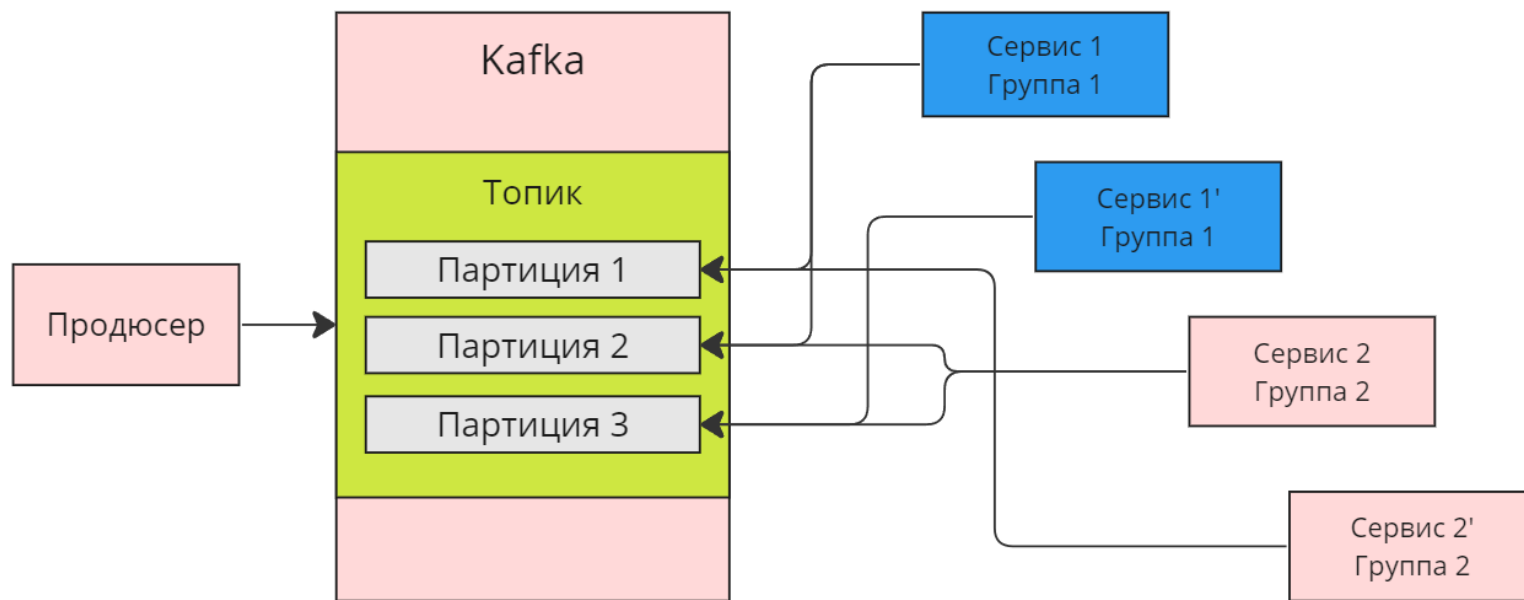


В чем смысл партиций в Kafka?

