ClickHouse для инженеров и архитекторов БД

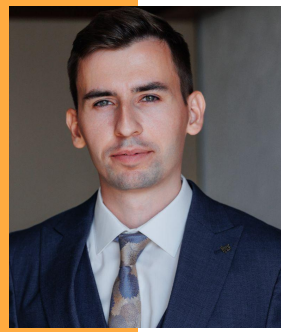
otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

Интеграция с Kafka и подводные камни



Алексей Железной

Senior Data Engineer/Architect
Магистратура - ФКН ВШЭ

*Руководитель курсов **DWH Analyst, ClickHouse для инженеров и архитекторов БД** в OTUS*

*Преподаватель курсов **Data Engineer, DWH Analyst, PostgreSQL** и пр. в OTUS*

LinkedIn

Правила вебинара

Активно
участвуем

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Маршрут вебинара

- Apache Kafka
- Интеграция с ClickHouse
- Подводные камни
- Рефлексия

Цели вебинара

1. Познакомиться с инструментом Apache Kafka
2. Изучить различные варианты интеграции с БД
3. Рассмотреть подводные камни

Поток данных

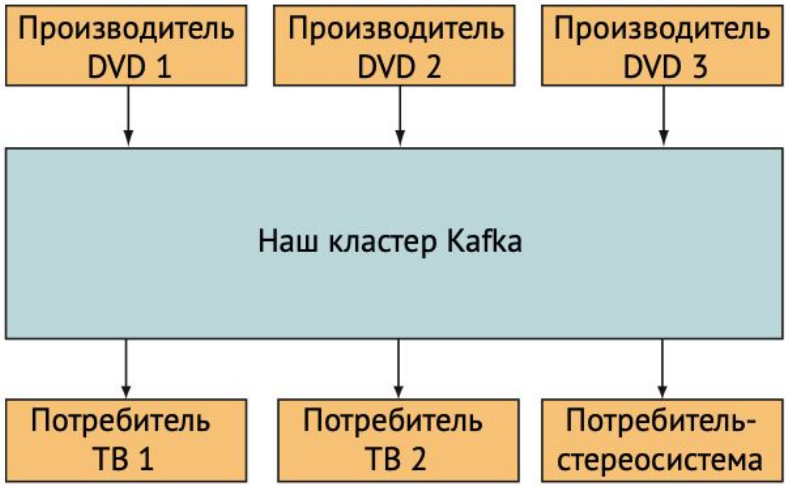
Поток данных

- **Поток данных** (stream) — это именованный набор сообщений.
- **Поток** — неограниченный набор записей
- **Пакет** — конечный набор записей

Потребности в потоковой обработке

- Ориентация на *сейчас*
  - Финансовые транзакции
  - Действия в онлайн-магазинах
  - Перемещения пользователей
  - Социальные сети
  - Промышленные датчики
- Результат нужен практически в режиме реального времени

Место Kafka в потоке данных



Источники данных, такие как спутниковая антенна, проигрыватели DVD или Blu-ray дисков, в Kafka называются производителями

Ресивер – это кластер Kafka, управляющий данными и позволяющий производителям и потребителям действовать независимо друг от друга

Телевизоры и стереосистемы, использующие данные, в Kafka называются потребителями

Apache Kafka

Что такое Kafka?

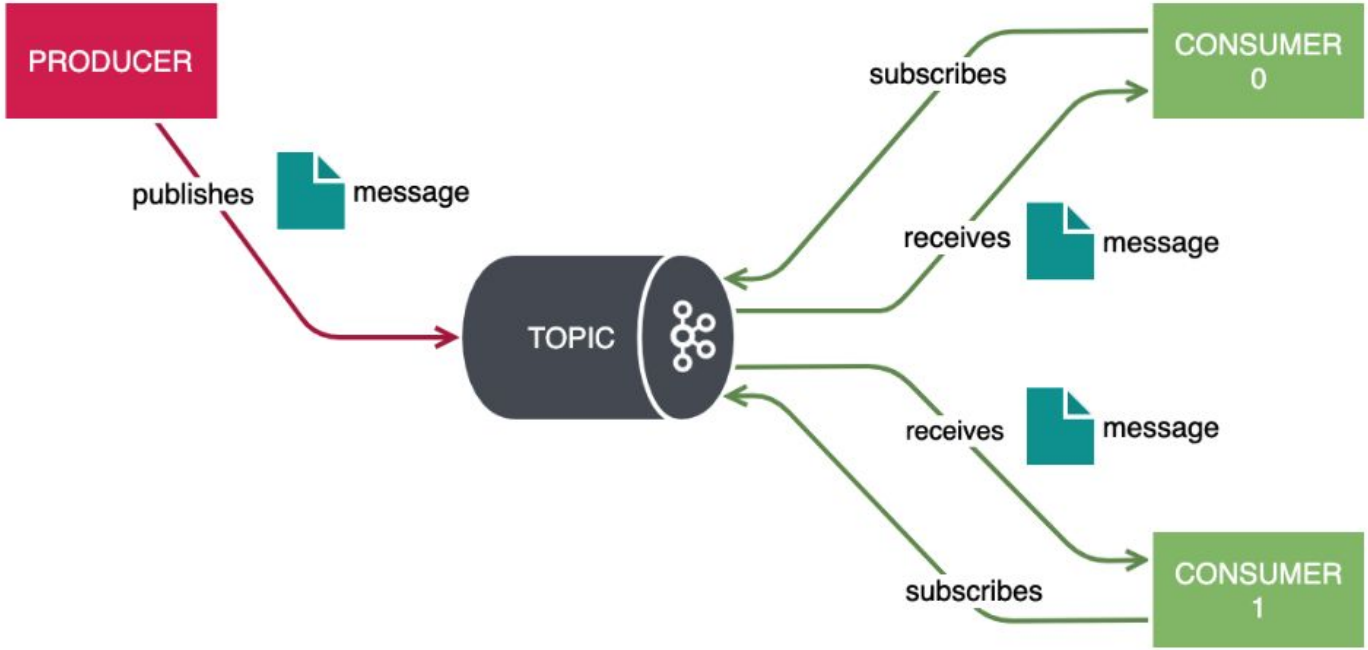
Kafka – это распределенная потоковая платформа, которая умеет:

- публиковать записи и подписываться на очереди сообщений
- хранить записи с отказоустойчивостью
- обрабатывать потоки по мере их возникновения

Платформа:

- API:
  - Producer
  - Consumer
  - Admin
- Kafka Streams
- Kafka Connect
- ksqlDB

Publish / Subscribe



Основные концепции

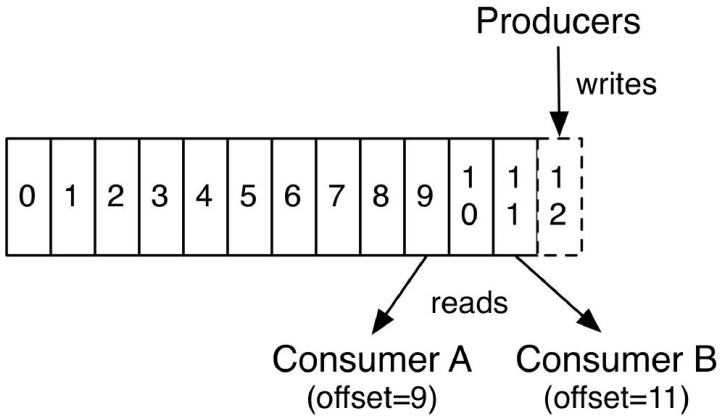
Record – элемент данных типа ключ-значение

