OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

Брокеры сообщений. История появления и развития. Решаемые задачи

Заигрин Вадим

Ведущий эксперт по технологиям, Сбербанк

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

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

Маршрут вебинара

- Знакомство
- О курсе
- Брокеры сообщений
- Поток данных
- Примеры использования
- Рефлексия

Знакомство

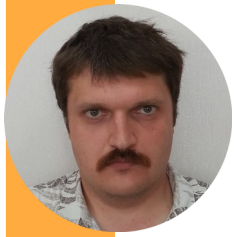
Преподаватель

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.

Команда курса



Евгений Непомнящий

C++ и Java разработчик.
В отрасли с 2006 года. Долгое время занимался разработкой под микроконтроллеры на C++, последние 3 года увлекся Java.

Разработчик в IT-Sense

Михаил Кузнецов

Fullstack developer и тимлид в компании Zalando, Germany. Проектирует и поддерживает процессы WEB-разработки. Окончил химфак МГУ.

Опыт в разработке - более 10 лет.
Консультировал по разработке, занимался популяризацией фронтенд-фреймворка Vue и SvelteJS.

Fullstack developer и тимлид в Zalando

Валентин Шилин

Выпускник СПбГУ ПМ-ПУ по специальности "Прикладная математика и физика".
Профессиональный опыт: 15 лет программирования на C++, C#, JavaScript, Python, Scala, Java.
Эксперт в обработке больших данных с помощью Scala/Spark и Hadoop Cloudera.

Старший программист/аналитик данных
Deutsche Telekom IT GmbH

Александра Чащина

Выпускница магистерской программы “Informatique pour la Science de Données (Big Data)” университета Париж-Сакле.

С 2018 года специализируется на больших данных. Участвует в проектах разработки аналитических платформ данных, а также платформ обработки данных в режиме реального времени на 10000+ пользователей.

Расскажите о себе

Напишите, пожалуйста, в чат или скажите голосом

Как вас зовут?

Какой опыт в IT?

Какие ожидания
от курса?

Заполните информацию
в разделе «О себе»
в личном кабинете

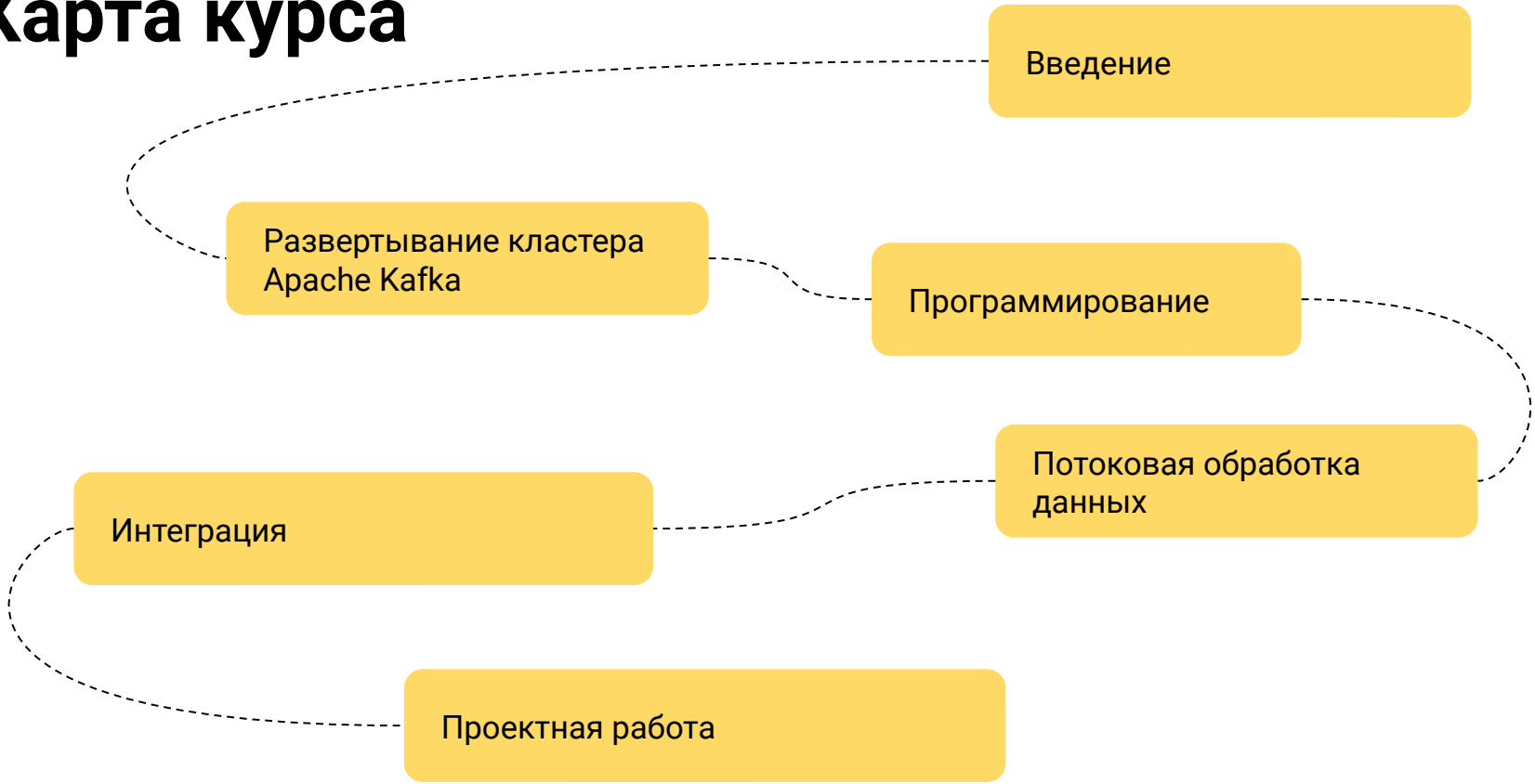
О курсе

О курсе

Курс адресован:

- разработчикам, которые хотят научиться грамотно организовать работу микросервисов и повысить общую надежность системы;
- инженерам данных, которые хотят научиться строить комплексные системы обработки данных;
- DevOps, SRE Engineer, архитекторам, которые хотят использовать всю мощь Kafka в работе.

Карта курса


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

Результат от курса

- **После обучения вы сможете:**
  - Развернуть Kafka;
  - Настроить брокеры и темы;
  - Использовать базовые API;
  - Разрабатывать программы на высокоуровневых фреймворках Kafka Streams, Spring, Akka;
  - Интегрировать Kafka с другими системами;
  - Настроить мониторинг и безопасность
- **Дополнительно на курсе вы:**
  - Познакомитесь с работой с топиками на SQL (ksqlDB);
  - Научитесь работать со Schema Registry;
  - Познакомитесь с REST API для работы с Kafka.

