OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

Kafka Connect

Заигрин Вадим

Ведущий эксперт по технологиям, Сбербанк

vzaigrin@yandex.ru
https://t.me/vzaigrin

Преподаватель

Вадим Заигрин

Более 30 лет в ИТ:
- Big Data
  - Data Engineer
  - Data Science
- Разработка
  - Scala, Java, Python, C, Lisp
- IT Infrastructure
  - Администрирование
  - Сопровождение
  - Архитектура

Big Data проекты в банках, телекоме и в рознице.

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Telegram

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

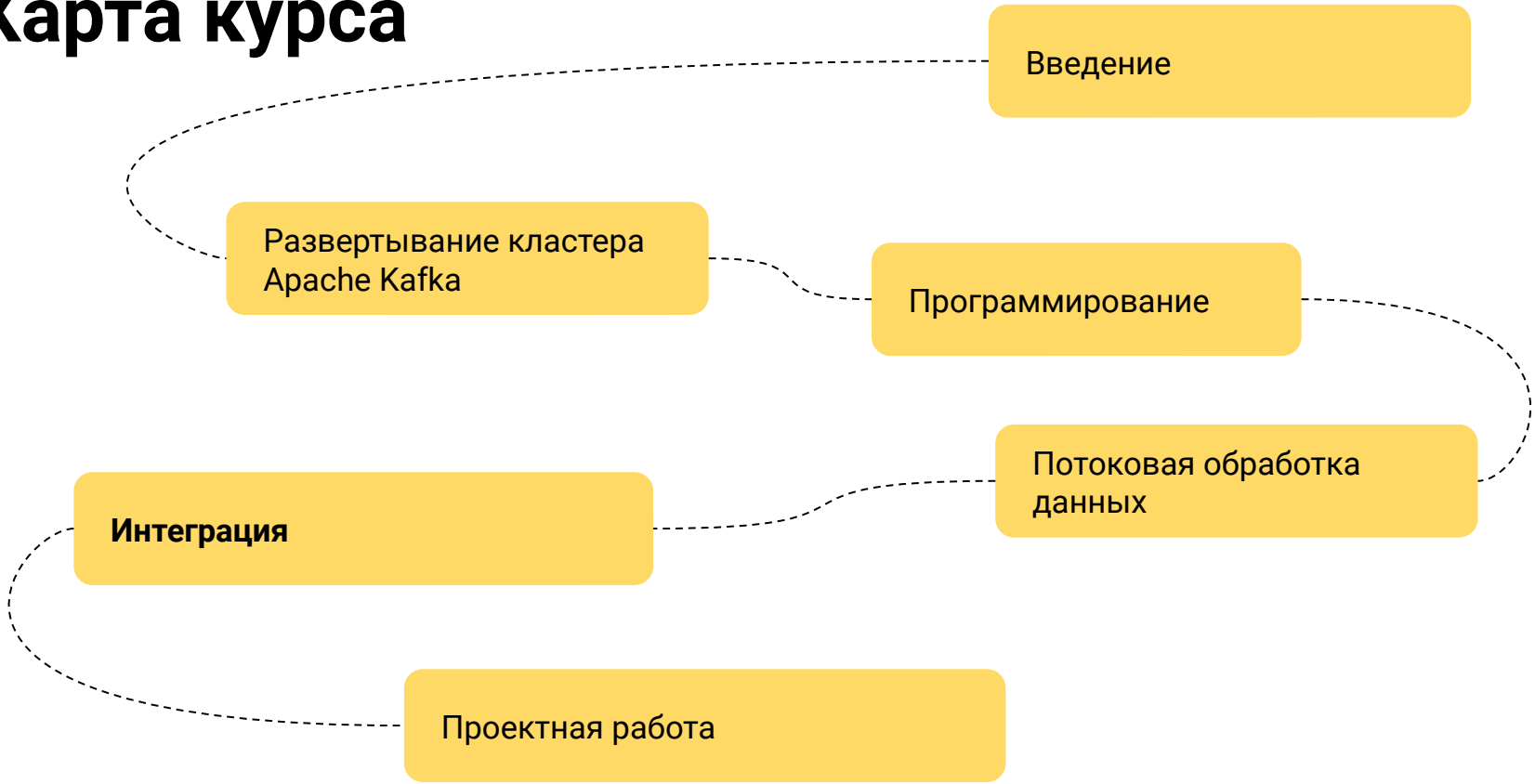
Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Карта курса