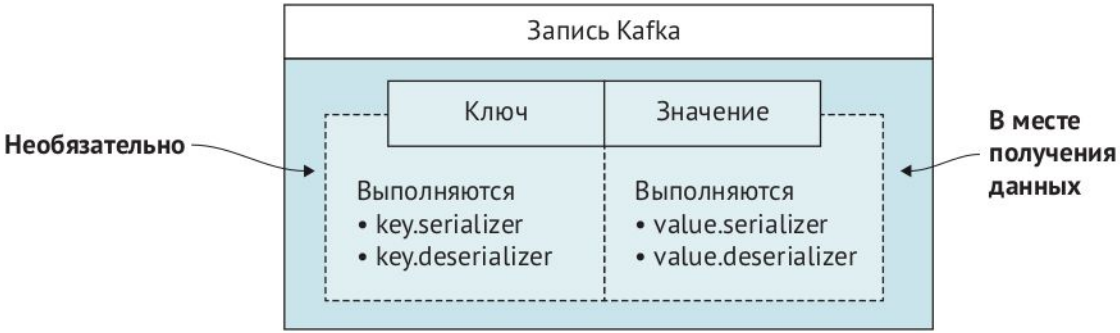
Topic – имя потока с данными

Offset – позиция записи

Producer – процесс, публикующий записи

Consumer – процесс, читающий записи



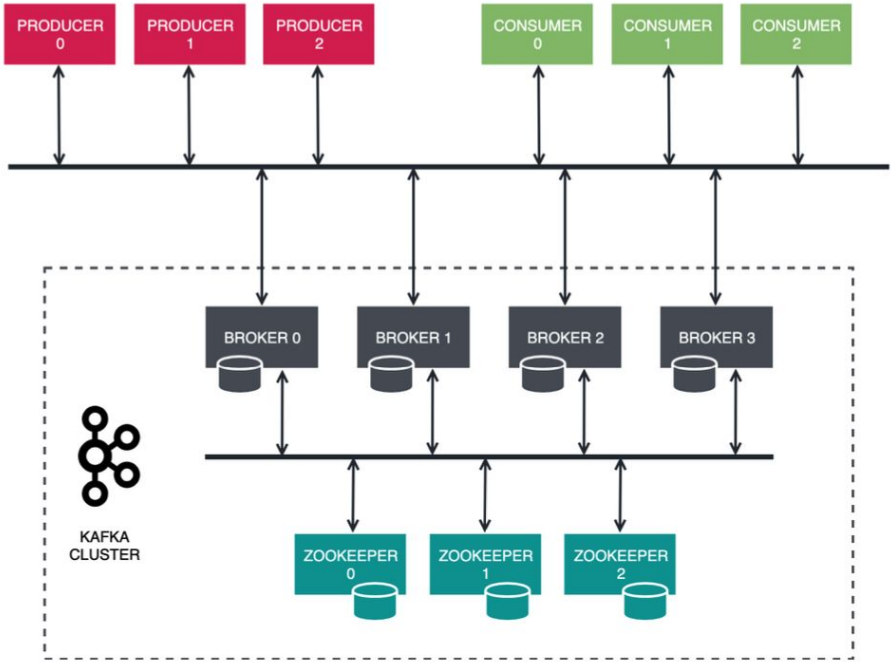


Архитектура Kafka

Broker – управление данными, взаимодействие с клиентами

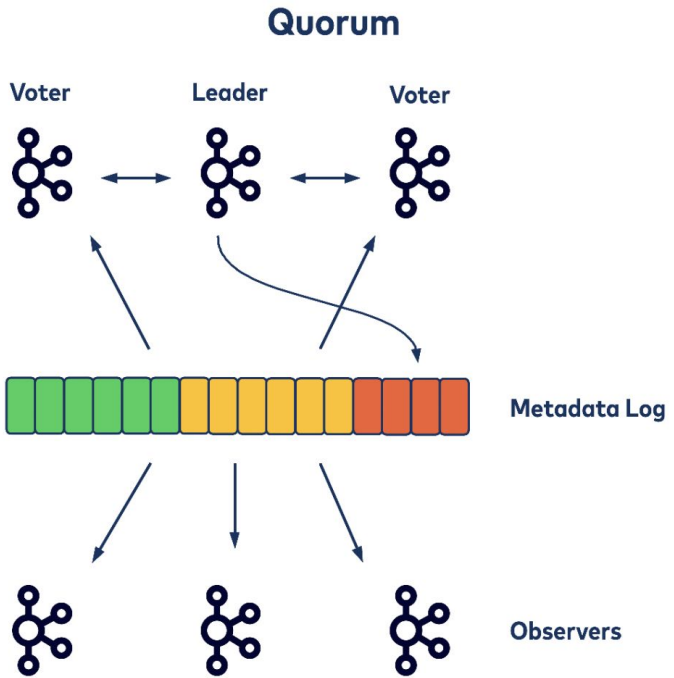
Zookeeper – членство брокеров в кластере, выборы контроллера

Контроллер – это брокер, который отвечает за выбор ведущих реплик для разделов



Kafka с KRaft

- Kafka сервер может быть:
  - broker
  - controller (3 или 5)
  - broker, controller (для разработки)
- Узлы контроллера - кворум Raft
- Журнал метаданных - информация о каждом изменении метаданных кластера
- Активный контроллер - лидер Raft

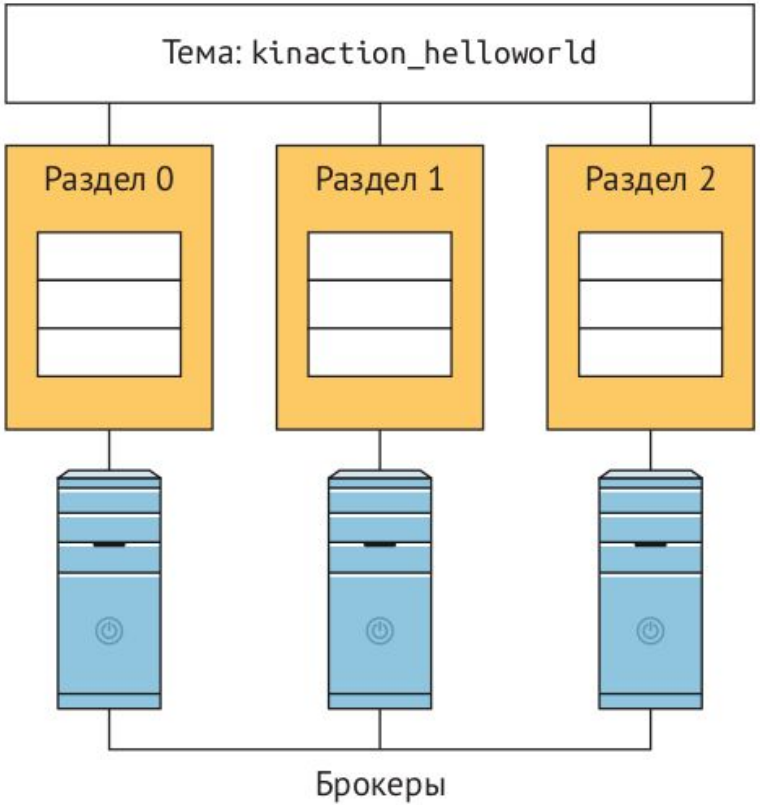


Еще раз - что такое Kafka

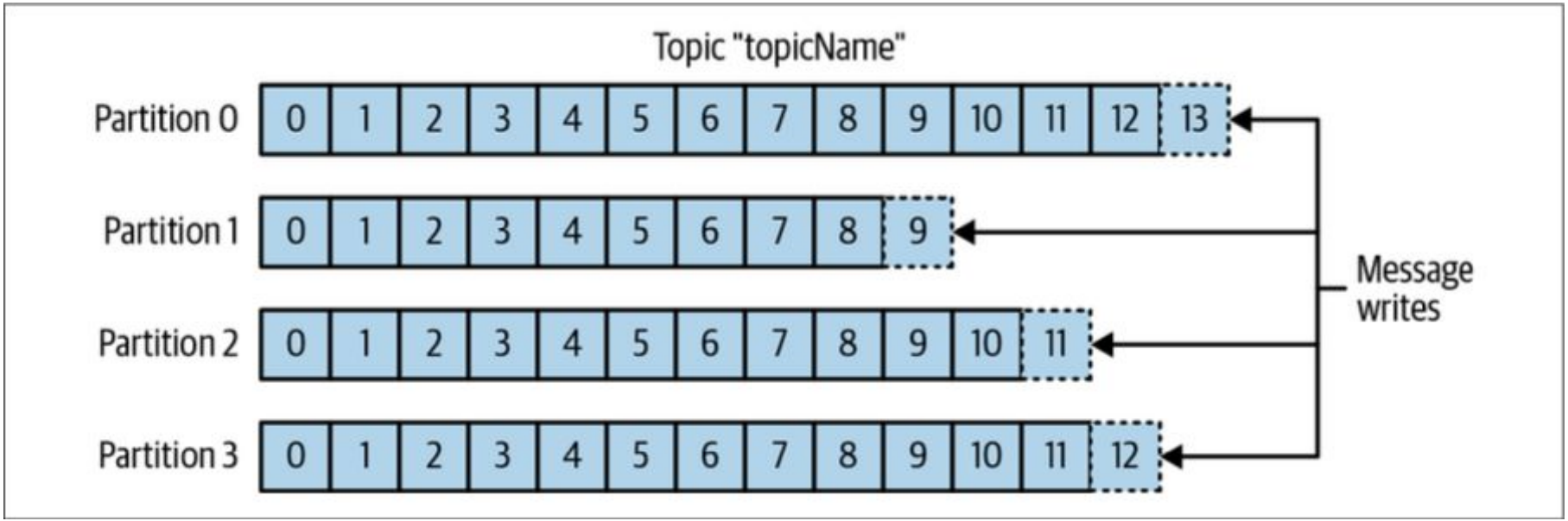
- масштабируемая шина сообщений
- используется для построения Data Pipelines и ETL процессов
- лучше всего принцип работы продемонстрирован в данной визуализации
- основные понятия
  - **Record** – Запись, состоящая из ключа и значения
  - **Topic** – категория или имя потока куда публикуются записи
  - **Producer** – процесс публикующий данные в топик
  - **Consumer** – процесс читающий данные из топика
  - **Consumer group -** группа читателей из одного топика для балансировки нагрузки по чтению
    - Разные consumer groups читают независимо друг от друга.
  - **Offset** – позиция записи
  - **Partition** – шард топика