Как будем учиться?

Вебинары

понедельник, пятница, 20:00.
(запись и материалы выкладывают, как правило, на следующий день после вебинара)

Домашние задания

1 дз на модуль.
Дедлайна нет, кроме окончания курса :)
Типовой срок проверки: 2-3 дня

Чат в телеграм

Задавайте вопросы, обменивайтесь инсайтами.

Отзывы

Вебинары во многом адаптивны.
Мы следим за результатами опросов, обсуждениями и ДЗ.

Всегда рады вашим
конструктивным отзывам :)

Лайфхаки

Сделайте упор на тему, которая вам важна

Старайтесь полученные знания применять на практике

Задавайте вопросы, так материал лучше усваивается

Регулярное выполняйте ДЗ (наверстать пропуски тяжело)

Организационные вопросы

Про технические моменты:

- Рекомендуется использовать Linux
- Можно в облаке

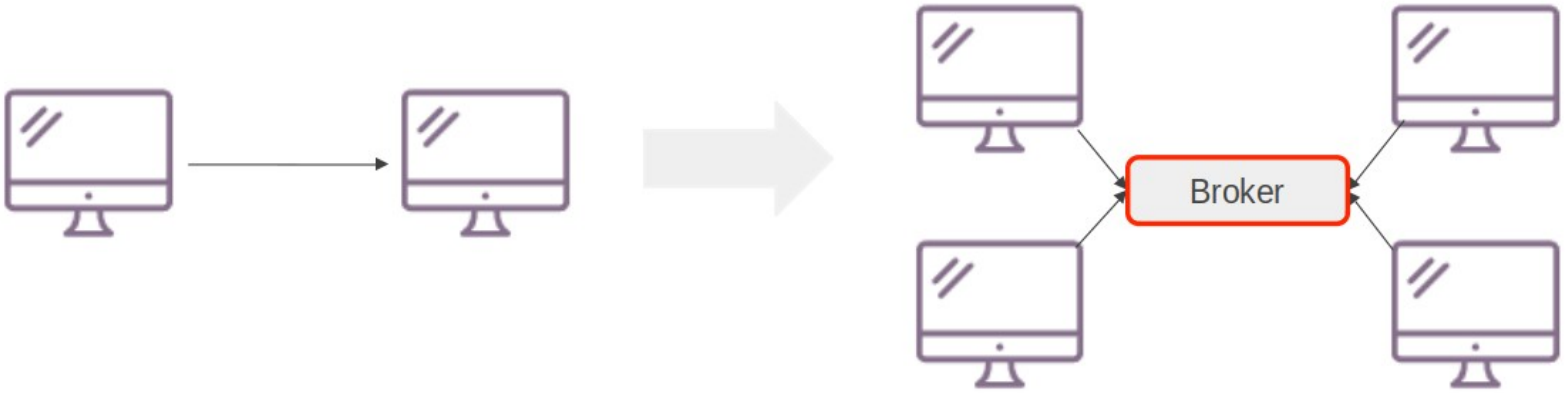
Какие инструменты будут использовать:

- Github
- Docker

Ваши вопросы?

Брокеры сообщений

Обмен сообщениями



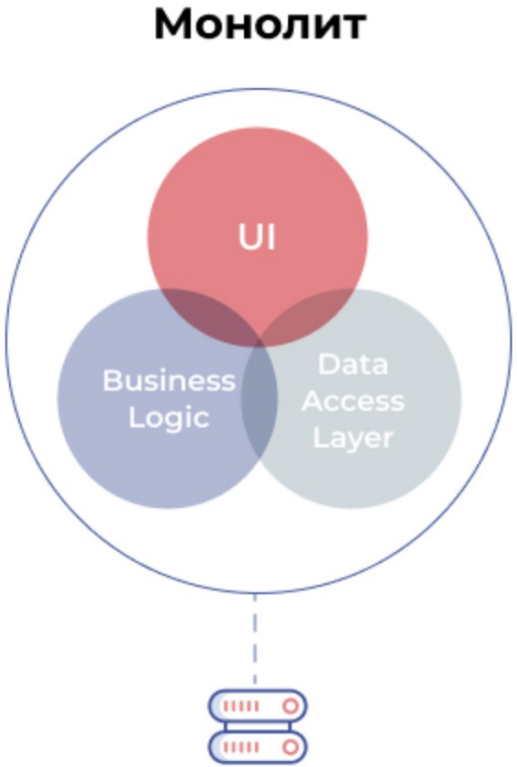
Монолитная архитектура

Достоинства:

