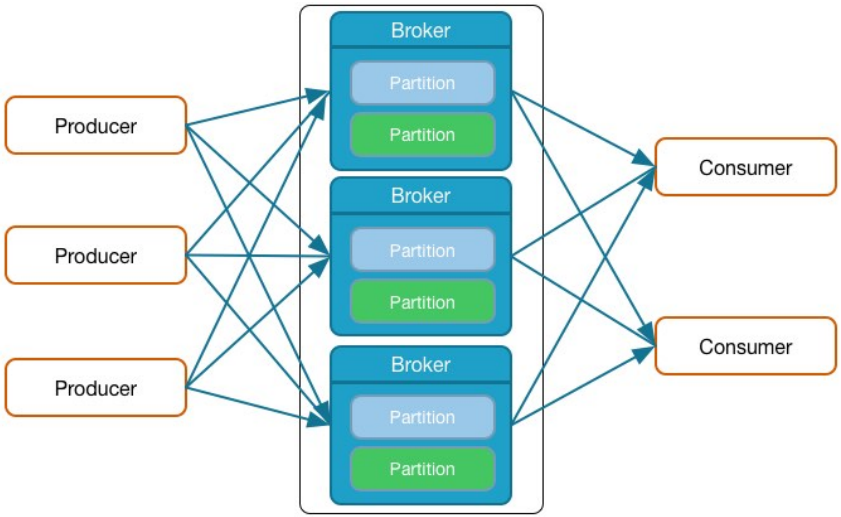
Apache Kafka - Producers

- Producers use a partitioning strategy to assign each message to a Partition
  - Default is hash key of message key

- Partitioning is used for
  - Load balancing → share load across brokers
  - semantic partioning →user-specified key allows locality-sensitive message processing

Apache Kafka - Brokers

- Receive and store the messages send by Producers
- A Kafka cluster has typically 3 or more brokers
  - Each can handle hundreds of thousands, or millions, of messages per second

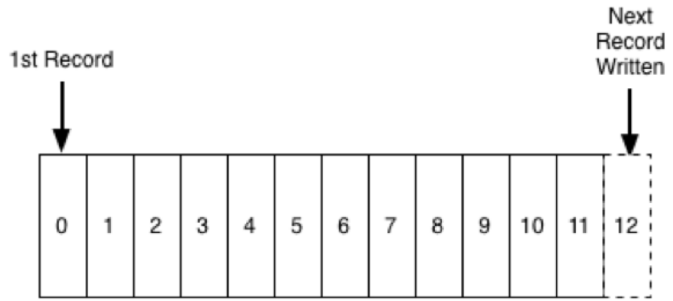


Apache Kafka - Brokers

- Messages in a Topic are spread across Partitions in different Brokers

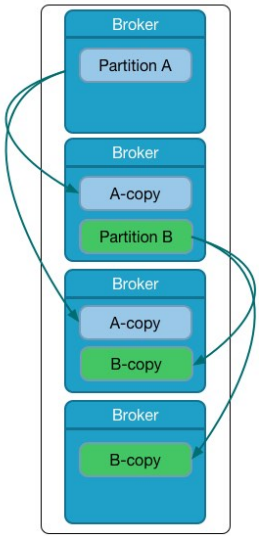
- Each Partition is stored on the Broker's disk as one or more log files
  - Do not get messed up with log4j logfiles
- Each message in the log is identified by its offset number

- Retention policy for messages to manage log file growth
  - Per Topic



Apache Kafka – Broker

- Partitions can be replicated across multiple Brokers
- Fault tolerance if one Broker goes down
  - Automatically handled by Kafka

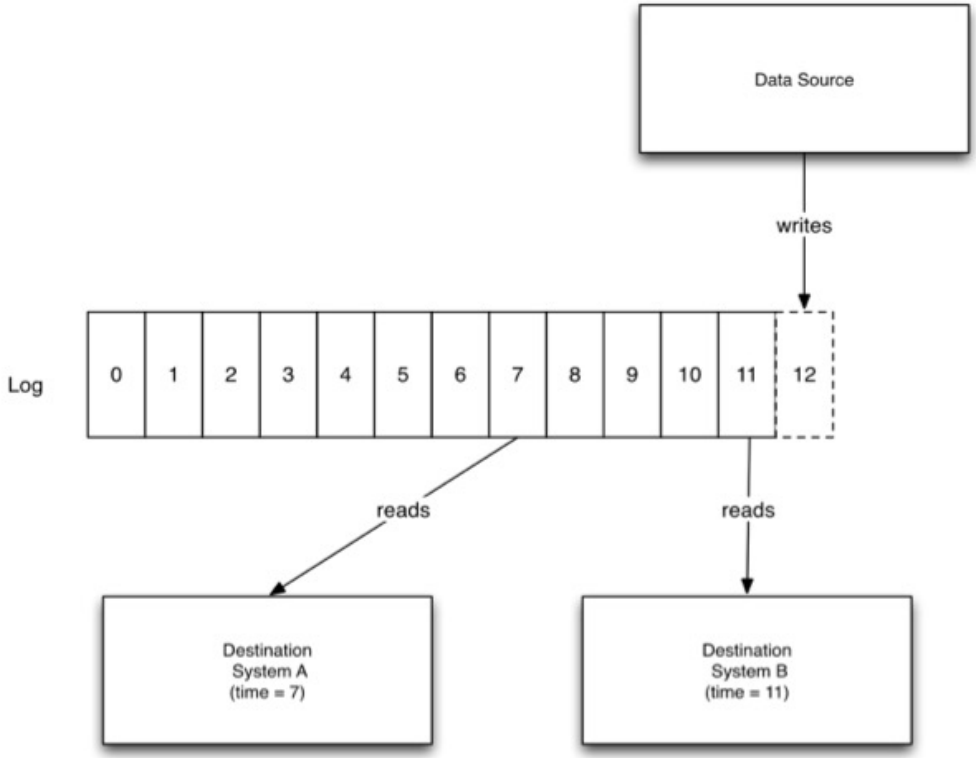


Apache Kafka - Consumer

- Consumers pull messages from one or more Topics in the cluster
  - Consumer will retrieve messages as they are written to the topic
- The Consumer Offset keeps track of the latest message read
  - Consumer Offset can be changed (if necessary) to reread messages
- The Consumer Offset is stored in a special Kafka Topic
- A command-line Consumer tool exists to read messages from the cluster