Элементы архитектуры

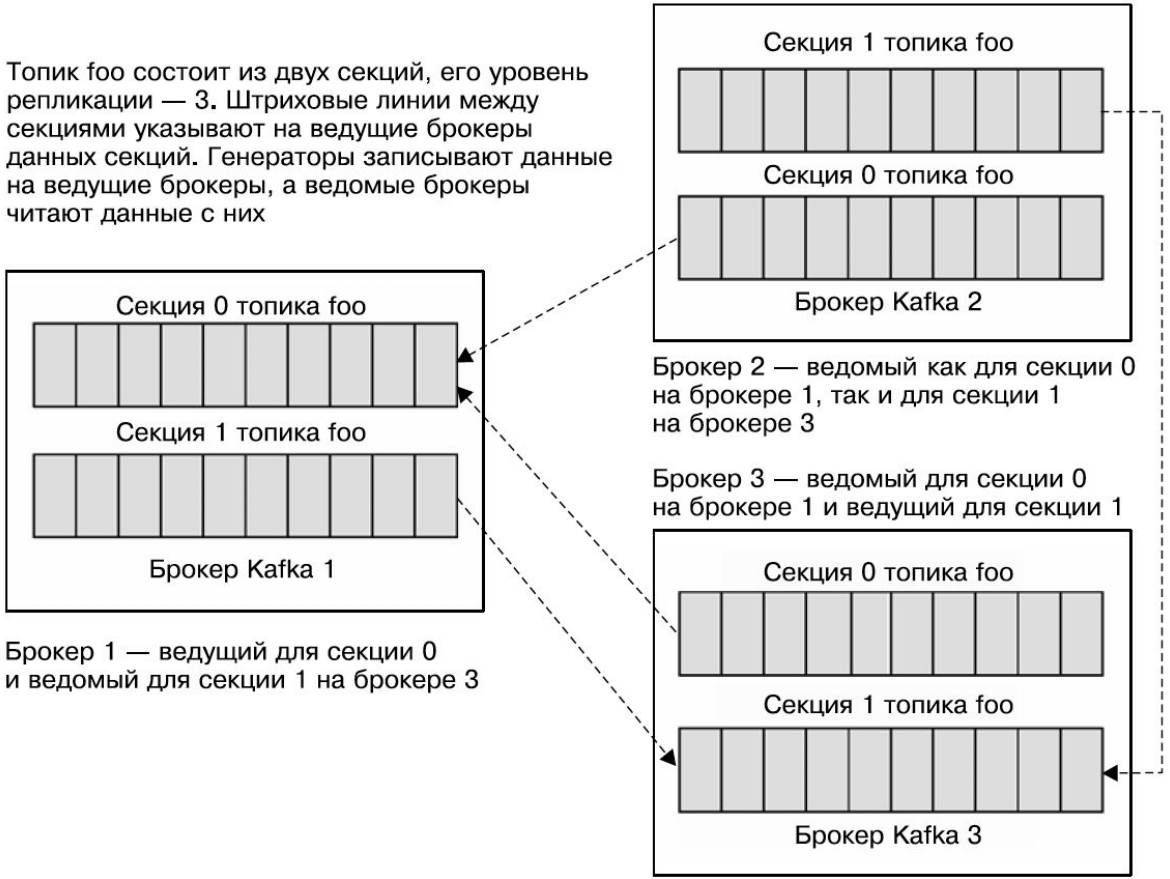
Темы и разделы (секции, partitions)



Запись в тему с разделами



Репликация

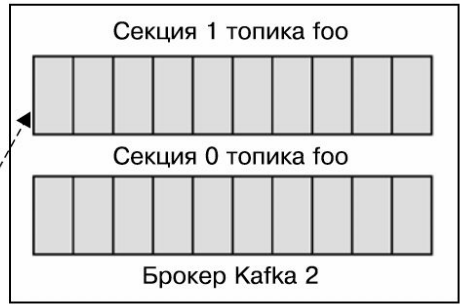

Топик foo состоит из двух секций, его уровень репликации — 3. Штриховые линии между секциями указывают на ведущие брокеры данных секций. Генераторы записывают данные на ведущие брокеры, а ведомые брокеры читают данные с них
Секция 1 топика foo
Секция 0 топика foo
Брокер Kafka 2
Брокер 2 — ведомый как для секции 0 на брокере 1, так и для секции 1 на брокере 3
Брокер 3 — ведомый для секции 0 на брокере 1 и ведущий для секции 1
Секция 0 топика foo
Секция 1 топика foo
Брокер Kafka 1
Брокер 1 — ведущий для секции 0 и ведомый для секции 1 на брокере 3
Секция 0 топика foo
Секция 1 топика foo
Брокер Kafka 3

Поведение при отказе брокера

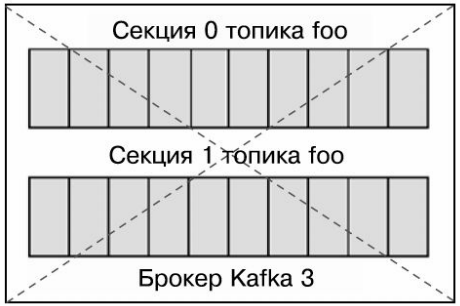
Топик foo состоит из двух секций, его уровень репликации — 3. Изначально у него следующие ведущие и ведомые брокеры:
Брокер 1 — ведущий для секции 0
и ведомый для секции 1
Брокер 2 — ведомый для секции 0 и секции 1
Брокер 3 — ведомый для секции 0
и ведущий для секции 1

Брокер 3 перестал реагировать
на контрольные сигналы ZooKeeper

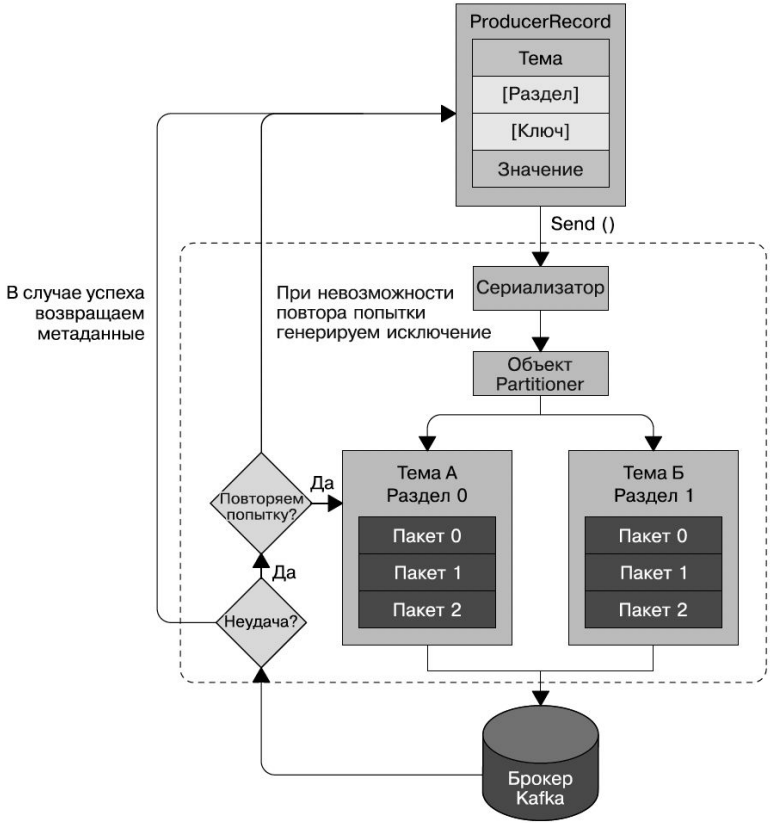
Шаг 1: будучи ведущим, брокер 1 обнаружил сбой брокера 3