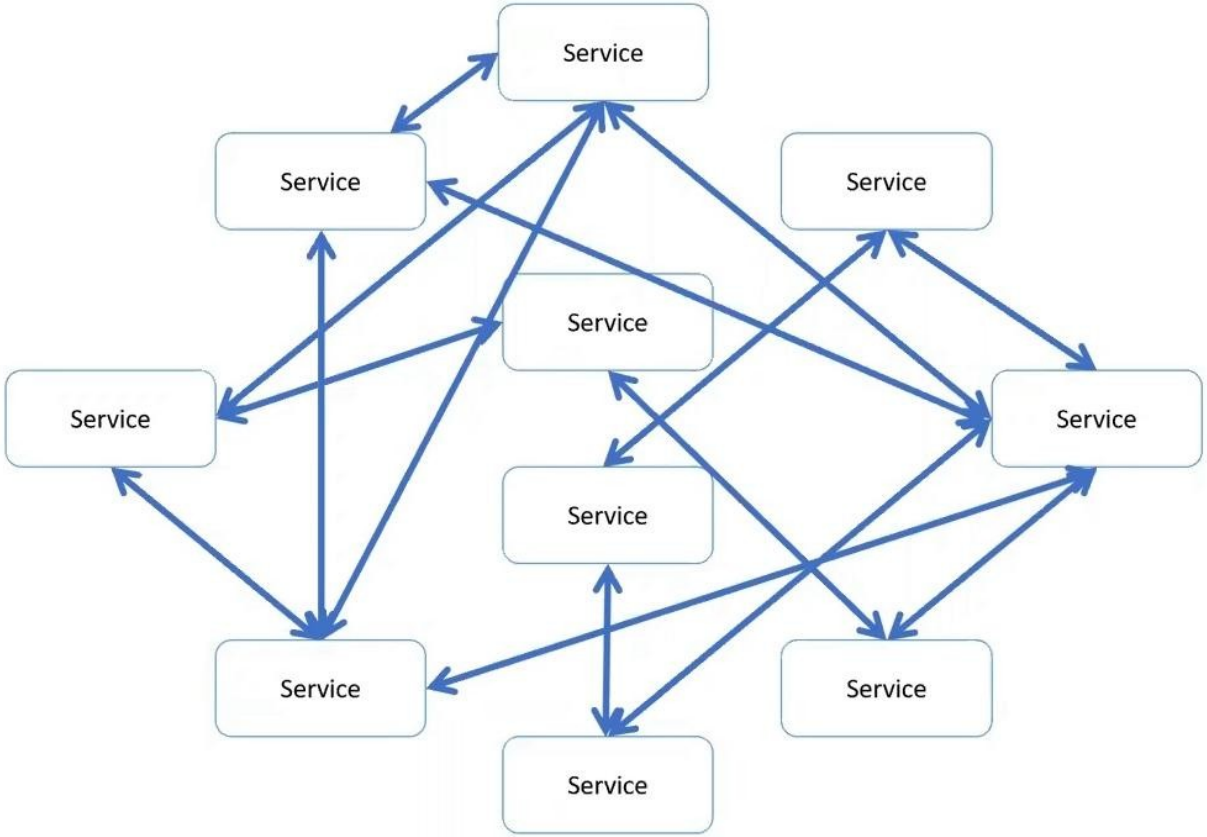
- Легко реализовать
- Легко тестировать

Недостатки:

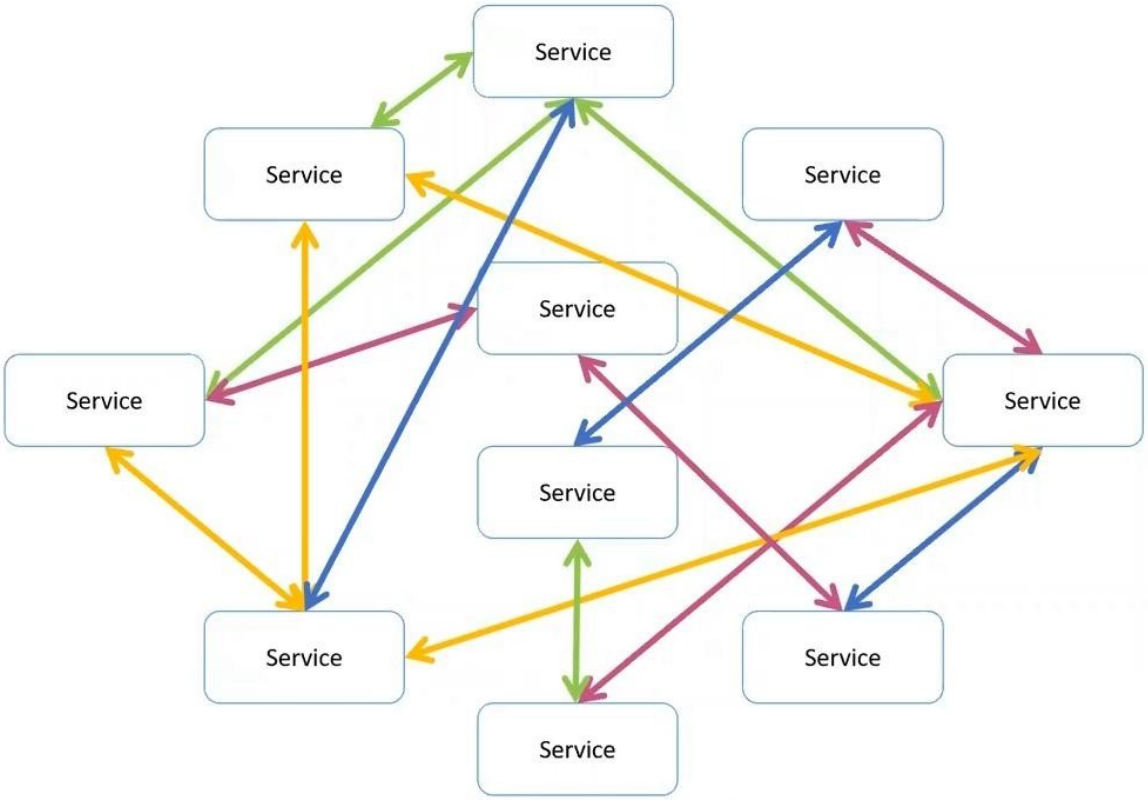
- Сложно добавлять функционал
- Нет изоляции между компонентами
- Невозможно масштабировать отдельные части