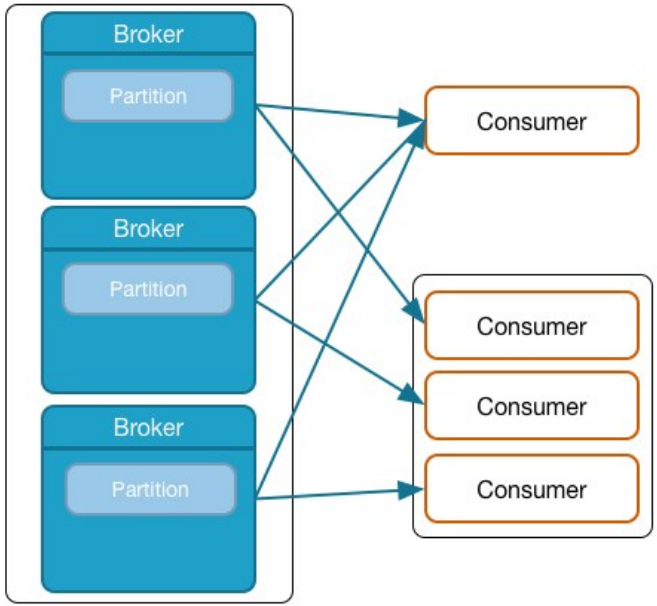
```
kafka-console-consumer --zookeeper localhost:2181 --topic test --from-beginning

kafka-console-consumer --bootstrap-server localhost:9092 --from-beginning --topic test
```

Apache Kafka - Consumers

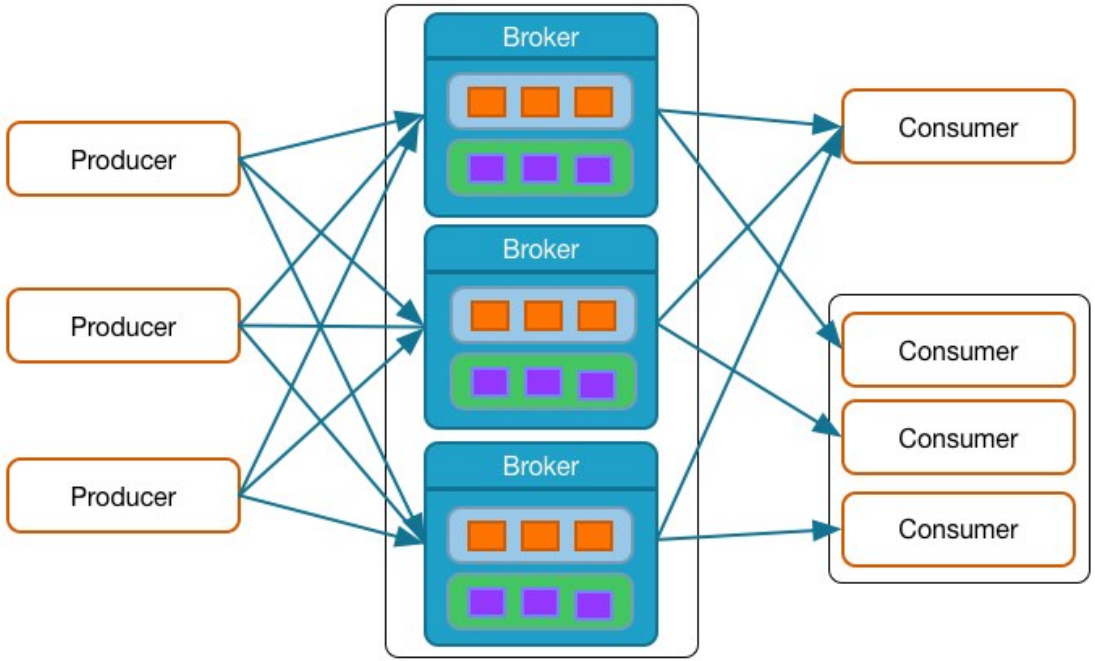


Apache Kafka - Consumers

- Different Consumers could read from the same Topic
- Multiple Consumers can be combined in a **Consumer Group**
  - Scaling
  - Each Consumer uses a subset of partitions



Apache Kafka – Cluster overview



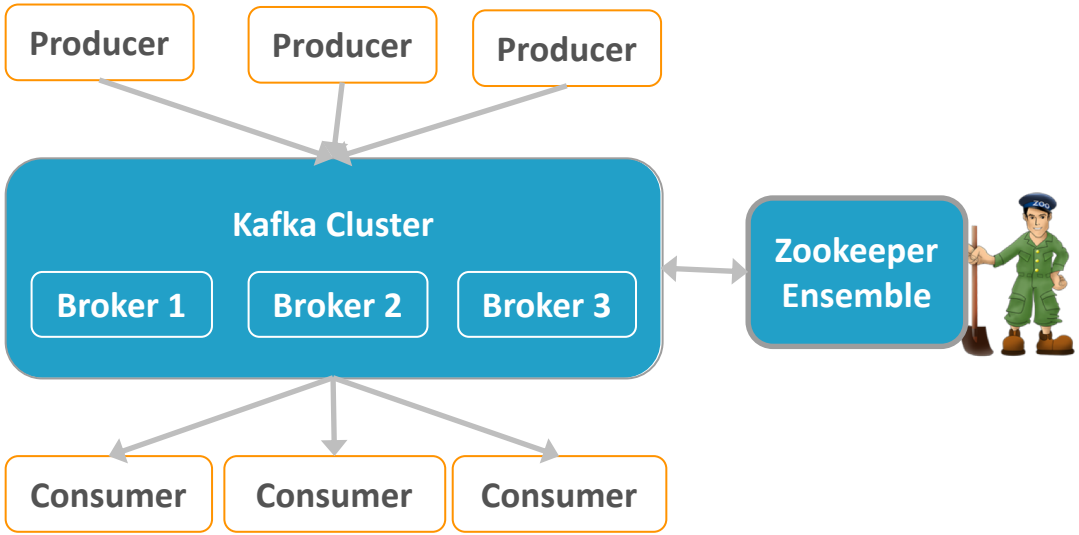
Apache Kafka - Zookeeper

- Zookeeper is a centralized service which can be used by distributed applications
  - Open source Apache project
  - Distributed synchronisation
  - Enables highly reliable distributed coordination
  - Maintains configuration infos