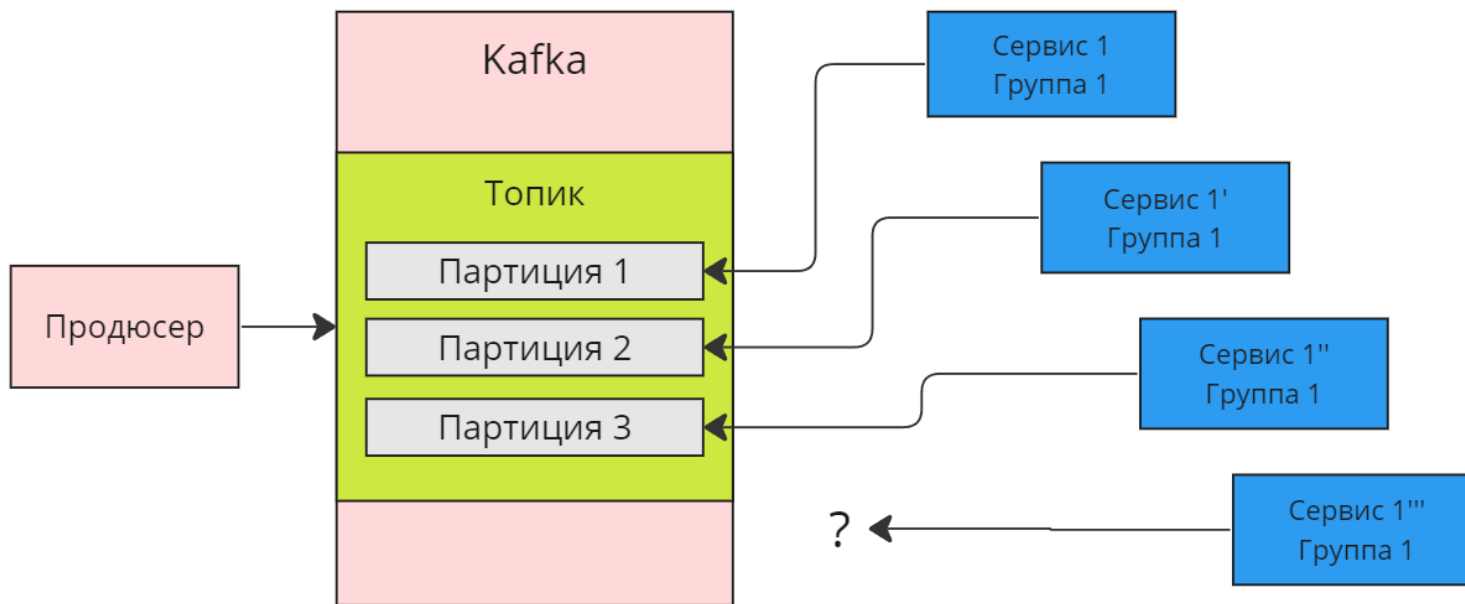
1. Параллельная обработка и увеличение пропускной способности: каждая партиция может обрабатываться независимо, что позволяет консьюмерам в рамках консьюмерской группы читать и обрабатывать сообщения параллельно. Это повышает пропускную способность системы.
2. Обработка по ключам: при наличии ключей в сообщениях можно управлять распределением сообщений на партиции таким образом, чтобы сообщения с одним и тем же ключом попадали в одну и ту же партицию. Это позволяет гарантировать сохранение порядка обработки для определенных ключей.
3. Сохранение порядка: внутри каждой партиции сообщения сохраняют порядок, в котором они были записаны. Это обеспечивает гарантию сохранения последовательности сообщений внутри одной партиции.
4. Хранение и управление данными: каждая партиция хранит свои сообщения и может быть управляема отдельно. Это позволяет лучше управлять временем хранения данных и их архивацией.
5. Изоляция ошибок: если одна из партиций перегружена или испытывает проблемы, это не влияет на работу других партиций, что обеспечивает изоляцию и устойчивость системы.

Давайте закрепим на практике. Допустим, есть топик с тремя партициями и два сервиса, которые хотят быстро вычитывать сообщения, данных очень много. Схематично это будет выглядеть так:



В группе 1 консьюмеры разделились так - первый читает партицию 1 и 2, а второй партицию 3. В группе 2 один консьюмер читает партицию 2 и 3, а второй партицию 1. Таким образом один из экземпляров каждого сервиса не нагружен полностью, и успевает читать сообщения. Так как партиции три, мы могли бы еще добавить по одному экземпляру сервиса. Тогда получили бы еще больший прирост в производительности.

Но важно то, что четвертый экземпляр к каждому сервису подключить в данную схему уже не получится. Партиции 4 нет, и новому консьюмеру просто нечего было бы читать, он просто будет запущен "без дела". Пример:

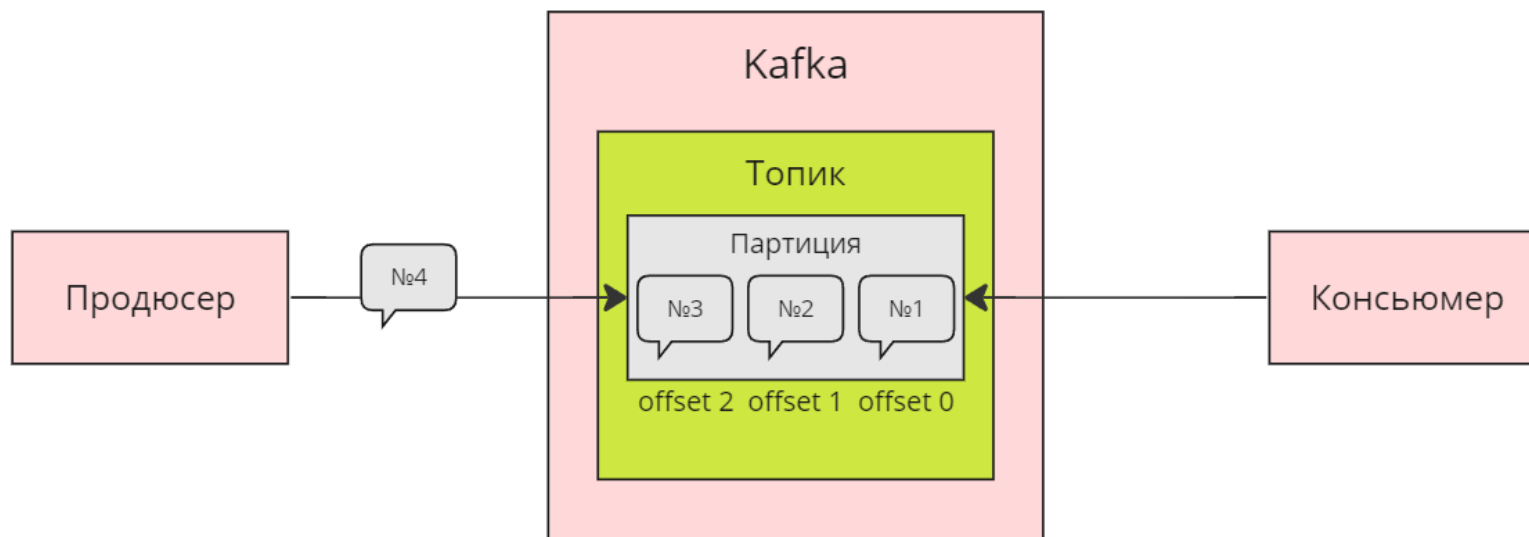


Поэтому в данном схеме при добавлении четвертого инстанса одной консьюмер группы нужно добавить еще одну партицию.

Подведем итог - один сервис может читать из всех партиций, а несколько инстансов сервиса (одна консьюмер группа) из разных.

Как происходит запись и чтение сообщений.

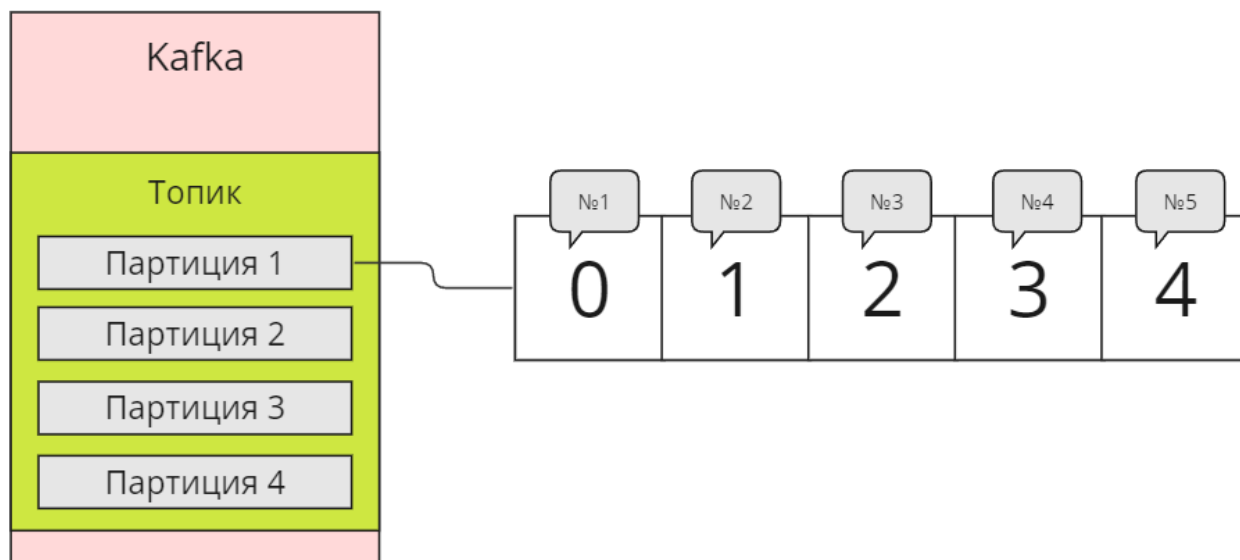
Общая схема работы выглядит так:



Продюсер отправляет сообщение, оно записывается в Kafka в строгой очередности и назначается оффсет.

Также в строгой очередности и читает консьюмер. Однако консьюмер может запрашивать и другие сообщения, обозначая оффсет (указатель).