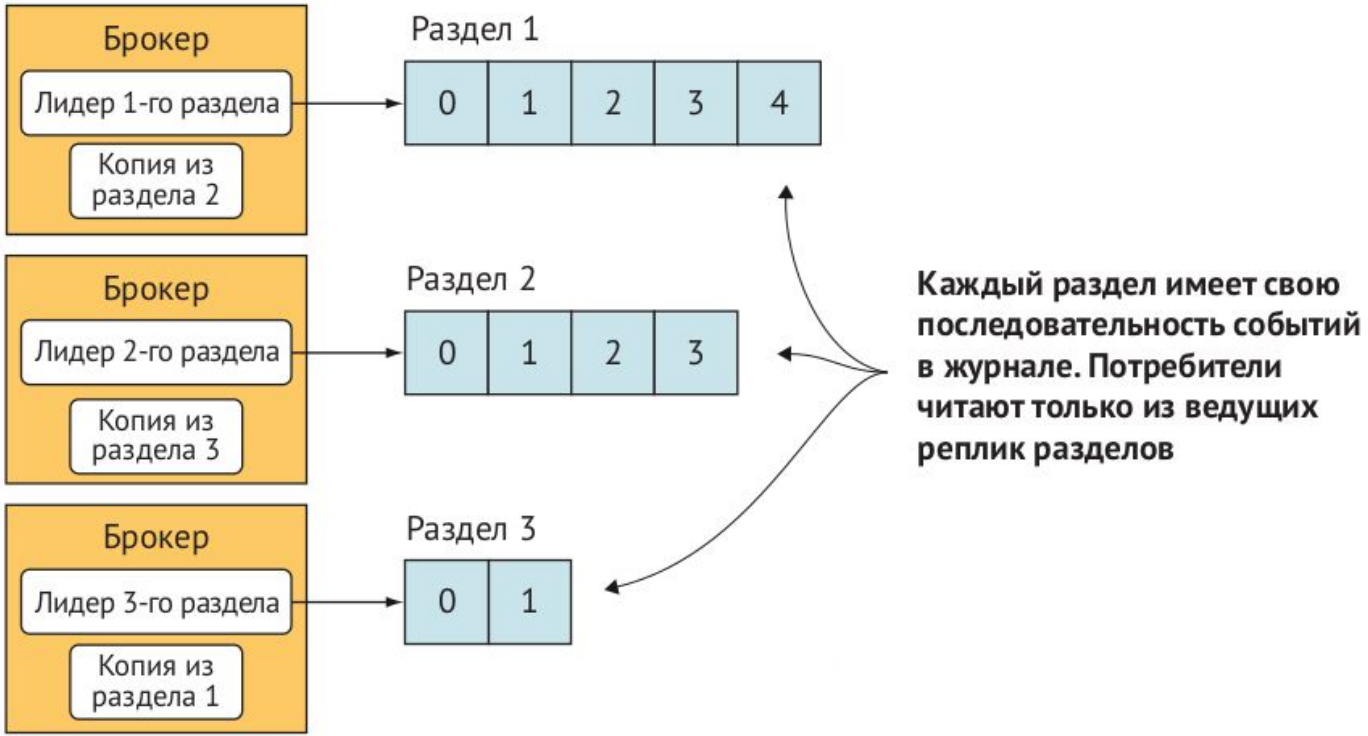
Шаг 2: контроллер делает ведущим для секции 1 брокер 2 вместо брокера 3. Все записи секции 1 теперь будут попадать на брокер 2, а брокер 1 будет потреблять сообщения для секции 1 с брокера 2