- Typically consists of 3 or 5 servers in a quorum

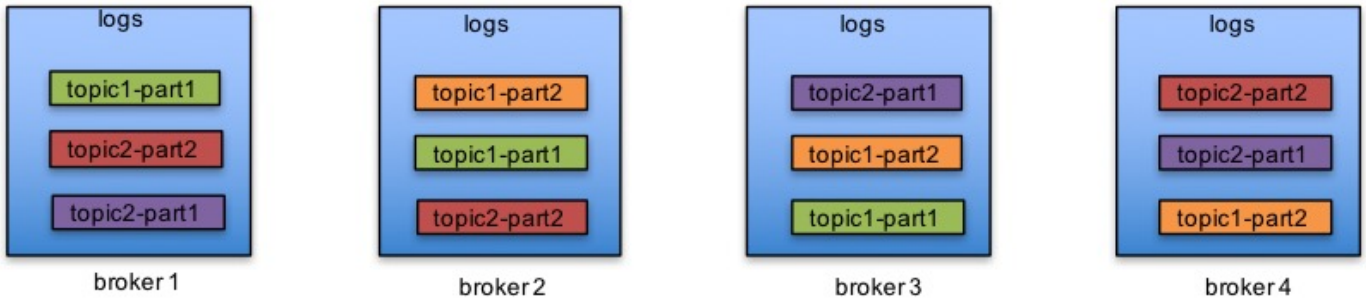
Apache Kafka - Zookeeper

- Kafka Brokers use Zookeeper for important internal features
  - Cluster Management
  - Failure detection and recovery
  - Access Control List storage



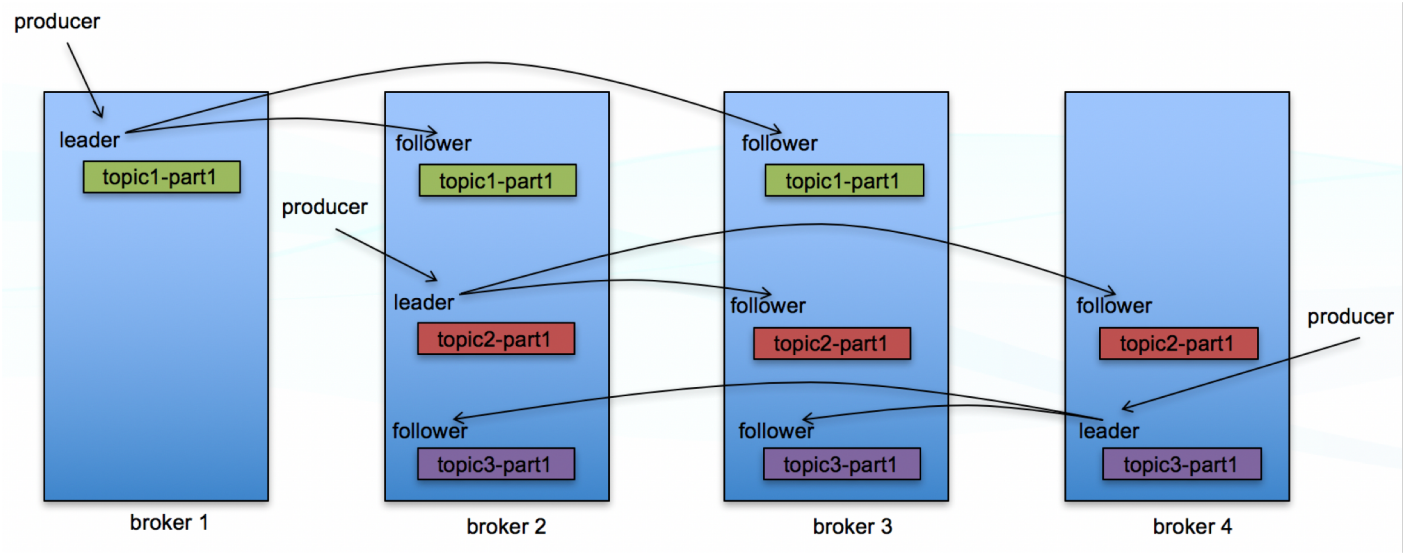
Apache Kafka – Replication and Durability

- Each Partition can have replicas
- Replicas will be placed on different Brokers
- Replicas are spread evenly among the Brokers



Apache Kafka – Replication and Durability

- Distributed partition leaders
  - Ideally spreaded evenly across the cluster



Apache Kafka – Replication and Durability

- Increase the replication factor for higher durability
- For auto-created Topics
  - default.replication.factor (Default: 1)
  - Configuration on each broker (server.properties)
- For manually created topics

```
kafka-topics --create --zookeeper zk_host:2181 --partitions 2 --replication-factor 3 --topic my_topic
```