Введение
Развертывание кластера Apache Kafka
Программирование
Потоковая обработка данных
Интеграция
Проектная работа

Маршрут вебинара

- Конвейеры данных
- Kafka Connect
- Примеры коннекторов
- Разработка коннектора
- Рефлексия

Цели вебинара

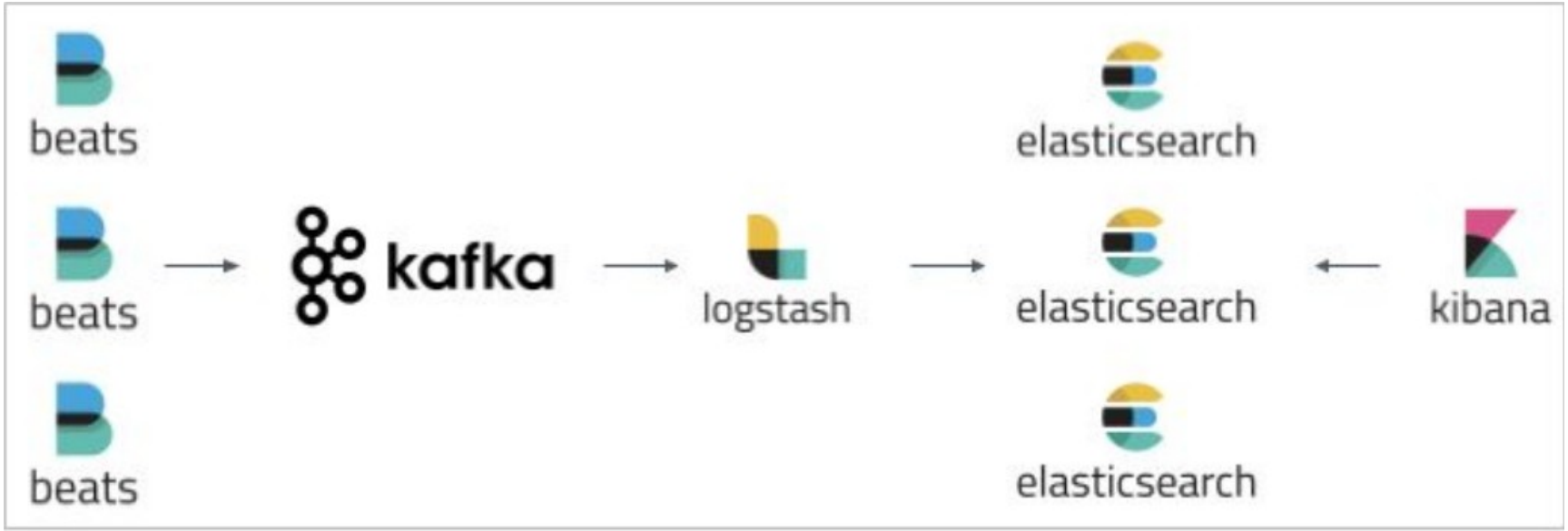
1.	Познакомимся с конвейерами данных
2.	Узнаем что такое Kafka Connect
3.	Рассмотрим создание своих коннекторов

