ОНЛАЙН-ОБРАЗОВАНИЕ

Не забыть включить запись!

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Правила вебинара

Активно участвуем

Задаем вопрос в чат или голосом

slack

Off-topic обсуждаем в Slack #канал группы или #general

Вопросы вижу в чате, могу ответить не сразу

OTUS

Занятие 14.
Confluent Platform.

Задорожный Антон

Цели вебинара

После занятия вы сможете:

1. выбирать компоненты платформы Kafka /Confluent
2. выбирать подходящие API для вашего стриминг приложения
3. разрабатывать простые приложения на KSQL

Краткое содержание

OTUS

- Введение в Apache Kafka и дистрибутив Confluent
- Модель данных и базовый API
- Архитектура кластера
- Управление схемами
- RESTful API
- Kafka Connect
- Kafka Streams
- KSQL
- Проприетарные компоненты

Введение в Apache Kafka / Confluent

Apache Kafka

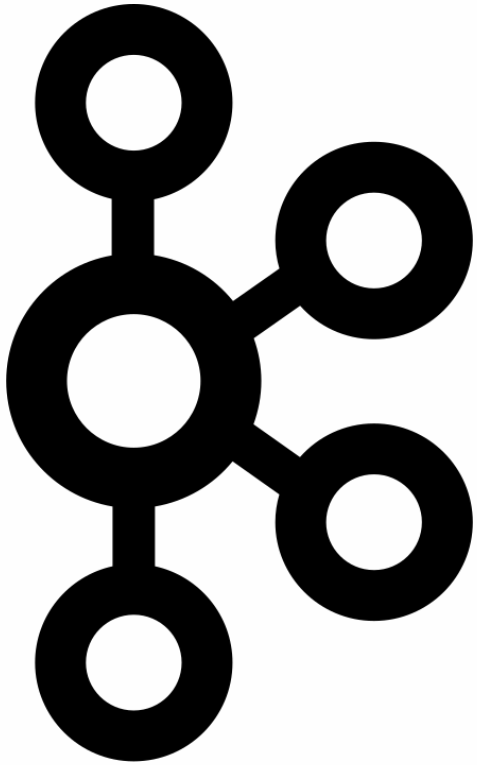
OTUS

- Распределенный персистентный брокер сообщений по архитектуре commit log
- Появился в LinkedIn, open source – 2011, Apache top tier проект – 2012
- Ядро написано на Scala, также много Java кода
- Стандарт де-факто для крупных стриминговых приложений
- Назван в честь Франца Кафки

Поставляется в составе нескольких дистрибутивов:

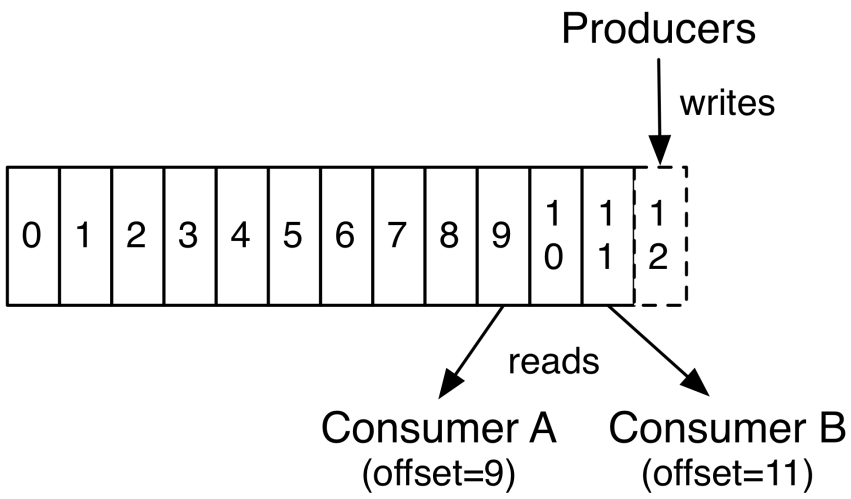
- Hortonworks Data Flow (HDF)
- Cloudera
- Amazon Managed Streaming for Kafka (MSK)
- Confluent

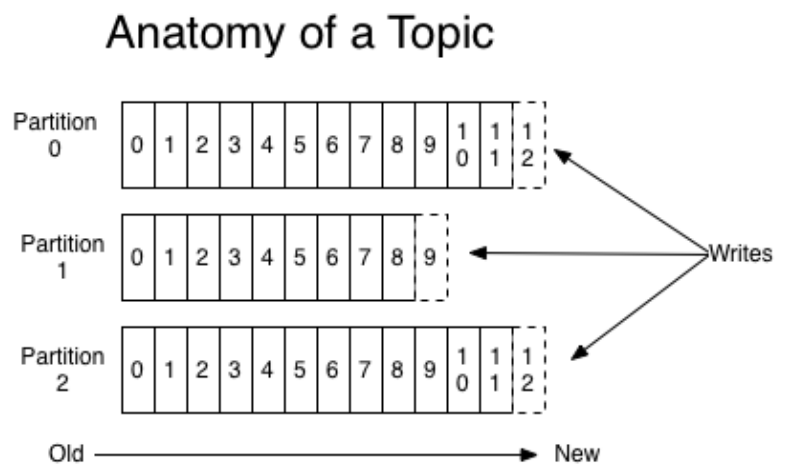
- Confluent – компания авторов Kafka



Основные концепции

- Record – Запись, состоящая из ключа и значения
- Topic – Топик: категория или имя потока куда публикуются записи
- Producer – Продюсер: процесс публикующий данные в топик
- Consumer – Консьюмер: процесс читающий данные из топика
- Offset – Смещение: позиция записи
- Partition – Партиция: единица параллелизма для топика





