**Kafka**

**Apache Kafka** — распределённый программный [брокер сообщений](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%BE%D0%BA%D0%B5%D1%80_%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B9) с потоковой обработкой, проект с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)

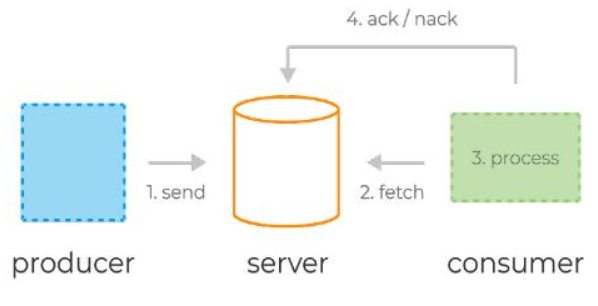
Спроектирован как [распределённая](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F), [горизонтально масштабируемая](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D1%80%D0%B8%D0%B7%D0%BE%D0%BD%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D0%BE%D1%81%D1%82%D1%8C) система, обеспечивающая наращивание пропускной способности как при росте числа и нагрузки со стороны источников, так и количества систем-подписчиков. Подписчики могут быть объединены в группы. Поддерживается возможность временного хранения данных для последующей [пакетной обработки](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0). Одной из особенностей реализации инструмента является применение техники, сходной с [журналами транзакций](https://ru.wikipedia.org/wiki/%D0%96%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%82%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D0%B9), используемыми в [системах управления базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)

Как и сервисы обработки очередей, Kafka условно состоит из трёх компонентов:

1) сервер (по-другому ещё называется брокер),  
2) продюсеры — они отправляют сообщения брокеру,  
3) консьюмеры — считывают эти сообщения, используя модель pull.

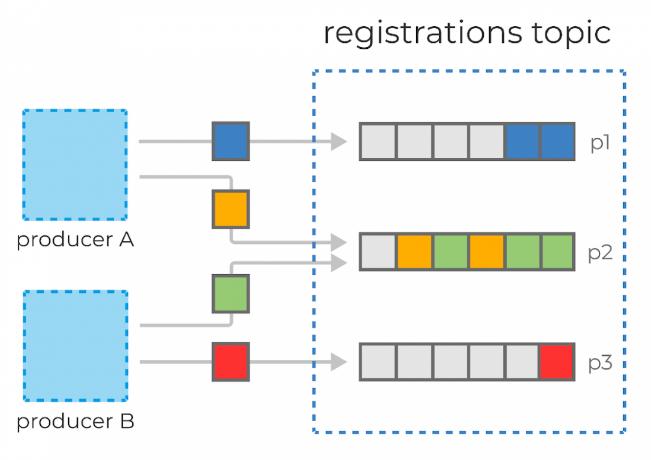
\* Сообщения в Kafka **не удаляются** брокерами по мере их обработки консьюмерами — данные в Kafka могут храниться днями, неделями, годами.

\* Благодаря этому одно и то же сообщение может быть обработано **сколько угодно раз** разными консьюмерами и в разных контекстах.