Смысл

1.	Сможем запускать Kafka Connect
2.	Сможем разрабатывать коннекторы

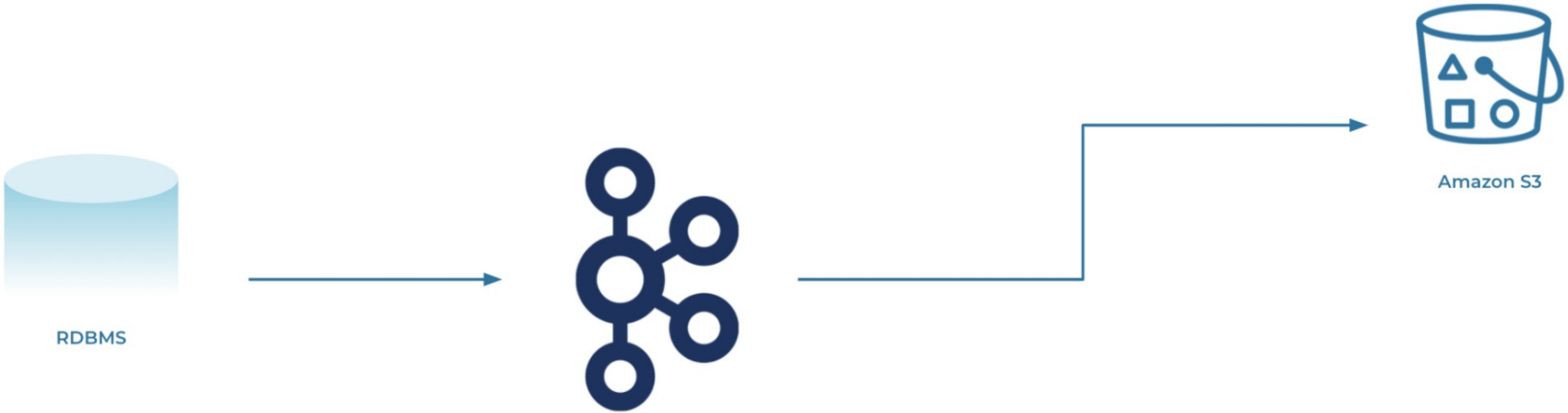
Конвейеры данных

Пример использования Kafka



Потоковый конвейер данных

Потоковый конвейер данных — это место, где мы отправляем данные из источника в целевую систему по мере их поступления.



Создание конвейеров данных

- **Своевременность** — соответствие различным требованиям к времени
- **Надёжность:**
  - быстрое автоматическое восстановление после сбоев
  - гарантия доставки
- **Высокая/переменная нагрузка —** распараллеливание работы
- **Форматы данных**:
  - согласование форматов
  - поддержка схем
- **Преобразования** — ETL или ELT
- **Безопасность** — шифрование, аутентификация, авторизация
- **Обработка сбоев** — что делать с «плохими» данными

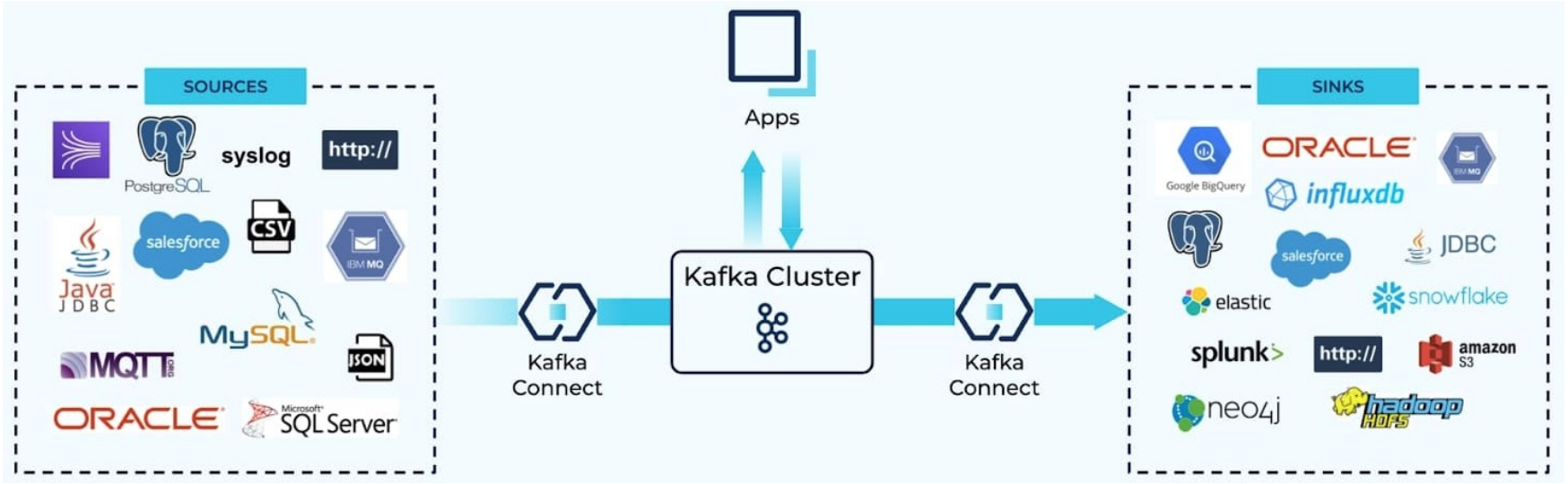
Что использовать?