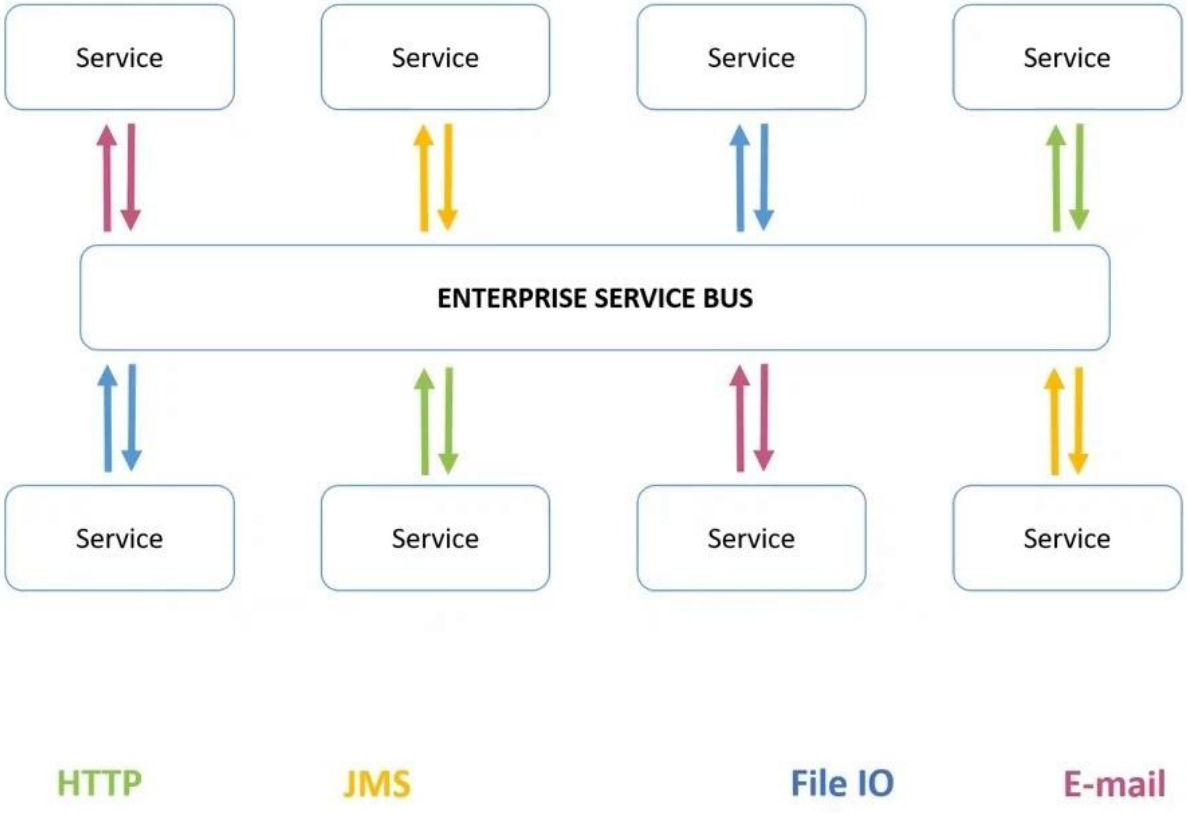
Сервисы


Service
Service
Service
Service
Service
Service
Service
Service
Service
Service

Много сервисов — много проблем



Enterprise Service Bus



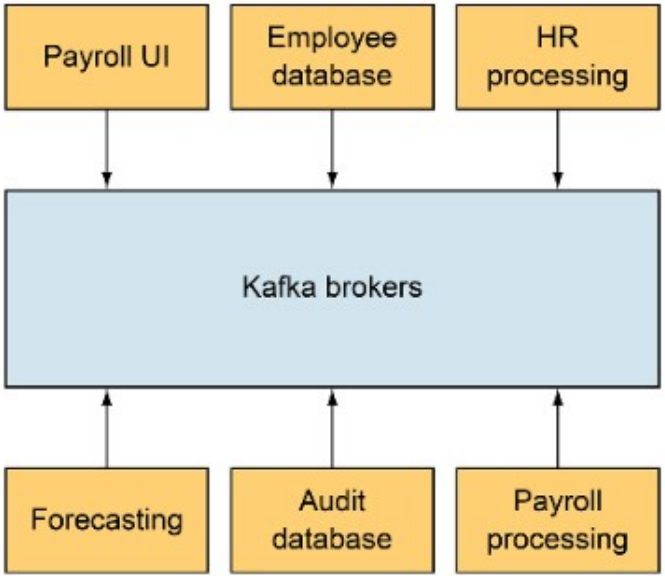
Enterprise Service Bus

Сервисная шина предприятия (enterprise service bus, ESB) — связующее программное обеспечение, обеспечивающее централизованный и унифицированный событийно-ориентированный обмен сообщениями между различными информационными системами на принципах сервис-ориентированной архитектуры.

- поддержка синхронного и асинхронного способа вызова служб;
- использование защищённого транспорта, с гарантированной доставкой сообщений, поддерживающего транзакционную модель;
- статическая и алгоритмическая маршрутизация сообщений;
- доступ к данным из сторонних информационных систем с помощью готовых или специально разработанных адаптеров;
- обработка и преобразование сообщений;
- оркестровка и хореография служб;
- разнообразные механизмы контроля и управления (аудиты, протоколирование).

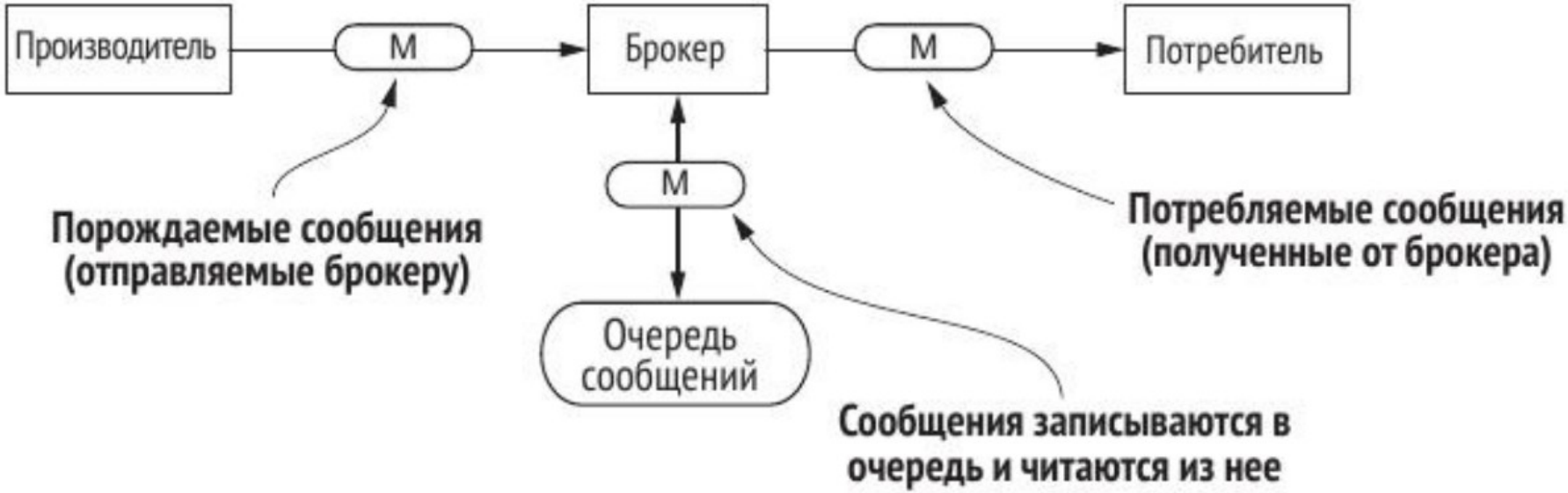
Message Brokers

Same data source and sinks. The interface between applications is now only Kafka.



Besides application decoupling of interfaces, now the applications have the chance to produce and consume data at their own pace and not be tied to a specific application uptime or availability.