Модель данных и базовый API

Модель данных

- Внутри брокера модель данных сообщения это Pair<ByteArray, ByteArray>
- Все знание о типах данных и сериализации/десериализации возложено на клиентские приложения *

*) Есть подпорка в виде регистра схем

Producer API

- Документация
- Producer требует подмножества брокеров для инициализации (bootstrap)
- Producer используется вместе с сериализатором для ключей и значений
- Распределение записей по партициям управляется объектом класса Partitioner
- DefaultPartitioner распределяет по key.hash() % numPartitions если ключ не null, и round-robin для ключей null
- Свойство acks контролирует количество подтверждений от реплик (0/1/all)
- Два режима повышенных гарантий: идемпотентный (отсутствие дубликатов в одной партиции/топике) и транзакционный (поддержка транзакционной записи сразу в несколько партиций и топиков)

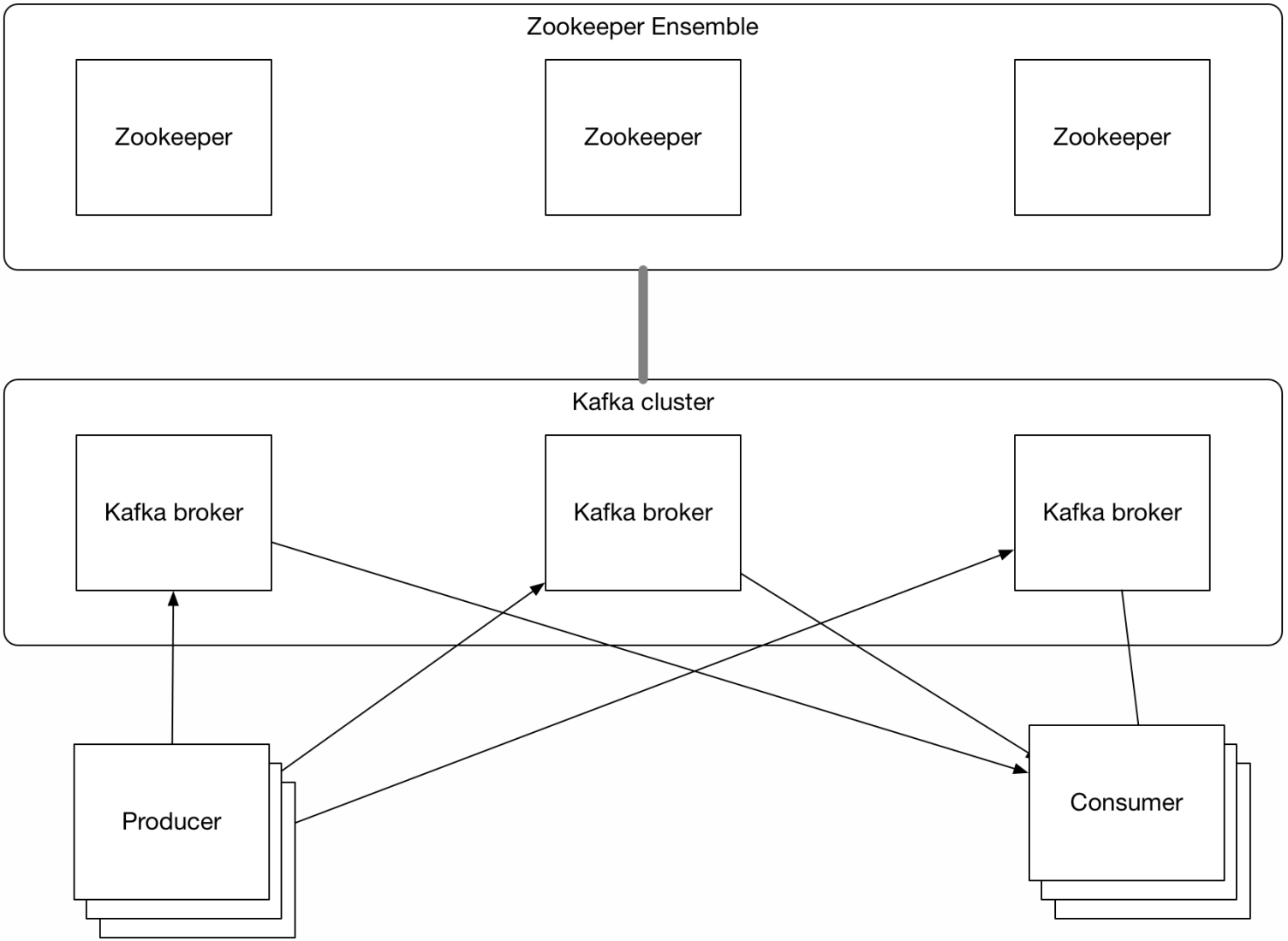
- Сериализатор, Partitioner – это настройки клиентского приложения, они никак не контролируются со стороны брокера!

