OTUS

# Онлайн образование

otus.ru


**Проверить, идет ли запись**

# Меня хорошо видно && слышно?

Тема вебинара

# Schema Registry

**Чащина Александра**

Big Data Engineer

# Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Телеграм

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

## Условные обозначения

Индивидуально

Время, необходимое
на активность

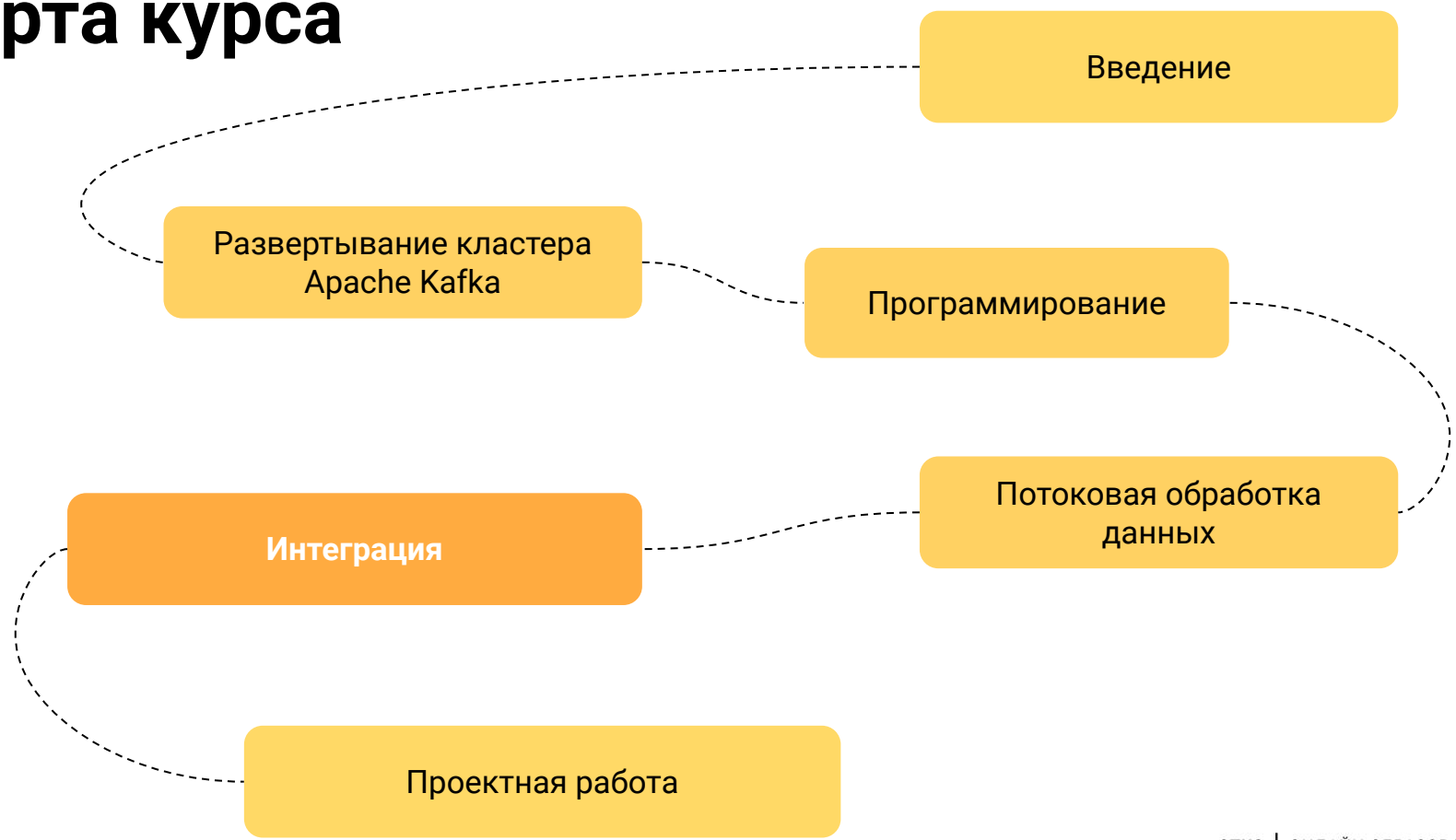
Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

# Карта курса


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

# Маршрут вебинара

- Введение
- Форматы сообщений
- Schema Registry
- Практика
- Рефлексия

# Цели вебинара

## После занятия вы сможете

1. Объяснить, что такое схемы в Kafka и зачем они нужны
2. Использовать Kafka Schema Registry
3. Писать и читать сообщения с помощью схем Kafka