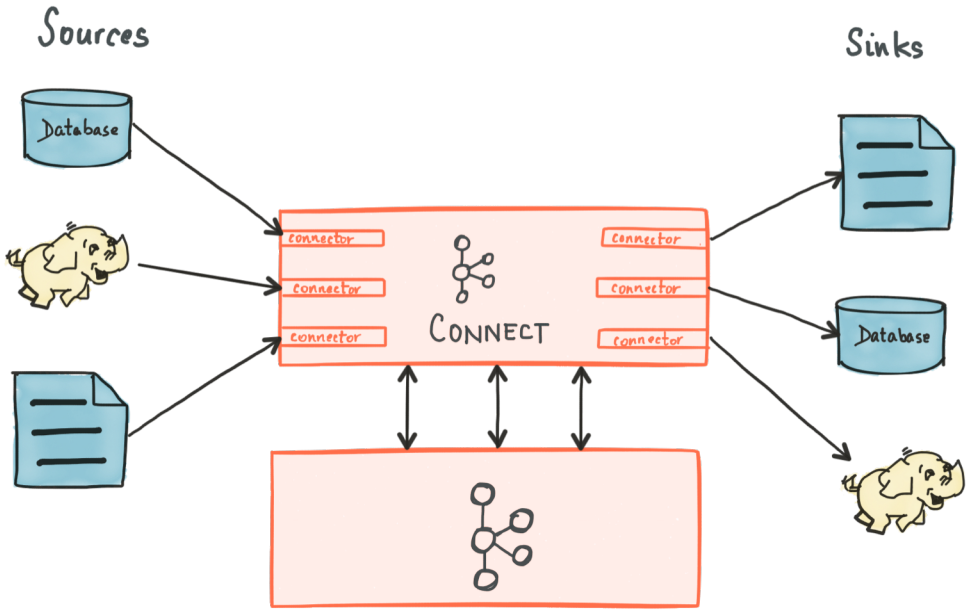
Apache Kafka – Replication and Durability

- Producers can control acknowledgement setting

Value	Latency	Durability	Description
0	No network delay	low	Producer doesn‘t wait for leader
1 (default)	1 network roundtrip	medium	Producer waits for Leader Leader sends ack No wait for follower
all(-1)	2 network roundtrips	high	Producer waits for Leader Leader sends ack when all In-Sync replicas have send the ack

Kafka Connect

Kafka Connect



Source: https://www.confluent.io/

Kafka Connect

- Framework for streaming data between Kafka and other systems
  - Open source
- Useful for
  - Stream an entire SQL database to Kafka
  - Stream Kafka topics into HDFS
- Kafka Connect has benefits over „do-it-yourself“ Producers and Consumers
  - Tested Connectors

Kafka Connect

CONNECTOR	TAGS	DEVELOPER/SUPPORT	DOWNLOAD
Amazon S3 (Sink)	S3	Confluent	Confluent
Elasticsearch (Sink)	search, Elastic, log, analytics	Confluent	Confluent
HDFS (Sink)	HDFS, Hadoop, Hive	Confluent	Confluent
JDBC (Source)	JDBC, MySQL	Confluent	Confluent
JDBC (Sink)	JDBC, MySQL	Confluent	Confluent

Kafka Connect

CONNECTOR	TAGS	DEVELOPER/SUPPORT	DOWNLOAD
Attunity (Source)	CDC	Attunity	Attunity
Azure IoTHub (Source)	IoT, messaging	Microsoft	Azure
Couchbase (Source)	Couchbase, NoSQL	Couchbase	Couchbase
Couchbase (Sink)	Couchbase, NoSQL	Couchbase	Couchbase
Dbvisit Replicate (Source)	CDC, Oracle	Dbvisit	Dbvisit
IBM Data Replication (Source)	CDC	IBM	IBM
JustOne (Sink)	Postgres	JustOne	JustOne
Kinetica (Source)	GPU, RDBMS	Kinetica	Kinetica
Kinetica (Sink)	GPU, RDBMS	Kinetica	Kinetica
Oracle GoldenGate	CDC, Oracle	Oracle	Oracle
SAP HANA (Sink)	HANA, RDBMS	SAP	SAP
SAP HANA (Source)	HANA, RDBMS	SAP	SAP
Striim (Source)	CDC, MS SQLServer, Oracle, MySQL	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
VoltDB (Sink)	VoltDB, NewSQL	VoltDB	VoltDB
Xenon (Sink)	Xenon	Levyx	Xenon

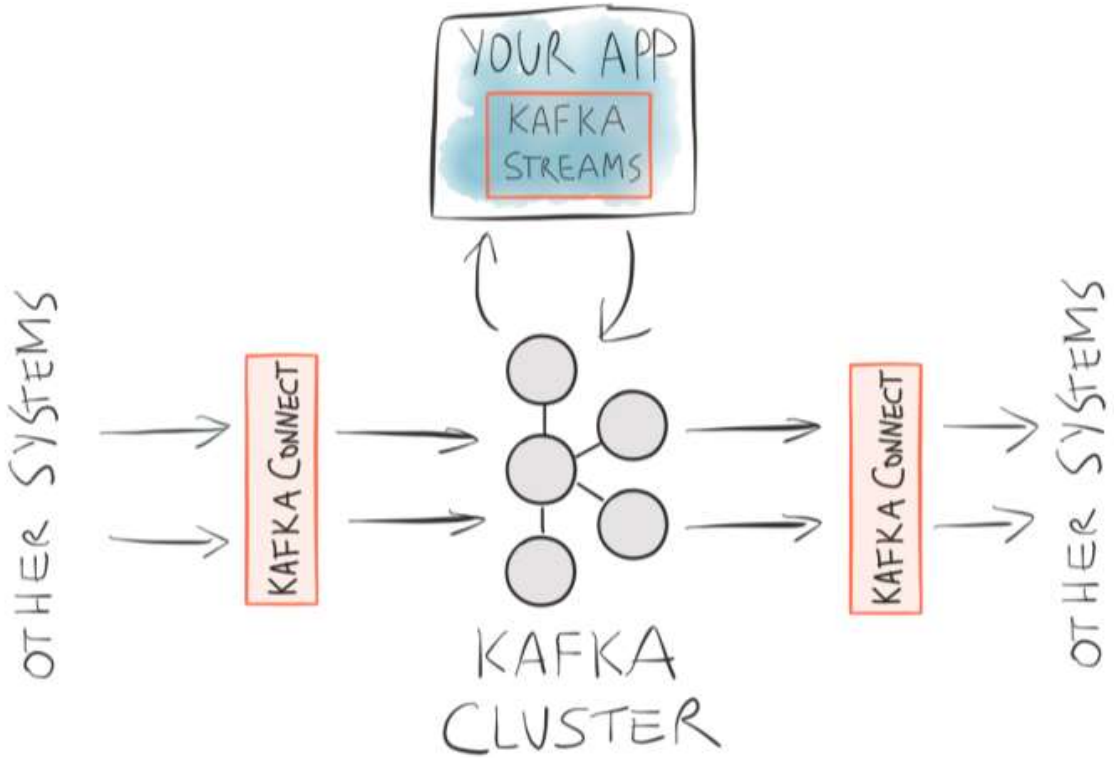
Source: https://www.confluent.io/product/connectors/