Consumer API

- Документация
- Аналогично продюсеру, консьюмер должен знать модель данных
- Консьюмер может использовать хранение смещений в Kafka, а может хранить снаружи
- Консьюмеры опрашиваются брокерами и в случае отсутствия heartbeat партиции передаются другим консьюмерам
- Транзакционные консьюмеры должны указывать уровень изоляции и работать с транзакционным логом, в топиках могут быть пропуски сообщений

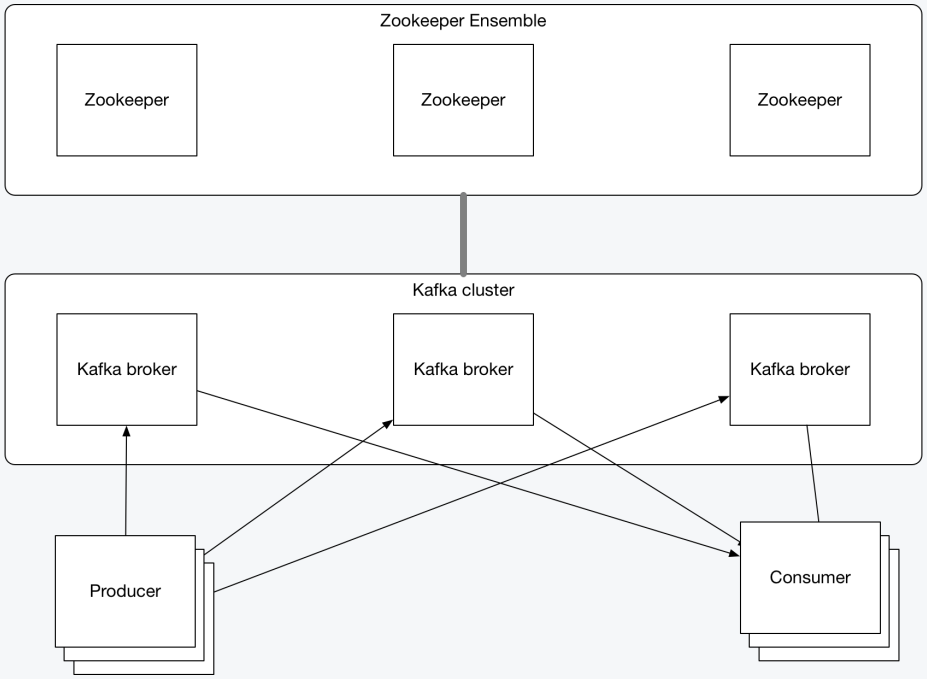
Архитектура кластера

Архитектура кластера Apache Kafka

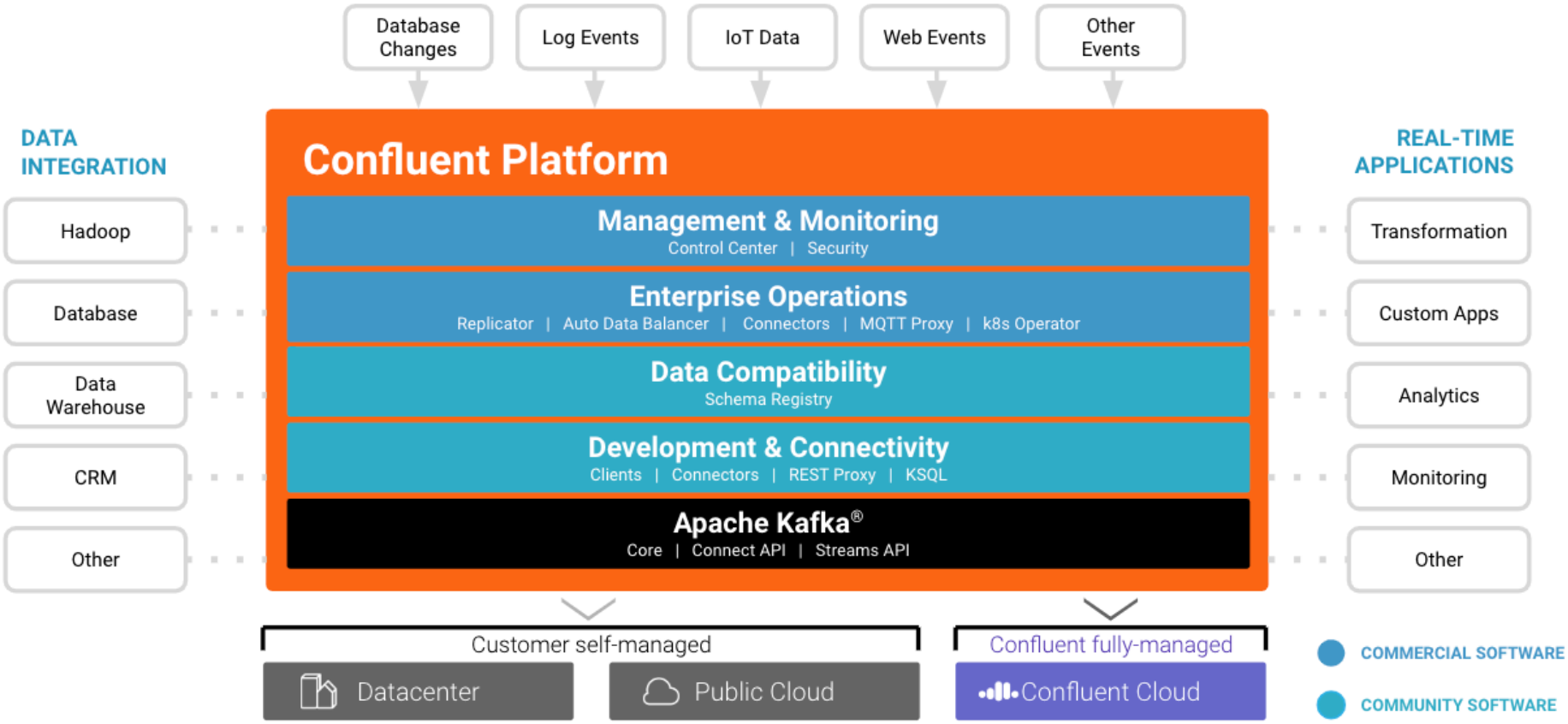


Компоненты кластера

- Брокеры Kafka: управление данными, взаимодействие с клиентами
- Брокеры не исполняют бизнес-логику
- Apache Zookeeper: членство брокеров в кластере, выборы контроллера
- Ранее Zookeeper использовался для хранения метаданных о топиках, смещений клиентов и консьюмеры общались с Zookeeper напрямую, сейчас это не так



Структура Confluent Platform



Дополнительные сервисы

- Schema Registry: управление схемами
- Kafka REST Proxy: RESTful API для использования Kafka
- Kafka Connect: сервис интеграции Kafka с другими системами
- KSQL: стриминговый фреймворк и сервер приложений от Confluent

Управление схемами

Schema Registry

- Сервис для коммуникации продюсеров и консьюмеров относительно схем данных, хранимых в Kafka