# Смысл

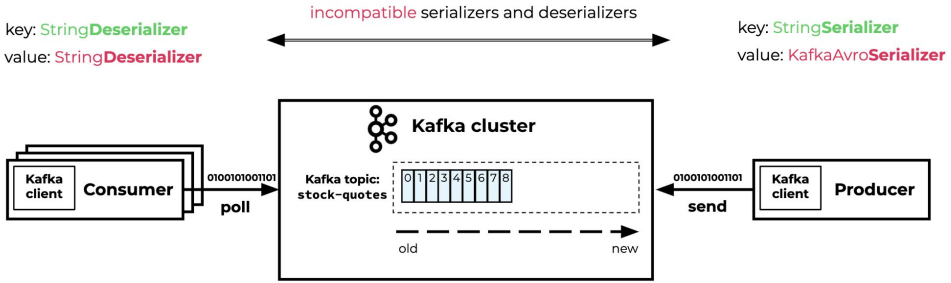
## Зачем вам это уметь

1. Schema Registry - важная составляющая кластера Kafka
2. Использовать схемы является хорошей практикой

# Введение

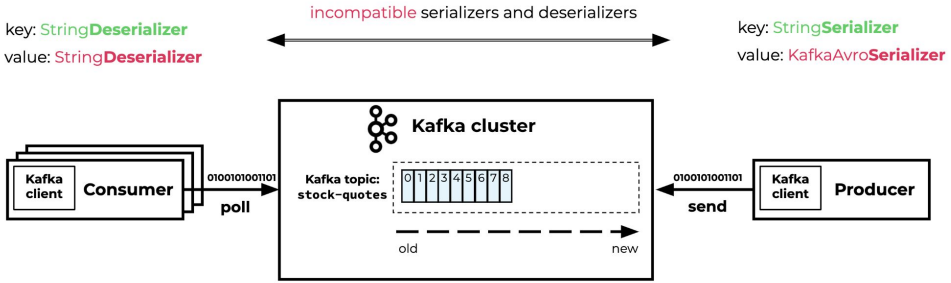
# Структура сообщений

- Сообщения Kafka передаются и хранятся в брокере в бинарном виде



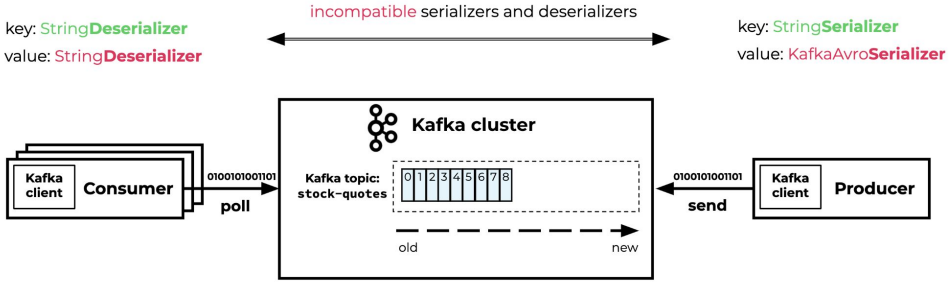
# Структура сообщений

- Сообщения Kafka передаются и хранятся в брокере в бинарном виде
- Самый простой вариант - передавать сообщения в текстовом формате `message.toString()`



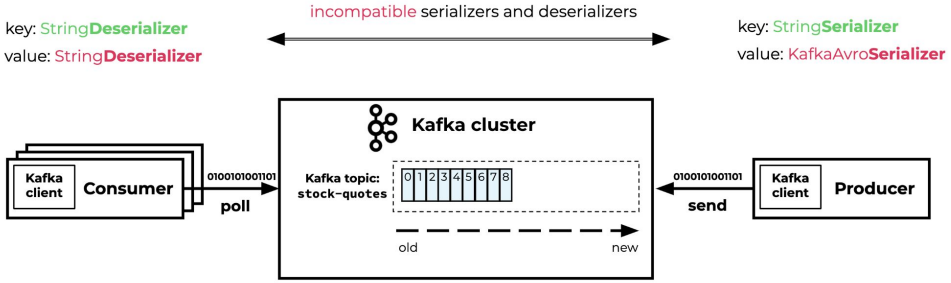
# Структура сообщений

- Сообщения Kafka передаются и хранятся в брокере в бинарном виде
- Самый простой вариант - передавать сообщения в текстовом формате `message.toString()`
    - Данные хранятся неэффективно