Kafka Streams

Kafka Streams

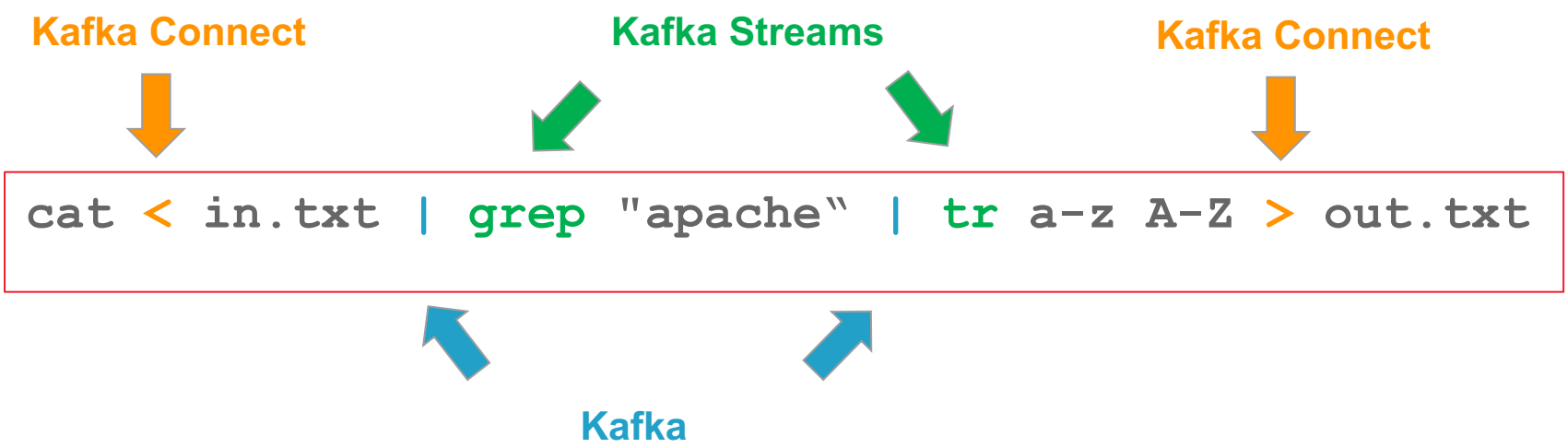
- Powerful easy-to use Java library
- Part of open source Apache Kafka
- Build your own stream processing applications that are
  - Scalable
  - Fault-tolerant
  - Stateful
  - Distributed
  - able to handle late-arriving, out-of-order data