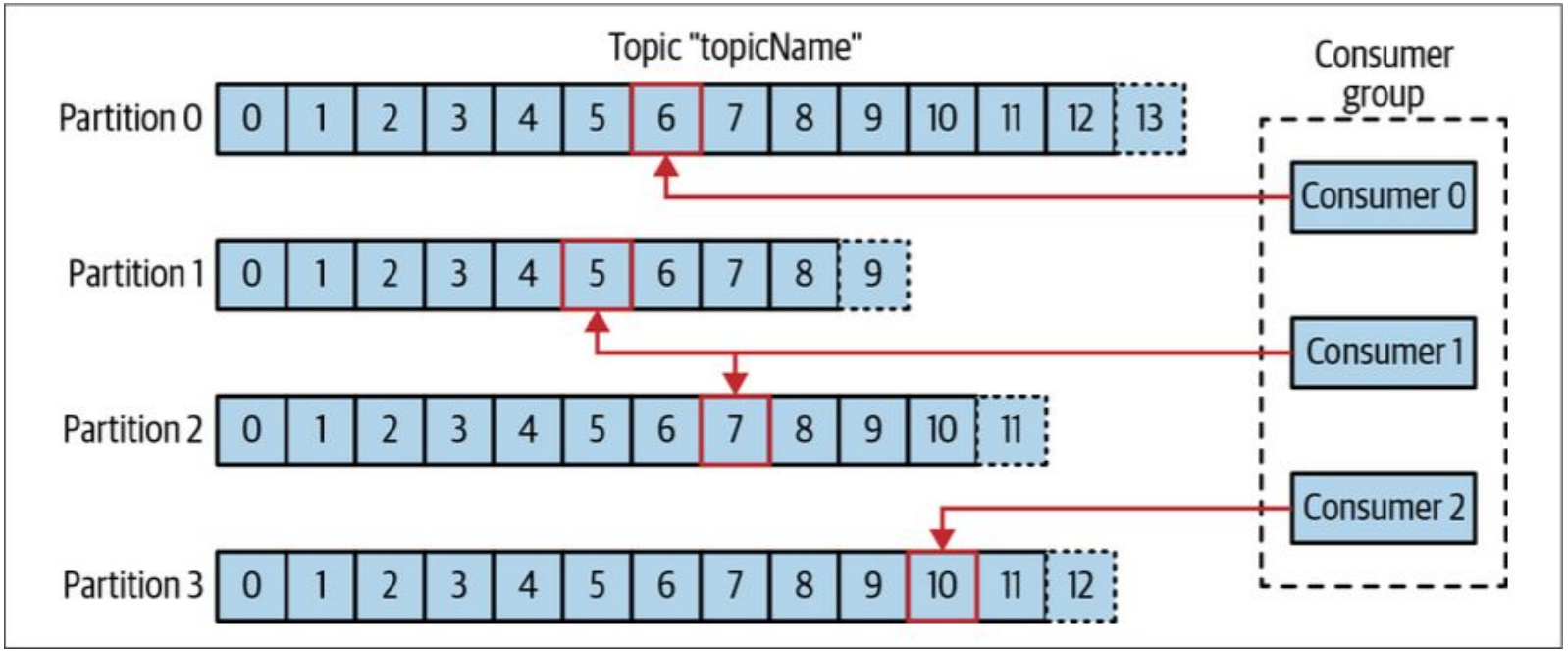
Producer – запись сообщений в Kafka



Consumer – чтение сообщений из Kafka

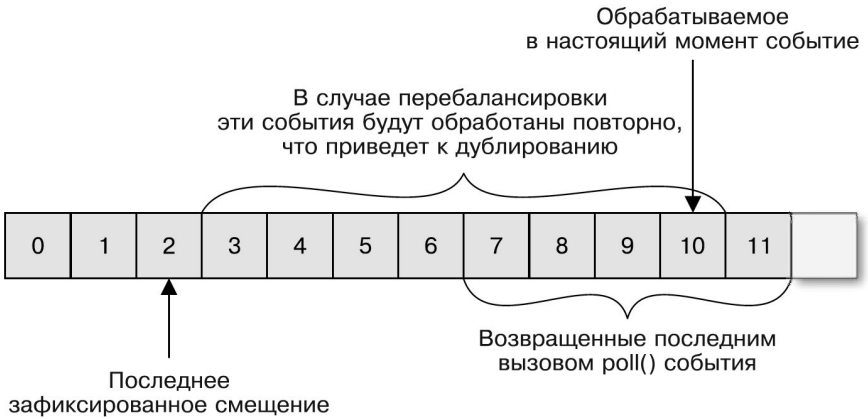


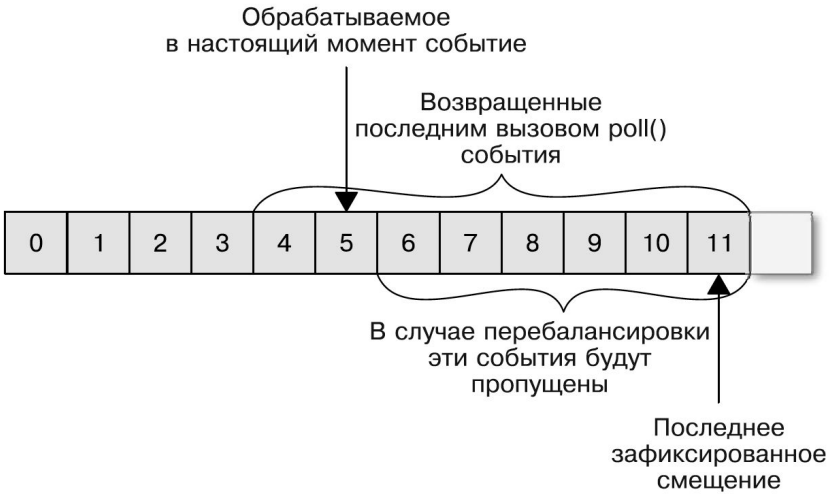
Consumer – чтение сообщений из Kafka



Фиксация и смещение

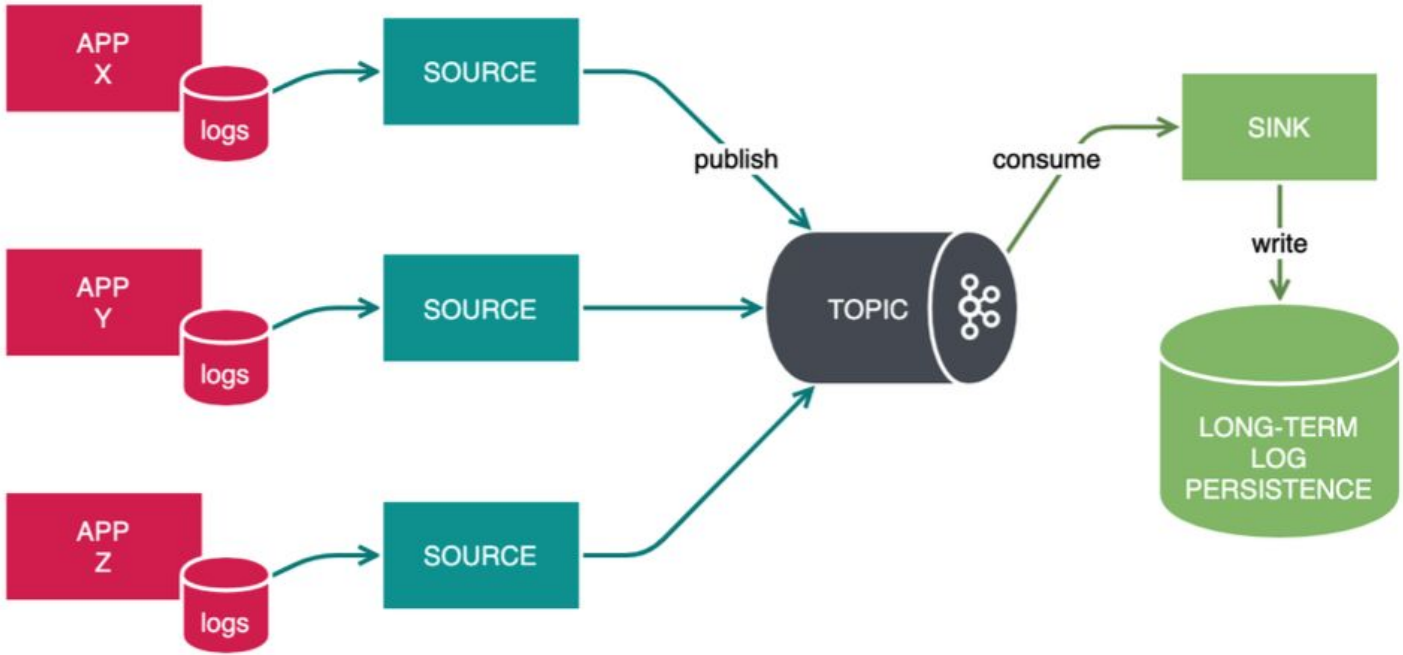
- Брокер Kafka не отслеживает чтение потребителями
- Потребители могут использовать Kafka для сохранения позиции (*смещения*)
- *Фиксация* (commit) – действие по обновлению текущей позиции потребителя
- Потребители отправляют в специальную тему `__consumer_offsets` сообщение, содержащее смещение для каждого раздела
- Аварийный сбой потребителя или присоединение нового потребителя инициирует перебалансировку





