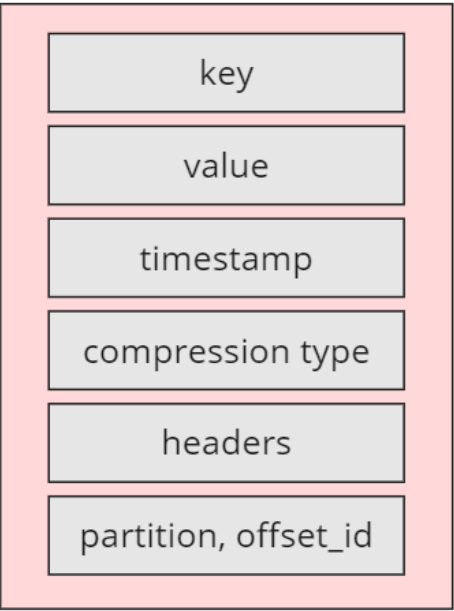


Поговорим про сообщения в Kafka.

Сообщение в Kafka - это некая единица информации, которая имеет отметку времени и не может быть изменена. Именно поэтому модель взаимодействия называется журнал. Если возвращаться к теме применения Kafka, то ее можно использовать как хранилище. Но использовать как типичную базу данных, в которой можно изменять записи - нельзя.

Из чего состоит сообщение Kafka? Если отображать схематично, то уже записанное сообщение состоит из следующих компонентов:



1. Key - это как раз partition key, куда вы передаете ключ для определения, в какую партицию положить сообщение. Может быть пустым.
2. Value - содержание сообщения, записываем сюда свою информацию. Может быть пустым.
3. Timestamp - отметка о времени, записывается Kafka самостоятельно. В настройках можно выбрать, указывать время создания сообщения (продюсером) или время добавления события в партицию.
4. Compression type - можно указать, с помощью какого алгоритма сжатия сокращать размер сообщения. Например, gzip. Опциональное поле.

5. Headers - заголовки, по аналогии как у HTTP. Опциональное поле. Используются для реализации дополнительной логики.

6. Системные поля `partition` и `offset_id` - информация о партиции, где находится сообщение и об оффсете (позиции в партиции). Значения этих полей Kafka записывает самостоятельно. Напомним, оффсет начинается отсчет с нуля.

Пример сообщения при отправке в брокер:



Для передачи сообщений доступны следующие форматы данных:

- JSON
 - Легко читается, много где используется. Но размер сообщения может быть в 2 раза больше чем Avro и в 3-10 раз больше чем Protobuf.
- Avro
 - Тяжело читается, но мало занимает места и быстро обрабатывается.
- Protobuf

- Тяжело читается, но занимает еще меньше места чем Avro и еще быстрее обрабатывается.

Формат данных обычно определяет разработчик, но вы можете повлиять на это решение. Например, если вам не требуется повышенная производительность, можно убедить оставить формат передачи JSON.

Schema Registry и изменение структуры сообщений (значение ключа value).

Давайте представим, что мы согласовали структуру сообщения для нашего сервиса заданий. Допустим, структура сообщения такая (JSON):