Kafka Streams

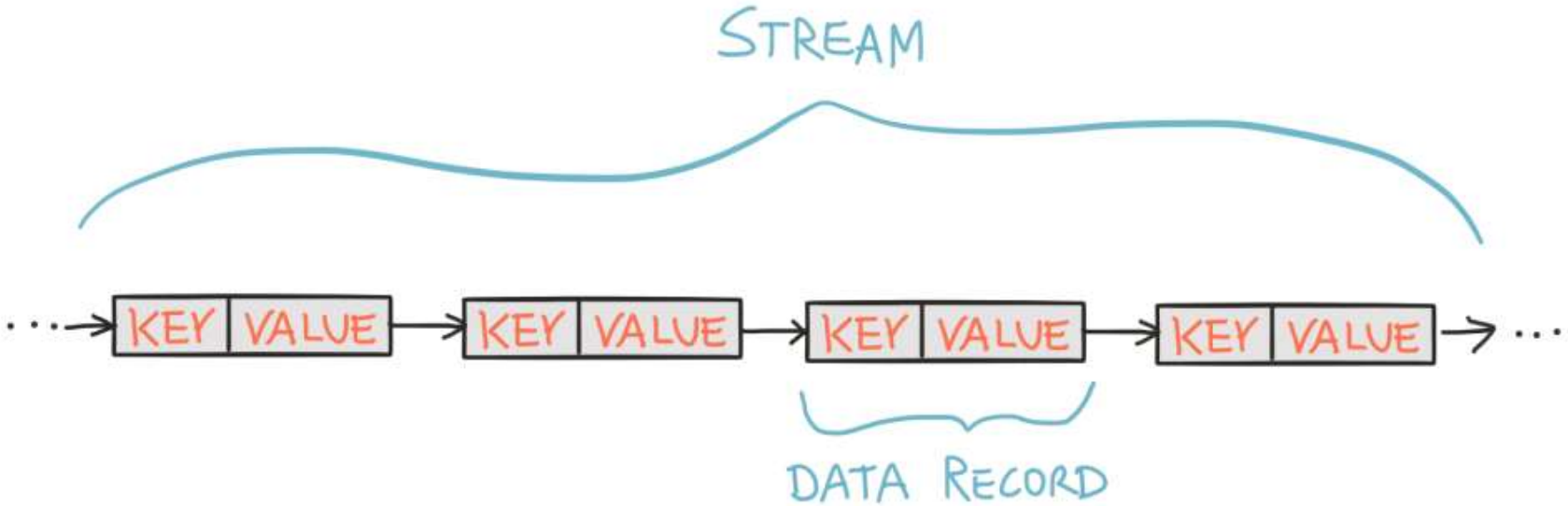


Source: https://www.confluent.io/

Kafka Streams – Unix

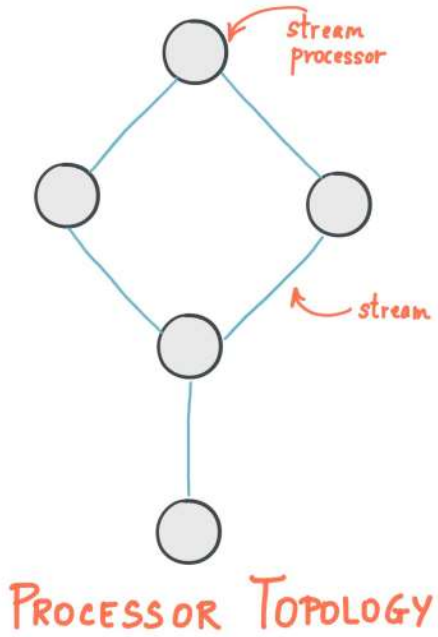


Kafka Streams



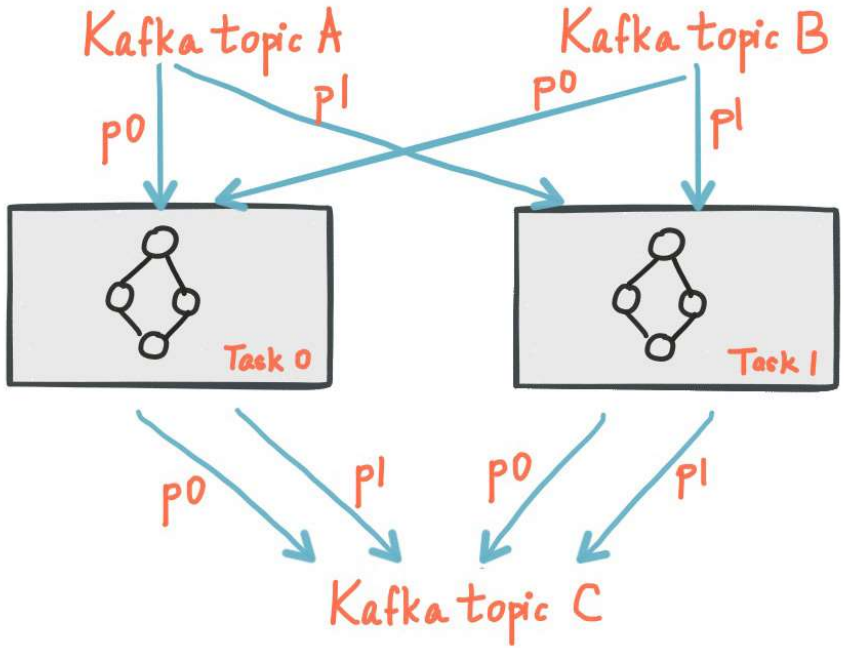
Source: https://www.confluent.io/

Kafka Streams - Topology



Source: https://www.confluent.io/

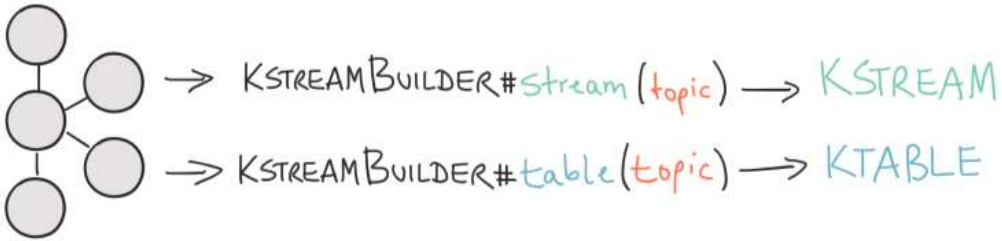
Kafka Streams – Partition and Tasks



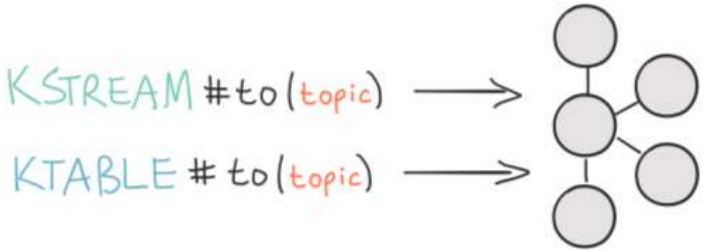
Source: https://www.confluent.io/

Kafka Streams

- Reading data from Kafka

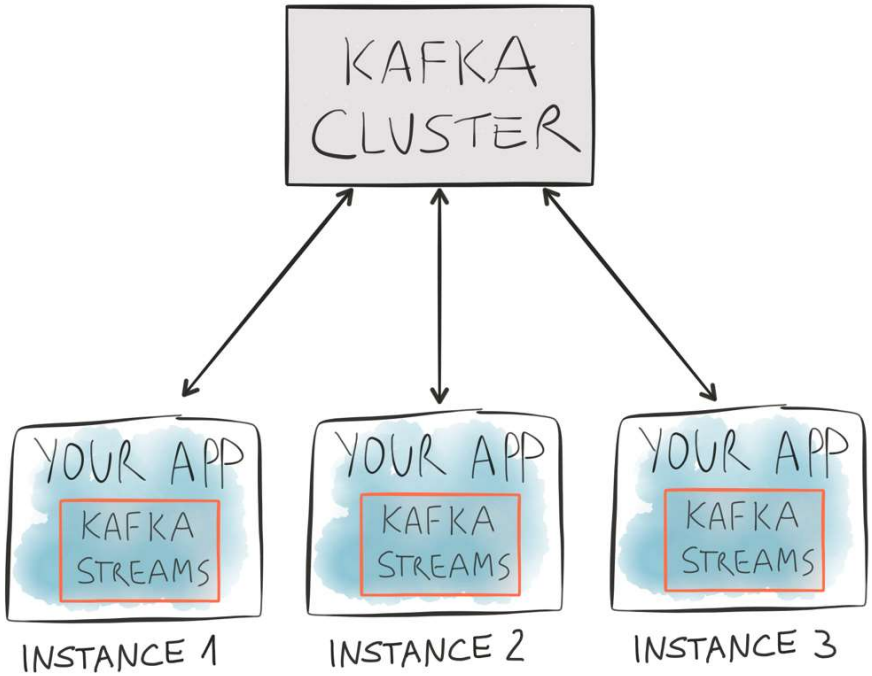


- Writing data to Kafka



Source: https://www.confluent.io/

