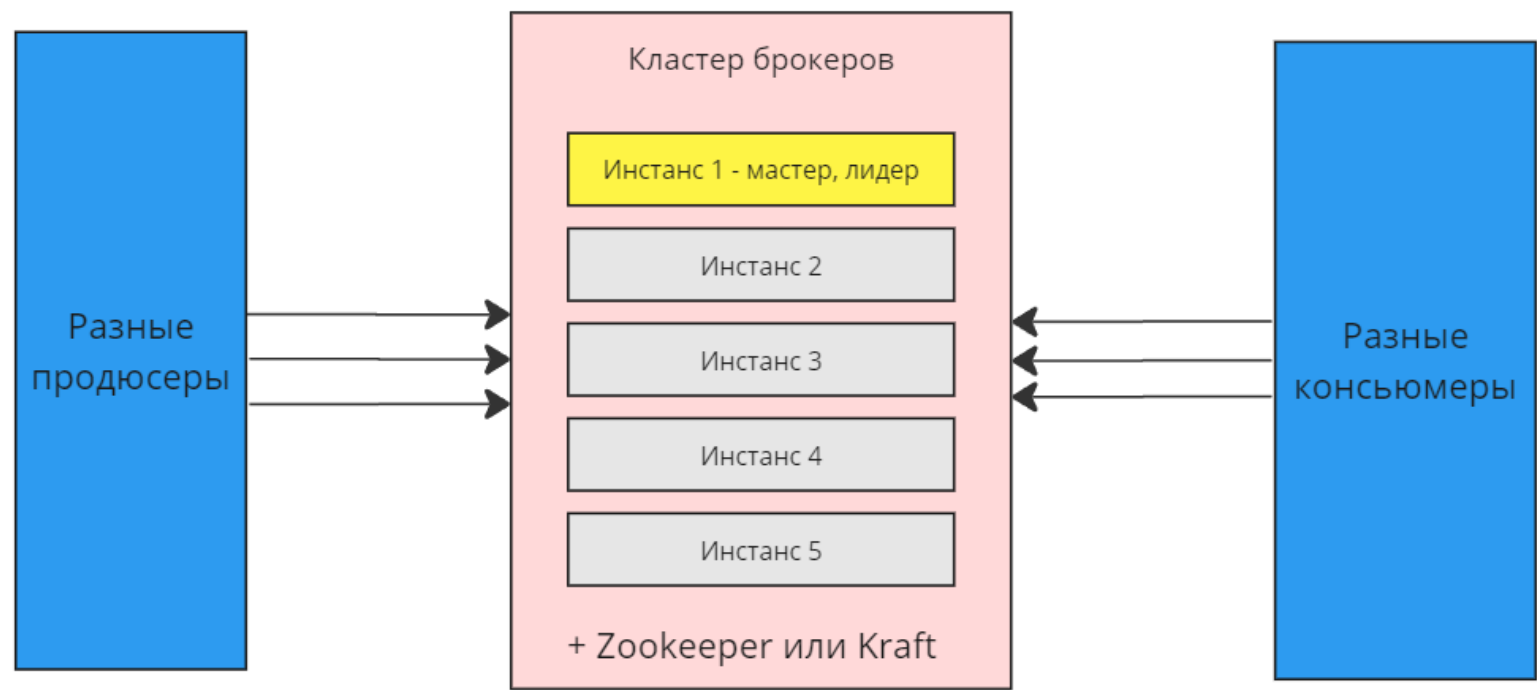


**Поговорим про физическое устройство Kafka и масштабирование.**

Инстансы брокера в Kafka объединяются в кластер, этот способ масштабирования мы рассматривали ранее.