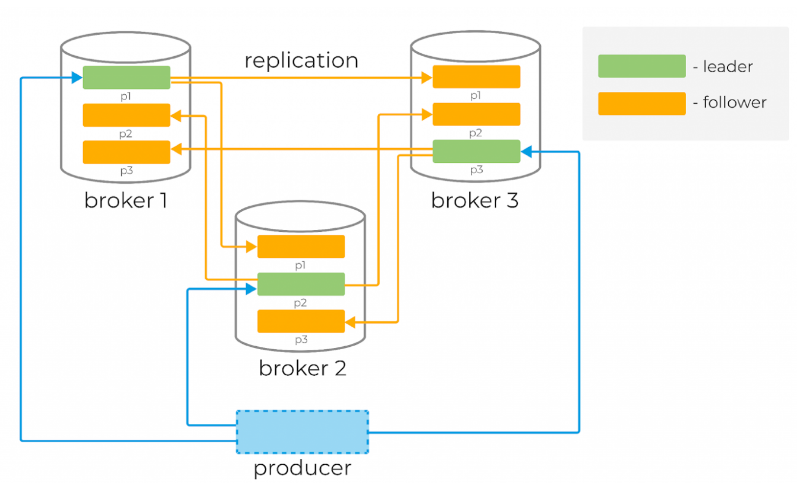


Каждое сообщение (event или message) в Kafka состоит из ключа, значения, таймстампа и опционального набора метаданных. Сообщения в Kafka организованы и хранятся в именованных топиках (Topics), каждый топик состоит из одной и более партиций (Partition), распределённых между брокерами внутри одного кластера. Подобная распределённость важна для горизонтального масштабирования кластера, так как она позволяет клиентам писать и читать сообщения с нескольких брокеров одновременно.

Когда новое сообщение добавляется в топик, оно записывается в одну из партиций этого топика. Сообщения с одинаковыми ключами всегда записываются в одну и ту же партицию, гарантируя очередность или порядок записи и чтения.

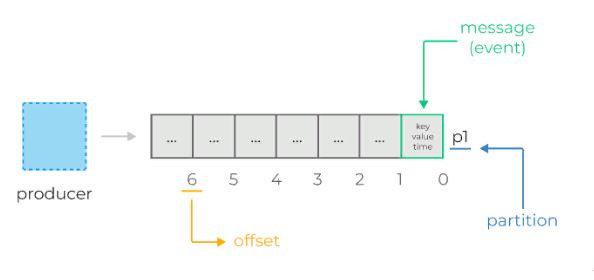


Для гарантии сохранности данных каждая партиция в Kafka может быть реплицирована n раз, где n — replication factor. Таким образом гарантируется наличие нескольких копий сообщения, хранящихся на разных брокерах.

У каждой партиции есть (Leader) — брокер, который работает с клиентами. Лидер работает с продюсерами и в общем случае отдаёт сообщения консьюмерам. К лидеру осуществляют запросы (Follower) — брокеры, которые хранят реплику всех данных партиций. Сообщения всегда отправляются лидеру и, в общем случае, читаются с лидера.  


Чтобы понять, кто является лидером партиции, перед записью и чтением клиенты делают запрос метаданных от брокера. Причём они могут подключаться к любому брокеру в кластере.

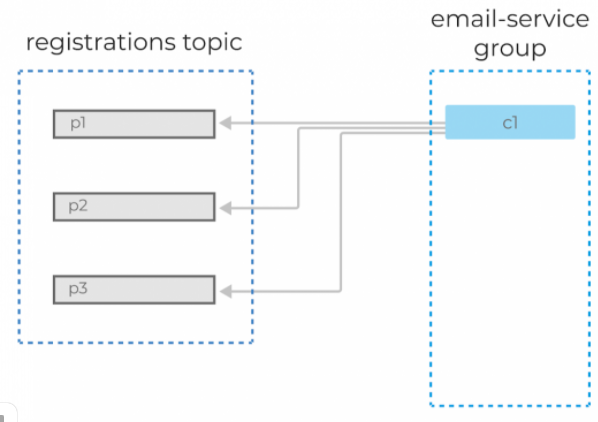
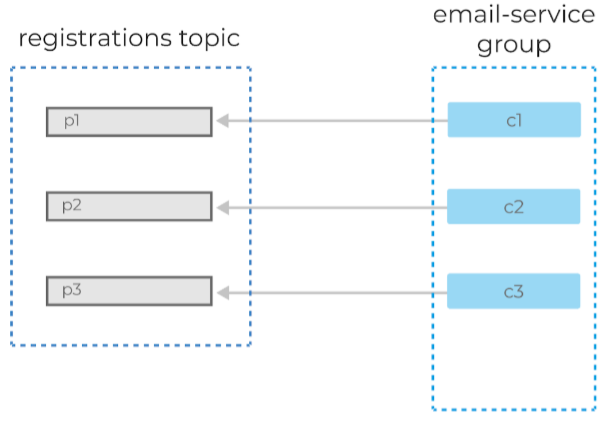
Основная структура данных в Kafka — это распределённый, реплицируемый лог. Каждая партиция — это и есть тот самый реплицируемый лог, который хранится на диске. Новое сообщение, отправленное продюсером в партицию, сохраняется в «голову» этого лога и получает уникальный, монотонно возрастающий offset (64-битное число, которое назначается самим брокером).