Kafka Streams



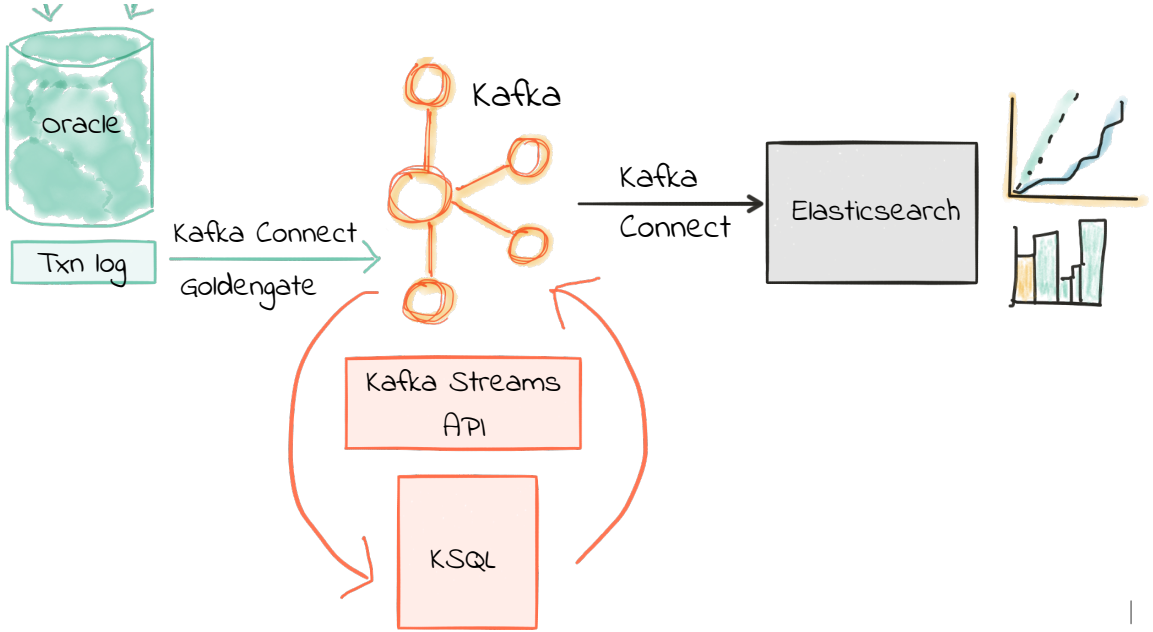
Source: https://www.confluent.io/

KSQL

KSQL

- Open Source
- Enables stream processing with zero coding required
- The simplest way to process streams of data in real-time
- Powered by Kafka: scalable, distributed, battle-tested
- All you need is Kafka

KSQL



Source: https://www.confluent.io/

KSQL

- Declare Table based on Topics

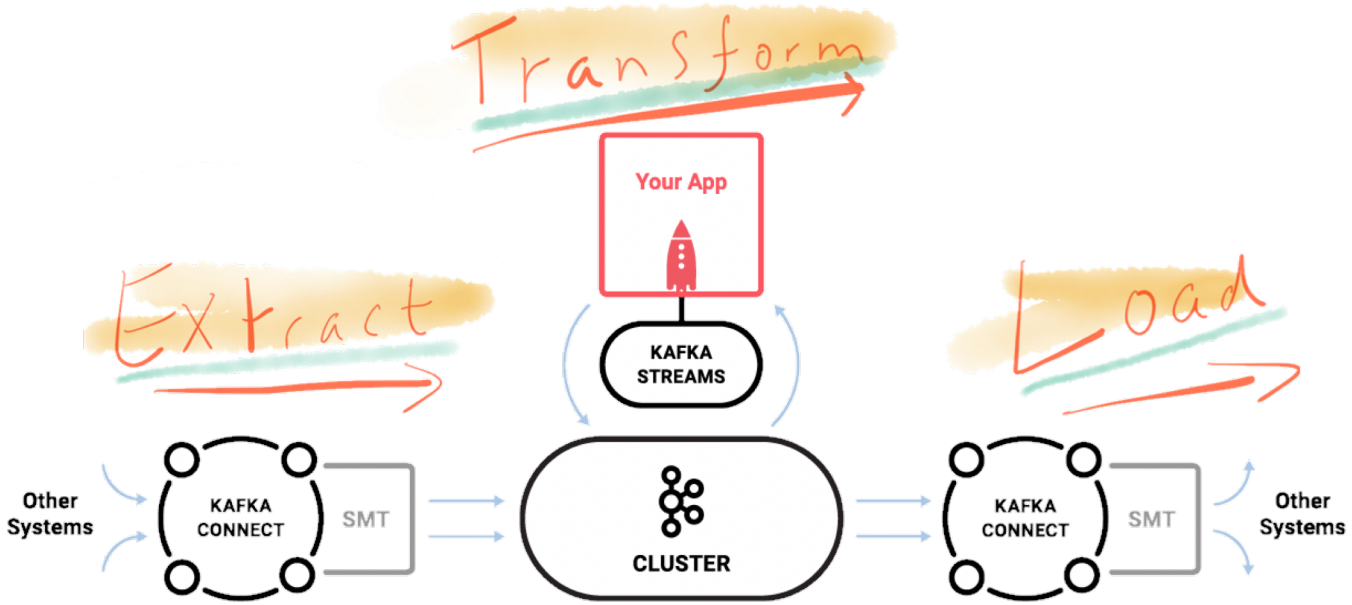
```
ksql> CREATE TABLE CUSTOMERS
(CUSTOMER_ID INT,
CUST_FIRST_NAME STRING,
CUST_LAST_NAME STRING, CUSTOMER_CLASS STRING)
WITH
(KAFKA_TOPIC=,my_ora_stream',
VALUE_FORMAT='JSON');
```