Брокер сообщений



Брокеры сообщений

- Amazon Web Services (AWS) Amazon MQ
- Amazon Web Services (AWS) Kinesis
- Apache ActiveMQ
- **Apache Kafka**
- Apache Pulsar
- Google Cloud Pub/Sub (Google)
- IBM MQ
- JBoss Messaging (JBoss)
- Microsoft Azure Service Bus (Microsoft)
- Microsoft BizTalk Server (Microsoft)
- RabbitMQ
- TIBCO Enterprise Message Service
- WSO2 Message Broker

https://ru.wikipedia.org/wiki/Брокер_сообщений

Kafka

История создания

- Разработана в LinkedIn для решения задачи организации конвейеров
- Существующие системы не могли обработать требуемый объём сообщений
- Нацелена на обеспечение высокопроизводительной системы обмена сообщениями
- Основные цели:
  - Разъединить производителя и потребителя посредством модели издатель/подписчик
  - Обеспечить сохраняемость сообщений
  - Оптимизировать систему для обеспечения высокой пропускной способности
  - Обеспечить горизонтальное масштабирование

Мифы о Kafka

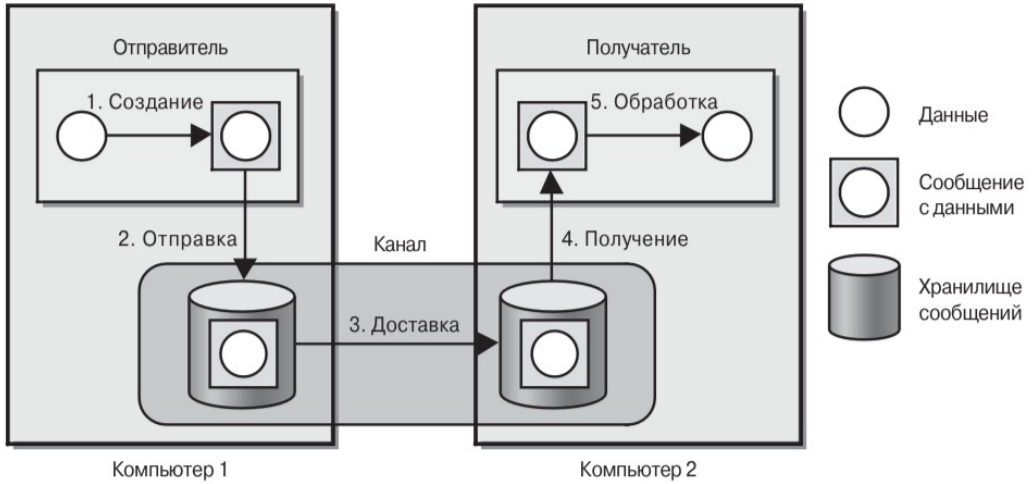
- Kafka работает только с Hadoop
- Для работы Kafka нужна HDFS
- Kafka ничем не отличается от других брокеров сообщений

Поток данных

Система потоковой обработки

Система потоковой обработки – система нежёсткого реального времени, которая делает данные доступными, когда клиентское приложение хочет их видеть.

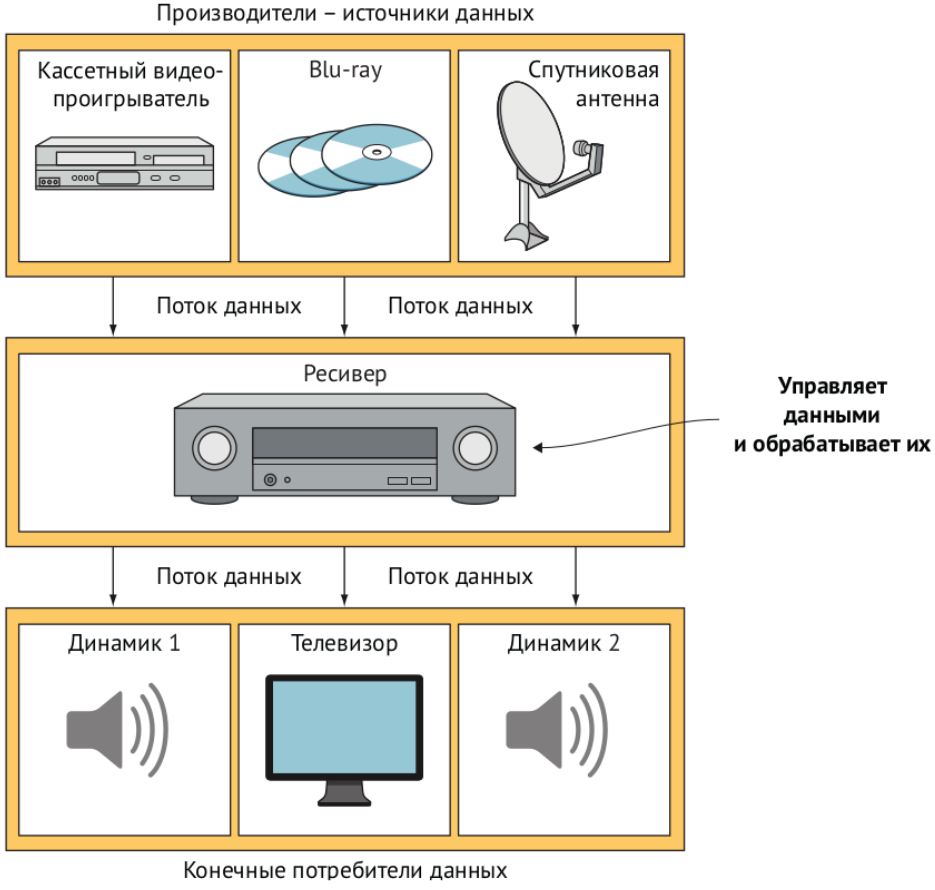
- Слабая связанность
- Асинхронное взаимодействие
- Рассогласование во времени
- Регулирование нагрузки
- Надёжное взаимодействие
- Удалённое взаимодействие