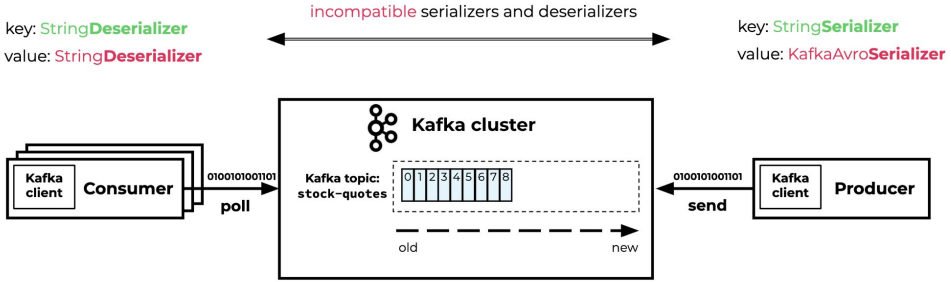
# Структура сообщений

- Сообщения Kafka передаются и хранятся в брокере в бинарном виде
- Самый простой вариант - передавать сообщения в текстовом формате `message.toString()`
  - Данные хранятся неэффективно
  - Нет типизации



# Структура сообщений

- Сообщения Kafka передаются и хранятся в брокере в бинарном виде
- Самый простой вариант - передавать сообщения в текстовом формате `message.toString()`
  - Данные хранятся неэффективно
  - Нет типизации
  - Невозможно заранее определить структуру сообщений и ее изменения



# Схема данных

- Описание структуры сообщения (поля и их типы)

```
{
  "type": "record",
  "name": "Student",
  "namespace": "ru.otus.kafka",
  "fields": [
    {
      "name": "Id",
      "type": "int"
    },
    {
      "name": "Name",
      "type": "string"
    },
    {
      "name": "Surname",
      "type": "string"
    }
  ]
}
```

# Схема данных

- Описание структуры сообщения (поля и их типы)
- Контракт между Producer и Consumer

```
{
  "type": "record",
  "name": "Student",
  "namespace": "ru.otus.kafka",
  "fields": [
    {
      "name": "Id",
      "type": "int"
    },
    {
      "name": "Name",
      "type": "string"
    },
    {
      "name": "Surname",
      "type": "string"
    }
  ]
}
```

# Схема данных

- Описание структуры сообщения (поля и их типы)
- Контракт между Producer и Consumer
- Эффективная сериализация / десериализация

```
{
  "type": "record",
  "name": "Student",
  "namespace": "ru.otus.kafka",
  "fields": [
    {
      "name": "Id",
      "type": "int"
    },
    {
      "name": "Name",
      "type": "string"
    },
    {
      "name": "Surname",
      "type": "string"
    }
  ]
}
```

# Схема данных

- Описание структуры сообщения (поля и их типы)
- Контракт между Producer и Consumer
- Эффективная сериализация / десериализация
- Версионирование

```
{
  "type": "record",
  "name": "Student",
  "namespace": "ru.otus.kafka",
  "fields": [
    {
      "name": "Id",
      "type": "int"
    },
    {
      "name": "Name",
      "type": "string"
    },
    {
      "name": "Surname",
      "type": "string"
    }
  ]
}
```

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

# Форматы сообщений

# Avro

- Проект с открытым кодом от Apache


```
{
  "type": "record",
  "name": "Student",
  "namespace": "ru.otus.kafka",
  "fields": [
    {
      "name": "Id",
      "type": "int"
    },
    {
      "name": "Name",
      "type": "string"
    },
    {
      "name": "Surname",
      "type": "string"
    }
  ]
}
```

# Avro

- Проект с открытым кодом от Apache
- Компактный, быстрый бинарный формат данных


```
{
  "type": "record",
  "name": "Student",
  "namespace": "ru.otus.kafka",
  "fields": [
    {
      "name": "Id",
      "type": "int"
    },
    {
      "name": "Name",
      "type": "string"
    },
    {
      "name": "Surname",
      "type": "string"
    }
  ]
}
```

# Avro

- Проект с открытым кодом от Apache
- Компактный, быстрый бинарный формат данных
- Поддерживается на многих языках


```
{
  "type": "record",
  "name": "Student",
  "namespace": "ru.otus.kafka",
  "fields": [
    {
      "name": "Id",
      "type": "int"
    },
    {
      "name": "Name",
      "type": "string"
    },
    {
      "name": "Surname",
      "type": "string"
    }
  ]
}
```