Время гарантированного хранения данных можно контролировать. Длительность хранения сообщений при этом не влияет на общую производительность системы. Поэтому нормально хранить сообщения в Kafka месяцами или даже годами.

**Consumer Groups**

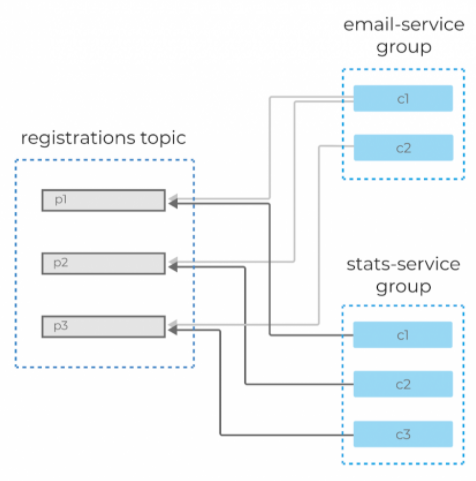
У сервиса отправки писем есть своя собственная консьюмер-группа с одним консьюмером *c1* внутри. Значит, этот консьюмер будет получать сообщения из всех партиций топика.

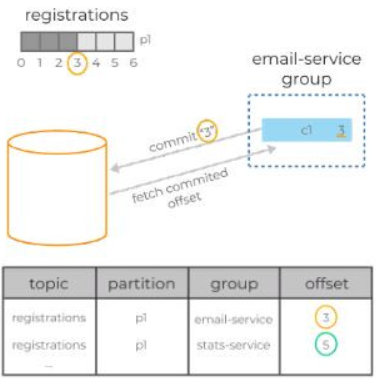
Добавив еще 2-ух консьюмеров, то партиции автоматически распределятся между ними, мы добьёмся идеального распределения нагрузки, и каждый из консюмеров в этой группе будет читать данные из одной партиции. А вот если мы добавим в группу ещё одного консьюмера (c4), то он не будет задействован в обработке сообщений вообще.

Механизм партиционирования является нашим основным инструментом масштабирования Kafka. Группы являются инструментом отказоустойчивости.

Помимо этого, механизм групп позволяет иметь несколько несвязанных между собой приложений, обрабатывающих сообщения.



В примере консьюмер в группе email-service-group, читающий партицию p1 в топике registrations, успешно обработал три сообщения с офсетами 0, 1 и 2. Для сохранения позиций консьюмер делает запрос к брокеру, коммитя офсет 3. В случае рестарта консьюмер запросит свою последнюю закоммиченную позицию у брокера и получит в ответе 3. После чего начнёт читать данные с этого офсета.



Консьюмеры вольны коммитить любой офсет (валидный, который действительно существует в этой топик-партиции) и могут начинать читать данные с любого офсета, двигаясь вперёд и назад во времени, пропуская участки лога или обрабатывая их заново.

Ключевой для понимания факт: в момент времени может быть только один закоммиченный офсет для топик-партиции в консьюмер-группе. Иными словами, мы не можем закоммитить несколько офсетов для одной и той же топик-партиции, эмулируя каким-то образом выборочный acknowledgment (как это делалось в системах очередей).

Так же kafka можно рассматривать как некую СУБД. Кафка предоставляет возможности избирательного доступа к данными с помощью [KSQL](https://www.bigdataschool.ru/wiki/ksql) – SQL-движка на базе [API Kafka Streams](https://www.bigdataschool.ru/blog/advantages-api-kafka-streams-devops-big-data.html).  Кроме того, сама [эта клиентская библиотека](https://www.bigdataschool.ru/blog/about-kafka-streams-big-data.html) для разработки приложений и микросервисов, в которых входные и выходные данные хранятся в кластерах Кафка, также включает множество функций обработки потоковых данных в режиме онлайн.

Ссылки:

https://habr.com/ru/company/southbridge/blog/550934/