Поток данных

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

Потоки в реальной жизни



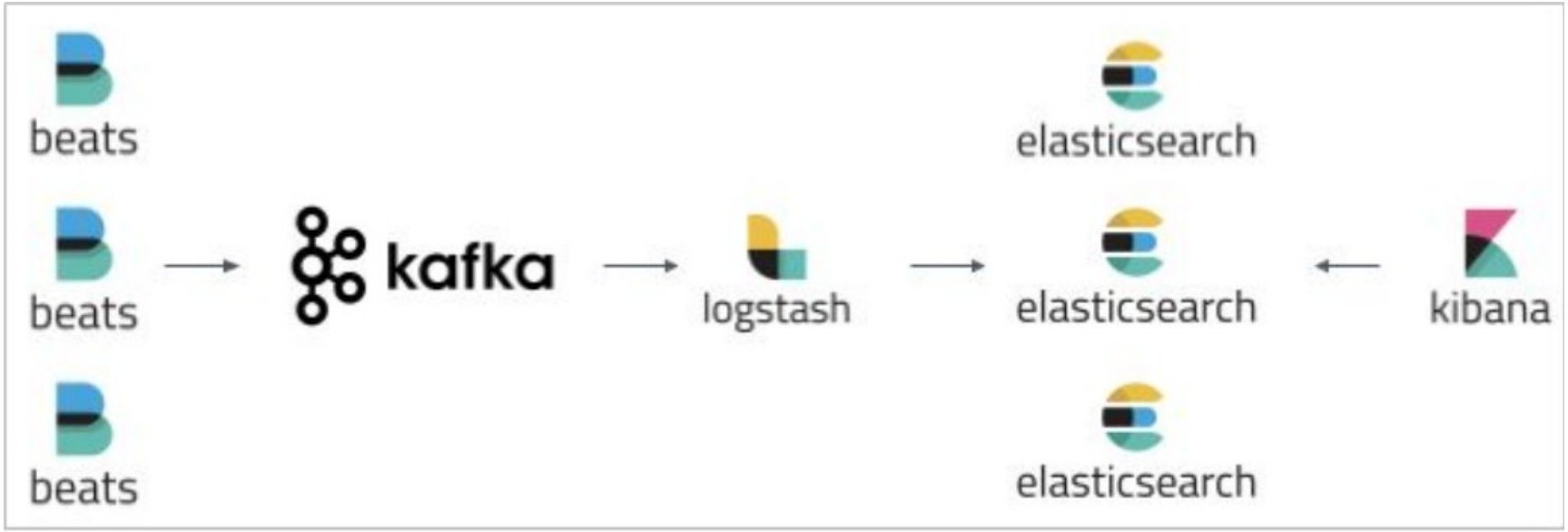
Потребности в потоковой обработки

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

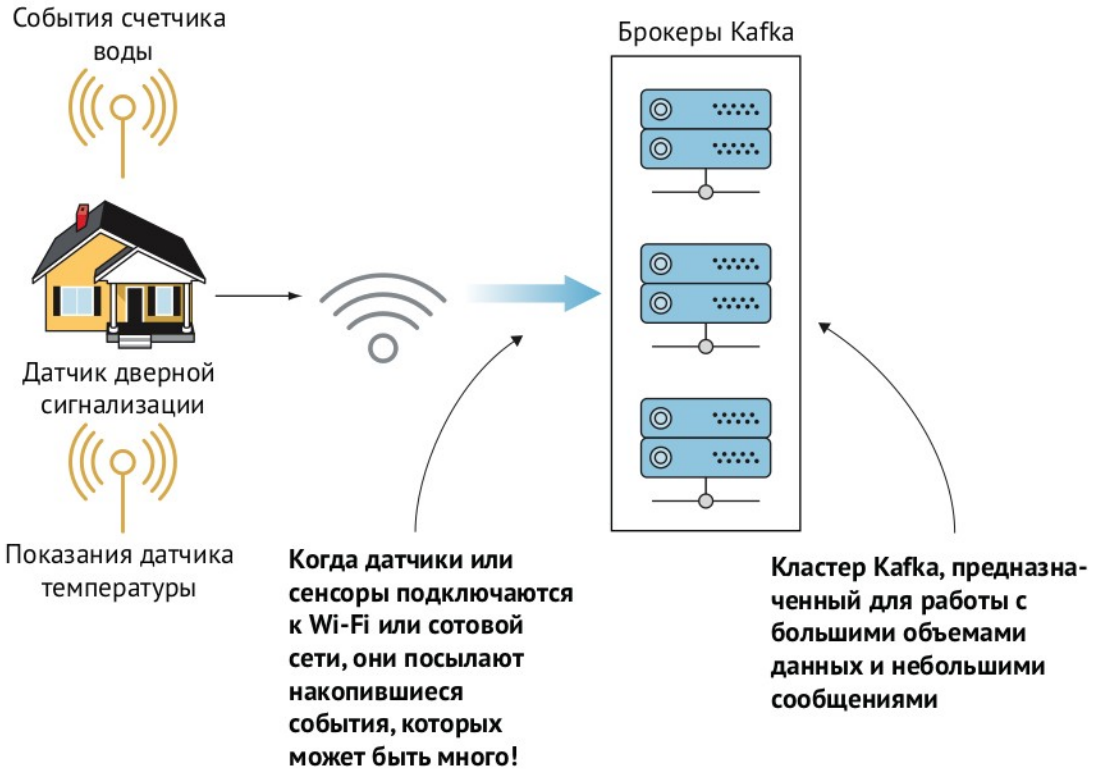
Примеры использования

Обработка журналов

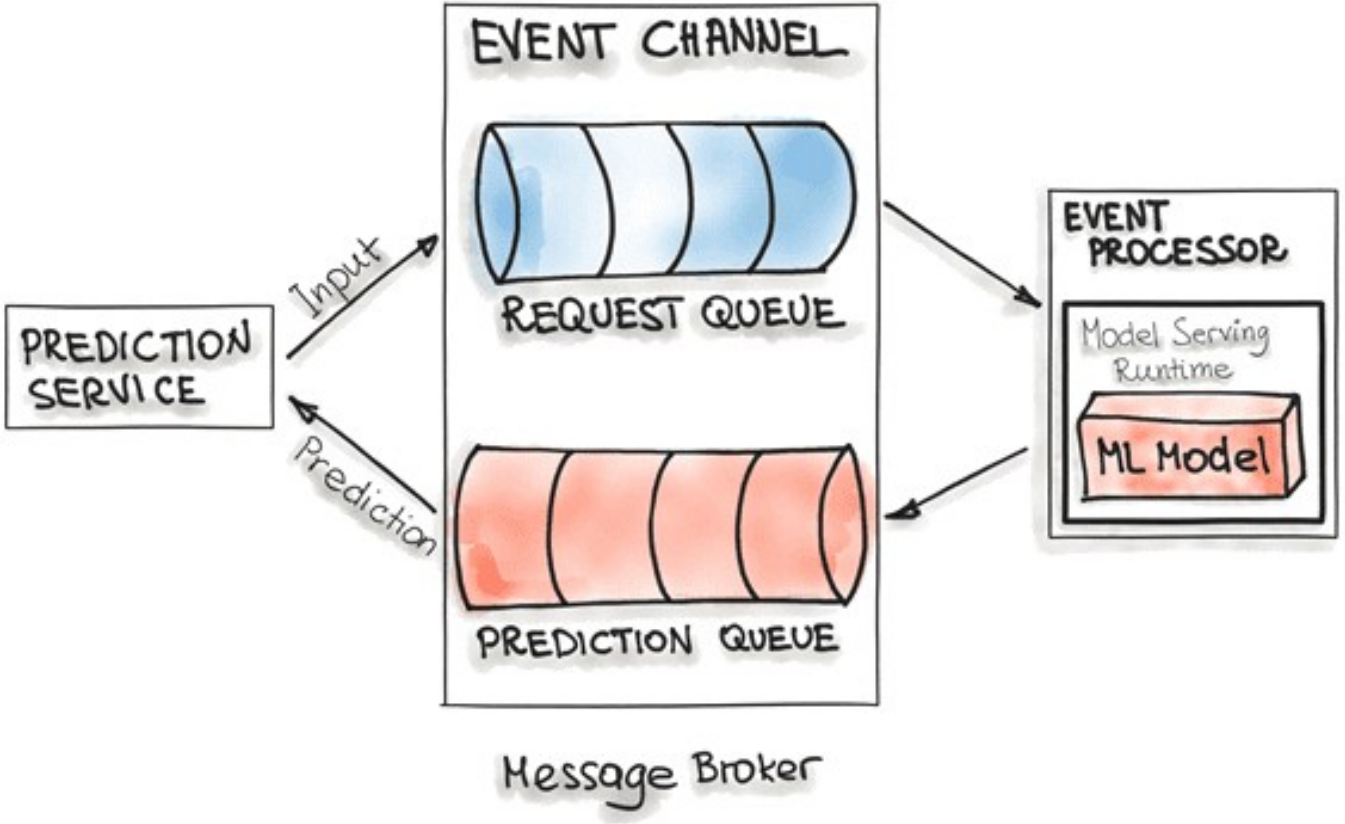
Обработка журналов



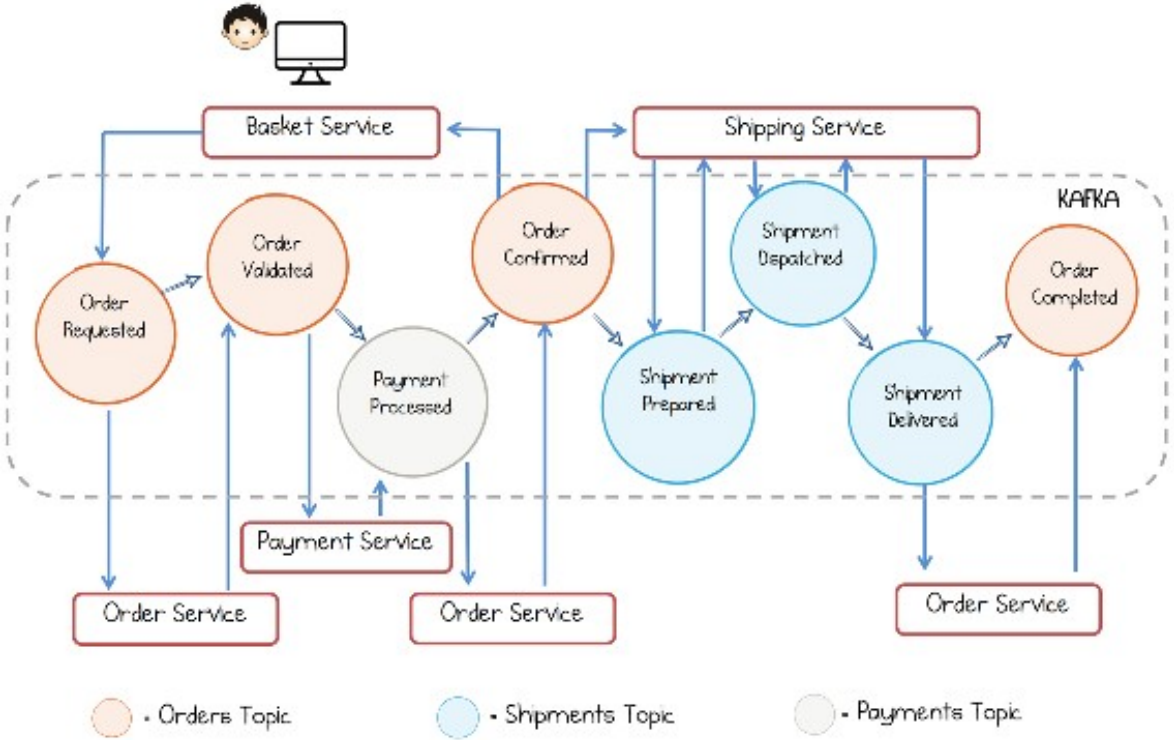
Интернет вещей



Model-on-Demand