# Avro

- Проект с открытым кодом от Apache
- Компактный, быстрый бинарный формат данных
- Поддерживается на многих языках
- Схема данных описывается в формате JSON


```
{
  "type": "record",
  "name": "Student",
  "namespace": "ru.otus.kafka",
  "fields": [
    {
      "name": "Id",
      "type": "int"
    },
    {
      "name": "Name",
      "type": "string"
    },
    {
      "name": "Surname",
      "type": "string"
    }
  ]
}
```

# Avro

- Проект с открытым кодом от Apache
- Компактный, быстрый бинарный формат данных
- Поддерживается на многих языках
- Схема данных описывается в формате JSON
- Можно генерировать объекты в ЯП


```
{
  "type": "record",
  "name": "Student",
  "namespace": "ru.otus.kafka",
  "fields": [
    {
      "name": "Id",
      "type": "int"
    },
    {
      "name": "Name",
      "type": "string"
    },
    {
      "name": "Surname",
      "type": "string"
    }
  ]
}
```

# Avro: создание сообщений

Есть несколько вариантов создания и обработки Avro-сообщений:

| Метод | Создание схемы JSON | Создание объекта (класс Java) |
|---|---|---|
| Generic | Вручную | Вручную |
| Reflection | Автоматически | Вручную |
| Specific | Вручную | Автоматически |

# Типы данных в Avro

- `boolean`, `int`, `long`, `float`, `double`, `string`, `bytes`, `null` - простые типы

# Типы данных в Avro

- `boolean`, `int`, `long`, `float`, `double`, `string`, `bytes`, `null` - простые типы
- `record` - поле, состоящие из нескольких вложенных полей, корень сообщения

# Типы данных в Avro

- `boolean`, `int`, `long`, `float`, `double`, `string`, `bytes`, `null` - простые типы
- `record` - поле, состоящие из нескольких вложенных полей, корень сообщения
- `union` - поле имеет несколько типов

# Типы данных в Avro

- `boolean`, `int`, `long`, `float`, `double`, `string`, `bytes`, `null` - простые типы
- `record` - поле, состоящие из нескольких вложенных полей, корень сообщения
- `union` - поле имеет несколько типов
- `array` - список с определенным типом элементов

# Типы данных в Avro

- `boolean`, `int`, `long`, `float`, `double`, `string`, `bytes`, `null` - простые типы
- `record` - поле, состоящие из нескольких вложенных полей, корень сообщения
- `union` - поле имеет несколько типов
- `array` - список с определенным типом элементов
- `enum` - перечисление

# Типы данных в Avro

- `boolean`, `int`, `long`, `float`, `double`, `string`, `bytes`, `null` - простые типы
- `record` - поле, состоящие из нескольких вложенных полей, корень сообщения
- `union` - поле имеет несколько типов
- `array` - список с определенным типом элементов
- `enum` - перечисление
- Можно задавать значение по умолчанию (`default`) и документацию (`doc`)

# Другие форматы

- Protobuf
- JSON

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

# Schema Registry

# Что такое регистр схем?

- Централизованный менеджер схем сообщений в кластере Kafka

# Что такое регистр схем?

- Централизованный менеджер схем сообщений в кластере Kafka
- Обеспечивает версионирование схем

# Что такое регистр схем?

- Централизованный менеджер схем сообщений в кластере Kafka
- Обеспечивает версионирование схем
- Обеспечивает совместимость между версиями