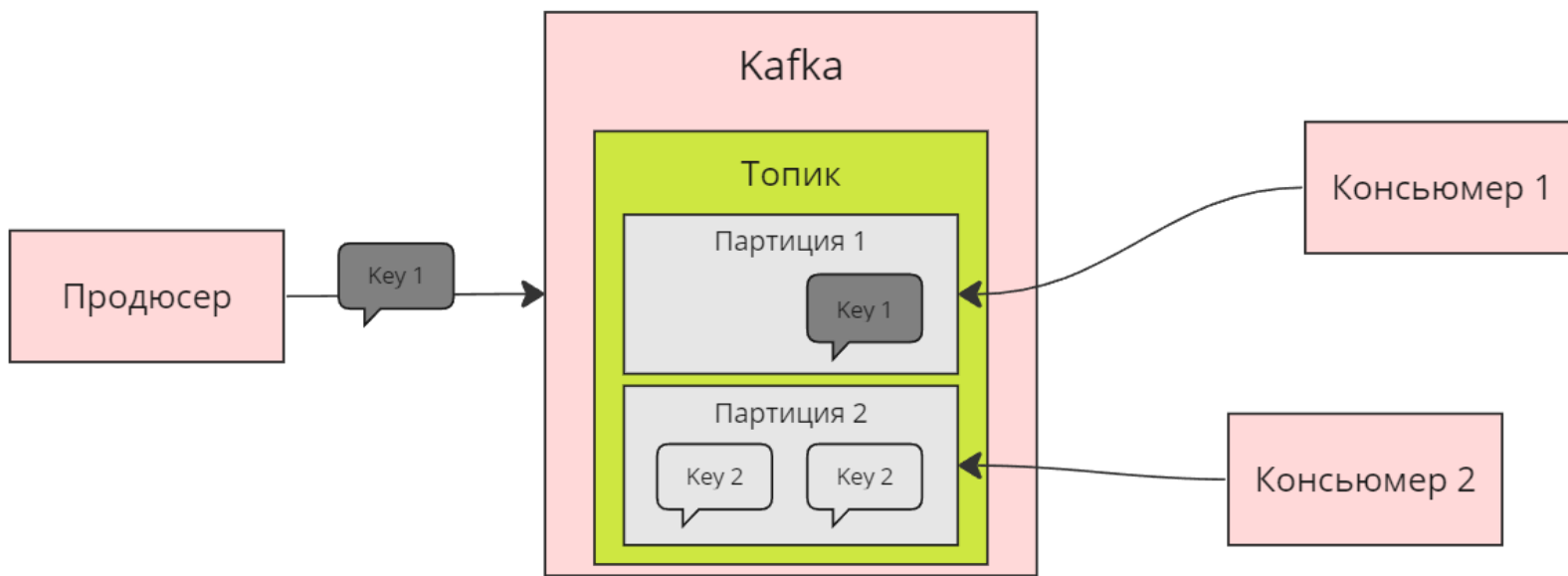
Остановимся подробнее на оффсете. У каждого консьюмера есть свой оффсет для каждой партиции, который показывает где остановился консьюмер. А у консьюмер группы общий оффсет. Разница лишь в том, как управляется позиция чтения сообщений: либо совместно внутри консьюмерской группы (общий оффсет), либо независимо для каждого консьюмера (обычный оффсет). В любом случае, хранится информация насколько "далеко" продвинулся тот или иной консьюмер в определенной партиции. При чтении каждого сообщения оффсет для определенной партиции, для определенного консьюмера, сдвигается на 1. А расчет оффсета начинается с 0. Наглядная схема для понимания:



В данном случае первое сообщение будет храниться под оффсетом 0, а четвертое под оффсетом 3. Для примера - один консьюмер может остановиться на позиции 2, это третье сообщение. А другой на позиции 4, это пятое сообщение. Еще один консьюмер может запросить сообщения начиная со второго, указав оффсет 1 при запросе.