Архитектура, ориентированная на события



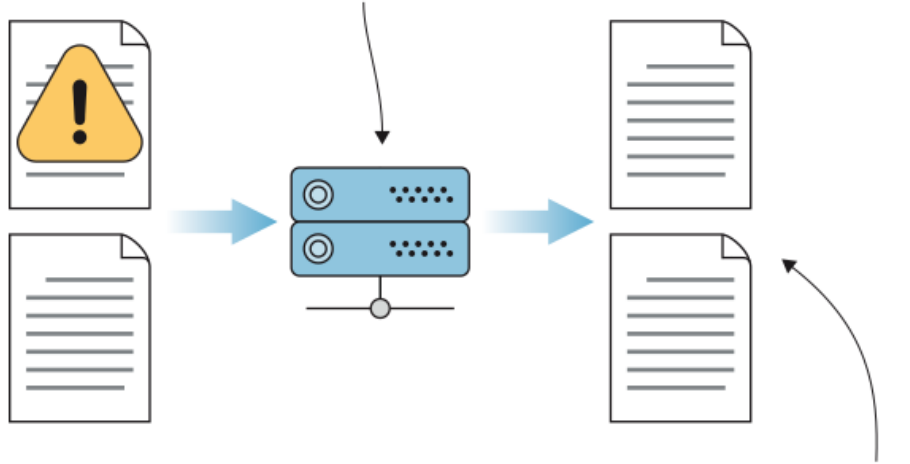
Семантика доставки

Семантика доставки

- **At-least-once** (*Не менее одного раза*) - сообщение не может потеряться, но может быть обработано несколько раз
- **At-most-once** (*Не более одного раза*) - сообщение может потеряться, но никогда не будет обработано дважды
- **Exactly-once** (*Ровно один раз*) - сообщение не может потеряться и обрабатывается ровно один раз