- Прежде всего рассчитан на использование Avro
- Позволяет задавать схемы для ключа и значения топика, поддерживает эволюцию схем
- Имеет широкий набор интеграций: Kafka Connect, KSQL, Spark/Flink
- Реализуется отдельным приложением

RESTful API

Kafka REST Proxy

- RESTful API для приложений где нельзя использовать нативный клиент
- Позволяет получить конфигурацию кластера, брокеров, топиков
- Продюсер/консьюмер: обеспечивает запись и чтение из Kafka в интеграции с регистром схем
- Форматы данных: поддерживает запись и чтение JSON, массивов байт (base64), и Avro сообщений сериализованных в JSON
- Реализуется отдельными процессами
- Поддерживает кластеризацию (нужен внешний балансировщик)

Kafka Connect

Kafka Connect

- Фреймворк и сервер приложений для интеграции Kafka с другими системами, например базами данных, другими брокерами, файловыми системами
- Архитектурно предназначен для технической интеграции систем, а не для бизнес-логики
- Реализуется внешним приложением с двумя режимами: одиночный и распределенный
- Confluent Hub – магазин коннекторов для Kafka Connect

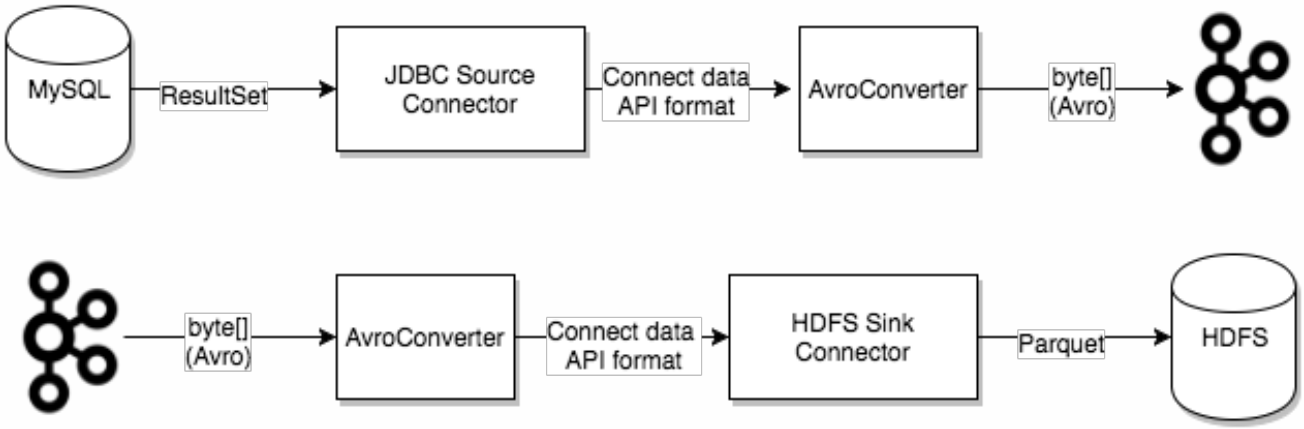
Основные концепции Kafka Connect

- Connector – плагин для интеграции с определенной платформой

Оперирует двумя типами коннекторов:

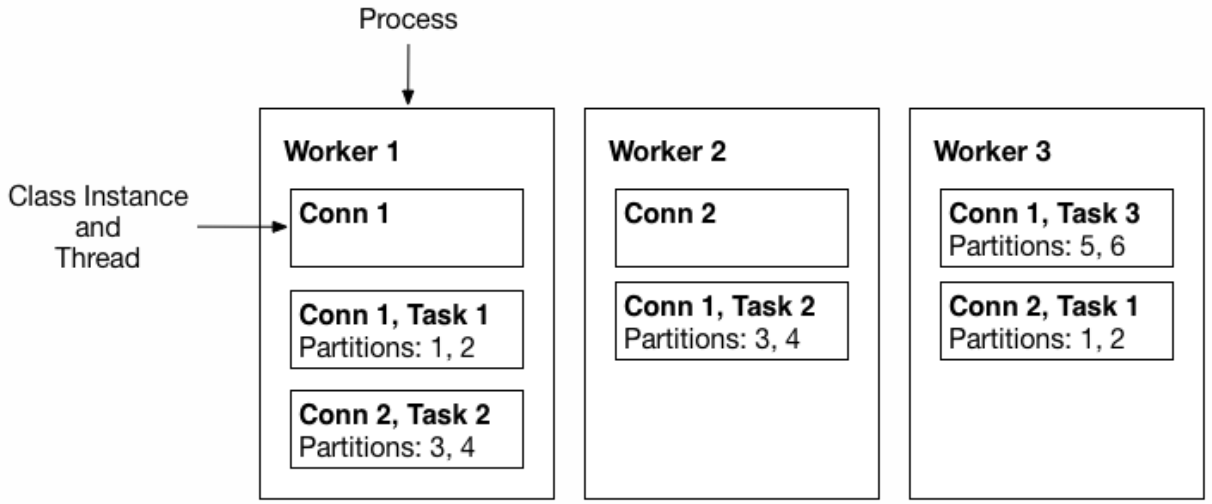
- Source connector – коннектор для передачи данных в Kafka
- Sink connector – коннектор для передачи данных из Kafka

- Converter – конвертер конкретного формата в/из формат Connect data API
- Transform – опциональная трансформация записи внутри Connect



Архитектура приложения Kafka Connect

- Connector instance – логическая задача связывающая конкретный кластер Kafka с конкретным экземпляром другой платформы
- Worker – экземпляр сервера приложений Kafka Connect
- Task – единица параллелизма connector instance



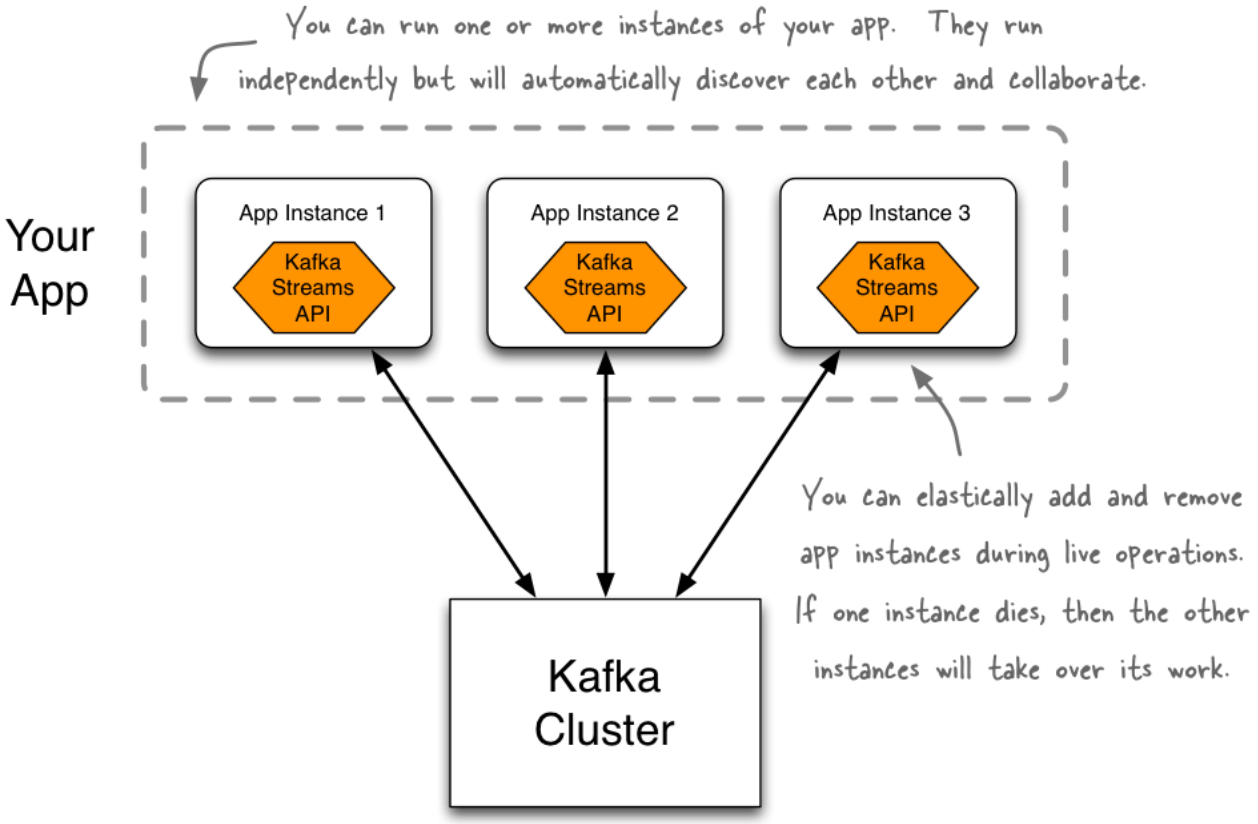
Лучшие практики

- Kafka Connect vs streaming framework
- Kafka Connect это отдельное выделенное приложение – не конфликтует за ресурсы с вашим Spark/k8s/YARN кластером, не имеет зависимостей по запуску
- Avro everywhere!
- Опасаться бизнес-логики в коннекторах