- **Producer / Consumer API** — возможность модифицировать код приложения, к которому надо подключиться
- **Kafka Connect** — подключение к существующим системам
- **Любой** — когда нужно подключиться к системе, для которой нет коннектора

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 понятия Kafka Connect

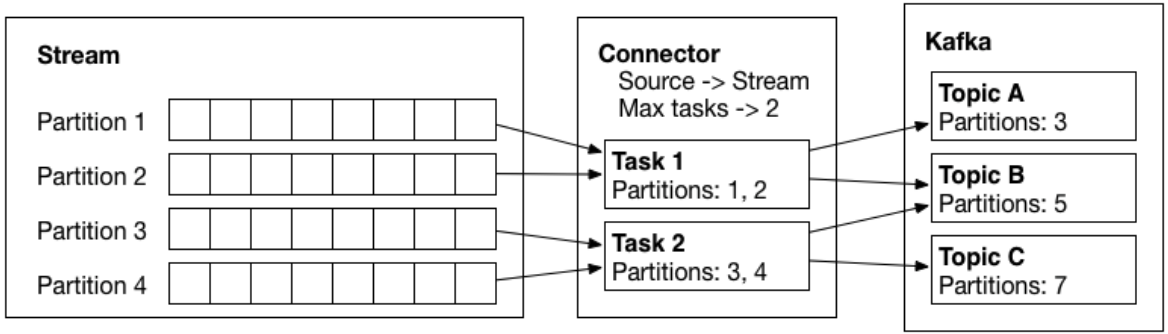
- **Connector** (Коннектор) – координатор потоковой передачи данных, запускает задачи и управляет ими
- **Task** (Задача) – реализация способа копирования данных в/из Kafka
- **Worker** (Исполнитель) – процесс, выполняющий коннекторы и задачи
- **Convertor** (Конвертер) – готовый компонент, преобразующий форматы данных
- **Transform** (Преобразователь) – готовый компонент с простой логикой изменения одного сообщения
- **Dead Letter Queue** (Очередь недоставленных сообщений) – место хранения сообщений, при обработке которых возникли ошибки

Коннектор (Connector)

Коннектор отвечают за:

- Определение числа задач для коннектора
- Разделение работы по копированию данных между задачами
- Получение от исполнителей настроек для задач

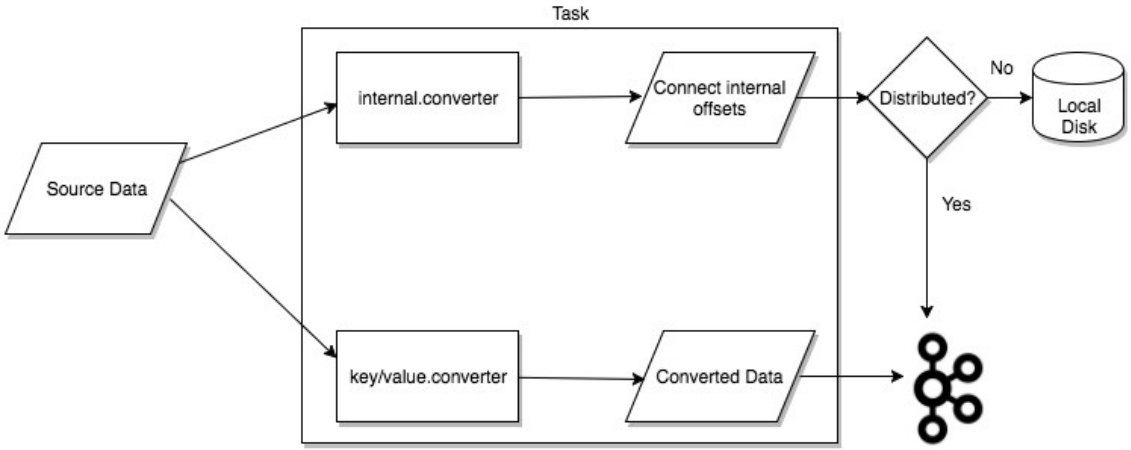
Экземпляр коннектора — это задание (Job), которое отвечает за управление копированием данных между Kafka и другой системой



Задача (Task)

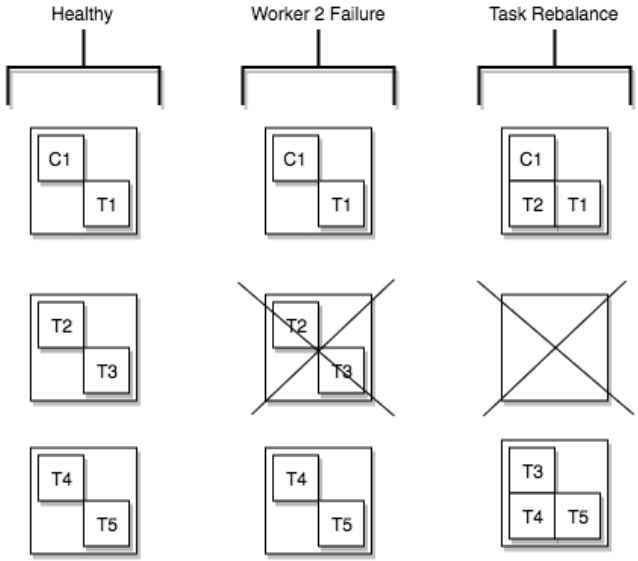
Задачи отвечают за получение данных из и отправку данных в Kafka