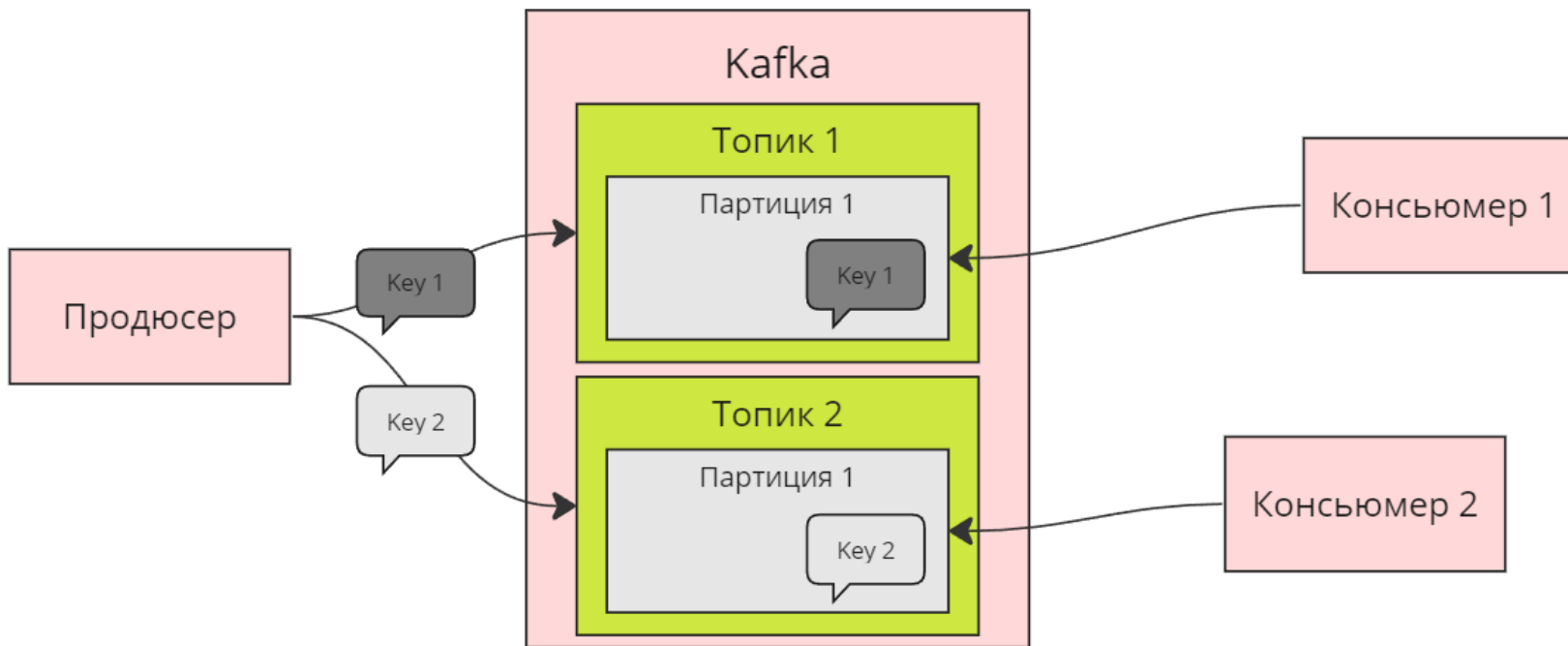
Запись сообщений от продюсера, как мы ранее обсуждали, происходит в указанную партицию или случайную, если продюсер ее не указал. Для распределения нагрузки на партиции рекомендуется не указывать партицию, а предоставлять этот выбор брокеру. Используя механизм Round robin будет стараться равномерно распределять сообщения. Но что делать, если мы хотим разделить партиции в рамках бизнес-логики?

Когда нам необходимо распределять типовые сообщения по партициям, нужно использовать настройку Partition Key. Она позволит отправлять брокеру одни сообщения в одну партицию, а другие в другую:



Согласно данной схеме, консьюмер 1 получит сообщения только с key 1, а консьюмер два сообщения с key 2. Таким образом вы можете спроектировать "умную" параллельную обработку сообщений. Однако заметим, лучше партии использовать только для масштабирования, т.е. для подключения новых консьюмеров. Бизнес-логику лучше закладывать на уровне топиков, как мы приводили пример ранее. Это позволит вам не зависеть от партий и масштабировать любую часть своего проекта без усилий.

Как бы могли это реализовать? Пример схемы:



В данной схеме:

1. Мы дорабатываем продюсер чтобы он сам определял какие сообщения в какой топик отправлять. Но при отправке не указываем в какую партицию отправлять сообщение, пусть это делает Kafka. Если будет много партиций, таким образом распределим нагрузку.
2. Мы создаем второй топик для отдельной бизнес-логики
3. Консьюмеры вычитывают информацию из разных топиков и выполняют разную бизнес-логику

При масштабировании (допустим, захотим добавить еще инстансы для всех консьюмеров) вам нужно создать в каждом топике по новой партиции и настроить консьюмеры. Других действий выполнять не потребуется.