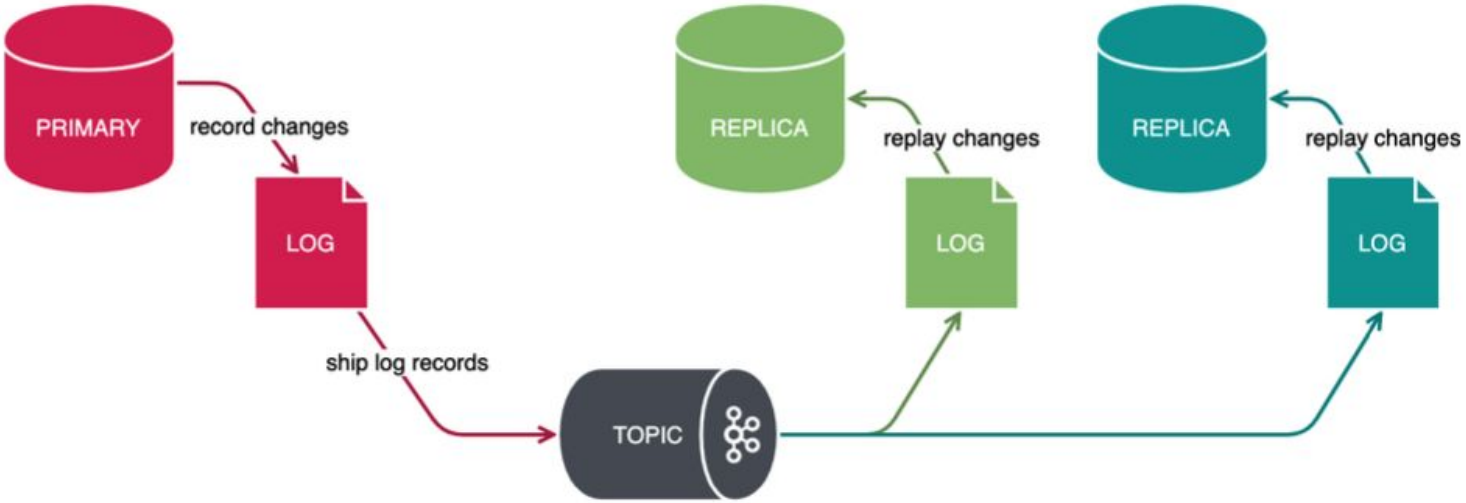
Варианты применения

Агрегация журналов

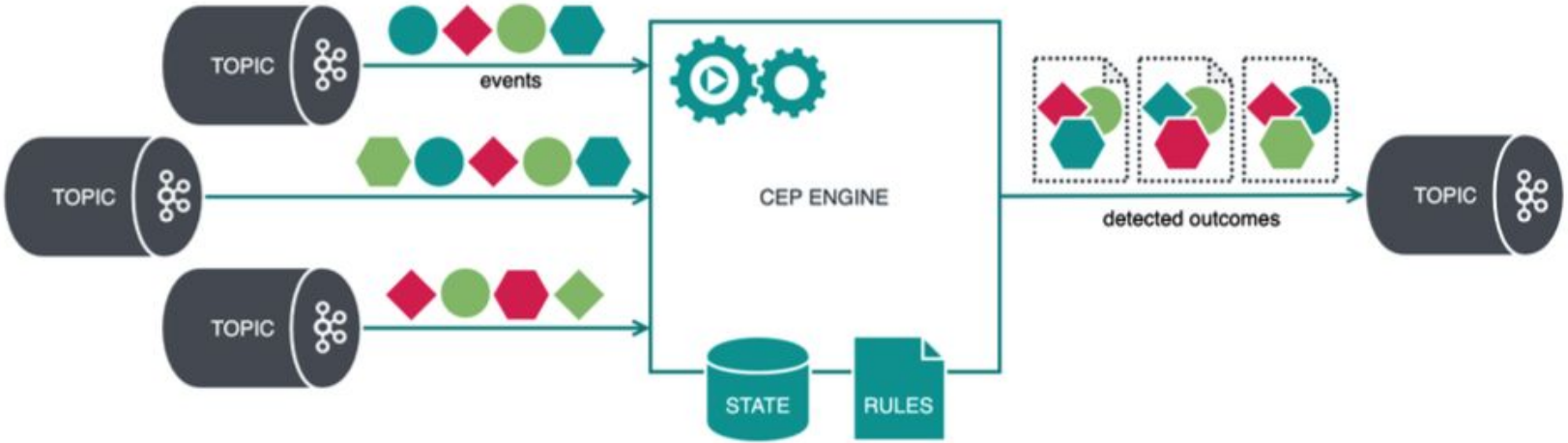


Обработка журналов

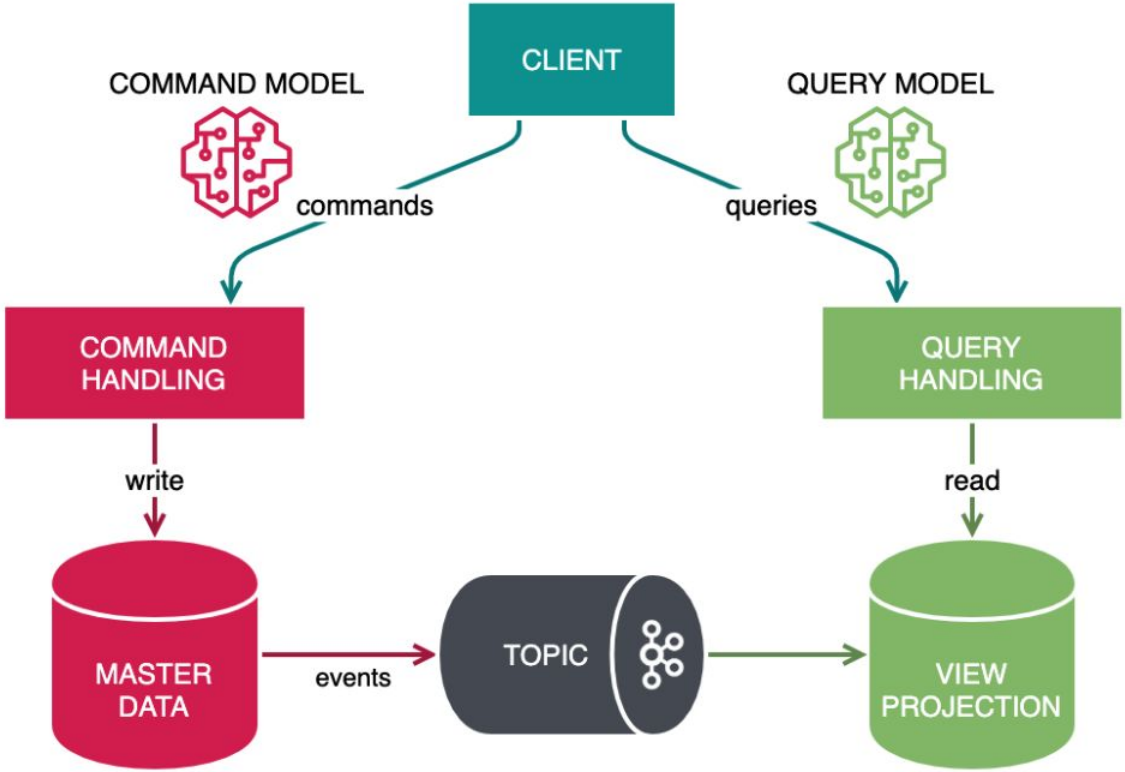
Доставка журналов (log shipping)



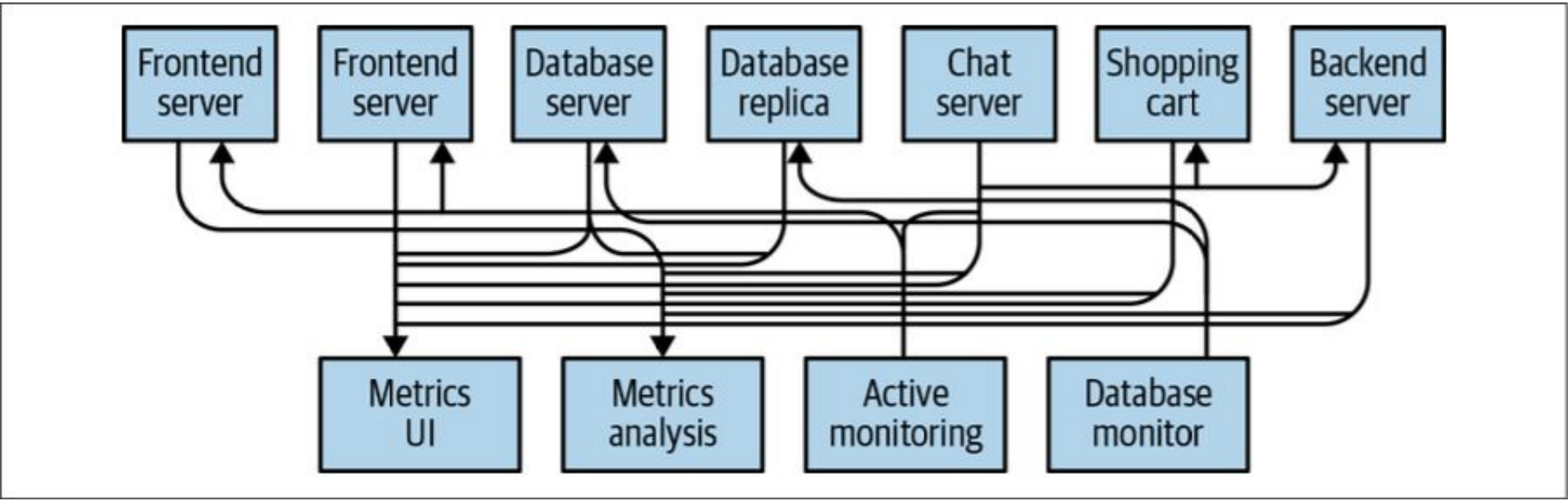
Сложная обработка событий (Complex Event Processing)



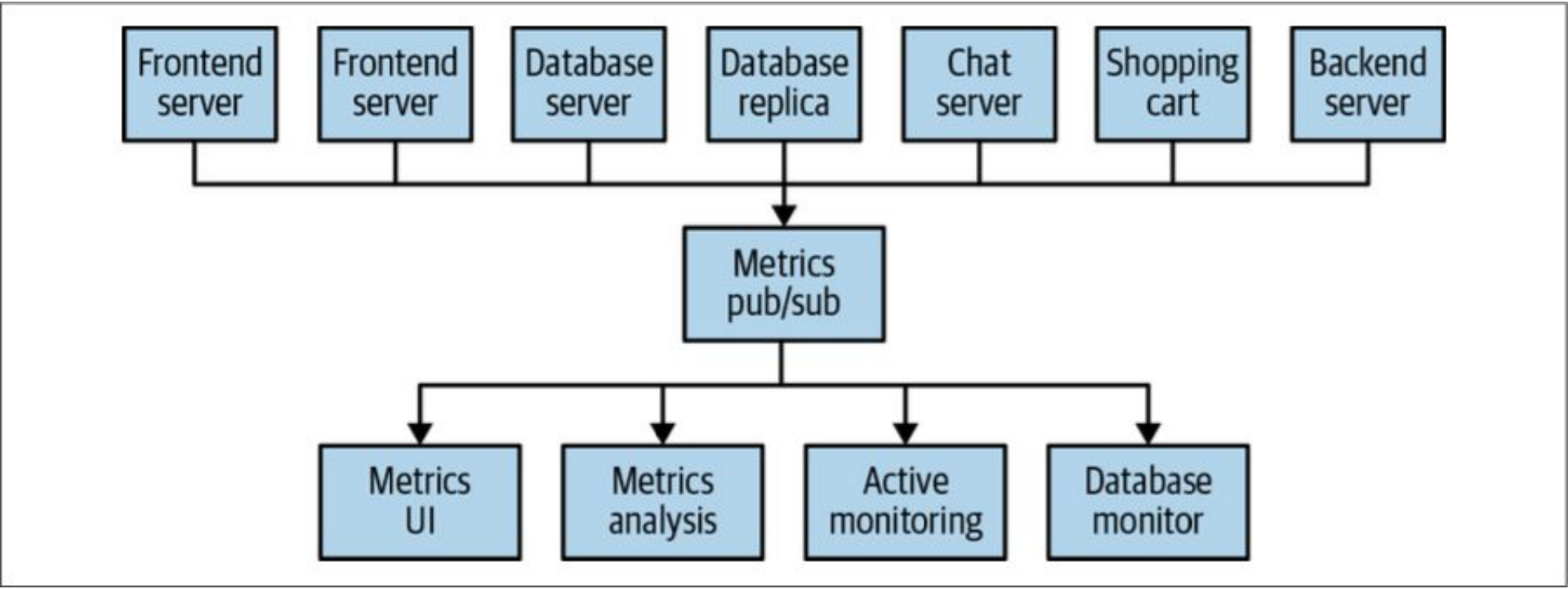
Command-Query Responsibility Segregation (CQRS)