Kafka Streams

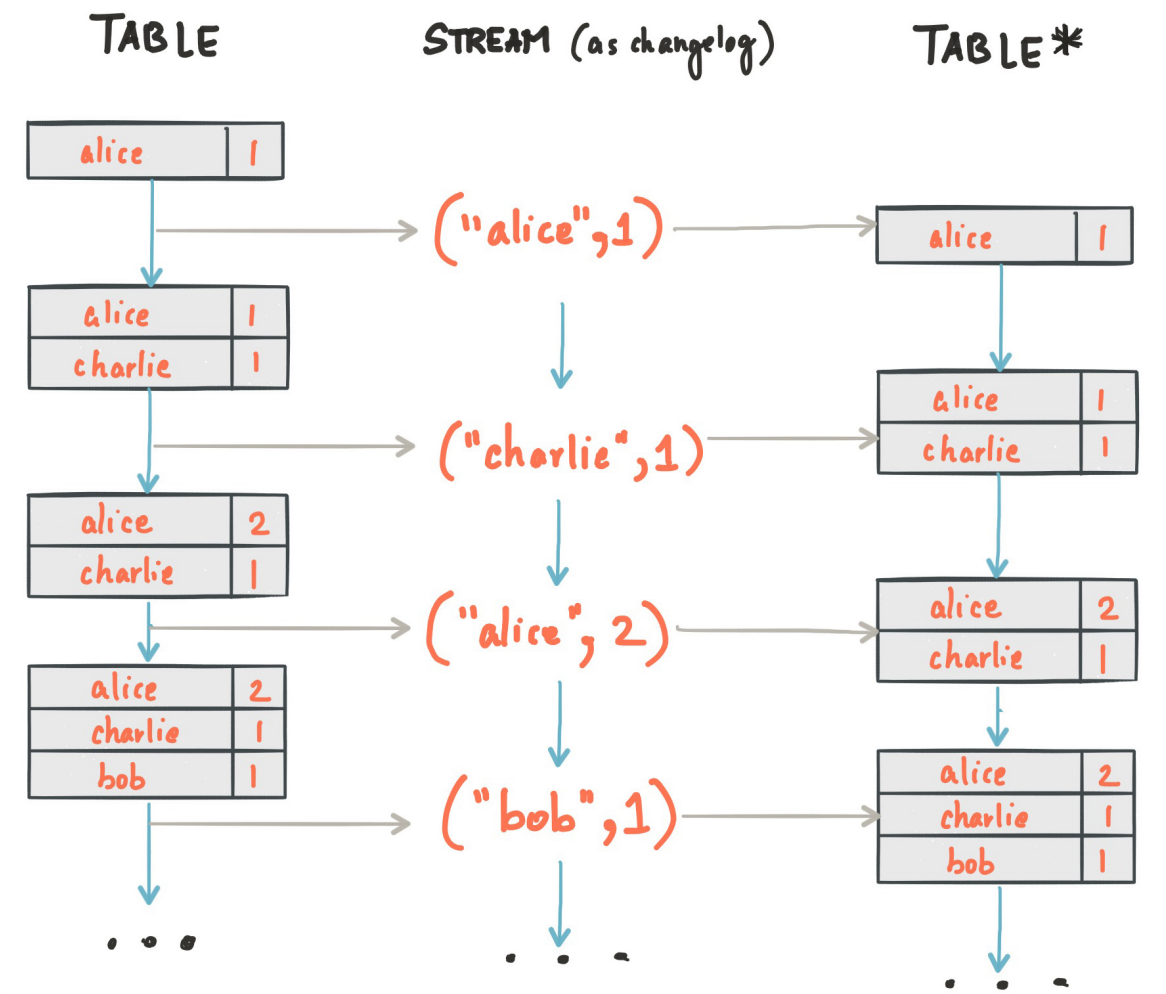
Kafka Streams

- Помните «базовый API»?
- Kafka Streams – не базовый API для написания приложений поверх Kafka
- Kafka Streams это фреймворк со своим API и DSL, но не сервер приложений, поэтому разработчикам нужно самим решать задачи развертывания и масштабирования приложения