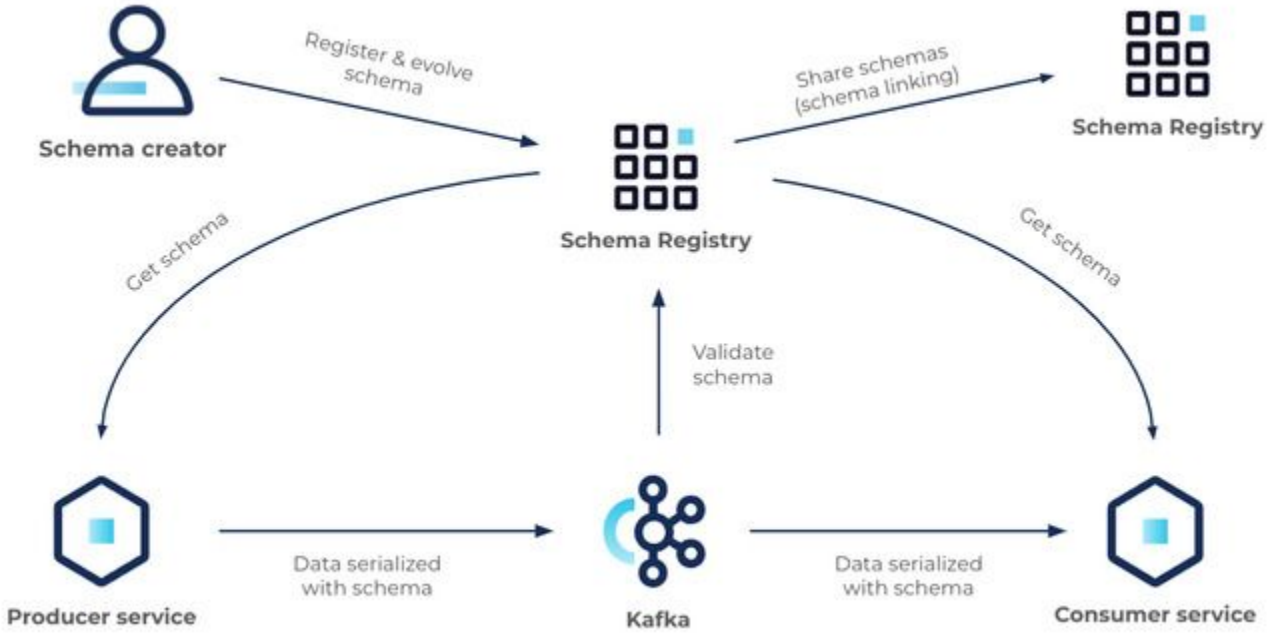
# Что такое регистр схем?

- Централизованный менеджер схем сообщений в кластере Kafka
- Обеспечивает версионирование схем
- Обеспечивает совместимость между версиями
- Доступен через REST API и Java API

# Что такое регистр схем?

- Централизованный менеджер схем сообщений в кластере Kafka
- Обеспечивает версионирование схем
- Обеспечивает совместимость между версиями
- Доступен через REST API и Java API
- Каждая схема имеет уникальный ID

# Schema Registry



# Стратегия совместимости схем

| Стратегия | Описание | Кто обновляет схему первым |
|---|---|---|
| BACKWARD | Consumer, используя новую версию, может прочитать сообщения с предыдущей схемой | Consumer |
| FORWARD | Consumer, используя предыдущую версию, может прочитать сообщения с новой схемой | Producer |
| FULL | BACKWARD и FORWARD одновременно | Любой порядок |
| NONE | Нет проверки совместимости | Любой порядок |

# Subject

- **Subject** - рамка, в которой определяются версии и совместимость схемы в Schema Registry

# Subject

- **Subject** - рамка, в которой определяются версии и совместимость схемы в Schema Registry
- Есть разные стратегии присваивания названия **subject** (subject naming strategy)

# Subject

- **Subject** - рамка, в которой определяются версии и совместимость схемы в Schema Registry
- Есть разные стратегии присваивания названия **subject** (subject naming strategy)
- Настраивается `key.subject.name.strategy` и `value.subject.name.strategy`

# Стратегии subject naming

# Стратегии subject naming

1. **TopicNameStrategy** (default): `<subject-name> = <topic>-key | <topic>-value`. Один топик = одна схема.

# Стратегии subject naming

1. **TopicNameStrategy** (default): `<subject-name>  = <topic>-key | <topic>-value`. Один топик = одна схема.
2. **TopicRecordNameStrategy**: `<subject-name> = <topic>-<type>-key | <topic>-<type>-value`. Type - это полное имя класса (fully qualified name). Один топик = несколько схем.

# Стратегии subject naming

1. **TopicNameStrategy** (default): `<subject-name>  = <topic>-key | <topic>-value`. Один топик = одна схема.
2. **TopicRecordNameStrategy**: `<subject-name> = <topic>-<type>-key | <topic>-<type>-value`. Type - это полное имя класса (fully qualified name). Один топик = несколько схем.
3. **RecordNameStrategy**: `<subject-name> = <type>-key | <type>-value.` Схемы определяются вне зависимости от топиков.

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

# Практика

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

# Тезисы

## Подведем итоги

1. Самые распространенные форматы данных в Kafka - JSON, Avro, Protobuf
2. Схема данных позволяет контролировать структуру сообщений между Producer и Consumer
3. Schema Registry - эффективный менеджер схем

# Рефлексия

# Цели вебинара

## После занятия вы сможете

1. Объяснить, что такое схемы в Kafka и зачем они нужны
2. Использовать Kafka Schema Registry
3. Писать и читать сообщения с помощью схем Kafka

# Рефлексия

С какими основными мыслями
и инсайтами уходите с вебинара?

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

# Следующий вебинар

## Confluent Rest API

- 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

**Чащина Александра**

Big Data Engineer