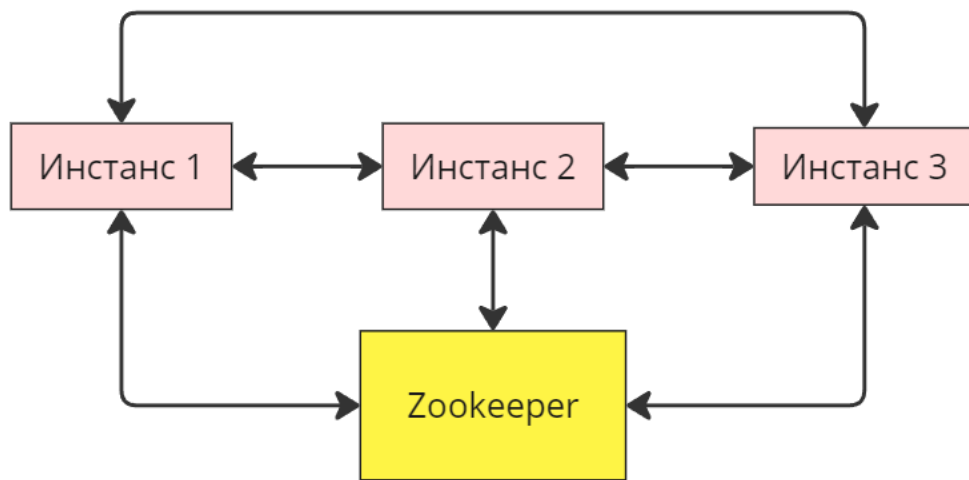
Для управления кластером используются модули Zookeeper или KRaft. Почему существуют эти модули? Надо определять кто мастер, кто нет, хранить информацию о расположении партиций и топиков по кластерам, помогать с перевыбором. Это делает не сам один инстанс, а с помощью модулей.

Общая схема взаимодействия продюсеров и консьюмеров с кластером:



В кластере существует, как мы ранее разбирали, мастер и фолловеры. Все инстансы объединяются в один кластер, у которого есть ID.

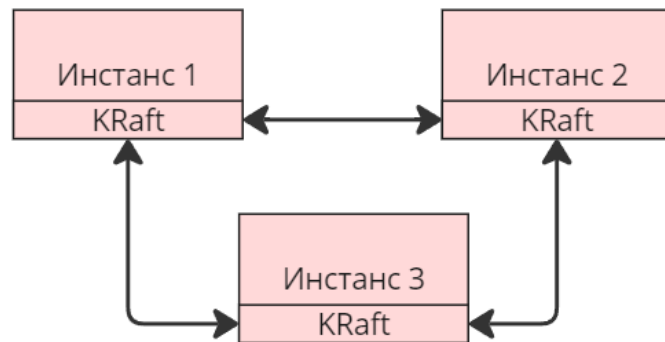
Как выглядит кластер с управлением Zookeeper:



- Каждый инстанс общается между собой
- Каждый инстанс также взаимодействует с Zookeeper
- Zookeeper хранит все метаданные о каждом инстансе в кластере

Сейчас Kafka поддерживает новый модуль, который позволяет уменьшить количество соединений и соответственно перевыбор и обмен данными происходят быстрее (обсуждали ранее, на время перевыбора система работает медленнее или вообще может стать недоступна).

Как выглядит кластер с управлением KRaft:



Технология вышла в 2021 году, и пока ее использование не так масштабно, однако плюсы очевидны:

- Метаданные хранятся в самой Kafka

- Не требуются дополнительные соединения с отдельным модулем (как в случае с Zookeeper)
- Перевыбор происходит намного быстрее

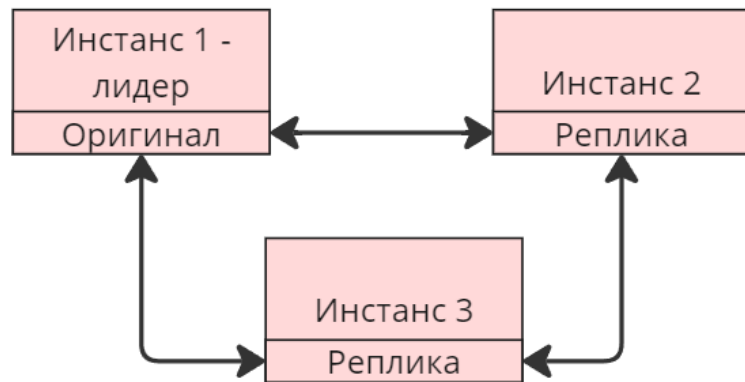
Выбор модуля зависит от опыта разработчиков и готовности работать с довольно новым модулем.

### Про репликацию в кластере.

Что такое репликация в IT? Репликация - это процесс, под которым понимается копирование данных из одного источника в другой.