- Задачи являются элементами, из которых строятся задания (Job)
- Коннектор разбивает одно задание на множество задач, обеспечивая параллелизм и масштабируемость при копировании данных
- Каждая задача выполняется в своем собственном потоке (java thread)



Перебалансировка задач

- Перераспределение задач между исполнителями (worker) чтобы у них был примерно одинаковый объем работы
- При выходе исполнителя из строя, его задачи перераспределяются между оставшимися активными исполнителями
- При сбое отдельной задачи перебалансировка не запускается, поскольку сбой задачи считается исключительным случаем
  - Такие задачи не перезапускаются автоматически и должны быть перезапущены через REST API

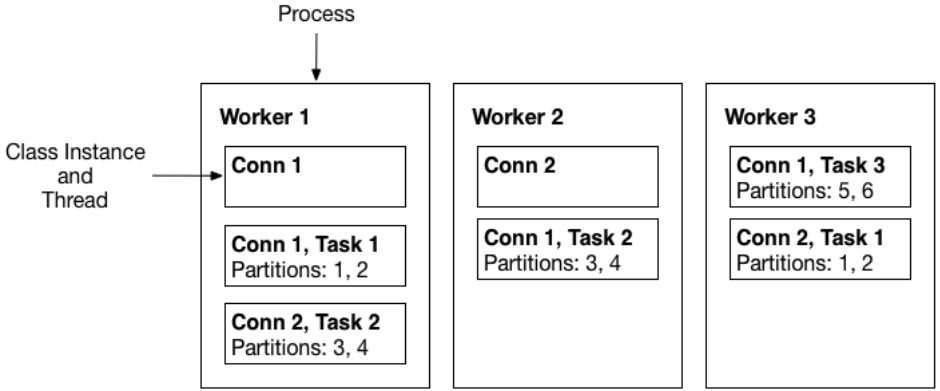


Исполнитель (Worker)

Исполнители — это контейнеры для коннекторов и задач (логические единицы работы)

Отвечают за:

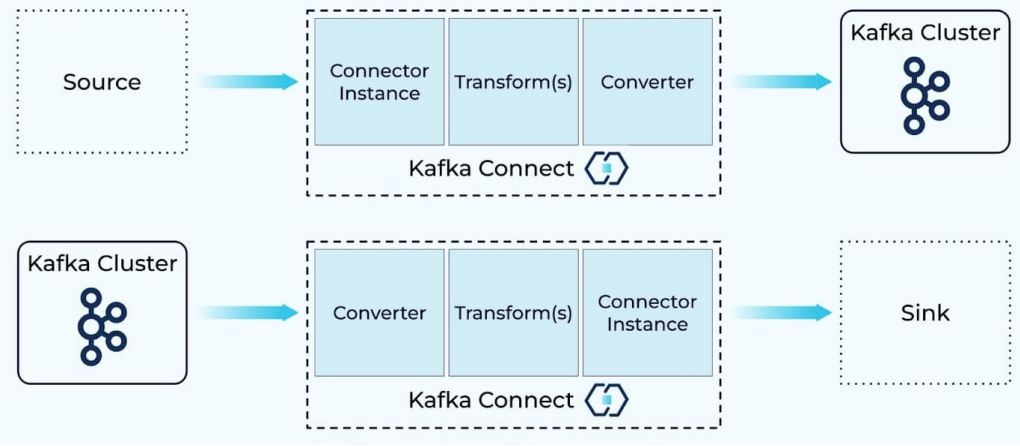
- Обработку REST-запросов для настройки коннекторов
- Хранение настроек коннекторов
- Запуск коннекторов и задач
- Переназначение коннекторов при сбоях и присоединении исполнителей
- Автоматическую фиксацию смещений