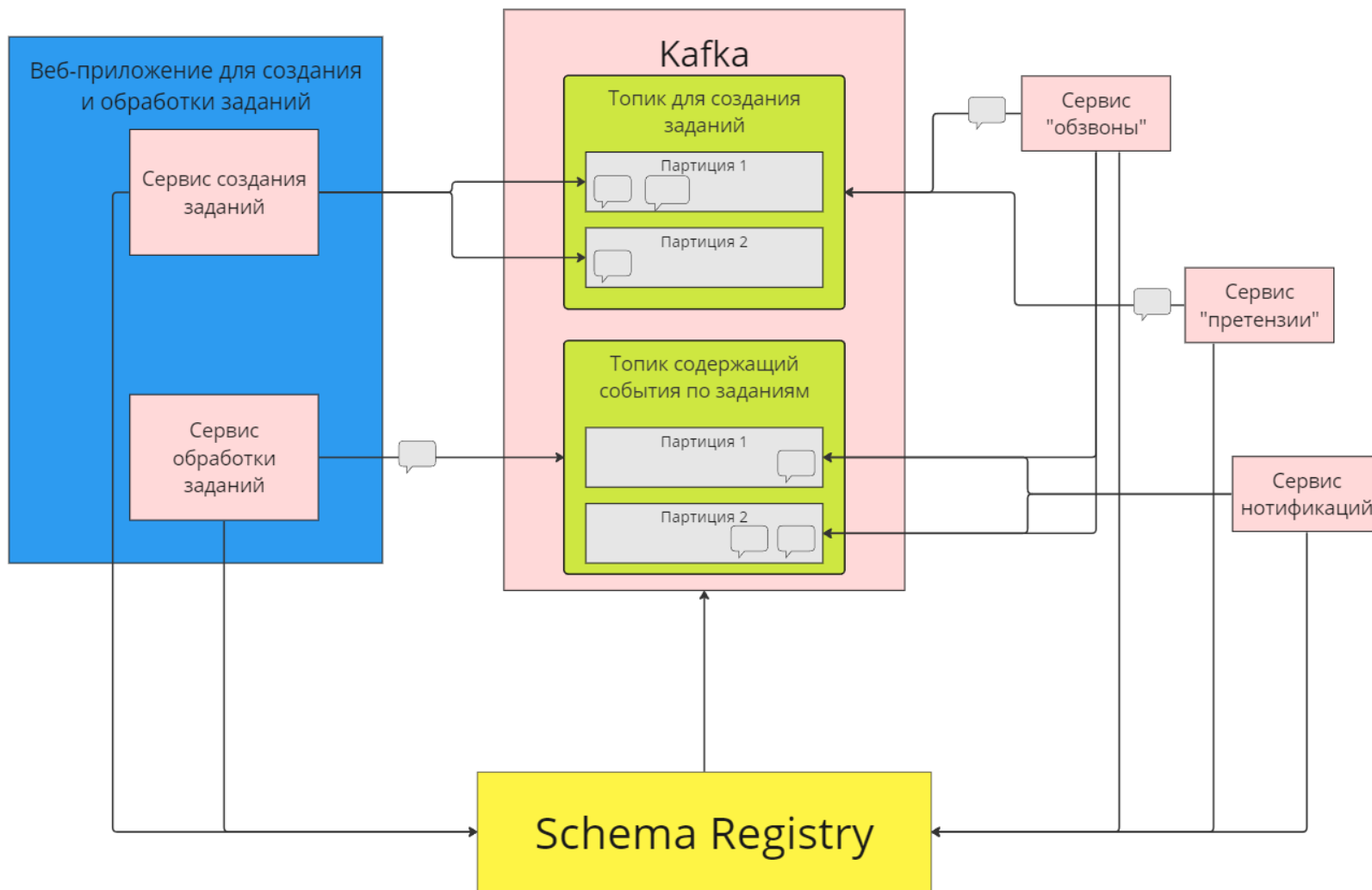
```
{"desk":"hello", "type":"TASK2", "id":32, "status":"В работе"}
```

Но появилась новая версия сервиса заданий, которая требует добавления нового поля. Допустим, у нас появился новый ключ "createdDate". Новая структура выглядит так:

```
{"desk":"hello", "type":"TASK2", "id":32, "status":"В работе", "createdDate":"2023-07-30"}
```

Если добавить поле в одностороннем порядке, то консьюмеры, которые читают ваши сообщения, должны будут доработаться на своей стороне. Так как они просто не поймут, что за новый ключ они получают в сообщении. А что если этих консьюмеров множество? А если надо добавлять новые поля каждый месяц?

Для решения вопроса структуры сообщений был разработан сервис Schema Registry. Данный сервис хранит в себе схемы и позволяет консьюмеру и продюсеру получать данные для "понимания" структуры сообщения. Схематично это работает так:



Это все тот же пример нашего сервиса заданий. Каждый продюсер, консьюмер и Kafka подключены к сервису хранения схем. И при передаче сообщения брокеру продюсер добавляет поле `schema_id`, опираясь на которое, консьюмер сможет прочитать сообщение по актуальной схеме. Сервис хранения схем хранит в себе схемы, которые описывают структуру ваших данных. В итоге вам достаточно создать новую схему и передать корректное сообщение брокеру, указав `id` новой схемы.

Подытожим, какие распространенные проблемы решает Schema Registry:

- Несогласованность данных: реестр гарантирует, что все данные соответствуют согласованным схемам. Это снижает риск несогласованности данных и повышает качество данных.

- Несовместимые форматы данных: с несколькими производителями и потребителями данных разные приложения могут использовать разные форматы данных. Реестр схем решает эту проблему, обеспечивая централизованное управление схемой и ее проверку, чтобы гарантировать совместимость всех данных сообщений.
- Эволюция схемы: схемы часто меняются со временем, что может вызвать проблемы совместимости между разными версиями схемы. Реестр схем поддерживает управление версиями схемы, гарантируя, что разные версии схемы могут использоваться одновременно, не вызывая проблем совместимости.
- Проверка схемы: реестр схем проверяет, что данные, созданные для брокера, используют действительный идентификатор схемы в реестре схем. Это гарантирует соответствие данных стандартному формату, снижая риск потери или повреждения данных.
- Управление данными: реестр схем — центральное место для управления и версий схем данных. Это упрощает управление, позволяя легко отслеживать изменения схемы, сохранять историю эволюции схемы и обеспечивать соответствие данных нормативным требованиям.