At-least-once

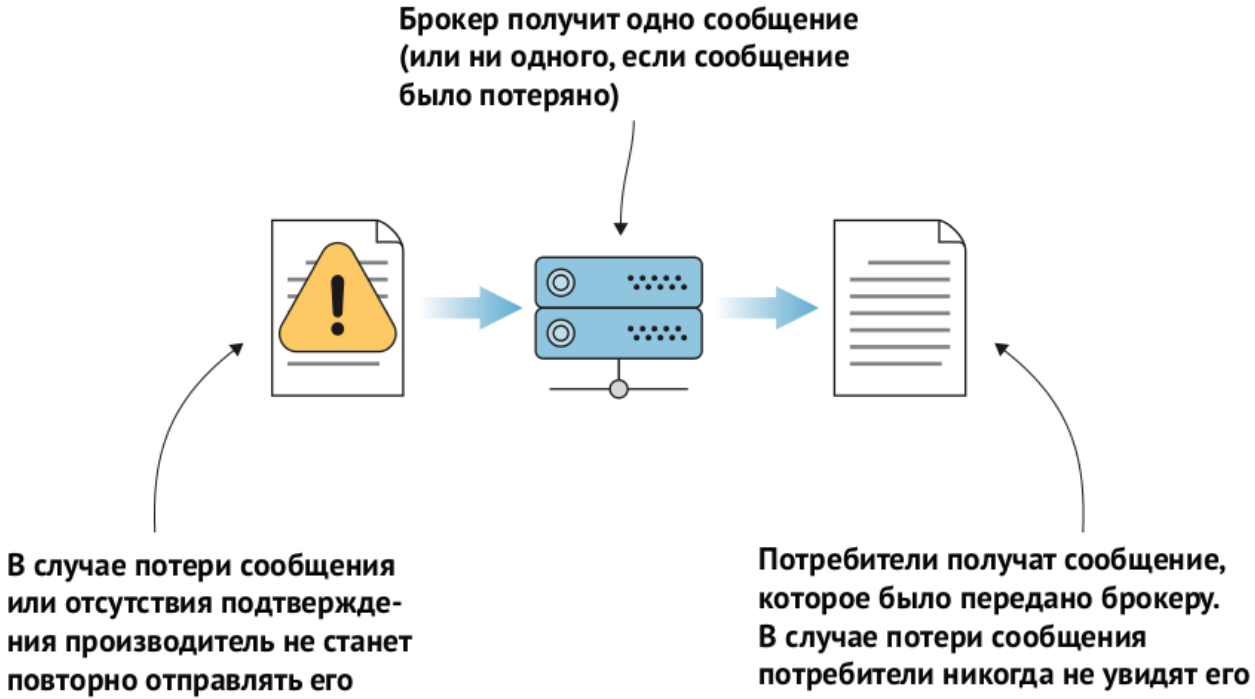
При использовании семантики «не менее одного раза» брокер получит два сообщения (или одно, если второе было потеряно)



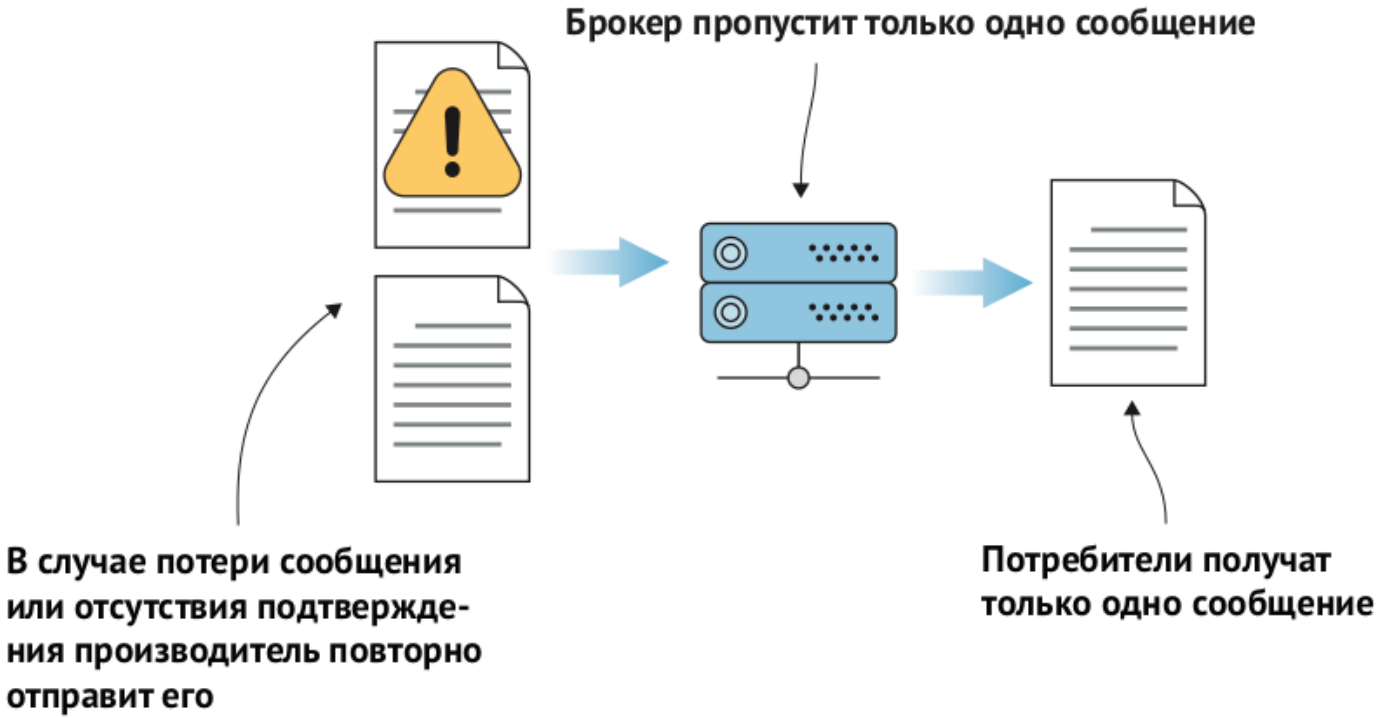
В случае потери сообщения или отсутствия подтверждения производитель повторно отправит его

Потребители получат столько сообщений, сколько получил брокер. Потребители могут получать повторяющиеся сообщения

At-most-once


Брокер получит одно сообщение (или ни одного, если сообщение было потеряно)
В случае потери сообщения или отсутствия подтверждения производитель не станет повторно отправлять его
Потребители получат сообщение, которое было передано брокеру. В случае потери сообщения потребители никогда не увидят его

Exactly-once



Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Литература

Список материалов для изучения

1. Apache Kafka. Потоковая обработка и анализ данных, 2-е издание
2. Kafka в действии
3. Kafka Streams в действии. Приложения и микросервисы для работы в реальном времени
4. Kafka Streams и ksqlDB: данные в реальном времени
5. Проектирование событийно-ориентированных систем в Apache Kafka
6. Потоковая обработка данных
7. Грокаем стриминг
8. Создание событийно-управляемых микросервисов
9. Современный язык Java. Лямбда-выражения, потоки и функциональное программирование
10. Микросервисы Spring в действии

Рефлексия

Рефлексия

С какими впечатлениями уходите с вебинара?

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

Следующий вебинар

5 мая 2023

Знакомство с Apache Kafka

- 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