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



Встроенные ковертеры

- Примитивные:
  - `org.apache.kafka.connect.converters.DoubleConverter`
  - `org.apache.kafka.connect.converters.FloatConverter`
  - `org.apache.kafka.connect.converters.IntegerConverter`
  - `org.apache.kafka.connect.converters.LongConverter`
  - `org.apache.kafka.connect.converters.ShortConverter`
- Confluent Platform:
  - `AvroConverter io.confluent.connect.avro.AvroConverter`
  - `ProtobufConverter io.confluent.connect.protobuf.ProtobufConverter`
  - `JsonSchemaConverter io.confluent.connect.json.JsonSchemaConverter`
  - `JsonConverter org.apache.kafka.connect.json.JsonConverter`
  - `StringConverter org.apache.kafka.connect.storage.StringConverter`
  - `ByteArrayConverter org.apache.kafka.connect.converters.ByteArrayConverter`

Очередь недоставленных сообщений

- Специальный топик для хранятся сообщения, которые не удалось обработать коннектору-приемнику (обработка ошибок)
- Очереди недоставленных сообщений применимы только для коннекторов-приемников
- `errors.tolerance = none` — ошибка или недопустимая запись приводит к немедленному завершению задачи коннектора и переходу коннектора в состояние сбоя
- `errors.tolerance = all` — все ошибки или недопустимые записи игнорируются, и обработка продолжается
- `erros.deadletterqueue.topic.name=badtopic` — топик для недоставленных сообщений
- `errors.deadletterqueue.context.headers.enable=true` — включать ли заголовки

Работа Kafka Connect

Kafka Connect выполняется в виде кластера процессов-*исполнителей* (worker process).

- На исполнителях устанавливаются плагины-*коннекторы*.
  - REST API используется для настройки и управления коннекторами.
- Коннекторы запускают *задачи* (tasks)
  - Задачи перемещают данные.
- *Преобразователи форматов* (convertors) обеспечивают преобразование данных в разные форматы

Запуск Kafka Connect

Запуск Kafka Connect

```
bin/connect-distributed.sh config/connect-distributed.properties
```

Свойства:

- `bootstrap.servers` — список брокеров Kafka
- `group.id` — id кластера Connect
- `plugin.path` — путь к папке с плагинами-коннекторами
- `key.converter` и `value.converter` — преобразователи формата для ключа и значения
- `rest.host.name` и `rest.port` — параметры подключения по REST API

Пример 1

- `bin/zookeeper-server-start.sh -daemon config/zookeeper.properties`
- `bin/kafka-server-start.sh -daemon config/server.properties`
- `bin/connect-distributed.sh -daemon config/connect-distributed.properties`
- `curl http://localhost:8083`
- `curl http://localhost:8083/connector-plugins | jq`

LIVE

Пример 2. Файловый источник и приёмник

- `docker-compose up -d`
- `kafka-topics.sh --create --topic data --bootstrap-server localhost:9092,localhost:9093,localhost:9094`
- `curl -X POST --data-binary "@source.json" -H "Content-Type: application/json" http://localhost:8083/connectors`
- `kafka-console-consumer.sh --topic data --bootstrap-server localhost:9092,localhost:9093,localhost:9094 —from-beginning`
- `curl -X POST --data-binary "@dump.json" -H "Content-Type: application/json" http://localhost:8083/connectors`

LIVE

Пример 3. PostgreSQL

- `docker-compose up -d`
- `CREATE TABLE clients (id int PRIMARY KEY, first_name text, last_name text, gender text, card_number text, bill numeric(7,2), created_date timestamp, modified_date timestamp);`
- `curl -X POST --data-binary "@clients.json" -H "Content-Type: application/json" http://localhost:8083/connectors | jq`
- `kafka-console-consumer.sh --topic postgres.clients --bootstrap-server localhost:9092,localhost:9093,localhost:9094 --from-beginning --property print.offset=true`
- `update clients set bill = 5000, modified_date = current_timestamp(0) where id = 262;`