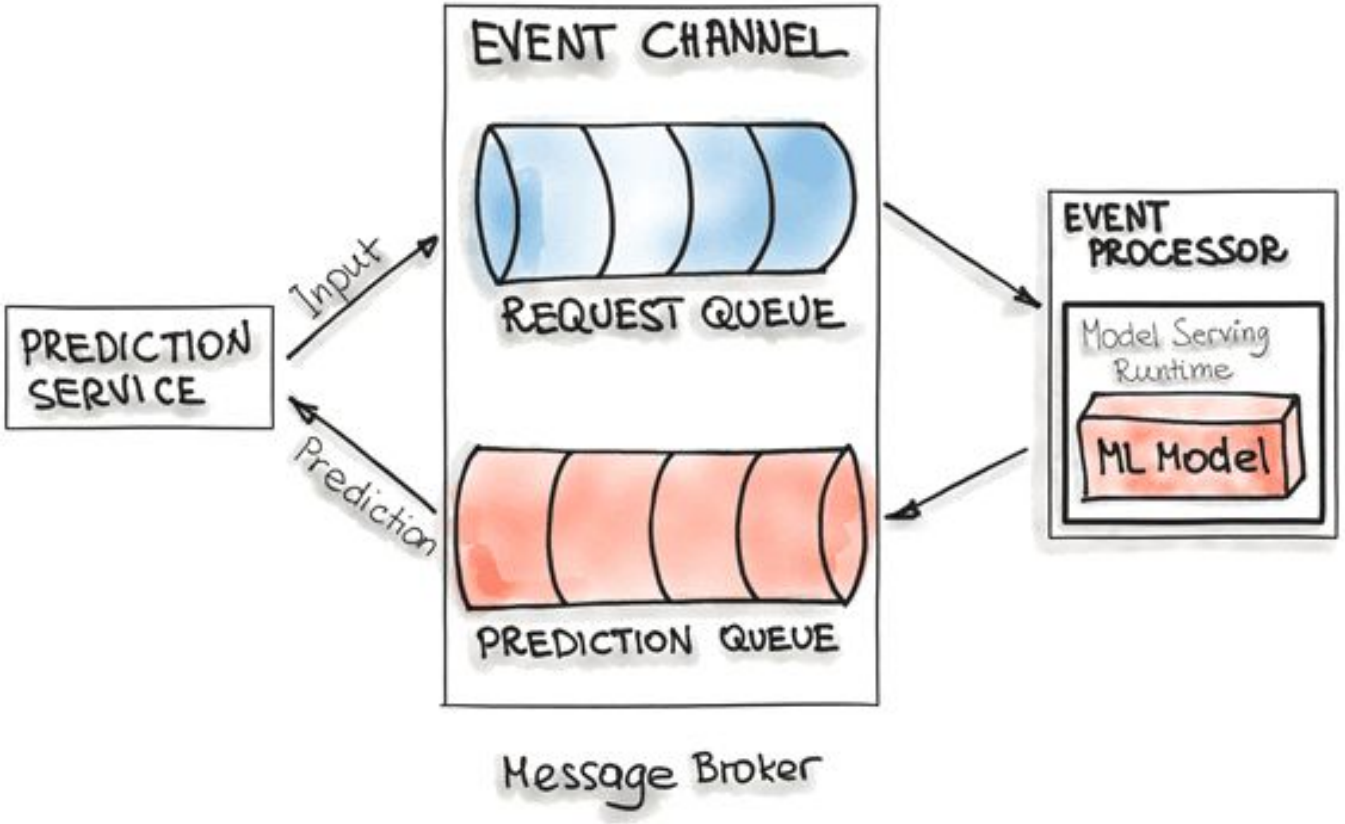
Пример традиционной архитектуры



Архитектура, ориентированная на события



Model-on-Demand



Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Варианты установки

Запускаем Kafka

1) Скачиваем и разворачиваем

```
$ tar -xzf kafka_2.13-3.6.1.tgz
$ cd kafka_2.13-3.6.1
```

2) Запускаем сервисы

```
$ bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
$ bin/kafka-server-start.sh -daemon config/server.properties
```

3) Создаём тему

```
$ bin/kafka-topics.sh --create --topic quickstart --bootstrap-server localhost:9092
$ bin/kafka-topics.sh --describe --topic quickstart --bootstrap-server localhost:9092
```

4) Запишем что-нибудь в тему

```
$ bin/kafka-console-producer.sh --topic quickstart --bootstrap-server localhost:9092
This is my first event
This is my second event
```

5) Прочитаем записи

```
$ bin/kafka-console-consumer.sh --topic quickstart --from-beginning ---bootstrap-server localhost:9092
This is my first event
This is my second event
```

Kafka в Docker

```
version: '2.1'

services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    ports:
      - "2181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000

  kafka:
    image: confluentinc/cp-kafka:latest
    ports:
      - "9092:9092"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: LISTENER_DOCKER_INTERNAL://kafka:19092,LISTENER_DOCKER_EXTERNAL://${DOCKER_HOST_IP:-127.0.0.1}:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: LISTENER_DOCKER_INTERNAL:PLAINTEXT,LISTENER_DOCKER_EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: LISTENER_DOCKER_INTERNAL
      KAFKA_LOG4J_ROOT_LOGLEVEL: INFO
      KAFKA_CONFLUENT_SUPPORT_METRICS_ENABLE: "false"
    depends_on:
      - zookeeper
```

Основные операции

- zookeeper-server-start.sh – запуск Zookeeper
- zookeeper-server-stop.sh – останов Zookeeper
- kafka-server-start.sh – запуск Kafka брокера
- kafka-server-stop.sh – останов Kafka брокера
- kafka-console-producer.sh – консольный Producer
- kafka-console-consumer.sh – консольный Consumer
- kafka-topics.sh – работа с темами (создать, удалить, изменить, посмотреть)

Варианты интеграции с ClickHouse

Взаимодействие Kafka и ClickHouse

Self-managed Kafka Connectivity

- Kafka Connect - это бесплатный компонент Apache Kafka с открытым исходным кодом, который работает как централизованный датахаб для простой интеграции данных между Kafka и другими системами данных.
- Vector - конвейер данных, не зависящий от источника. Благодаря возможности чтения из Kafka и отправки событий в ClickHouse он представляет собой надежный вариант интеграции.
- JDBC Connect Sink - позволяет экспортировать данные из топиков Kafka в любую реляционную базу данных с драйвером JDBC.

Взаимодействие Kafka и ClickHouse

Self-managed Kafka Connectivity

- Custom code - В случаях, когда требуется пользовательская обработка событий, может быть использован пользовательский код с использованием соответствующих клиентских библиотек для Kafka и ClickHouse.
- Kafka table engine обеспечивает нативную интеграцию ClickHouse (недоступна в ClickHouse Cloud).

Kafka Engine

Движок Kafka

- одна из самых частых интеграций
- не хранит данные самостоятельно
  - предназначен для подписки на потоки данных в конкретных топиках (consumer)
    - SELECT может прочесть запись только один раз
      - поэтому имеет смысл использовать MV
  - и для публикации данных в конкретные топики (producer)

Синтаксис

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = Kafka()
SETTINGS
    kafka_broker_list = 'host:port',
    kafka_topic_list = 'topic1,topic2,...',
    kafka_group_name = 'group_name',
    kafka_format = 'data_format'[,]
    [kafka_row_delimiter = 'delimiter_symbol',]
    [kafka_schema = '',]
    ….
```

Обязательные настройки движка Kafka

- **kafka_broker_list** — перечень брокеров, разделенный запятыми
- **kafka_topic_list** — перечень необходимых топиков Kafka, разделенный запятыми
- **kafka_group_name** — группа потребителя Kafka. Если необходимо, чтобы сообщения не повторялись на кластере, необходимо использовать везде одно имя группы.
- **kafka_format** — формат сообщений, например JSONEachRow.
- Опциональные параметры, такие как размер блока, количество потребителей на таблицу и потоков, подробно описаны в документации.

Пример организации пайплайна INSERT to -> kafka -> MV -> MergeTree

```
CREATE TABLE kafka_tbl (
     Message String,
     TimeStamp String,
     ChannelID String,
     Name String,
     Priority String
) ENGINE = Kafka
SETTINGS kafka_broker_list = 'localhost:9092',
kafka_topic_list = 'mytopic',
kafka_group_name = 'test-cg',
kafka_format = 'JSONEachRow';
```

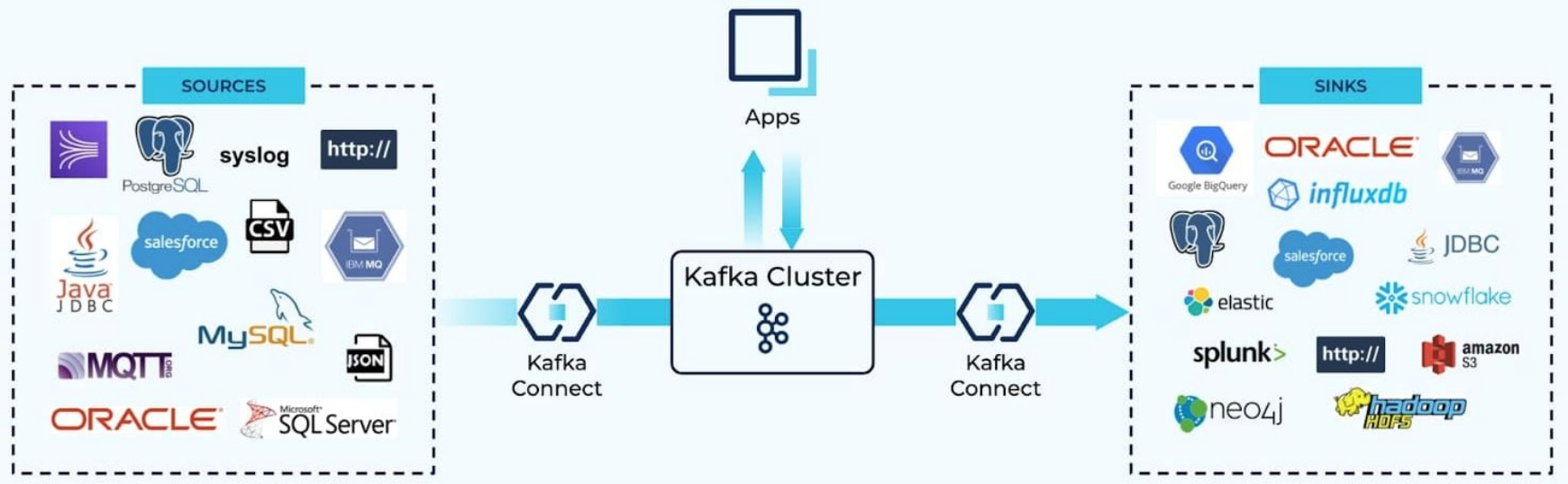
Пример организации пайплайна INSERT to -> kafka -> MV -> TinyLog