KSQL

- Query the live stream of Kafka

```
ksql> SELECT CUSTOMER_ID,
CUST_FIRST_NAME,
CUST_LAST_NAME,
CUSTOMER_CLASS
FROM CUSTOMERS
LIMIT 3;
75003 | karl | greene | Occasional
75010 | samuel | cook | Prime
75012 | paul | taylor | Occasional
```

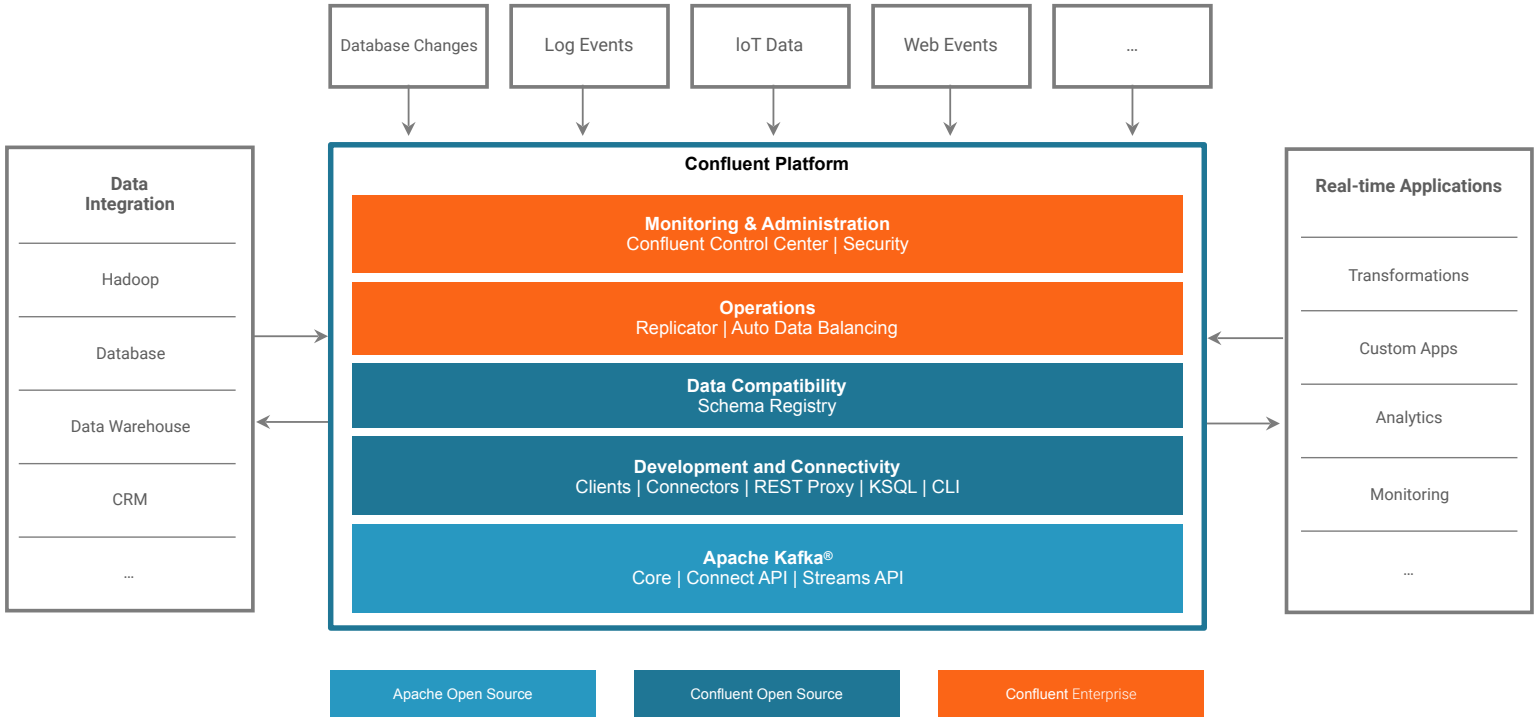
Kafka powered ETL platform



Source: https://www.confluent.io/

Confluent Platform

Confluent Platform



Source: https://www.confluent.io/

Cluster Installation

Cluster installation

- Install current Java JDK

```
sudo yum install java-1.8.0-openjdk -y
```

- Install Confluent

```
sudo rpm --import
http://packages.confluent.io/rpm/3.3/archive.key
```

Cluster installation

- Install Confluent Packages

```
sudo vi /etc/yum.repos.d/confluent.repo
[Confluent.dist]
name=Confluent repository (dist)
baseurl=http://packages.confluent.io/rpm/3.3/7
gpgcheck=1
gpgkey=http://packages.confluent.io/rpm/3.3/archive.key
enabled=1

[Confluent]
name=Confluent repository
baseurl=http://packages.confluent.io/rpm/3.3
gpgcheck=1
gpgkey=http://packages.confluent.io/rpm/3.3/archive.key
enabled=1
```

Cluster installation

Install the Packages

```
sudo yum install confluent-platform-oss-2.11
```

For a lab environment the above setting should be fine no need to change anything.
Start zookeeper

```
sudo zookeeper-server-start /etc/kafka/zookeeper.properties
```

Cluster installation

Start Kafka Broker

```
sudo kafka-server-start /etc/kafka/server.properties
```

Cluster is ready to use

```
sudo kafka-topics --list --zookeeper localhost:2181
_confluent.support.metrics
__consumer_offsets
```

http://kafka.apache.org/
https://www.confluent.io/
https://www.confluent.io/blog/
https://www.confluent.io/product/ksql/
http://blog.muehlbeyer.net/index.php/apache-kafka-installation-linux/