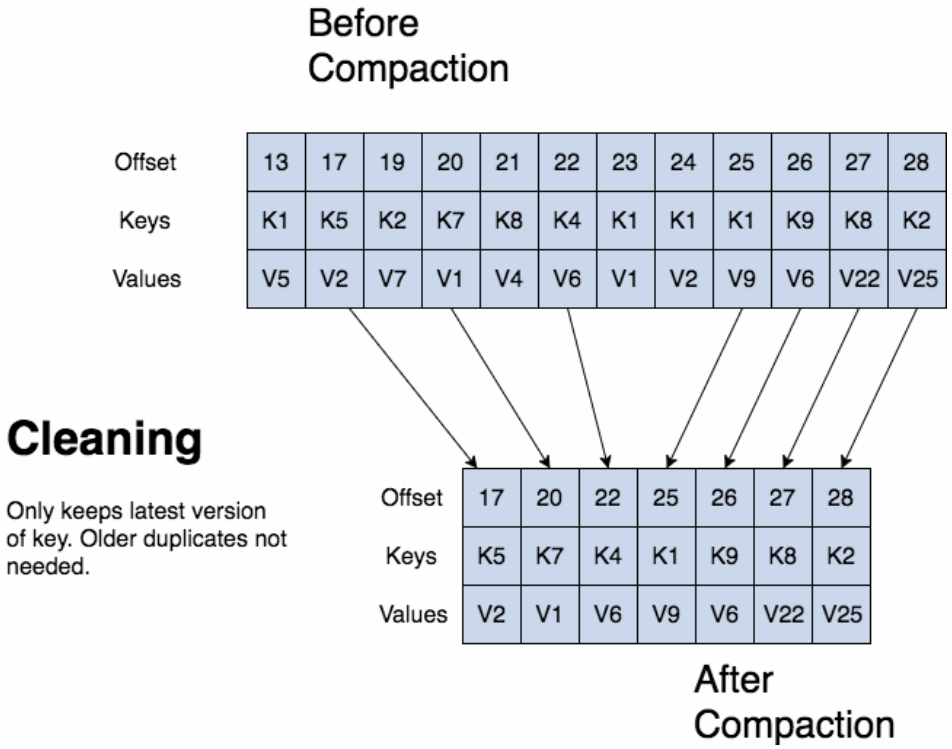
Основные концепции Kafka Streams

- KStream – записанный поток событий по ключам
- KTable – записанный поток изменений по ключам



Отступление о компакции топиков

- Для длительного хранения и KV-like функциональности Kafka предлагает технологию компакции топиков
- Используется для топиков с непустым ключом
- Старые записи в топике будут видны в простых API, но маскируются в Streams KTable
- Tombstone записи – ключ + пустое значение, используется для «удаления»



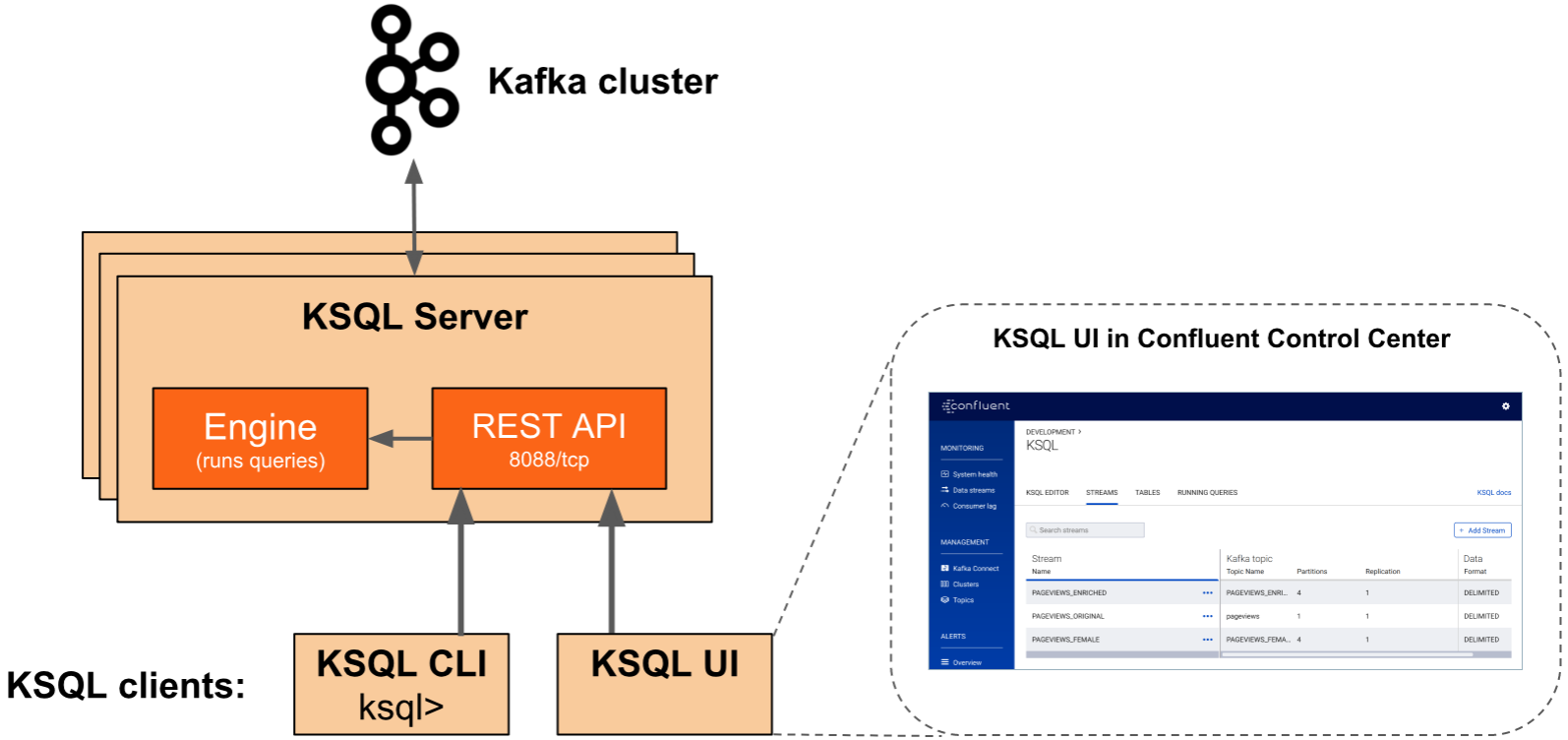
Kafka Streams DSL

- Domain Specific Language для Java и Scala вокруг Streams API
- Позволяет разрабатывать приложения в стиле похожем на Apache Beam или Flink