LIVE

Пример 4. PostgreSQL CDC

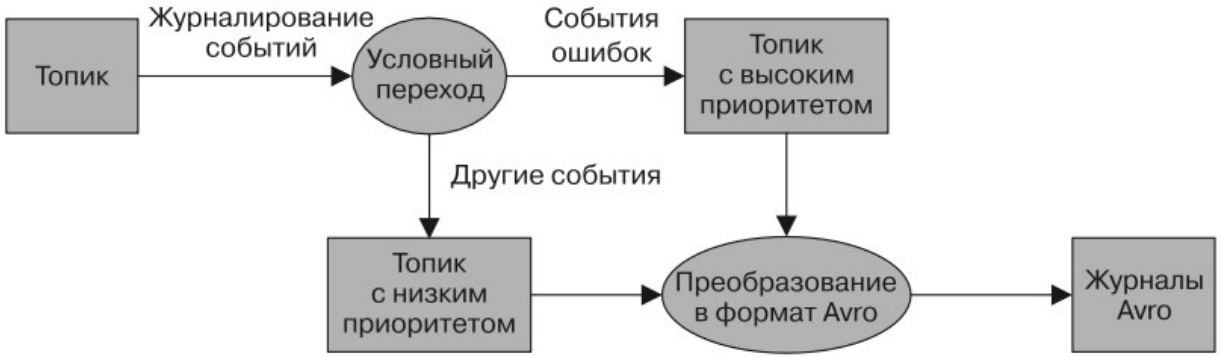
- `docker-compose up -d`
- `curl -X POST --data-binary "@inventory.json" -H "Content-Type: application/json" http://localhost:8083/connectors`
- `kafka-topics.sh --list --bootstrap-server localhost:9092,localhost:9093,localhost:9094`
- `docker exec -ti -e PGOPTIONS="--search_path=inventory" demo3-postgres-1 psql -U postgres`
- `update customers set first_name = 'Sarah' where id = 1001;`

LIVE

Преобразования

Обработка событий по отдельности

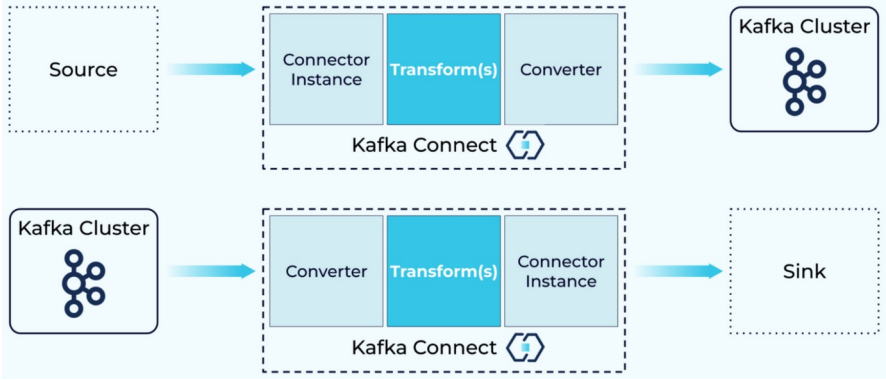
- *Паттерн отображения/фильтрации*
- Приложение читает события из потока, модифицирует каждое из них, после чего генерирует события в другой поток



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

Пример 5. PostgreSQL SMT

- `docker-compose up -d`
- `CREATE TABLE clients (id int PRIMARY KEY, first_name text, last_name text, gender text, card_number text, bill numeric(7,2), created_date timestamp, modified_date timestamp);`
- `curl -X POST --data-binary "@clients-smt.json" -H "Content-Type: application/json" http://localhost:8083/connectors | jq`
- `kafka-console-consumer.sh --topic postgres.clients --bootstrap-server localhost:9092,localhost:9093,localhost:9094 --from-beginning --property print.offset=true --property print.headers=true`

LIVE

Разработка коннекторов

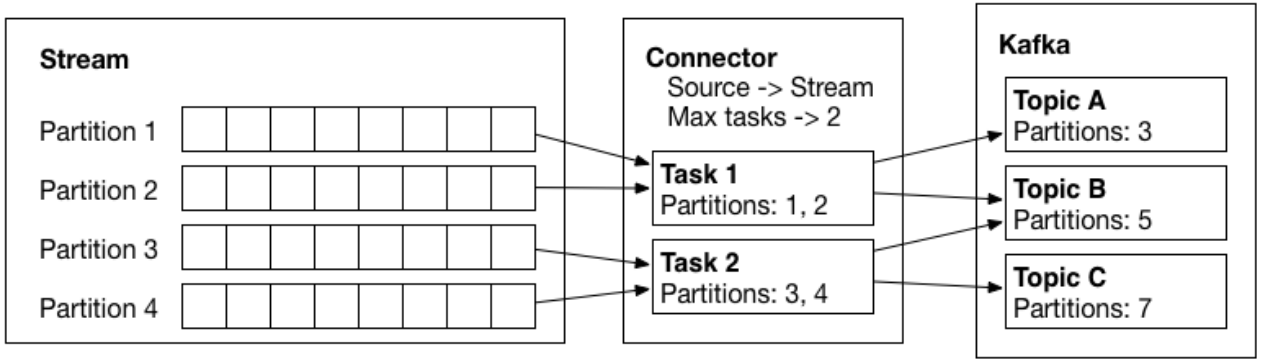
Connector

- `Connector` — управляет интеграцией Kafka Connect с другой системой
  - `SourceConnector` — реализуйте интерфейс Connector для извлечения данных из другой системы и отправки их в Kafka
  - `SinkConnector` — реализуют интерфейс Connector для отправки данных Kafka в другую систему