Зачем существует механизм репликации? Для повышения отказоустойчивости и более надежного хранения данных. Представьте, что у вас отключена репликация и брокер с определенным топиком просто упал. В таком случае консьюмеры не смогут получить данные, т.к. топик полностью недоступен. При репликации консьюмеры смогут получить данные из топиков, где будут копии сообщений из оригинального топика.

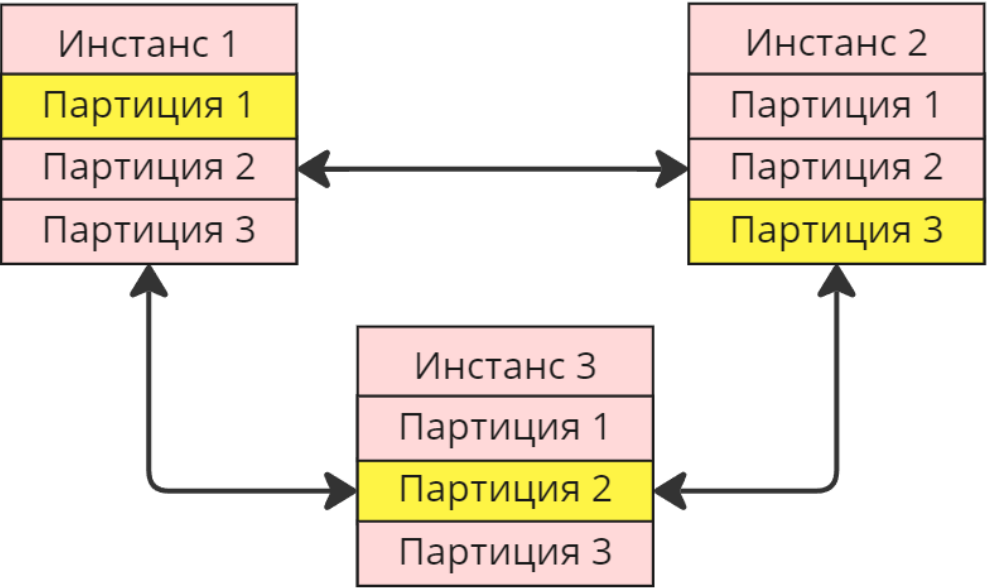
Работает это в Kafka так - есть оригинальный топик, в одном из инстансов, а есть реплики топика. Количество реплик настраивается в брокере и количество не может быть больше количества инстансов, но может быть равным нулю. Пример:



Инстанс 1 содержит топик, где хранятся оригинальные данные. Туда брокер передает сообщения. Т.е. продюсер не знает, где лидер, он просто передает на любой инстанс брокера данные. Инстанс в свою очередь, если не является мастером, передает данные мастеру, говоря "запиши, пожалуйста". Инстанс 2 и 3 хранят топик, являющиеся репликами. Туда отправляет данные инстанс 1, после того как совершил запись у себя. Подчеркнем, что мы не можем настроить для оригинального топика количество реплик = 3, т.к. дополнительных инстансов всего 2.

Как же выстраивать эффективную схему репликации? Проблема очевидна - один мастер в кластере будет всегда под нагрузкой, если всегда будет сам записывать данные для всех топиков и партиций. И если он упадет, мы полностью лишимся возможности записывать данные. Давайте погрузимся глубже, не забывая про существование партиций.

Схема с распределением нагрузки будет выглядеть так:



Для каждой партиции назначается свой лидер (выделен желтым). Остальные инстансы с такой же партицией будут репликами. Если инстанс 1 упадет, мы не сможем записывать данные всего лишь в партицию 1. Однако запись в партиции 2 и 3 будет доступна, что повышает отказоустойчивость. Когда инстанс 1 "оживет", то продолжится запись, или брокер выберет нового лидера, например инстанс 2 будет записывать в партицию 1 и в партицию 3. Такое распределение лидеров не только повышает отказоустойчивость, но и повышает производительность работы брокера.

Может случиться ситуация, когда Kafka назначит один инстанс лидером множества партиций. В таком случае есть дополнительная настройка, `reassignment`, которая позволяет переназначать лидеров.