KSQL

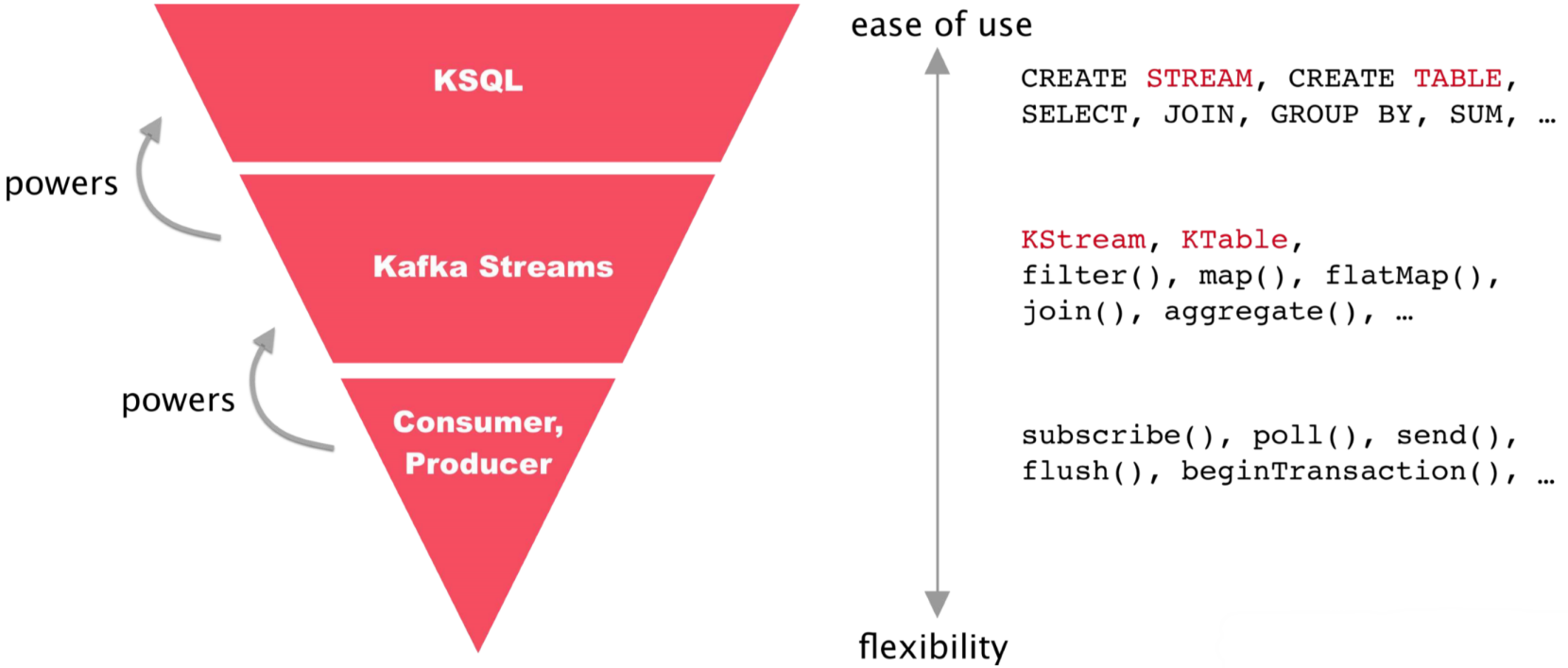
KSQL

- Язык и сервер приложений для разработки стриминговых приложений поверх Kafka Streams



KSQL и Kafka Streams

- KSQL построен на Kafka Streams
- В отличии от Kafka Streams предоставляет свой сервер приложений
- Confluent Vision – если приложение может быть просто реализовано на KSQL то нужно использовать его, во всех остальных случаях используйте Kafka Streams



Проприетарные компоненты

Проприетарные компоненты Confluent Platform

- Control Center: управляющий сервис
- Replicator: расширение Kafka Connect для репликации данных между несколькими кластерами
- Auto Data Balancer: балансировщик партиций между брокерами
- MQTT Proxy: прокси протокола MQTT для Internet of Things
- Connectors: дополнительные коннекторы для Kafka Connect (и поддержка)
- Security: дополнительные возможности по безопасности, например CORS и SAML
- (в будущем) Kubernetes operator

Следующий вебинар

Тема: Elasticsearch.

Среда 2019.07.24 в 20.00

Ссылка на вебинар будет в ЛК за 15 минут

Заполните, пожалуйста, опрос о занятии

Спасибо
за внимание!

ОНЛАЙН-ОБРАЗОВАНИЕ