- Задачи Connector:
  - Создание конфигураций для набора задач (tasks), которые разделяют обработку данных. Например, Connector базы данных может создавать задачи, равномерно распределяя набор таблиц между задачами
  - Мониторинг входных данных на предмет изменений, требующих реконфигурации. Например, Connector может периодически проверять наличие новых таблиц и уведомлять Kafka Connect о добавлениях и удалениях

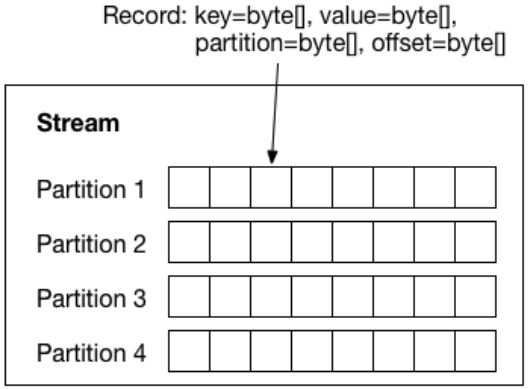
Task

- `Task` — содержат код, который фактически копирует данные в/из другой системы.
  - `SourceTask` — задача, которая извлекает записи из другой системы для отправки их в Kafka
  - `SinkConnector` — задача, которая берет записи из Kafka, и отправляет их в другую систему
- Данные, которые копирует Connector, должны быть представлены в виде секционированного потока (**partitioned stream**), аналогично модели топиков Kafka, где каждая партиция представляет собой упорядоченную последовательность записей со смещениями.
- Каждой задаче назначается подмножество партиций для обработки.



Partitions и Records

- Каждая партиция представляет собой упорядоченную последовательность записей типа ключ-значение
- Типы данных:
  - Стандартные типы Java: `java.lang.Integer`, `java.lang.Map`, `java.lang.Collection`
  - `org.apache.kafka.connect.data` — классы для представления данных и схем
  - `Struct` — для структурированная запись, содержащая набор именованных полей со значениями



Пример: FileStreamSinkConnector

- `FileStreamSourceConnector`:
  https://github.com/apache/kafka/blob/3.5/connect/file/src/main/java/org/apache/kafka/connect/file/FileStreamSourceConnector.java
- `FileStreamSourceTask`:
  https://github.com/apache/kafka/blob/3.5/connect/file/src/main/java/org/apache/kafka/connect/file/FileStreamSourceTask.java
- `FileStreamSinkConnector`:
  https://github.com/apache/kafka/blob/3.5/connect/file/src/main/java/org/apache/kafka/connect/file/FileStreamSinkConnector.java
- `FileStreamSinkTask`:
  https://github.com/apache/kafka/blob/3.5/connect/file/src/main/java/org/apache/kafka/connect/file/FileStreamSinkTask.java

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Литература

Список материалов для изучения

1. Apache Kafka Connect https://kafka.apache.org/documentation/#connect
2. Confluent Kafka Connect https://docs.confluent.io/platform/current/connect/index.html
3. Connector Hub https://www.confluent.io/hub/
4. Debezium https://debezium.io
5. Confluent Connector Developer Guide
   https://docs.confluent.io/platform/current/connect/devguide.html
6. 4 Steps to Creating Apache Kafka Connectors with the Kafka Connect API
   https://www.confluent.io/blog/create-dynamic-kafka-connect-source-connectors/
7. Maven quick start for building Kafka Connect connectors
   https://github.com/jcustenborder/kafka-connect-archtype

Рефлексия

Рефлексия

С какими впечатлениями уходите с вебинара?

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

Следующий вебинар

Schema Registry

- Ссылка на вебинар будет в ЛК за 15 минут
- Материалы к занятию в ЛК — можно изучать
- Обязательный материал обозначен красной лентой

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары

Заигрин Вадим

Ведущий эксперт по технологиям, Сбербанк

vzaigrin@yandex.ru
https://t.me/vzaigrin