```
CREATE TABLE log_target (id UInt64, msg String)
ENGINE = TinyLog ORDER BY (id);

CREATE MATERIALIZED VIEW kafka_mv TO log_target
AS SELECT
     id,
     msg
FROM kafka_tbl;

INSERT INTO kafka_tbl SELECT number as id, concat('test',number) as msg FROM
numbers(10) FORMAT JSONEachRow;

SELECT * FROM log_target;
```

Kafka Connect

Kafka Connect

Kafka Connect — это часть Kafka, обеспечивающая масштабируемый и гибкий способ копирования данных между Kafka и другими системами



Возможности Kafka Connect

Kafka Connect предоставляет:

- Среду выполнения для запуска плагинов-коннекторов
- Общий фреймворк для коннекторов

Возможности:

- Управление настройками
- Хранение смещений
- Распараллеливание (масштабируемость)
- Обработка ошибок
- Автоматическое восстановление
- Поддержка различных типов данных
- REST API

Основные параметры коннекторов

Параметры зависят от типа коннектора, но есть основные:

- name — уникальное имя коннектора
- connector.class — Java класс коннектора
- tasks.max — максимальное количество задач, которые должны быть созданы для этого коннектора
- key.converter (опционально) — переопределение преобразователя ключей
- value.converter (опционально) — переопределение преобразователя значений

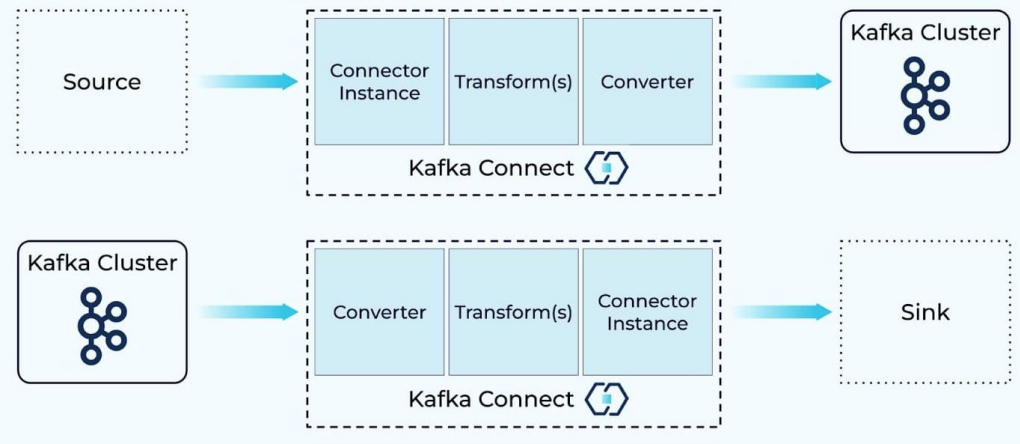
У коннекторов-приёмников есть дополнительные параметры.

- Topics — список тем, разделенных запятыми
- topics.regex — регулярное выражение для фильтрации

Преобразователи форматов (Converter)

Конвертеры необходимы для обеспечения одинакового формата данных при записи в и чтении из Kafka

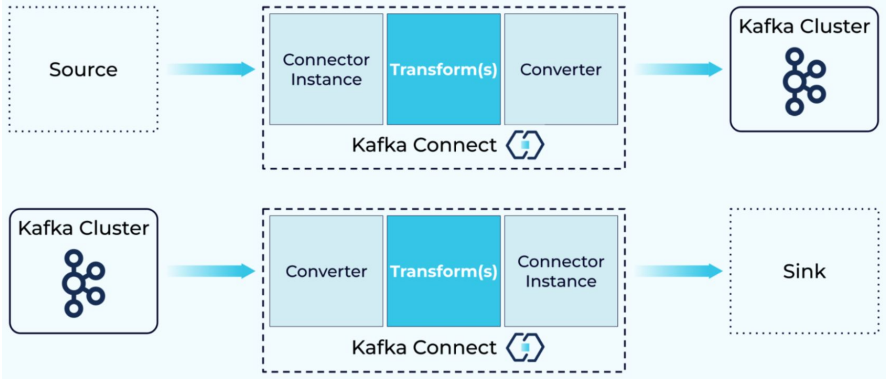
- Задачи используют конвертеры для изменения формата данных с массива байтов во внутренний формат данных Connect и наоборот



Single Message Transforms

Типичное применение SMT:

- Удаление полей
- Добавление метаданных
- Изменение типов данных полей
- Переименование полей



Преобразования одиночных сообщений

Kafka Connect включает следующие SMT:

- *Cast* — изменение типа данных поля
- *MaskField* — замена содержимого поля на null
- *Filter* — удаление или включение сообщений по условию
- *Flatten* — преобразование вложенной структуры в плоскую
- *HeaderFrom* — перемещение или копирование полей из сообщения в заголовок
- *InsertHeader* — добавление строки в заголовок каждого сообщения
- *InsertField* — добавление нового поля в сообщение
- *RegexRouter* — изменения топика назначения
- *ReplaceField* — удаление или переименование поля в сообщении
- *TimestampConverter* — изменение формата времени
- *TimestampRouter* — изменение топика на основании временной метки сообщения

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Подводные камни

Подводные камни

- керберос аутентификация
  - ошибки коннекта из-за разных версий кербероса
  - протухание кеша и криво работающие процессы по его перезапросу
  - необходимость хранить кейтаб на сервере
- потери данных
  - иногда не exactly once (чаще at most once)
  - https://clickhouse.com/blog/kafka-connect-connector-clickhouse-with-exactly-once
  - https://github.com/ClickHouse/ClickHouse/discussions/52298
- высокая нагрузка на сервер и слабая скорость через Kafka Engine
  - https://github.com/ClickHouse/ClickHouse/discussions/36419
  - https://clickhouse.com/blog/measure-visaualize-minimize-kafka-latency-clickhouse

Подводные камни

- свои consumer-сервисы хороши, но вызывают проблемы при написаниии и решении проблем
- проблемные версии
  - https://github.com/ClickHouse/ClickHouse/issues/57132
- кривая обработка ошибок
  - выбрали не тот формат данных - получили пустые строки)))
  - надеетесь на логи - зря)
- разные выводы при проверке селекта из kafka engine и mergetree

Рефлексия

Список материалов для изучения

- Kafka и Clickhouse - как организовать взаимодействие // Демо-занятие курса «Apache Kafka»
- How to use Kafka Connect
- clickhouse_sinker
- ClickHouse Kafka Connect Sink
- https://github.com/AlexeyFerum/teaching_time/wiki/Sbornaya-solyanka
- https://clickhouse.com/docs/knowledgebase/kafka-to-clickhouse-setup

Рефлексия

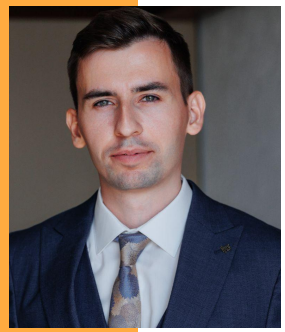
С какими впечатлениями уходите с вебинара?

Как будете применять на практике то,
что узнали на вебинаре?

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары



Алексей Железной

Senior Data Engineer
Магистратура - ФКН ВШЭ

*Руководитель курсов **DWH Analyst, ClickHouse для инженеров и архитекторов БД** в OTUS*

*Преподаватель курсов **Data Engineer, DWH Analyst, PostgreSQL** и пр. в OTUS*

LinkedIn